T.C.
ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
DOKTORA TEZİ

KUANTUM BİLGİSAYARLARI VE BAZI UYGULAMALAR

Mustafa ŞAHİN
Fizik Anabilim Dalı

Tezin Sunulduğu Tarih: **13/08/2014**

Tez Danışmanı:
Prof. Dr. İhsan YILMAZ

Eş Danışman:
Doç. Dr. İbrahim TÜRKYILMAZ

ÇANAKKALE

Mustafa ŞAHİN tarafından Prof. Dr. İhsan YILMAZ yönetiminde hazırlanan ve **13/08/2014** tarihinde aşağıdaki jüri karşısında sunulan "**Kuantum Bilgisayarları ve Bazı Uygulamalar**" başlıklı çalışma, Çanakkale Onsekiz Mart Üniversitesi Fen Bilimleri Enstitüsü **Fizik Anabilim Dalı**'nda **DOKTORA TEZİ** olarak oybirliği ile kabul edilmiştir.

JÜRİ

Prof. Dr. İhsan YILMAZ ……

Başkan

Prof. Dr. Hüsnü BAYSAL ……

Üye

Prof. Dr. A. Mesut RAZBONYALI ……

Üye

Yrd. Doç. Dr. Can AKTAŞ ……

Üye

Yrd. Doç. Dr. Ali Murat TİRYAKİ ……

Üye

Sıra No:……

İNTİHAL (AŞIRMA) BEYAN SAYFASI

Bu tezde görsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, tez içinde yer alan ancak bu çalışmaya özgü olmayan tüm sonuç ve bilgileri tezde kaynak göstererek belirttiğimi beyan ederim.

Mustafa ŞAHİN

TEŞEKKÜR

Bu tezin gerçekleştirilmesinde, çalışmam boyunca benden bir an olsun yardımlarını esirgemeyen saygı değer danışman hocam Prof. Dr. İhsan YILMAZ, çalışma süresince tüm zorlukları benimle göğüsleyen eşime ve hayatımın her evresinde bana destek olan değerli aileme sonsuz teşekkürlerimi sunarım.

Mustafa ŞAHİN

Çanakkale, Ağustos 2014

SİMGELER VE KISALTMALAR

$\mathcal{H}$	Hilbert Uzayı
$\|\psi\rangle$	Kuantum sistemin durumu
CNOT	Kontrollü Değil Kapısı (Controlled-Not Gate)
RAM	Rastgele Erişim Belleği (Random Access Memory)
QRAM	Quantum Random Access Memory
QDil	Quantum Dynamically Interpreted Language
QIL	Kuantum Ara Dili (Quantum Intermediate Language)
QIL-G	Kuantum Ara Dil Oluşturucu (Quantum Intermediate Language Generator)
QVM	Kuantum Sanal Makinesi (Quantum Virtual Machine)
QDM	Kuantum Donanım Yöneticisi (Quantum Device Manager)
QOPINT	QOperator Yorumlayıcısı (Qoperator Interpreter)
$\mathcal{E}$	Null eleman
QVM-OE	Quantum Virtual Memory-Object Eraser
$\mathcal{X}$	Pauli $\mathcal{X}$ matrisi
Y	Pauli Y matrisi
Z	Pauli Z matrisi
$\oplus$	Modül ikiye göre toplama
$\otimes$	Tensörel Çarpım
Γ	Boş sembol içermeyen alfabe
URI	Birleşik Kaynak Tanımyayıcı (Unified Resource Identifier)
DAG	Yönlü Çevrimsiz Graf (Directed Acyclic Graph)

ÖZET

KUANTUM BİLGİSAYARLARI VE BAZI UYGULAMALAR

Mustafa ŞAHİN
Çanakkale Onsekiz Mart Üniversitesi
Fen Bilimleri Enstitüsü
Fizik Anabilim Dalı Doktora Tezi
Danışman : Prof. Dr. İhsan YILMAZ
13/08/2014, 107

Kuantum bilgisayarları atom ve atom altı parçacıkların davranışlarından yararlanılarak laboratuvar ortamında geliştirilmeye çalışılmaktadır. Bu bilgisayarlar kübit adlandırılan iki seviyeli kuantum mekaniksel sistemleri bilgi depolama ve temel işlem birimi olarak kullanmaktadır. Klasik bitlere göre mümkün olmayan aynı anda iki durumu birden sergileyebilme kübitler için mümkündür ve bu durum hesaplamalarda kullanılabilmektedir. Bu özellik kuantum bilgisayarlara üstün işlem yeteneği kazandırmaktadır. Klonlanamama kübitlerin kopyasının alınmasını engellemekte ve kuantum bilgisayarların doğasına veri güvenliğini katmaktadır. Farklı yeteneklere sahip bu bilgisayarların kullanılabilmesi için programlanması ve neler yapması gerektiği söylenmelidir. Tezimde kuantum bilgisayarların üstün işlem yeteneklerinin kullanılmasına, mevcut kuantum algoritmalarının yazılmasına imkan sağlayan yeni bir kuantum programlama dili geliştirilmiştir.

Anahtar sözcükler: Kuantum Bilgisayarlar, Kuantum Algoritmalar, Kuantum Programlama Dilleri.

ABSTRACT

QUANTUM COMPUTERS AND SOME APPLICATIONS

Mustafa ŞAHİN

Çanakkale Onsekiz Mart University

Natural and Applied Sciences

Doctoral Dissertation in Physics

Advisor : Prof. Dr. İhsan YILMAZ

13/08/2014, 107

Implementation of quantum computers is based on the behavior of atomic and subatomic particles and those computers are being trying to be produced in laboratuaries. Quantum computer is a computer design which uses two-level quantum mechanical systems called as qubit for information storage and data processing. Qubits may be two-possible state simultaneously and this feature gives the ability to perform high parallel processing. No-cloning therom adds internal security concepts in the quantum computers. In order to use these different features, quantum computers must be programmed. In this thesis, a new quantum programming language was developed that allows the use of quantum computers with high processing cability and writing present quantum algorithms.

Keywords: Quantum Computers, Quantum Algorithms, Quantum Programming Languages.

İÇİNDEKİLER

Sayfa No

ŞEKİLLER DİZİNİ

Sayfa No

ÇİZELGELER DİZİNİ

Sayfa No

BÖLÜM 1
GİRİŞ

Mevcut klasik bilgisayarlarda saklanan ve üzerinde hesaplama yapılan en temel bilgi dijital bittir. Bit 0 veya 1 değerlerinden sadece birine sahip olabilmektedir.

Günümüzde silikon teknolojisinin sınırlarına gelinmesi bizleri atom boyutlarında çalışmaya zorlamaktadır. Bu boyutlarda klasik fizik yerine kuantum mekaniği geçerli olmaktadır. Kuantum mekaniği kurallarına dayanarak bilgi işleme ve iletişim fikrinin ortaya atılmasıyla kuantum bilgisayarları kavramı da ortaya çıkmıştır.

Kuantum bilgisayarlar atom ve atom altı parçacıkların davranışlarından yararlanılarak gerçekleştirilmeye çalışılmaktadırlar. Kuantum bilgisayarlar; iyon veya atomların tuzaklanması, atom ve atom altı parçacıkların manyetik alanda davranışlarının belirlenmesi, atomlarda elektronların farklı yörüngelerde olması, ışığın polarizasyonu gibi çeşitli yöntemler kullanılarak gerçekleştirilmeye çalışılmaktadır. Fiziksel gerçekleştirim kuantum bilgisayardaki bilgiyi ifade edecek kübitlerin nasıl olacakları ve nasıl işleneceklerini belirlemektedir.

Kuantum bilgisayarları, kübit (qubit: Quantum Bit) olarak adlandırılan iki seviyeli kuantum mekaniksel sistemleri bilgi depolama ve temel bilgi işleme elemanları olarak kullanan bir bilgisayar tasarımıdır. Kuantum bilgisayarlarda bilgiyi ifade edecek birimlere kübit denilmektedir. Her bir kübit klasik 1 ve 0 değerlerini belirli olasılık genlikleriyle aynı anda içerebilmektedir.

Kuantum mekaniğine göre, herhangi bir fiziksel sistem olası durumların sadece birinde olmayabilir. Bu durumlardan birçoğunda aynı anda belirli olasılık genlikleriyle bulunabilir. Buna üst üste gelme veya süper pozisyon prensibi denilmektedir. Bu özellik, kuantum hesaplamaya yüksek paralel işlem yapma yeteneği kazandırmaktadır.

Örneğin, 32 kübit üzerinde işlem yapan kuantum bilgisayarı iki üzeri otuz iki yani aynı anda 4 milyar veriyi tek bir seferde işleyebilmektedir. Bu ise aynı anda 4 milyar klasik bilgisayarın eş zamanlı olarak çalışmasına eşdeğerdir.

Shor (1994,1997) 'un geliştirmiş olduğu algoritma kuantum bilgisayarları alanındaki araştırmaların birden bire artmasına sebep olmuştur. Bu algoritma kuantum bilgisayarları ile bir sayının asal çarpanlarını bulmakta ve bu işlemi inanılmaz bir hızla yapmaktadır. İnternetteki bilgilerin şifrelenmesinde ve şifrelenmiş iletişimde RSA isimli bir algoritma

kullanılmaktadır. Bu algoritma ile şifrelemede kullanılan anahtar çok büyük iki sayının çarpımından oluşan, 128 (256 veya daha fazla) bit ile gösterilebilen bir sayıdır. Günümüz klasik bilgisayarlarında bu anahtarın elde edilmesi ve çarpanlarına ayrılması çok uzun bir süre aldığı için RSA algoritmasının güvenli olduğu düşünülmektedir. Bununla birlikte, yeterli kübite sahip bir kuantum bilgisayarı çarpanlarına ayırma işlemini çok kısa sürelerde başarmaktadır. Buna benzer olarak kuantum bilgisayarları alanında Grover'ın arama algoritması Grover (1998), Deutch (1985) ve Simon (1997) algoritmaları gibi farklı amaçlar için geliştirilen birçok algoritma bulunmaktadır.

Güvenli iletişimin sağlanması son derece önemlidir. Kuantum mekaniğinin doğasında bulunan klonlanamama gibi özellikler kuantum bilgisayarlarda verilerin daha güvenilir iletilmesini sağlamaktadır.

Bahsedilen gelişmelerin kullanılabilmesi için kuantum bilgisayarların programlanması gerekmektedir. Programlama dillerinin gelişimi çerçevesinde kuantum programlama ve kuantum programlama dilleri yeni bir kavramdır. Kuantum mekaniğindeki üst üste binme ilkesi, dolanıklık ve klonlanamama gibi farklı özellikler kuantum programlama dillerinin klasik programlama dillerinden farklı olması gerektiğini göstermektedir.

Yeni geliştirdiğimiz kuantum programlama dili, kuantum bilgisayarlar ile klasik bilgisayarların aynı anda çalışmasını sağlamaktadır ve kuantum bilgisayarlarının fiziksel gerçekleştiriminden (iyon veya atomların tuzaklanması, atom ve atom altı parçacıkların manyetik alanda davranışlarının belirlenmesi, atomlarda elektronların farklı yörüngelerde olması, ışığın polarizasyonu v.b.) bağımsızdır. Bu amacı gerçekleştirmek için yazılan kodlar kuantum ara diline (QIL-Quantum Intermediate Language) çevrilmekte ve kuantum sanal makinesinde (QVM-Quantum Virtual Machine) çalıştırılmaktadır.

Kuantum yada klasik sistemin dinamik olarak verdiği geri dönüş değerleri ya da hata durumlarına göre kuantum sistemin çalışması farklılaşabilir. Bu nedenle kuantum programlama dilinde çalışan kodun, çalışma zamanında değiştirilmesi gerekebilir. Bu durumu sağlamak için qoperator isminde yeni bir veri yapısı geliştirilmiştir. Bu veri yapısının çalıştırılması, çalışma zamanında yorumlanarak yapılmaktadır; böylece kuantum programlama dilimiz dinamiklik kazanmaktadır.

1.1. Kuantum Mekaniği

Kuantum mekaniği çok küçük parçacıkların davranışlarını açıklamak için çıkmıştır. Bir parçacığın durumu $\psi(x,t)$ ile gösterilen dalga fonksiyonuyla tanımlanır. Bu fonksiyona olasılık dalga fonksiyonu denilir.

Bir parçacığın zaman içerisindeki durumu ise Schrödinger diferansiyel denklemi ile aşağıdaki gibi tanımlanır.

$$i\hbar \frac{\partial \psi(x,t)}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2}\psi(x,t) + V(x,t)\psi(x,t) \qquad \textbf{(1.1)}$$

Burada $\hbar = \frac{h}{2\pi}$ ve $\psi(x,t)$ potansiyeldir (Griffiths, 2004). Daha ayrıntılı bilgi için temel kuantum mekaniği kitaplarına bakılabilir (Griffiths, 2004).

1.1.1. Kuantum mekaniğinin temel postulatları

Kuantum mekaniği deneylerle uyum içerisindeki postülatlar temelinde çalışmaktadır. Bu postülatları kısaca aşağıdaki gibi özetleyebiliriz:

Fiziksel sistemin durumu $\mathcal{H}$ Hilbert uzayında $|\psi>$ şeklinde gösterilen bir vektör ile temsil edilir.

$$|\psi> = \sum_{i=1}^{n} \alpha_i |v_i> \qquad \alpha_i \in \mathbb{C} \qquad \textbf{(1.2)}$$

$$\sum_{i=1}^{n} |\alpha_i|^2 = 1 \qquad \textbf{(1.3)}$$

Burada $|v_i>$ ler uzayın bazlarıdır (Nielsen ve Chuang, 2000).

Fiziksel sistemin ölçülebilen (gözlenebilen) değerleri(momentum, açısal momentum, enerji, spin, vb.) operatörler ile temsil edilir. Bu operatörler aşağıdaki gibi hermityen özelliğine sahiptir (Nielsen ve Chuang, 2000).

$$A^{\dagger} = A \qquad \textbf{(1.4)}$$

Bu operatörlerin öz vektörleri ise Hilbert uzayında ($\mathcal{H}$) baz teşkil eder. Bir sistemin durumu şağıdaki şekilde tanımlanan öz değer denklemi ile kolayca anlaşılabilir (Nielsen ve Chuang, 2000).

$$A|a_i>=\lambda_i|a_i> \qquad \textbf{(1.5)}$$

Burada λ_i lere öz değer ve $|a_i>$ ler bu öz değere karşılık gelen öz vektörlerdir. Sistemin durumu bu operatörün öz vektörleri cinsinden yazılır. A operatörü ölçülürse; ölçüm sonucunda A operatörünün öz değerlerinden biri (λ_i) ölçülmüş olur ve sistemin yeni durumu ölçülen öz değere karşılık gelen öz vektör ($|a_i>$) olur (Griffiths, 2004).

1.1.2. Zaman içerisinde bir kuantum sistemin durumunun değişimi

Bilindiği gibi $|\psi>$ durumundaki bir sistemin zaman içerisindeki değişimi

$$i\hbar\frac{\partial|\psi>}{\partial t}=H|\psi> \qquad \textbf{(1.6)}$$

şeklindedir (Nielsen ve Chuang, 2000). Buradaki H sistemin toplam enerji operatörü olan Hamiltonyen operatörüdür. Zamandan bağımsız Hamiltonyenler için bu diferansiyel denklemin çözümü

$$i\hbar\frac{\partial|\psi>}{\partial t}=H|\psi> \qquad \textbf{(1.7)}$$

şeklindedir (Nielsen ve Chuang, 2000). Burada sistem durumunun zaman ile değişimini sağlayan operatör

$$i\hbar\frac{\partial|\psi>}{\partial t}=H|\psi> \qquad \textbf{(1.8)}$$

şeklinde tanımlanır (Nielsen ve Chuang, 2000). Bu operatör birimsel bir operatördür. Eğer H zamana bağlıysa çözüm

$$|\psi(t)>=\tau e^{-\frac{i}{\hbar}\int_0^t H(t)dt}|\psi(0)> \qquad \textbf{(1.9a)}$$

$$\tau[A(t_1)B(t_2)] = \begin{cases} A(t_1)B(t_2), & t_1 > t_2 \\ B(t_2)A(t_1), & t_1 \le t_2 \end{cases} \quad \textbf{(1.9b)}$$

Burada τ zaman-sıralama operatörüdür (Nielsen ve Chuang, 2000).

1.1.3. Üst üste gelme prensibi ve ölçme

Kuantum mekaniğine göre, herhangi bir fiziksel sistem olası durumların sadece birinde olmayabilir. Bu durumlardan birçoğunda aynı anda belirli olasılık genlikleriyle bulunabilir. Buna üst üste gelme ya da süper pozisyon prensibi denilir. Bu özellik, kuantum hesaplamaya yüksek paralel işlem yapma yeteneği kazandırmaktadır. Aynı anda birden fazla durumda bulunan fiziksel sistemde herhangi bir ölçme işlemi gerçekleştiğinde, sistemin bu süper pozisyon hali bozulur ve durumlardan sadece birisinin değeri elde edilebilir.

1.1.4. EPR deneyi ve dolanıklık

Einstein, Podolsky ve Rosen (1935) kuantum mekaniğinin tutarsızlığını göstermek için bir deney tasarlamışlardır. Bu deneye göre birbirleriyle etkileşime geçen iki parçacığın durumları aralarındaki mesafeden bağımsız olarak ölçülebilir ve bu ölçümler birbirleriyle ilişkilidir. Parçacıklardan biri üzerinde yapılan bir ölçme işlemi ile diğeri hakkında bir bilgi elde edilir.

Bu durumu basitçe elektronların $1/2$-spinleri ile gösterelim. Bir EPR durumu aşağıdaki gibi gösterilebilir (Einstein, Podolsky ve Rosen, 1935).

$$|\psi_{12}>=\frac{|0_1 0_2> + |1_1 1_2>}{\sqrt{2}} \quad \textbf{(1.10)}$$

Burada alt indis elektron numarasını göstermektedir. Burada iki elektron birbirleriyle etkileşime geçmiş ve yukarıdaki dolanık durumu oluşturmuştur. Burada herhangi bir kübitte yapılan ölçmenin diğer kübitin durumunu belirlediğini görebiliriz.

Dolanık durumlar birçok kuantum algoritmasına temel teşkil etmektedirler ve halen yoğun şekilde araştırılan bir konudur.

1.2. Kuantum Bilgi Teorisi

Kuantum bilgi teorisinin temel amacı kuantum mekaniğinin bize sunmuş olduğu imkanları kullanarak klasik bilgi teorisi ile başaramadığımız algoritmaları ve iletişim protokollerini gerçekleştirmektir.

Moore (1965) kanununa göre bilgisayarın işlemcisi üzerindeki transistör sayısı her iki yılda iki katına çıkmaktadır. Bu durumda hesaplama aygıtları küçülerek gelebilecek en küçük fiziksel sınırlarına gelmiştir. Daha küçük seviyelere indiğimizde klasik fizik kuralları yerine kuantum fiziği kuralları kullanılmaktadır. Böylece gelecekteki bilgisayarların tasarımlarında ister istemez kuantum seviyedeki sistemlerin çalışmaları düşünülecektir.

Kuantum algoritmaları ve iletişim protokolleri, kuantum sistemlerinin nasıl kontrol edilebileceğini ve istediğimiz hesaplamaların daha yüksek performansla ya da daha güvenlikli olarak nasıl gerçekleştirilebileceğini gösterirler.

Benioff (1980), ilk kez klasik bilgisayarların temeli olan Turing makinesinde bulunan teyp yerine iki-durumlu kuantum durumlarını kullanılabileceğini önermiştir. Turing makinesindeki okuma/yazma kafası yerine spin durumlarını değiştirebilen kuantum mekaniksel etkileşimi önermiştir. Turing makinesinin kuralları yerine Schrödinger denklemi kullanılmıştır.

Feynman (1982,1985) ise kuantum hesaplamaların bir gün kuantum bilgisayarların temelini oluşturabileceğinden bahsetmiştir.

Bennett ve Brassard (1984), Feynman'dan iki yıl sonra ilk kuantum kriptografi protokolünü tanımlamışlar ve kuantum fiziğinin bilgi teorisindeki yeni etkilerini göstermişlerdir.

Deutsch (1985) tarafından ilk Kuantum Turing Makinesi geliştirilmiştir. Bu modelde kuantum veriler süper pozisyon şeklinde bulunabilir ve işlemciler ile hesaplamaya girebilirler. Deutsch bu makelesinde ayrıca kuantum bilgisayarların klasik bilgisayarların gerçekleştiremeyeceği işlemleri başarabileceğini de ortaya koymuştur. Böylece kuantum paralelliğini kullanan ilk kuantum algoritmayı da tanımlamıştır.

Deutsch(1985) geliştirmiş olduğu algoritma ile bir ikili fonksiyonun sabit ya da dengeli bir fonksiyon olup olmadığını bulmanın klasik bilgisayarlardan daha hızlı bir şekilde hesaplanabileceğini göstermiştir. Deutsch ve Jozsa (1992), önceki algoritmayı daha da geliştirmişlerdir.

Ekert (1991), kuantum ölçeklerinde karşılaştığımız dolanık durumların iletişim protokollerinde kullanılabileceğini göstermiştir. Dah asonraları, uzak mesafeler arasında kuantum kriptografi kullanan iletişim protokolleri gerçekleştirilmiştir (Rosenberg D., Peterson C. G. ve ark., 2009), (Jouguet P., Kunz-Jacques S., ve ark., 2012).

Simon (1994) kendinden önceki yaklaşımları kullanarak herhangi bir klasik algoritmadan üstel olarak daha hızlı şekilde $f:\{0,1\}^n \to \{0,1\}^m$ fonksiyonunda sadece ve sadece $x = y \oplus s$ için $f(x) = f(y)$ yi sağlayan gizli $s \in \{0,1\}^n$ i bulan bir algoritma geliştirmiştir.

Kuantum bilgisayarlara ve kuantum bilgi teorisine dikkatleri çeken algoritma Shor tarafından 1994 de geliştirilmiştir. Shor (1994), bir kuantum bilgisayarın inanılmaz hızlı bir şekilde çok büyük bir tamsayının asal çarpanlarına nasıl ayrılabileceğini göstermiştir. Ayrıca 1997 de ayrık logaritmaların hesaplanmasıyla ilgili bir makale yayınlamıştır (Shor,1997). Kuantum bilgisayara sahip olunduğunda bu algoritmalar sayesinde RSA kripto yöntemi kolaylıkla kırılabiliyor, böylece internetin ve bilgilerin güvenliği kalmamaktadır. Bu nedenle kuantum algoritmalara ve bilgisayarlara olan ilginin son derece artmasını sağlamıştır.

Kuantum algoritmalarının gelişim sağladığı başka bir problem alanı da arama işlemleridir. Grover (1997) düzensiz bilgiler içerisinden bir bilginin polinomsal hızlanma sağlanarak bulunabileceğini göstermiştir. Bu algoritma, kuantum olasılık genliklerinin arttırılması yöntemiyle çalışmaktadır.

Kuantum hesaplama teorisi kullanılarak klasik rastgele yürümenin kuantum modeli Kempe(2003) ve Koŝik(2003) tarafından gösterilmiştir.

AND-OR ağaçları iki oyunculu kombinasyon oyunlarında ve oyun teorisinde karşımıza çıkan bir problemdir. Bu problemin kuantum bilgisayar algoritması ilk kez Farhi ve arkadaşları(2007) tarafından tanımlanmış ve daha sonra Ambainis ve arkadaşları (2007) tarafından da daha iyi bir hale getirilmiştir.

Kuantum algoritmaları ile ilgili daha ayrıntılı bilgiler (Mosca ve Smith, 2011) ve (Childs ve van Dom, 2010) dan elde edilebilir.

1.2.1. Kuantum bilgisayarlar ve kuantum bilginin ifadesi

Kuantum mekaniği kurallarına dayanarak bilgi işleme ve iletişim fikrinin ortaya atılmasıyla kuantum bilgisayarları kavramı da ortaya çıkmıştır (Feynman,1982,1985). Kuantum bilgisayarlar, kübit (qubit:Quantum Bit) olarak adlandırılan iki seviyeli kuantum mekaniksel sistemi bilgi depolama ve temel bilgi işleme elemanları olarak kullanan bir bilgisayar tasarımıdır (Kaye, Laflamme ve Mosca, 2011).

Kuantum bilgisayarlarda da klasik hesaplamadaki gibi bilgiyi ifade edecek birimlere ihtiyaç vardır. Kuantum bilgisayarlarda ve hesaplamada kullanılan , bilgi tutan birimlere kübit denilmektedir. $|0>$ kuantum durumu mantıksal (0) a, $|1>$ kuantum durumu da mantıksal (1) e karşılık gelmektedir. $|0>$ ve $|1>$ ile ifade edilen kuantum durumları herhangi bir kuantum sisteminin farklı iki durumunu göstermektedir (Nielsen ve Chuang, 2000).

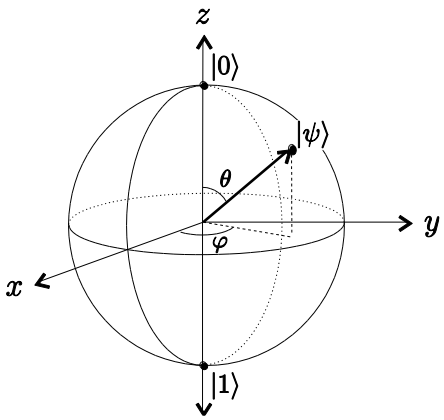
Örneğin (Nielsen ve Chuang, 2000):

Spin-1/2 parçacık için; spin-yukarı durumu: $|0>$, spin-aşağı durumu: $|1>$ e karşılık gelebilir.

Fotonlar için; yatay polarizasyon: $|0>$, dikey-polarizasyon: $|1>$ e karşılık gelebilir. Daha öncede bahsedildiği gibi kuantum sistemlerde bir parçacığın durumu birden fazla durumun süper pozisyonu olabilir. İki durumlu kuantum sistemleri için $\mathcal{H}$ Hilbert uzayının baz vektörleri ($|0>$ ve $|1>$) seçilebilir. Bu bazlara, kuantum bilgisayarlarda hesaplama bazları denilmektedir (Nielsen ve Chuang, 2000). Bir kübitin durumu :

$$|\psi>=\alpha|0>+\beta|1> \qquad \alpha,\beta\in\mathbb{C} \tag{1.11}$$

şeklinde ifade edilir. Bir kübit faz uzayında Bloch küresi üzerinde aşağıdaki şekilde gösterilebilir (Nielsen ve Chuang, 2000).



Şekil 1.1. Bloch küresi üzerinde bir kübitin durumu (Nielsen ve Chuang, 2000, syf.15)

$$|\psi>=\cos(\frac{\theta}{2})|0>+e^{i\varphi}\sin(\frac{\theta}{2})|1> \tag{1.12}$$

Hesaplama bazlarını oluşturan kübitlerin $\{|0>$ ve $|1>\}$ şeklindeki gösterimine Dirac gösterimi de denilmektedir(Nielsen ve Chuang, 2000). Buna ilaveten kübitler vektörel olarak aşağıdaki şekilde gösterilebilirler.

$$|0>=\begin{pmatrix}1\\0\end{pmatrix}, \qquad |1>=\begin{pmatrix}0\\1\end{pmatrix} \tag{1.13}$$

Birden fazla kübitten oluşan sistemler olabilir. Bu sistemlere kuantum yazmaçları denilmektedir (Nielsen and Chuang, 2000). Birden fazla kübitten oluşan sistemi vektörel olarak ifade edebilmek için tensörel çarpım kullanılmaktadır. Örneğin (Kaye, Laflamme ve Mosca,2011);

$$|0>\otimes|0>=|00>=\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\\0\\0\\0\end{pmatrix} \tag{1.14}$$

$$|0>\otimes|1>=|01>=\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}0\\1\\0\\0\end{pmatrix} \tag{1.15}$$

Birden fazla kübitten oluşan sistemin durumu, sistemi oluşturan her kübitin durumunu tanımlayan uzayların birleşimiyle oluşan daha büyük bir uzaydır. Örneğin iki alt sistemden oluşan bir uzay aşağıdaki şekilde ifade edilebilir (Nielsen and Chuang, 2000).

$$\begin{aligned}
&|\psi_1> = \alpha_1|0>+\beta_1|1> \qquad |\psi_1>\in\mathcal{H}_1\\
&|\psi_2> = \alpha_2|0>+\beta_2|1> \qquad |\psi_2>\in\mathcal{H}_2\\
&|\psi>=|\psi_1>\otimes|\psi_2> \qquad |\psi>\in\mathcal{H}=\mathcal{H}_1\otimes\mathcal{H}_2
\end{aligned} \tag{1.16}$$

$$\begin{aligned}
&|\psi>=\alpha_1\alpha_2|0>\otimes|0>+\alpha_1\beta_2|0>\otimes|1>+\beta_1\alpha_2|1>\otimes|0>+\beta_1\beta_2|1>\otimes|1>\\
&=\alpha_1\alpha_2|00>+\alpha_1\beta_2|01>+\beta_1\alpha_2|10>+\beta_1\beta_2|11>\\
&=c_0|00>+c_1|01>+c_2|10>+c_3|11>
\end{aligned} \tag{1.17}$$

$$\sum_{i=0}^{3}|c_i|^2=1 \tag{1.18}$$

Burada $c_0 = \alpha_1\alpha_2, c_1 = \alpha_1\beta_2, c_2 = \beta_1\alpha_2, c_3 = \beta_1\beta_2$ dır. Bununla birlikte iki durumun tensör çarpımı şeklinde tanımlanamayan durumlarda vardır. Bu durumlar daha önce bahsedildiği gibi dolanık durumlardır.

$$|\beta_{00}>=\frac{|00>+|11>}{\sqrt{2}}, \quad |\beta_{10}>=\frac{|00>-|11>}{\sqrt{2}}$$
$$|\beta_{01}>=\frac{|01>+|10>}{\sqrt{2}}, \quad |\beta_{11}>=\frac{|01>-|10>}{\sqrt{2}} \qquad \textbf{(1.19)}$$

şeklinde gösterilirler ve Bell bazları adlandırılırlar (Nielsen ve Chuang, 2000). Bu durumlar kuantum teleportasyon ve kriptografi gibi birçok algoritma ve iletişim yöntemi için temel teşkil etmektedir (Bennett ve Brassard, 1984), (Bennett ve Wiesner, 1992).

1.3. Hesaplanabilirlik

Klasik hesaplama kavramı çeşitli modellerle ifade edilmektedir. Bu modellerin hepsi aynı hesaplamayı ifade edebilmektedir. Bu modellerin en önemlileri şunlardır:

a) Turing Makinesi: Alan Turing (1937) tarafından keşfedilmiştir. Turing bir bilgisayarın matematik modelini gerçekleştirmiştir. Bu model bilgisayarların neler yapıp yapamayacağının gösteriminde ve karmaşıklık analizlerinde kullanılmaktadır.

b) Rastgele Erişim Makinesi (RAM): Bu yazmaçlara sahip bir bilgisayarın örneğidir. Yazmaçlar hesaplamalarda kullanılmak üzere verileri saklarlar. Hesaplamalar sonucundaki veriler yine yazmaçlara kaydedilir. Günümüz bilgisayarları bu modelin temel özelliklerini içerir ve bu model programlama dillerinin temel hesaplama modelidir (Cook ve Reckhow, 1973), (Savage J.E., 2008).

c) Mantık Devreleri: Günümüz bilgisayarlarının temel birimleridir. Sınırlı belleğe sahip tüm bilgisayarlar mantık devreleri kullanılarak üretilebilirler. $f:\{0,1\}^n \to \{0,1\}^m$ şeklinde bir ikili (binary) fonksiyonun hesaplanmasında kullanılırlar. Burada n girdi m çıktı bulunmaktadır (Savage J.E., 2008).

d) Lambda Calculus: Church (1936) yılında tanımlanmıştır. Lisp, Scheme, ML gibi birçok fonksiyonel programlama dilinin temelidir.

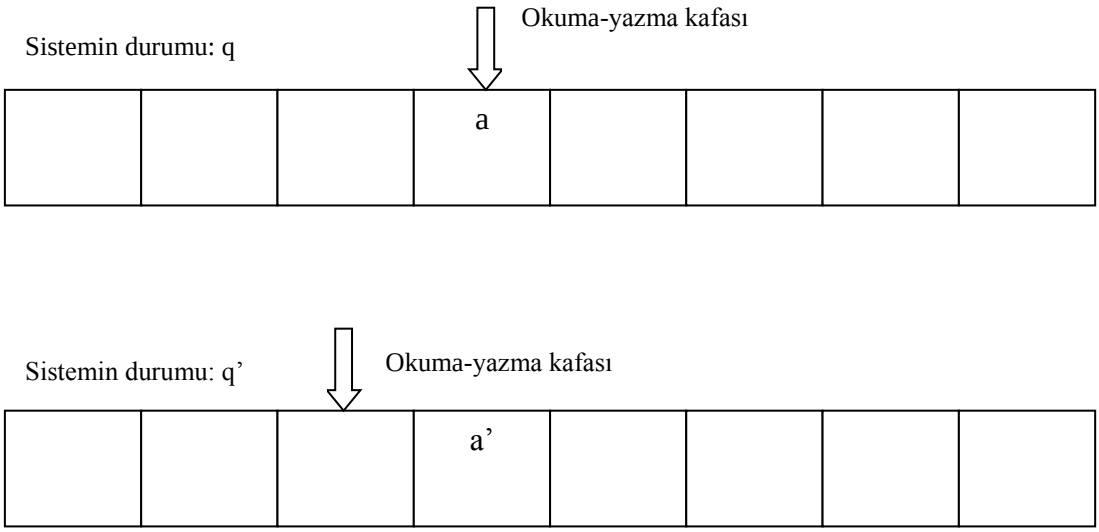
e) Evrensel Programlama Dilleri: Yaygın kullanılan hesaplama modelidir (Sebesta, 2008).

Herhangi bir hesaplama modelinde gerçekleştirilebilen bir problem diğer tüm modellerde de gerçekleştirilebilirdir.

Church-Turing tezi bize doğal olarak hesaplanabilir görülen her fonksiyonun evrensel Turing makinesi tarafından hesaplanabileceğini söylemektedir. Yani bir problemin hesaplanabilir olması için Turing makinesi tarafından hesaplanabilir olması gerekmektedir (Kaye, Laflamme ve Mosca,2011). Kuantum bilgi teorisi Church-Turing tezinin Deutsch(1985) tarafından genişletilmesiyle gelişmiştir.

1.3.1. Turing makineleri

Alan Turing (1937) tarafından önerilmişlerdir. Standart Turing makinesi 6 elemandan oluşan bir kümedir. $M=(\Gamma,\beta,Q,\delta,s,h)$. Burada Γ, boş sembol(β) içermeyen teyp alfabesidir. Q turing makinesinin durum kümesidir. $\delta: Q\times(\Gamma\cup\{\beta\})\to(Q\cup\{h\})\times(\Gamma\cup\{\beta\})\times\{L,N,R\}$ bir sonraki durum fonksiyonudur. s başlangıç durumu ve $h\notin Q$ kabul edilen sonlandırma durumudur. Eğer Turing makinesinin okuma-yazma kafası a kelimesinde ve sistemin durumu da q ise bir sonraki durum fonksiyonu incelenir. Eğer $\delta(q,a)=(q',a',C)$ ise sistemin kontrol birimi q' durumuna geçiş yapar ve yazma-okuma kafası bulunduğu hücreye a' yazar. C nin L,R ve ya N değerine göre sol tarafa, sağ taraf ilerle ya da ilerlemez.



Şekil 1.2. Turing makinesi durum geçişi (Savage J.E., syf:119)

Γ^*, Γ alfabesinden oluşturulan istenilen uzunluktaki kelimeler kümesi olmak üzere; turing makinesi bir q durumuyla ve $w\in\Gamma^*$ şeklinde verilen girdi kelimesiyle çalışmaya başlar. Bu girdi kelimesi Turing makinesinin hücrelerine yazılır. Durum geçişleriyle herhangi bir sonlandırma $h\notin Q$ durumuna geldiğinde çalışmasını sonlandırır. Eğer

istenilen sonlandırma elemanına gelinebiliyorsa Turing makinesi bu girdiyi tanıyor denilir (Savage J.E., 2008).

1.3.1.1. Kuantum turing makinesi

Deutsch (1985) tarafından önerilmiştir. Tüm hesaplama modellerine eştir fakat kuantum algoritmalarının tanımlanmasına uygun değildir. Bunun nedeni kuantum algoritmalarında okuma-yazma kafasının ve sistem durumlarının süper pozisyon halinde gösterilen bir durum vektörüyle tanımlanmasıdır.

Kuantum turing makinesi iki temel elemandan oluşur, sonlu işlemci ve sonsuz uzunluğa sahip bellek. İşlemci M adet 2-durumlu gözlemlenebilirlerden oluşur (Deutsch 1985).

$$\{\hat{n}_i\} \qquad (i \in \mathbb{Z}_M) \tag{1.20}$$

Burada $\mathbb{Z}_M$, 0 dan $M-1$ e kadar tamsayıların oluşturduğu kümedir. Bellek sonsuz uzunluktadır ve aşağıdaki gözlemlenebilirlerden oluşmaktadır (Deutsch 1985).

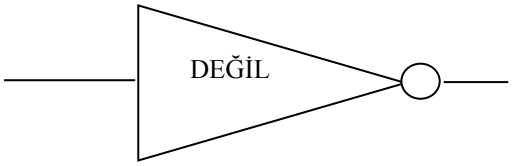
$$\{\hat{m}_i\} \qquad (i \in \mathbb{Z}) \tag{1.21}$$

x gözlemlenebiliri şu anki okuma-yazma kafasının pozisyonu olmak üzere, makinenin durumu $|\Psi(t)\rangle = |x; n_0, n_1, \cdots; m\rangle$ ile gösterilmektedir. $t=0$ anında makinenin durumu $|\Psi(0)\rangle = \sum_m a_m |0; 0, 0, \cdots; 0 \cdots 0\rangle$, $\sum_i |a_i|^2 = 1$ şeklindedir. Bu sistemin zaman evrimi H da verilen U birimsel dönüşümüyle tanımlanmaktadır (Deutsch, 1985). Böylece Deutsch (1985) 2-durumlu gözlemlenebilirlerin kümesinin $\mathbb{Z}_2 = \{0,1\}$ alınarak klasik bilgisayarlardaki bit kavramına karşılık gelineceğini göstermiştir.

1.3.2. Mantık kapıları

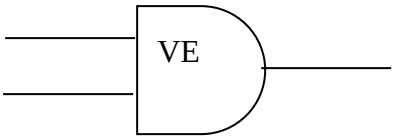
Mantık devreleri yönlü döngüsüz graflardır (DAG). Düğümler işlenecek olan mantık fonksiyonları (mantık kapıları) ile etiketlenirler. Her mantık devresi $f : \{0,1\}^n \to \{0,1\}^m$ şeklinde bir fonksiyonu hesaplamaktadır (Nielsen ve Chuang, 2000).

Bu hesaplamada VE (AND), VEYA(OR), DEĞİL(NOT) ve DIŞLAMALI YA DA(XOR) kapıları kullanılmaktadır. Bu kapılar $\wedge, \vee$ ve $\sim$ şeklinde etiketlenmektedir. Bu mantık kapılarının doğruluk diyagramları aşağıdaki gibi verilebilir.



DEĞİL (NOT)	
GİRDİ	ÇIKTI
0	1
1	0

Şekil 1.3. Değil kapısı ve doğruluk tablosu (Nielsen ve Chuang,2000 syf:21)



VE (AND)		
GİRDİ		ÇIKTI
0	0	0
0	1	0
1	0	0
1	1	1

Şekil 1.4. Ve kapısı ve doğruluk tablosu (Nielsen ve Chuang, 2000 syf:21)

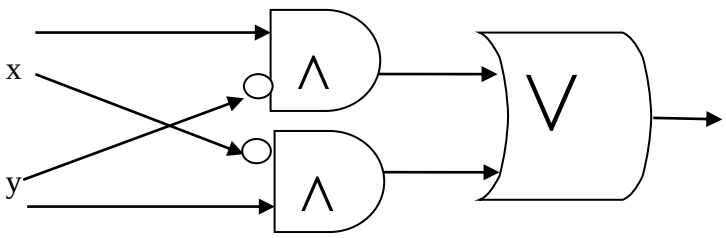
VEYA (AND)		
GİRDİ		ÇIKTI
0	0	0
0	1	1
1	0	1
1	1	1

Şekil 1.5. Veya kapısı ve doğruluk tablosu (Nielsen ve Chuang, 2000 syf:21)

DIŞLAMALI YADA (XOR)		
GİRDİ		ÇIKTI
0	0	0
0	1	1
1	0	1
1	1	0

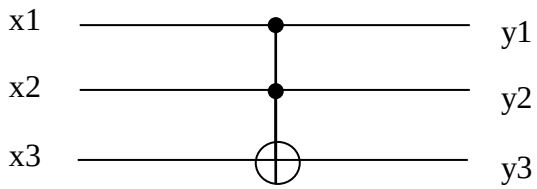
Şekil 1.6. Dışlamalı yada kapısı ve doğruluk tablosu (Nielsen ve Chuang, 2000 syf:21)

Basit bir mantık devresi aşağıdaki şekilde gösterilmektedir.



Şekil 1.7. Bir mantık devresi

Mantıksal devreler de terslenebilir kapılar önemlidir. Terslenebilir kapı, çıktı değerleri kullanılarak girdi değerlerinin elde edilebildiği kapı demektir Toffoli (1981). Bir mantıksal devrede kullanılan tüm kapılar terslenebilir halleriyle değiştirildiğinde devre terslenebilir devre olmaktadır. Toffoli (1981) kendi adıyla anılan terslenebilir Toffoli kapısının evrensel bir kapı olduğunu yani tüm mantık kapılarının bu kapı yardımıyla elde edilebildiğini göstermiştir. Toffoli kapısının devre şeması aşağıda gösterilmiştir.



Şekil 1.8. Toffoli kapısı devre diyagramı (Nielsen ve Chuang, 2000 syf:30)

Terslenebilirlik kavramı kuantum hesaplamada son derece önemlidir. Kuantum sistemde gerçekleştirilen tüm işlemler birimsel dönüşümlere karşılık gelmektedir. Birimsel dönüşüm terslenebilirlik özelliğine sahiptir.

1.3.3. Kuantum devre modeli

Deutsch (1989) tarafından önerilmiştir ve mantıksal kapı modelinin kuantum halidir. Kuantum algoritmalarının tanımlanmasında ve gösterilmesinde en fazla kullanılan yöntemdir. Bu yöntemde kuantum bitler kablolar, kapılar (birimsel dönüşümler) kutular şeklinde gösterilmektedir.

Kuantum devrelerde kullanılan girdi ve çıktı sayısı eşittir. Bilindiği gibi her kuantum kapı birimsel özelliğe sahiptir. Birimsel işlemler kuantum sisteminin zaman içerisinde evrimini ifade ederler. Kuantum sisteminin tamamında gerçekleştirilecek olan işlem daha basit elementer kapılar şeklinde ifade edilir. Bu elementer kapılar tüm sistemde gerçekleştirilecek olan herhangi bir birimsel işlemi yapabiliyorsa, bu elementer kapıların kümesine evrensel kuantum kapı kümesi denilir (Kaye P., Laflamme R., Mosca M., 2011).

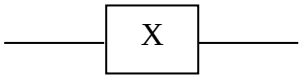
1.3.3.1. Tek kübite etki eden kapılar

Klasik mantık devrelerinde olduğu gibi kuantum devre modelinde de tek kubite etki eden ve durumunu değiştiren temel kapılar bulunmaktadır. Bu kapıları ve hesaplama bazlarına ($|0\rangle,|1\rangle$) olan etkilerini ve devre şekillerini aşağıdaki gibi gösterebiliriz.

a) DEĞİL Kapısı: Pauli X operatörüdür (Kaye P., Laflamme R., Mosca M., 2011).

$$|0>=\begin{pmatrix}1\\0\end{pmatrix}, \quad |1>=\begin{pmatrix}0\\1\end{pmatrix} \tag{1.22}$$

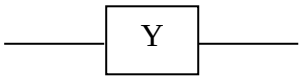
$$X=\sigma_x=\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}, \quad \begin{matrix}X|0>=|1>\\X|1>=|0>\end{matrix} \tag{1.23}$$



Şekil 1.9. Pauli X kapısının kuantum devre şekli (Kaye P., Laflamme R., Mosca M., 2011, syf:62)

b) Y Kapısı: Pauli Y operatörüdür (Kaye P., Laflamme R., Mosca M., 2011).

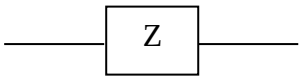
$$Y=\sigma_y=\begin{pmatrix}0 & -i\\i & 0\end{pmatrix}, \quad \begin{matrix}Y|0>=i|1>\\Y|1>=-i|0>\end{matrix} \tag{1.24}$$



Şekil 1.10. Pauli Y kapısının kuantum devre şekli(Kaye P., Laflamme R., Mosca M., 2011, syf:62)

c) Faz Kaydırma Kapısı: Pauli Z operatörüdür (Kaye P., Laflamme R., Mosca M., 2011).

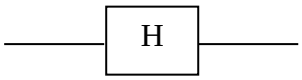
$$Z=\sigma_z=\begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}, \quad \begin{matrix}Z|0>=|0>\\Z|1>=-|1>\end{matrix} \tag{1.25}$$



Şekil 1.11. Pauli Z kapısının kuantum devre şekli(Kaye P., Laflamme R., Mosca M., 2011, syf:62)

d) Hadamard Kapısı: Qubiti aşağıdaki şekilde süper pozisyon durumuna getiren kapıdır (Kaye P., Laflamme R., Mosca M., 2011).

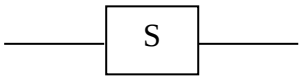
$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \begin{matrix} H|0> = \frac{|0>+|1>}{\sqrt{2}} \\ H|1> = \frac{|0>-|1>}{\sqrt{2}} \end{matrix} \qquad \textbf{(1.26)}$$



Şekil 1.12. Hadamard kapısının kuantum devre şekli (Kaye P., Laflamme R., Mosca M., 2011, syf:68)

e) Faz Kapısı: $|1\rangle$ Kübitine $e^{i\frac{\pi}{2}}$ fazını uygulayan kapıdır (Kaye P., Laflamme R., Mosca M., 2011).

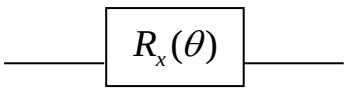
$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad S(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle + i\beta|1\rangle \qquad \textbf{(1.27)}$$



Şekil 1.13. S kapısının kuantum devre şekli (Kaye P., Laflamme R., Mosca M., 2011, syf:69)

f) Döndürme Kapıları (Kaye P., Laflamme R., Mosca M., 2011).

$$\begin{aligned} R_x(\theta) &= e^{-i\frac{\theta}{2}X} = \cos(\frac{\theta}{2})I - i\sin(\frac{\theta}{2})X \\ R_y(\theta) &= e^{-i\frac{\theta}{2}Y} = \cos(\frac{\theta}{2})I - i\sin(\frac{\theta}{2})Y \\ R_z(\theta) &= e^{-i\frac{\theta}{2}Z} = \cos(\frac{\theta}{2})I - i\sin(\frac{\theta}{2})Z \end{aligned} \qquad \textbf{(1.28)}$$



Şekil 1.14. X eksenine göre döndürme kapısının kuantum devre şekli (Kaye P., Laflamme R., Mosca M., 2011, syf:62)

Kuantum devre modelinde tek bir kübite etki eden herhangi bir $U \in U(2)$ kapısı

$$U = e^{ia}R_z(b)R_y(c)R_z(d) \qquad \textbf{(1.29)}$$

şeklinde ifade edilebilir ve bu kapı için $\{a,b,c,d\}$ sayıları bulunabilir (Divincenzo, 1995). Buradaki $U:\mathbb{C}^2 \to \mathbb{C}^2$ ye bir eşlemedir.

Bu şekilde kuantum bilgisayarlarda yapacağımız hesaplama işlemlerinde ihtiyacımız olan tek kübitlik herhangi bir kapıyı bu kapılar cinsinden yazabiliriz.

1.3.3.2. İki kübite etki eden kapılar

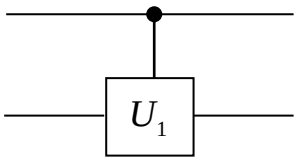
Girdi olarak iki kübit alarak iki kubit çıktı veren kapılardır. Bu kapılar $U:\mathbb{C}^4 \to \mathbb{C}^4$ e bir birimsel eşlemedir.

a) Kontrollü Kapı: $U_1, 2\times 2$ birimsel bir kapı olsun (Nielsen ve Chuang,2000, syf:178).

$$|1\rangle\langle 1| \otimes U_1 + |0\rangle\langle 0| \otimes I \tag{1.30}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.31}$$

İki kübite etki eden kontrollü- U_1 kapısı adlandırılır. Birinci yani kontrol bitinin $|1\rangle$ olduğu durumda ikinci kübite U_1 kapısını uygular. Devre diyagramı aşağıdaki gibi gösterilebilir.

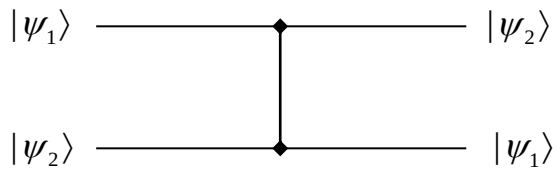


Şekil 1.15. Kontrollü-U_1 kapısının kuantum devre şekli(Kaye P., Laflamme R., Mosca M., 2011, syf:178)

Buradaki $U_1 = NOT$ alındığı duruma kontrollü- NOT (CNOT) kapısı denilir. Bu kapı iki kübitin dolanık durum getirilmesinde önemlidir. Dolanık durumlar birçok kuantum algoritmasının, özellikle iletişim protokollerinin temelini oluşturmaktadır.

b) SWAP Kapısı: İki kübitin değerlerini değiştirmede kullanılan bu kapıda kuantum algoritmalarında oldukça önemlidir (Nielsen ve Chuang, 2000).

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{1.32}$$



Şekil 1.16. SWAP kapısının kuantum devre şekli(Kaye P., Laflamme R., Mosca M., 2011, syf:xxv)

1.3.3.3. n>2 kübite etki eden kapılar

Yukarıdan görüleceği gibi n adet kübite etki eden bir birimsel kapı $U:\mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ şeklinde bir eşlemedir. $2^n \times 2^n$ boyutlu birimsel matrislerdir.

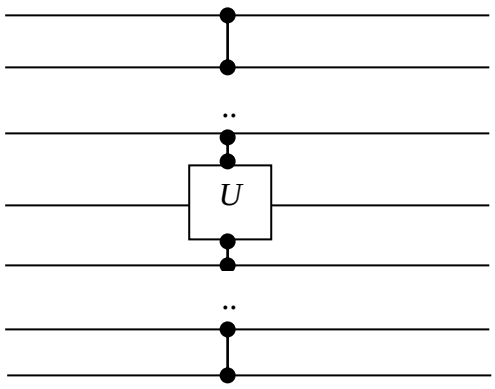
Teorem: $U:\mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ şeklinde verilen keyfi bir birimsel operator $(2^n)(2^n-1)/2$ adet aşağıdaki biçimde verilmiş matrislerin çarpımı şeklinde ifade edilebilir (Kitaev, 2002).

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & 0 & 1 & 0 & \cdots & \cdots & \vdots \\ 0 & \cdots & 0 & \begin{pmatrix} a & b \\ c & d \end{pmatrix} & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & 1 & 0 & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & 1 \end{pmatrix}, burada \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in U(2) \tag{1.33}$$

Yukarıdaki $2^n \times 2^n$-boyutlu birimsel kapı aslında (n-1)-kontrollü-U kapısıdır . $\wedge_{n-1}(U)$ şeklinde gösterilir. Bu kapıya genelleştirilmiş-Toffoli kapısıda denilir. Bu kapı tensörel olarak

$$|\underset{m-1}{1\cdots 1}\rangle\langle\underset{m-1}{1\cdots 1}|\otimes U\otimes|\underset{n-m}{1\cdots 1}\rangle\langle\underset{n-m}{1\cdots 1}|+\sum_{l\neq 1\cdots 1}|l\rangle\langle l|\otimes I \qquad \textbf{(1.34)}$$

şeklinde (Nielsen ve Chuang, 2000) ve devre diyagramı da



Şekil 1.17. $(n-1)-kontrollü-U$ kapısının kuantum devre şekli(Kaye P., Laflamme R., Mosca M., 2011, syf:181)

şeklinde gösterilebilir.

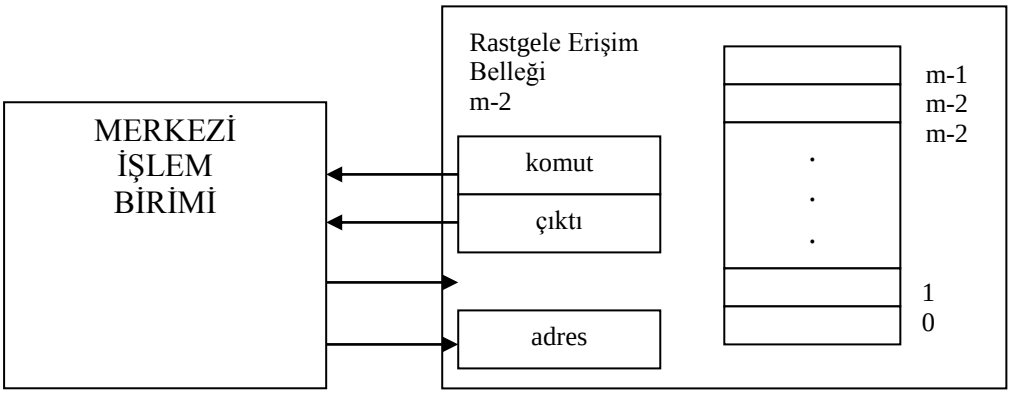
Teorem: Tüm tek-kübitlik kapıların ve CNOT kapısının oluşturduğu kapı kümesi evrensel bir kümedir. Tüm n-kübitlik kapılar bu kapıların kompozisyonu olarak yazılabilir (Nielsen ve Chuang, 2000).

Bu teoremle şunu söyleyebiliriz elimizde tek-kübitlik kapıların ve CNOT kapısının fiziksel gerçekleştirimi mevcutsa herhangi bir $f:\{0,1\}^n\rightarrow\{0,1\}^n$ fonksiyonunu gerçekleştirebilen kuantum devre bu kapılar kullanılarak gerçekleştirilebilir. Bu kapılar kullanılarak herhangi bir kuantum devresinin gerçekleştirimi için kullanılması gerekli olan kapı sayısının alt sınırı Shende ve ark. (2004) tarafından gösterilmiştir. Keyfi şekilde seçilecek olan evrensel kapı kümesinden verimli olarak basit kapıların elde edilmesini Möttönen ve ark. (2004) de çalışmışlardır. Ayrıca herhangi bir birimsel kapının nasıl temel kapılara ayrılacağını Vartiainen ve ark. (2004) incelemiştir.

1.3.4. Rastgele erişim makinesi (RAM)

Rastgele erişim makinesi iki adet sonlu durum makinesinin birleşiminden oluşmaktadır. Bunlardan biri merkezi işlem birimi (CPU), diğeri rastgele-erişim belleği (RAM)dir. Rastgele erişim belleği m adet bitten oluşan bellek hücresi içermektedir. Her bellek hücresinin belirli bir adresi vardır ve bu bellek hücreleri üzerindeki her tür işlem bu bellek adresleri kullanılarak gerçekleştirilmektedir. Ayrıca rastgele erişim belleğinde çıktı,

girdi, komut ve adres bölümleri bulunmaktadır. Komut bölümünde sadece OKU, YAZ ve İŞLEM-YOK olabilir. Komut OKU işlemiyse çıktı bölümü adres bölümünün gösterdiği bellek hücresinin verisiyle değiştirilir. Komut YAZ işlemiyse adres bölümünün gösterdiği bellek hücresindeki veri girdi bölümündeki veriyle değiştirilir. Bu klasik bilgisayarların çalışmasında kullanılan temel yöntemdir. Aşağıda RAM modelinin çalışma şekli gösterilmektedir (Savage J.E., 2008).



Şekil 1.18. Rastgele erişim makinesi (Savage J.E. , 2008, syf:110).

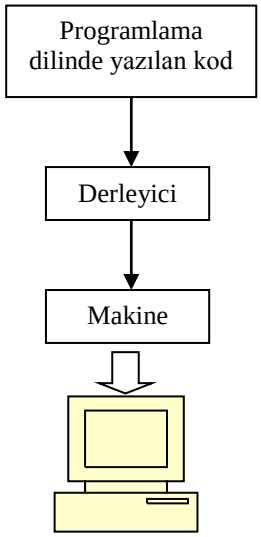
Programlama dillerini kullanmamızın temel amacı, bilgisayarın durumunu verdiğimiz komutlar aracılığıyla değiştirmek ve yapmak istediğimiz işlemleri gerçekleştirmesini sağlamaktır. Rastgele erişim makinesi bu amaca uygun bir zemini bizlere sunmaktadır.

Rastgele erişim makinesi modeli şu ana kadar geliştirilmiş bütün kuantum programlama dillerinde kullanılmıştır. Kullanılan modele QRAM (Quantum Random Access Machine) modeli denilmektedir. Bu model hesaplamayı gerçekleştiren kuantum kısmıyla, kontrolü yapan kısmı birbirinden ayırır.

BÖLÜM 2
ÖNCEKİ ÇALIŞMALAR

2.1. Kuantum Programlama Dilleri

Genel bir ifadeyle bir programlama dili, programcı ya da yazılım mühendisine bilgisayarda istediği işlemleri yapabilmesine olanak sağlayan bir özel dildir. Programlama dillerinde yazılan kodların çalışmasını aşağıdaki şekil ile kısaca gösterebiliriz.



Şekil 2.1. Programlama dillerinin klasik bilgisayarlarda çalıştırılması

Programlama dilinde yazılan kod bir derleyici aracılığıyla bilgisayarın anlayacağı makine koduna dönüştürülür. Bilgisayarda bu makine kodunu çalıştırır. Bu makine kodu bilgisayarın sahip olduğu merkezi işlem birimine gönderilecek olan emirlerden oluşmaktadır.

Bir programlama dilinin amacı hesaplamanın anlamını soyut bir şekilde ifade etmek ve hesaplamanın gerçekleştirilebilmesi için hesaplama aygıtına uygun gelen işlemler dizisini otomatik olarak oluşturmaktır. Bununla birlikte kuantum bilgisayarlar aşağıda belirtilen nedenlerden dolayı klasik bilgisayarlar gibi programlanamazlar.

a) Programlar çalışırken gözlemlenemezler. Kuantum mekaniğinin yapısı gereği gözlem yapılacak olan sistemin durumu gözlemleme (ölçme) işleminden etkilenir ve gözlem yapılan niceliği temsil eden operatörün öz vektörlerden birine çöker. Bu şekilde hesaplama yapılmak istenilen sistem ve değerlerin durumu bozulmuş olur. Buna tutarlılığın bozulması (decoherence) da denilmektedir.

b) Hesaplamalar terslenebilir olmalıdır. Terslenebilirlik, yapılan işlemler sonucunda elde edilen çıktılardan tekrar girdilerin elde edilebilirliği şeklinde basitçe tanımlanabilir. Kuantum mekaniğinde kapalı bir sistemin zaman içerisindeki evrimi birimsel operatörler tarafından tanımlanırlar. Birimsel bir operatörün mutlaka tersi vardır. Bu nedenle kuantum bilgisayarlarda yapılan işlemlerin tamamı birimsel olmak zorundadır.

c) Kuantum bilgisayarlardan ölçmeyle elde edilecek olan sonuçlar olasılıksaldır. Yani sonuçlar belli olasılıklarla elde edilir. Kuantum mekaniğinde bir sistem, uzayın bazları kullanılarak olasılık genlikleri denilen katsayılar oranında yazılabilir (Kaye P., Laflamme R. Ve Mosca M., 2011).

$$|\psi>=\sum_{\alpha} c_{\alpha}|\alpha>, \quad c_{\alpha}=<\alpha|\psi> \qquad c_{\alpha}\in\mathbb{C}$$
$$\sum_{\alpha}|c_{\alpha}|^2=1, \quad \sum_{\alpha}|\alpha><\alpha|=I \qquad \textbf{(2.1)}$$

Burada $|\alpha>$ durumunu ölçme olasılığı $|c_{\alpha}|^2$ dir. Kuantum bilgisayarda çalıştırılan programın çıktısı her zaman aynı olmayabilir. Bununla birlikte doğru sonucu verecek şekilde olasılık genlikleri ayarlanabilir. Olasılık genliklerinin doğru sonucu verecek şekilde değiştirilmesine yapıcı girişim, yanlış sonuçların genliklerinin azaltılmasına yıkıcı girişim denilmektedir.

Kuantum hesaplama, paralel veya dağıtık hesaplamaya bazı kavramlarda benzemektedir. Her iki yöntemde de belirli bir problemin çözümleri paralel olarak denenmektedir. Paralel işlem yapmada çözüm bulununcaya kadar denemeler aynı anda yapılmaya devam edilir. Örneğin, denenmesi gerekli olan bir milyon çözüm varsa en kötü durumda bu bir milyon çözümün tamamı denenir. Deneme işleminin kapasitesi yani aynı anda kaç tane çözümün deneneceği sistemimizde denemeleri yapacak işlem birimlerinin sayısı kadardır. Ayrıca denenecek olan verilerin parçalanması ve ilgili işlem birimine gönderilmeleri gerekmektedir. Bu da bir ek yük getirmektedir.

Kuantum bilgisayarlar deneme işlemlerinin tamamını süper pozisyon ilkesi ve paralellik özelliği ile bir kerede gerçekleştirebilir. n adet kübitin oluşturabileceği en genel süper pozisyon durumu aşağıdaki gibi gösterilebilir (Kaye P., Laflamme R. Ve Mosca M., 2011).

$$|\psi>=c_0|0>+c_1|1>+\cdots+c_n|2^n-1>$$
$$\sum_i |c_i|^2=1, \qquad c_i \in \mathbb{C} \tag{2.2}$$

Bu durumu kullanarak bir $f(x)$ fonksiyonunun değerini hesapladığımızda aşağıdaki gibi tüm $x \in \{0, 2^n\}$ değerleri için bu fonksiyon eş zamanlı olarak hesaplanmaktadır (Kaye P., Laflamme R. Ve Mosca M., 2011).

$$f(|\psi>)=c_0 f(|0>)+c_1 f(|1>)+\cdots+c_n f(|2^n-1>) \tag{2.3}$$

Bu bağlamda, mevcut kuantum programlama dillerinden kısaca bahsedelim.

Knill (1996), kuantum bilgisayarları için ilk kez psödo-kod geliştirmiştir. Knill'in çalışmalarına kadar kuantum bilgisayar programları matematiksel ifadelerle verilmekteydi. Bu ifadelerin programcı ve yazılım mühendislerince anlaşılması oldukça zordur. Bununla birlikte Knill'in çalışması kuantum bilgisayarların ve kuantum programların çalışma prensibi bakımından oldukça önemlidir.

Knill (1996) makelesinde kullandığı daha önceden tanımladığımız QRAM modeli diğer kuantum programlama dilleri tarafından da kullanılmaktadır.

Knill'in psödo-kodu için bir alt program örneği verelim. Kuantum yazmaçlar (birden fazla kübitten oluşan yapı) altı çizili şekilde ifade edilmektedir.

QINTROEXAMPLES()

$a \leftarrow 0^{\frown 5}$

C: 5 bit içeren klasik a yazmaçını istenilen ilk duruma getirir ve tüm bitlerin değerini 0 yapar.

$\underline{a} \leftarrow a$

C: Bu a yazmaçını herhangi bir işlem yapmadan bir kuantum yazmaçına çevirir.

$d \leftarrow 10$

C: İçerisinde 10 tamsayısı bulunan klasik d yazmaçı tanımlar.

$\underline{b} \leftarrow$ UNIFORMSUPERPOSITION(d)

C: d tamsayısına göre yeni bir kuantum yazmaçı oluşturur ve bu yazmaçı istenilen ilk duruma getirir.

$\underline{c} \leftarrow$ MULTIPLY($\underline{b}, d$)

C: Kuantum ve klasik değişken alan bir alt programdır. Geriye yeni bir $\underline{c}$ kuantum yazmaçı

çevirmektedir.

Şimdi işlemler sonunda ölçüm yapılan Fourier dönüşümünün psödo-kod ile nasıl ifade edildiğini görelim.

Girdi: d kübitten oluşan bir $\underline{a}$ kuantum yazmaçı. En değerli bitinin indisi $d-1$ dir.

Çıktı: $\underline{a}$ nın genlikleri $\mathbb{Z}_{2^d}$ üzerinde Fourier dönüşümüne tabi tutulmaktadır. Çıktıdaki en değerli bitin indisi 0 dır yani girdiye göre ters sıradadır. Girdi kuantum yazmacı süreç içerisinde klasik duruma geri döndürülür.

$\omega \leftarrow e^{i2\pi/2^d}$

$\phi \leftarrow 0$

for $i = d-1$ to $i = 0$

$\mathcal{R}_\phi(\underline{a}_i)$

$\mathcal{H}(\underline{a}_i)$

$a_i \leftarrow \underline{a}_i$
$\phi \leftarrow (\phi + a_i\pi)/2$

C:Bu atamanın en sağındaki ifade a_i nin klasik durumda olmasını gerekli kılar.

Ömer (2003) QCL adında ilk emirsel kuantum programlama dilini geliştirmiştir. QCL dilinin yazım şekli C diline benzemektedir. Programcılara az sayıdaki kübite sahip kuantum sistemlerinin benzetimini klasik bilgisayarlarda yapabilecekleri bir arayüze sahiptir.

Bettelli ve ark. (2003) C++ dili için Q dili adında bir kütüphane geliştirmiştir. Bu kütüphanedeki sınıflar ve fonksiyonlar, kuantum programlama yapılmasına imkan vermektedir. Bu makalesinde bir kuantum programlama dilinin ve özelliklerinin nasıl olması gerektiğini açıklamıştır. Q dilinde basit işlemler için QHadamard, QFourier, QNot, QSwap gibi sınıflar bulunmaktadır. Operatörler C++ 'ın sınıf mekanizması kullanılarak türetilmişlerdir.

Mlnarik (2007) tarafından geliştirilen LanQ dilininde C diline benzer bir yazımı bulunmaktadır. Temel geliştirilme amacı quantum iletişim protokollerini kodlayabilmektir.

Fork() komutu kullanılarak yeni süreçler başlatılabilir ve bu süreçler arası iletişim yönetilebilir.

Fonksiyonel programlama paradigmaları kullanılarak geliştirilen bazı kuantum programlama dilleri de bulunmaktadır. Bu alandaki ilk çalışmayı vanTonder (2004) kuantum lambda hesapla makalesiyle gerçekleştirmiştir. Lineer mantık temelinde tanımlarını test etmek için Scheme dilini kullanmıştır.

Selinger (2004) QPL isminde bir fonksiyonel programlama dili geliştirmiştir. Statik-tipli bir programlama dilidir. Derleme zamanında hataların tespitine imkan vermektedir. Özellikle klonlanamama özelliğini çalışma zamanında test etmemek için aynı kübitin farklı adlarla yazımına imkan vermektedir. Bununla birlikte programların kontrol akış diyagramları şeklinde yazılması klasik programlama alışkanlığına uygun gelmemektedir.

Mauerer (2005) QPL dili temelinde iletişim yetenekleri olan cQPL dilini geliştirmiştir. Bu dilin arka tarafında Ömer (2003) in geliştirmiş olduğu QCL kullanılmaktadır. cQPL dili de QCL yardımıyla az sayıda kübite sahip kuantum algoritmalarının benzetimini gerçekleştirebilmektedir.

Bahsedilen kuantum programlama dillerinde genellikle düşük seviyeli veri yapıları bulunmaktadır. Sadece QCL dilinde kullanılan qreg veri tipi kuantum belleğine erişim için kullanılmaktadır. cQPL de qint veri tipi geliştirilmiştir. Bu da 16 kübitten oluşan bir kübit dizisidir. Arka tarafında yine qreg veri tipi kullanılmaktadır. LanQ kuantum programlama dilinde de qnint veri yapısı bulunmaktadır. Buradaki n sayısı veri tipinin kaç kübitten oluşacağını göstermektedir.

Baker (1996), OGOL ile fonksiyonel programlama metodolojisi kullanarak kuantum bilgisayarını simüle etmiştir. Nesneye yönelik programlamanın avantajlarını kullanmıştır.

Sabry (2003) de kuantum bilgisayarlarda programlama için bir fonksiyonel dil olan Haskell kullanmıştır. Fonksiyonel diller sistemin genel durumunu değiştirmek yerine matematiksel fonksiyonların hesaplanması temelinde çalışmaktadır. Bu temelde çalışan tüm fonksiyonel dillere lamda-hesaplama yapan diller denilmektedir.

Maymin (1996) kuantum algoritmalarının lambda hesaplama yöntemiyle nasıl gerçekleştirilebileceği hakkında bir makale yayınlamıştır. Verilen hesaplamalar NP-tam (NonPolinomial-Complete) problemlerine çözüm getirseler de daha çok ifadeseldir ve fiziksel gerçekleştirilebilir bir kuantum bilgisayar temelli değildir.

Altenkrich ve Grattate. (2005) QML isminde fonksiyonel bir programlama dili olan ML temelli bir kuantum programlama dili geliştirmiştir. Bu dili gösterimsel ifadeler ile belirtmişlerdir.

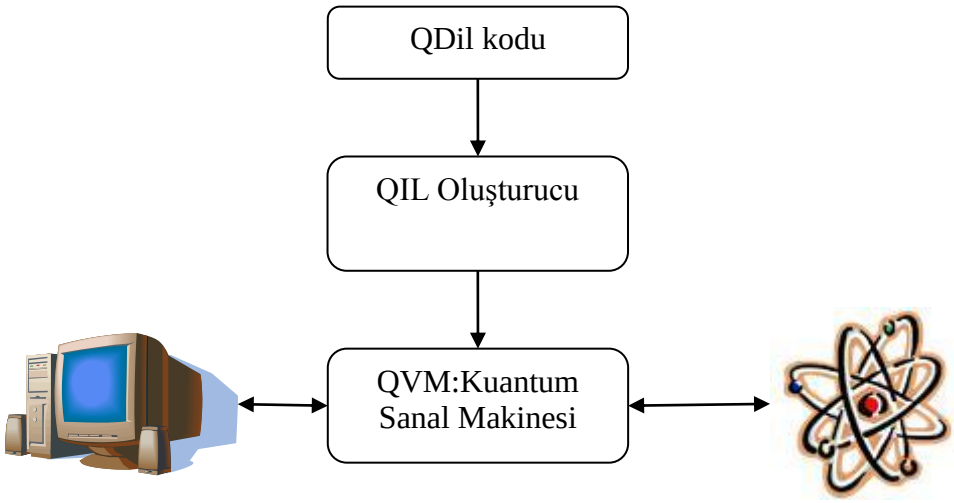
Green ve ark. (2013) quipper isminde Haskell temelli bir kuantum programlama dili geliştirmişlerdir. Bu dil fonksiyonel ve kuvvetli-tiplemeye sahip bir dildir. Algoritmaların otomatik olarak kuantum devrelerine dönüştüren bir alt yapıya sahiptir ve bir simülatör içermektedir.

Wecker ve Svore (2014) LIQUi|> adında bir kuantum hesaplama için bir dil geliştirmişlerdir. Microsoft F# dili ve .NET alt yapısını kullanmışlardır. QRAM modelinde geliştirilmiştir. Kontrol bilgisayarının microsoft işletim sistemiyle çalışmasını gerekli kılar. Tüm yazılan kodlar arka planda kuantum devrelerine dönüştürülür.

BÖLÜM 3
MATERYAL VE YÖNTEM

3.1. QDil (Quantum Dynamically Interpreted Language) Yeni Kuantum Programlama Dilinin Mimarisi

Kuantum bilgisayarın ve kübitlerin fiziksel gerçekleştirimleri üzerinde çeşitli araştırmalar yapılmaktadır ve yeni yaklaşımlar geliştirilmektedir. Fiziksel gerçekleştirimin değişebilir olması kuantum programlama dilinin de bu gerçekleştirimlerden bağımsız olmasını gerekli kılar. Daha önceden bahsedildiği gibi kuantum devre modeli kuantum bilgisayarın gerçekleştirebileceği tüm işlemleri gösterebilir bununla birlikte bazı kuantum algoritmalarında (Shor, 1994 ; Grover, 1997) var olan klasik kontrol kavramını içermezler. Bu yüzden QRAM modeli yani klasik bilgisayar tarafından kontrol edilen kuantum donanım ya da kuantum bilgisayar fikri tüm kuantum programlama dilleri tarafından tercih edilmektedir. Geliştirdiğimiz programlama dilinde hesaplama modeli olarak QRAM kullanılmıştır. QDil ile klasik bilgisayarlar için program geliştirebileceğiniz gibi kuantum bilgisayarlar için de programlar geliştirebilir ve kuantum bilgisayarların üstün paralel işlem yapma yeteneği ile dolanıklık gibi farklı yeteneklerinden faydalanabilirsiniz. QDil kuantum programlama dili kullanılacak olan klasik bilgisayarın ve kuantum donanımın sisteminden ve gerçekleştiriminden bağımsızdır. LIQUi|> ve Quipper gibi güncel ve mevcut eski birçok kuantum programlama dili, bir klasik programlama dilinin özellikleri kullanılarak bir kütüphane ya da eklenti şeklinde geliştirilmiştir ve bu dilin yetenekleriyle sınırlıdırlar. QDil nesne-yönelimli bir programlama dilidir ve baştan sona yeni tasarlanmıştır. Kodu açık olarak verildiği için daha sonraki güncel özelliklerin eklenilmesi ya da mevcut özelliklerin değiştirilmesi kolaydır. QDil, Java programlama dili kullanılarak geliştirilmiştir. Bunu sağlamak için QDil aşağıdaki mimaride geliştirilmiştir.

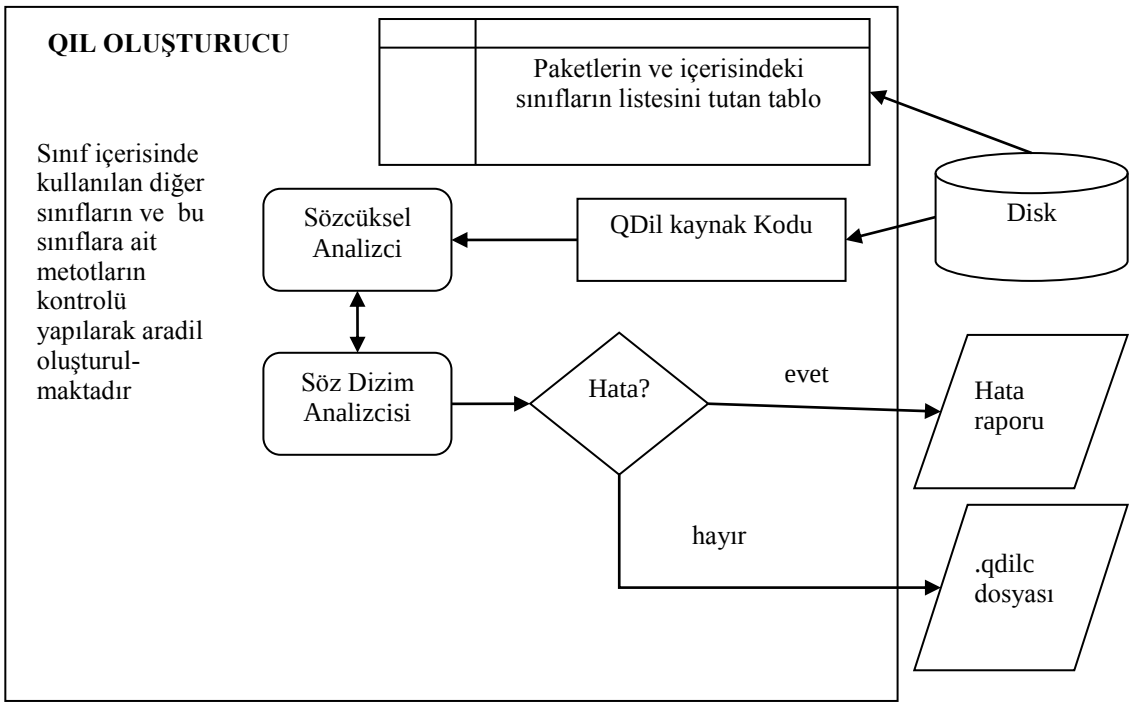


Şekil 3.1. QDil programlama dilinin temel mimarisi

Yukarıdaki birimlerden kısaca bahsedelim.

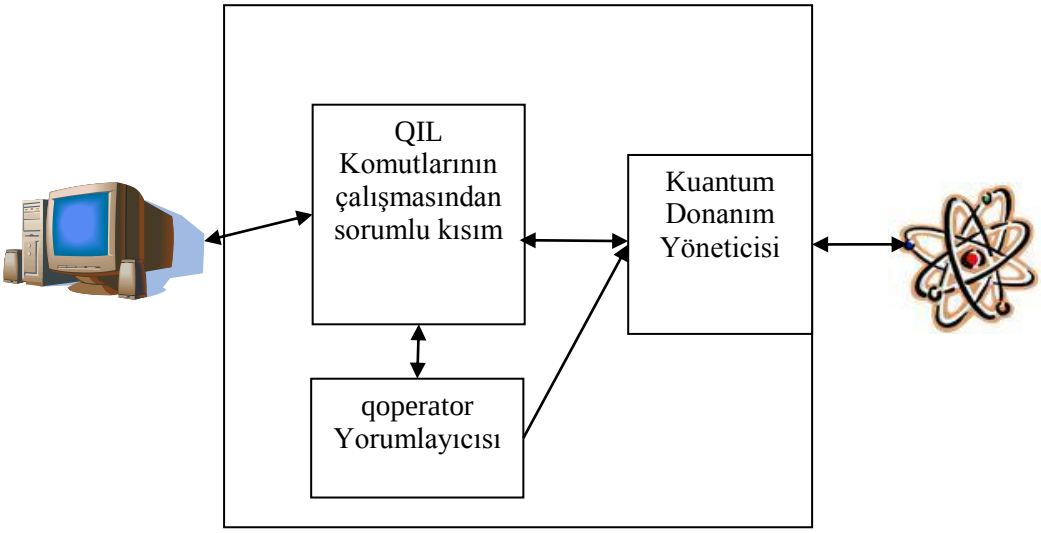
a) QDil Kodu: İçerisinde klasik bilgisayarda ve kuantum donanımda çalışacak olan kodların bir arada bulunduğu kaynak kodudur. Yazılımı geliştiriciler kodlarını sınıflar halinde tasarlarlar. Herhangi bir yazı editörü kullanılarak bu sınıfların kodları yazılabilir. Yazılan her sınıfın uzantısı **.qdil** şeklinde verilmelidir. QDil kodlarının nasıl yazılması gerektiği yazım kuralları bölümünde açıklanmıştır.

b) (QIL-G: Quantum Intermediate Language Generator) Kuantum Aradil Oluşturucu: Yazılan kodların klasik bilgisayardan ve kuantum donanımdan bağımsızlığını sağlayan en önemli özellik burasıdır. Yazılan kaynak kodu QIL'e çevrilmektedir. Bu ara dil işlemcilerin kullandıkları komut kümesine benzer bir komut kümesinden meydana gelmektedir. Bu komut kümesi daha sonraki bölümlerde açıklanacaktır. Bu komut kümesi içerisinde klasik bilgisayarda çalışacak olan komutlar ile kuantum donanımda çalışacak olan komutlar, sabitler ve diğer veriler bir arada bulunmaktadır. QDil kodunda geliştirilen her sınıf arakod dönüştürücüsü tarafından içerisinde QIL kodlarının bulunduğu sınıf ile aynı ada sahip uzantısı **.qdilc** olan bir dosyaya kaydedilir. QIL oluşturucunun dahili yapısını aşağıdaki şekilde daha ayrıntılı olarak görebiliriz.



Şekil 3.2. QIL oluşturucunun dahili yapısı

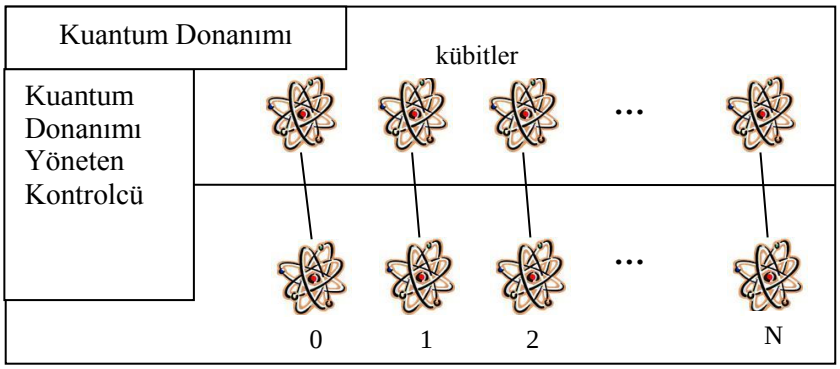
c) QVM (Quantum Virtual Machine) Kuantum Sanal Makinesi: QVM temel dört bölümden meydana gelmektedir. Temel amacı klasik bilgisayar ve kuantum donanımda çalışması gerekli olan QIL komutlarını çalıştırmaktır. QIL komutlarında gerçekleştirilecek olan işlemler farklı klasik ve kuantum sistemlerinde farklı olabilir. Bu nedenle yorumlayıcı esnek ve kullanıcı tanımlı fonksiyonlar temelinde gerçekleştirilmiştir. QVM içerisinde QIL kodlarına karşılık gelen fonksiyon prototipleri tanımlanmış, bunların fiziksel gerçekleştirimleri kullanıcıya ya da üretici firmalara bırakılmıştır. QVM kendi içerisindeki temel 4 bileşeni bir arada kullanarak verilen QIL kodunu çalıştırır ve sonuçları üreterek gerekli çıktıları sağlar. Aşağıdaki şekilde QVM nin genel yapısı gösterilmiştir.



Şekil 3.3. QVM nin genel yapısı

QVM nin alt kısımlarını kısaca açıklayalım

1) (QDM:Quantum Device Manager) Kuantum Donanım Yöneticisi: Kuantum sistem yöneticisi, RAM modelindeki rastgele erişim belleğine karşılık gelmektedir. Temel görevi kuantum donanımındaki gerçek kübitler üzerinde her türlü işlemi gerçekleştirmektir. Fiziksel kuantum donanımı içerisindeki kübitleri Şekil 23 de gösterelim. Şekilden görüldüğü gibi kuantum donanımının içerisinde hesaplamalarda kullanılmak üzere N adet kübit bulunacaktır. Bu kübitlerin adreslenebilir olduğunu kabul ediyoruz, yani kuantum donanımı her kübite belirli bir indis sayısıyla ulaşır ve tüm işlemleri bu sayı ile gerçekleştirir.



Şekil 3.4. Kuantum donanımının varsayılan dahili yapısı

Kuantum donanımını farklı üreticiler farklı kübit gerçekleştirim yöntemleri kullanarak geliştirmiş olabilir. QDil programlama dilinin bu üretimlerden bağımsızlığını sağlayan, kendi içerisinde tüm kuantum donanımlarının sağlaması gerekli olan temel

fonksiyonların arayüzlerini QDM tarafında tanımlanmasıdır. Üretici QDM nin arayüzünün kendisine sunduğu fonksiyonların kodunu geliştirecek ve QDM nin içerisine entegre edecektir. Bu şekilde QVM, QDM'nin fonksiyonlarını çağırdığında üretici firmanın geliştirdiği fonksiyonları çağırmış olacaktır. Üretici firmanın geliştirdiği fonksiyonlarda kuantum donanımının kontrolcüsünü çalıştırmakta ve istenilen işlemlerin donanım tarafından yapılmasını sağlamaktadır. QDM'nin sunduğu fonksiyonların arayüzü aşağıdaki çizelgede verilmiştir.

Çizelge 3.1. Kuantum donanım yöneticisinin (QDM) fonksiyon arayüzü

1	public abstract Qd_int AllocateQbit()	Kuantum donanımı, QVM tarafından kullanılmayan bir kübiti kullanılanlar listesine atar ve adres numarasını geriye çevirir. Tüm kübitler kullanılıyorsa geriye -1 değerini çevirir.
2	public abstract Qd_int[] AllocateQbit(Qd_int noqbits)	Kuantum donanımı, QVM tarafından kullanılmayan istenilen sayıdaki kübiti kullanılanlar listesine atar ve adreslerinin numaralarını geriye çevirir. Hata surumunda null değerini çevirir.
3	public abstract Qd_boolean DeallocateQbit (Qd_int qbitNo)	QVM' nin gönderdiği qbitNo lu kübiti kullanılanlar listesinden kullanılmayanlara aktarır. Gerekli işlemleri yerine getirir. Hata durumunda false geriye çevirir.
4	public abstract Qd_boolean DeallocateQbit (Qd_int[] qbits)	QVM' nin gönderdiği numaralara sahip kübitleri,kullanılanlar listesinden kullanılmayanlara aktarır. Gerekli işlemleri yerine getirir.
5	public abstract Qd_boolean SetQbit (Qd_int qbitNo, Qd_int value)	Verilen qbitNo lu kullanılan kübitin değerini value {|0> | |1>} yapar. Başarılı olursa geriye true, olmazsa false çevirir.

Çizelge 3.1.'in devamı

6	public abstract Qd_boolean ResetQbit (Qd_int qbitNo)	Verilen qbitNo lu kübitin değerini |0> yapar. Başarılı olursa geriye true, olmazsa false çevirir.
7	public abstract Qd_boolean Measure (Qd_int qbitNo)	Verilen qbitNo lu kübitte hesaplama bazlarında {|0>,|1>} bir ölçüm gerçekleştirir ve sonucu {true:1 | false:0} geriye çevirir.
8	public Qd_boolean SwapQbits (Qd_int qbit1, Qd_int qbit2)	Verilen iki kübitin değerlerini birbirlerine aktarır. Başarılı olursa geriye true, olmazsa false çevirir.
9	public abstract Qd_boolean RotationX (Qd_int qbitNo, Qd_double theta)	Verilen kübitte x yönünde theta değeri kadar döndürme gerçekleştirir. Başarılı olursa geriye true, olmazsa false çevirir.
10	public abstract Qd_boolean RotationY (Qd_int qbitNo, Qd_double theta)	Verilen kübitte y yönünde theta değeri kadar döndürme gerçekleştirir. Başarılı olursa geriye true, olmazsa false çevirir.
11	public abstract Qd_boolean RotationZ (Qd_int qbitNo, Qd_double theta)	Verilen kübitte z yönünde theta değeri kadar döndürme gerçekleştirir.
12	public abstract Qd_boolean X (Qd_int qbitNo)	Verilen kübite X kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
13	public abstract Qd_boolean inv_X (Qd_int qbitNo)	Verilen kübite X kapısının tersini uygular. Başarılı olursa geriye true, olmazsa false çevirir.
14	public abstract Qd_boolean Y (Qd_int qbitNo)	Verilen kübite Y kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
15	public abstract Qd_boolean inv_Y (Qd_int qbitNo)	Verilen kübite Y kapısının tersini uygular. Başarılı olursa geriye true, olmazsa false çevirir.

Çizelge 3.1.'in devamı

16	public abstract Qd_boolean Z (Qd_int qbitNo)	Verilen kübite Z kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
17	public abstract Qd_boolean Z (Qd_int qbitNo)	Verilen kübite Z kapısının tersini uygular. Başarılı olursa geriye true, olmazsa false çevirir.
18	public abstract Qd_boolean CNot (Qd_int controlQbit, Qd_int NotQbit)	Kontrol kübitinin değerinin |1> olması durumunda, NotQbit numarasıyla verilen kübite X kapısını uygular.
19	public abstract Qd_boolean Hadamard (Qd_int qbitNo)	Verilen kübite hadamard kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
20	public abstract Qd_boolean Toffoli (Qd_int[] controls, Qd_int qbit)	Kontrol kübitlerinin değerlerinin |1> olması durumunda kübite X kapısı uygular. Başarılı olursa geriye true, olmazsa false çevirir.
21	public abstract Qd_boolean S (Qd_int qbitNo)	Verilen kübite S kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
22	public abstract Qd_boolean inv_S (Qd_int qbitNo)	Verilen kübite S kapısının tersini uygular. Başarılı olursa geriye true, olmazsa false çevirir.
23	public abstract Qd_boolean T (Qd_int qbitNo)	Verilen kübite T kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
24	public abstract Qd_boolean T (Qd_int qbitNo)	Verilen kübite T kapısının tersini uygular. Başarılı olursa geriye true, olmazsa false çevirir.
25	public abstract Qd_boolean V (Qd_int qbitNo)	Verilen kübite V kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.

Çizelge 3.1.'in devamı

26	public abstract Qd_boolean inv_V (Qd_int qbitNo)	Verilen kübite V kapısının tersini uygular. Başarılı olursa geriye true, olmazsa false çevirir.
27	public abstract Qd_boolean OneControlledRotX (Qd_int[] controlQbits, Qd_int qbitNo, Qd_double theta)	Kontrol kübitlerinin |1> olduğu durumda qbitNo ya RotX kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
28	public abstract Qd_boolean OneControlledRotY (Qd_int[] controlQbits, Qd_int qbitNo, Qd_double theta)	Kontrol kübitlerinin |1> olduğu durumda qbitNo ya RotY kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
29	public abstract Qd_boolean OneControlledZ (Qd_int[] controlQbits, Qd_int qbitNo, Qd_double theta)	Kontrol kübitlerinin |1> olduğu durumda qbitNo ya RotZ kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
30	public abstract Qd_boolean ZeroControlledRotX (Qd_int[] controlQbits, Qd_int qbitNo)	Kontrol kübitlerinin |0> olduğu durumda qbitNo ya RotX kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
31	public abstract Qd_boolean ZeroControlledRotY (Qd_int[] controlQbits, Qd_int qbitNo)	Kontrol kübitlerinin |0> olduğu durumda qbitNo ya RotY kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
32	public abstract Qd_boolean ZeroControlledRotZ (Qd_int[] controlQbits, Qd_int qbitNo)	Kontrol kübitlerinin |0> olduğu durumda qbitNo ya RotZ kapısını uygular. Başarılı olursa geriye true, olmazsa false çevirir.
33	public abstract Qd_boolean Receive (Qd_int qbitNo, Qd_qbitMachine resource)	Verilen URI dan gelen kübit değeri qbitNo içerisine konulur. Bunun için öncelikle kübit değeri sıfırlanır. Başarılı olursa geriye true, olmazsa false çevirir.

Çizelge 3.1.'in devamı

34	public abstract Qd_boolean Send (Qd_int qbitNo, Qd_qbitMachine destination)	Verilen URI ya qbitNo nun gösterdiği kübit değeri gönderilir. Başarılı olursa geriye true, olmazsa false çevirir.
35	public abstract Qd_int AvailableQbitsCount()	Kullanılabilir kübitlerin sayısını geriye çevirir.
36	public abstract Qd_int TotalQbitsCount()	Toplam kübit sayısını geriye çevirir.
37	public Qd_boolean checkQbitsAvailable(Qd_int[] qbits)	Numaraları verilen kübitlerin kullanılabilir olup olmadığını belirtir.
38	public Qd_boolean checkQbitAvailable(Qd_int qbit)	Numarası verilen kübitin kullanılabilir olup olmadığını belirtir.

2) (QOPINT:qoperator Interpreter) qoperator Yorumlayıcısı: qoperator QDil içerisinde kullanılan bir veri türüdür. Diğer kuantum veri türleri olan qbit ve qregister' a parametre olarak gönderilir ve bu veri türlerine uygulanan birimsel dönüşümlere karşılık gelmektedir. Kendine has bir yazım biçimi vardır. qoperator tipinden bir değişken qbit ya da qregister tipinden bir değişkene uygulanmak istendiğinde qoperator yorumlayıcısı devreye girer ve yazılmış olan qoperatoru yazım hataları, birimsellik hataları gibi çeşitli hatalara karşı denetler. Herhangi bir hata bulmazsa, yorumlayıcı mevcut qoperatoru kübitlere uygulanabilecek şekilde temel kapılara ayrıştırır ve uygular. İstenilirse üretici firma qoperator yorumlayıcısının ayrıştırma kodunu değiştirebilir ve kendi ayrıştırıcısını donanıma özel olarak QVM içerisine dahil edebilir.

Örnek:

qoperator NOT= " < 0 , 1 ; 1 , 0 > ";

Açıklama: Matris biçiminde bir operatör tanımlanmıştır.

qbit a = new qbit();

Açıklama: QDM tarafından kullanılmayan bir kübit kullanılanlar listesine atanır ve QVM tarafından bu kübitin adres numarası ile a ismi ilişkilendirilir.

a.applyOperator(NOT);

Açıklama: NOT ile gösterilen qoperator yorumlanır ve hata yoksa kuantum kapılarına dönüştürülerek a kübitine uygulanır.

qoperator tipinden değişkenlerin yazımı daha sonra anlatılacaktır.

3) Kuantum kodlarının çalışmasından sorumlu kısım: QVM içerisindeki qbit ve qregister veri tiplerinin kendilerine ait metotları vardır. Bu metotlar QMM nin sahip olduğu kapılara karşılık gelmektedir. Bu metotlar QDil kodunda yazılmışlarsa QIL oluşturucu bunlara karşılık komutları **.qdilc** dosyası içerisine yazar. Bu kodlar QVM içerisindeki bu bölüm tarafıından kuantum donanımında işletilir.

   Örnek:

   qbit a= new qbit();

   a.H() ;

   Açıklama: bu kübite hadamard uygulanmasını sağlayan QIL komutunun .qdilc dosyası içerisine yazılmasını sağlar.

4) Klasik kodların çalışmasından sorumlu kısım: QIL içerisinde kontrolcü bilgisayarda çalıştırılacak olan QIL komutları da bulunur. QVM içerisindeki bu kısım, bu komutların çalıştırılmasından ve QVM nin klasik bilgisayardaki bellek yönetiminden sorumludur. QVM nin kendisi de bir program olduğu için klasik bilgisayardaki işletim sistemi üzerinde çalışmaktadır.

3.2. QDil'in Sözcüksel Analizcisi

Bir programlama dilinini geliştirilmesindeki ilk adım kullanılacak olan temel sözcüklerin belirlenmesi ve bu sözcükleri, sabitleri ve özel karakterleri ayırt edip sınıflandırabilecek bir sözcüksel analizcinin yazılmasıdır. QDil'in sözcüksel analizcisinin geliştirilmesi için JFlex (Klein, Rowe ve Decamps, 1998) uygulaması kullanılmıştır. Sözcüksel analizcide kullanılan dosya ekler bölümünde ayrıntılı olarak verilmiştir. JFlex, verilen ayar dosyası temelinde sözcüksel analiz yapabilen bir sınıf dosyasını otomatik olarak oluşturmaktadır. Daha sonraki çalışmalarda, sözcüksel analizcinin performansının arttırılması için tüm kodların C dilinde geliştirilmesi planlanmaktadır.

3.3. QDil'in Sözdizimsel Analizcisi

Bir programlama dilinin geliştirilmesindeki ikinci adım programlama dilinin yazım kurallarını denetleyerek yazım hatalarını tespit edebilen ve yazım hatası yok ise tüm programlama dili elemanlarından oluşan bir ağaç yapısı oluşturabilen sözdizimsel analizcinin yazılmasıdır. QDil kuantum programlama dilinin geliştirilmesinde JCUP (Hudson S.E ve ark., 1999) uygulaması kullanılmıştır. JCUP uygulaması kendisine verilen ayar dosyası temelinde bir sözdizimsel analizciyi oluşturmaktadır ve JFlex'in oluşturduğu sözcüksel analizci ile uyumlu şekilde çalışmaktadır.

BÖLÜM 4
ARAŞTIRMA BULGULARI VE TARTIŞMA

4.1. QDil Söz Dizim Kuralları

QDil'in söz dizim kuralları ve söz dizim analizcisi JCUP programının kuralları kullanılarak tanımlanmıştır. QDil'e ait ayrılmış kelimeler aşağıdaki gibidir.

abstract , and, AND, binary, boolean, break, breakall, byte, case, catch, char, class, complex, const, continue, do, double, each, else, finally, float, for, get, if, in, int, interface, isa, long, method, extern, new, qbit, qregister, qoperator, or, OR, others, set, short, super, package, return, when, while, this, throw, throws, try, until, use, def, true, false, null

Söz dizim kuralları da Çizelge 4.1. de sırasıyla verilmektedir. Tokenlar {" "} arasında yazılmıştır.

Çizelge 4.1. QDil'in sözdizim kuralları

Kural no	Yazım kuralı
1	goal ::= compile_unit;
2	lit ::= INT_LIT | FL_LIT| BOOL_LIT| CHR_LIT| STR_LIT| NULL_LIT
3	var_type ::= cl_type | ary_type
4	cl_type ::= id_name
5	ary_type ::= id_name dms
6	id_name ::= short_id_name | long_is_name | opr
7	short_id_name ::= ID_NAME | "method" | "byte" |"short" |"int" | "long" | "float" | "double" | "char" | "string" | "complex" | "binary" | "boolean" | "qbit" | "qregister" | "qoperator"
8	long_id_name ::= id_name "." short_id_name
9	opr ::= "@+" | "@-" | "@/" | "@%" | "@*" | "@&" | "@|" | "@xor"| "@~" | "@^" | "@>>" |"@<<" | "@>>>" | "@!" | "@in" | "@[]" |"@and" | "@or" | "@?=" | "@!="
10	compile_unit ::= pac_dec use_dec_o cl_or_int_dec
11	use_dec_o::= use_decs | $\mathcal{E}$
12	use_decs ::= use_dec | use_decs use_dec
13	use_dec ::= "use" id_name ";"

Çizelge 4.1.'in devamı

14	pac_dec ::= “package” id_name “;”
15	cl_or_int_dec ::= cl_dec | int_dec
16	access_types_opt ::= $\mathcal{E}$ | access_types
17	access_types ::= access_type | access_types access_type
18	access_type::= “+” | “-” | “#” | “const” | “abstract”
19	cl_dec ::= access_types_opt “class” id_name sup_and_int_opt cl_bdy
20	sup_and_int_opt::= $\mathcal{E}$ | sup_and_ints
21	sup_and_ints::= “:”sup_and_ints _list
22	sup_and_ints _list ::= cl_type | sup_and_ints _list “,” cl_type
23	cl_bdy ::= “{” cl_mem_decs “}” | “{” $\mathcal{E}$ “}”
24	cl_mem_decs::= cl_mem_dec | cl_mem_decs cl_mem_dec
25	cl_mem_dec::= fld_dec | mtd_dec
26	fld_dec::= var_type var_dec “{” fld_bdy “}”
27	fld_bdy::= get_dec set_dec | set_dec get_dec
28	get_dec ::= access_types_opt “get” mtd_bdy
29	set_dec ::= access_types_opt “set” mtd_bdy
30	var_decs ::= var_dec | var_decs var_dec
31	var_dec::= id_name | id_name “=” var_init
32	var_dec_id _id ::= id_name
33	var_init ::= expr | ary_init
34	mtd_dec ::= mtd_head mtd_bdy
35	mtd_head ::= access_types_opt mtd_name mtd_back
36	mtd_name ::= “def” id_name “(” formal_param_lst_opt “)” | “extdef” id_name “(” formal_param_lst_opt “)” | “oradef” id_name “(” formal_param_lst_opt “)”
37	formal_param_lst_opt ::= $\mathcal{E}$ | formal_param_lst
38	formal_param_lst ::= formal_param | formal_param_lst formal_param
39	formal_param ::= var_type var_dec_id | “const” var_type var_dec_id
40	mtd_back ::= $\mathcal{E}$ | mtd_ret_type | raise | mtd_ret_type raise
41	mtd_ret_type::= “:” var_type
42	raise ::= “raise” cl_type_lst
43	class_type_list ::= cl_type | cl_type_lst “,” cl_type

Çizelge 4.1.'in devamı

44	mtd_bdy ::= blk | ";" | cons_bdy
45	cons_body ::= "{" call_cons blk_stmts "}" | "{" call_cons "}"
46	call_cons ::= "self" "(" arg_lst_opt ")" ";" | "super" "(" arg_lst_opt ")" ";" |first_expr "." "self" "(" arg_lst_opt ")" ";" | first_expr "." "super" "(" arg_lst_opt ")" ";"
47	int_dec ::= "interface" id_name sup_and_int_opt int_bdy
48	int_bdy ::= "{" int_mem_decs_opt "}"
49	int_mem_decs_opt::= $\mathcal{E}$ | int_mem_decs
50	int_mem_decs::= int_mem_dec | int_mem_decs int_mem_dec
51	int_mem_dec::= int_fld_dec | mtd_dec
52	int_fld_dec ::= var_type var_dec ";"
53	ary_init ::= "{" var_inits "," "}" | "{" var_inits "}" | "{" ";" "}" | "{" "}"
54	var_inits ::= var_init | var_inits "," var_init
55	blk ::= "{" blk_stmts_opt "}"
56	blk_stmts_opt ::= $\mathcal{E}$ | blk_stmts
57	blk_stmts ::= blk_stmt | blk_stmts blk_stmt
58	blk_stmt ::= loc_var_dec ";" | stmt
59	loc_var_dec::= var_type var_decs | "const" var_type var_decs
60	stmt ::= stmt_no_sub_stmt | stmt_with_lbl | if_stmt | if_else_stmt | while_stmt | for_stmt | each_stmt
61	stmt_no_sub_stmt::= blk | no_stmt | expr_stmt | case_stmt | until_stmt | break_stmt | breakall_stmt | continue_stmt | ret_stmt | raise_stmt | try_stmt
62	no_stmt ::= ";"
63	stmt_with_lbl ::= ID_NAME ":" stmt
64	expr_stmt ::= stmt_expr ";"
65	stmt_expr ::= asgn | mtd_call | new_expr
66	if_stmt ::= "if" "(" expr ")" "{" blk_stmts_opt "}"
67	if_else_stmt ::= "if" "(" expr ")" "{" blk_stmts_opt "}" "else" "{"blk_stmts_opt "}"
68	case_stmt ::= "case" "(" id_name ")" case_blk

Çizelge 4.1.'in devamı

69	case_blk ::= "{" when_blk_stmt_groups others_blk "}" | "{" when_blk_stmt_groups "}"
70	when_blk_stmt_groups ::= when_blk_stmt_group | when_blk_stmt_groups | when_blk_stmt_group
71	when_blk_stmt_group ::= "when" "(" when_cnt_expr ")" block
72	when_cnt_expr ::= cnst_expr | in_expr
73	in_expr ::= "in" var_dec_id
74	others_blk ::= "others" blk
75	while_stmt ::= "while" "(" expr ")" blk
76	until_stmt ::= "do" blk "until" "(" expr ")" ";"
77	each_stmt ::= "each" "(" var_type var_dec_id ":" expr ")" blk
78	for_stmt ::= "for" "(" for_init_opt "," for_last_opt for_step_opt ")" blk
79	for_init_opt ::= asgn
80	for_last_opt ::= expr
81	for_step_opt ::= $\mathcal{E}$ | "," expr
82	id_opt ::= $\mathcal{E}$ | ID_NAME
83	break_stmt ::= "break" id_opt ";"
84	breakall_stmt ::= "breakall" ";"
85	continue_stmt ::= "continue" id_opt ";"
86	ret_stmt ::= "return" expr_opt ";"
87	raise_stmt ::= "raise" expr ";"
88	try_stmt ::= "try" blk catches | "try" blk catches_opt finally
89	catches_opt ::= $\mathcal{E}$ | catches
90	catches ::= catch | catches catch
91	catch ::= "catch" "(" formal_param ")" blk
92	finally ::= "finally" blk
93	first_expr ::= first_expr_no_ary | new_ary_init | new_ary_no_init
94	first_expr_no_ary::= lit | "self" | "(" expr ")" | new_expr | fld_var | mtd_call | ary_elmnt | id_name "." "self" | id_name "." "qdilc" | ary_type "." "qdilc"
95	new_expr ::= "new" cl_type "(" arg_lst_opt ")"
96	arg_lst_opt ::= $\mathcal{E}$ | arg_lst

Çizelge 4.1.'in devamı

97	arg_lst ::= expr | arg_lst "," expr
98	new_ary_no_init ::= "new" cl_type d_exprs dms_opt
99	new_ary_init ::= "new" cl_type dms ary_init
100	d_exprs ::= d_expr | d_exprs d_expr
101	d_expr ::= "[" expr "]"
102	d_opt ::= $\mathcal{E}$ | dms
103	dms ::= "[" "]"| dms "[" "]"
104	fld_var ::= first_expr "." ID_NAME | "super" "." ID_NAME | id_name "." "super" "." ID_NAME
105	mtd_call ::= id_name "(" arg_lst_opt ")" | id_name "(" arg_lst_opt ")' " | first_expr "." ID_NAME "(" arg_lst_opt ")" | first_expr "." ID_NAME "(" arg_lst_opt ")' " | "super" "." ID_NAME "(" arg_lst_opt ")" | "super" "." ID_NAME "(" arg_lst_opt ")' " | id_name "." "super" "." ID_NAME "(" arg_lst_opt ")" | id_name "." "super" "." ID_NAME "(" arg_lst_opt ")' "
106	ary_elmnt::= id_name "[" expr "]" | first_expr_no_ary "[" expr "]"
107	post_expr ::= first_expr | id_name
108	sgn_expr ::= "+" sgn_expr | "-" sgn_expr | un_exprs
109	un_exprs ::= post_expr | "~" sgn_expr | "!" sign_expr | type_ch_expr
110	type_ch_expr ::= "(" classical_type dms_opt ")" sgn_expr | "(" expr ")" un_exprs | "(" id_name dms ")" un_exprs
111	pow_expr ::= sign_expr | sign_expr "^" pow_expr
112	mul_div_expr ::= pow_expr | mul_div_mod_expr "*" pow_expr | mul_div_mod_expr "/"pow_expr | mul_div_mod_expr "%"pow_expr
113	add_sub_expr ::= mul_div_mod_expr | add_sub_expr "+"mul_div_mod_expr | add_sub_expr "-"mul_div_mod_expr

Çizelge 4.1.'in devamı

114	shft_expr ::= add_sub_expr | shft_expr "<<"add_sub_expr | shft_expr ">>"add_sub_expr | shft_expr ">>>"add_sub_expr
115	rel_expr ::= shft_expr | rel_expr "<"shft_expr | rel_expr ">"shft_expr | rel_expr "<="shft_expr | rel_expr ">="shft_expr | rel_expr "in" ref_type | rel_expr "isa" ref_type
116	eq_or_noteq_expr ::= rel_expr | eq_or_noteq_expr "?="rel_expr | eq_or_noteq_expr "!="rel_expr
117	bitand_expr ::= eq_or_noteq_expr | bitand_expr "&"eq_or_noteq_expr
118	xor_expr ::= bitand_expr | xor_expr "xor" bitand_expr
119	bitor_expr ::= xor_expr | bitor_expr "|" exor_expr
120	and_expression ::= bitor_expr | and_expr "and" bitor_expr | and_expression "AND" bitor_expr
121	or_expr ::= and_expr | or_expr "or" and_expr | or_expr "OR" and_expr
122	asgn_expr ::= or_expr | asgn
123	asgn ::= l_expr "=" asgn_expr
124	l_expr ::= id_name | fld_var | ary_elmnt
125	expr_opt ::= $\mathcal{E}$ | expr
126	expr ::= asgn_expr
127	cnst_expr ::= expr

4.2. QDil'in Temel Kavramları

Bir programlama dilinin geliştirilmesinde, dil hakkında temel programlama dili kavramlarının belirlenmesi gerekmektedir. Bu kavramlar bir dili diğer programlama dillerinden ayırır. QDil için tercih edilen temel programlama dili kavramları aşağıda verilmektedir.

4.2.1. QDil metodolojisi

QDil saf nesne-yönelimli bir programlama dilidir. Programlama dilindeki her tip bir sınıftır. Programlar, sınıflardan oluşturulan nesnelerin birbirleriyle mesajlaşmasıyla işlemektedir. Nesnelerin, sınıfların oluşturulması ve yönetimi, bellek yönetimi, mesajlaşmalar ve diğer tüm görevlerden QVM sorumludur. Nesneler programlama dili tarafından otomatik olarak silinir.

4.2.2. QDil değişken, metot, sınıf ve arayüz adlandırılması

QDil programlama dilinde kullanılacak olan değişkenlerin, metodların, sınıf ve arayüzlerin adlandırılmasında aşağıdaki tercihler kullanır.

a) Adlar harf ile başlar ve harf,sayı ya da "_" ile devam edebilir.
b) Dile ait ayrılmış kelimeler değişken, metot, sınıf ve arayüzlerin adlandırılmasında kullanılmazlar.
c) Kullanılacak adlar büyük-küçük harf duyarlıdır.

4.2.3. QDil değişkenlerinin özellikleri

QDil programlama dilinde kullanılacak olan değişkenlere ait temel özellikler aşağıdaki gibidir.

a) Değişkenlerin adlandırılmasında 4.2.2. deki yöntemler kullanılır.
b) Değişkenlerin bellek yönetimi QVM tarafından yapılacaktır. Bu yönetim daha sonra ayrıntılı olarak açıklanacaktır.
c) Kuantum tiplerin yönetimi QVM tarafından yapılacaktır. **qbit, qregister** tiplerinin kullanımı için kuantum sistemden gerekli fiziksel kübitlerin ayrılmasını, geri verilmesini ve yönetimini QVM'nin QMM birimi gerçekleştirir. Bu tipler sadece yerel değişkenler olarak kullanılabilir.
d) QDil içerisindeki her değişkenin QIL oluşturulmadan önce belirli bir tipi vardır. Bu tip QIL oluşturumunda ilk hata denetiminde kullanılır. **method** veri tipi sınıfların içerisindeki metotları gösterebilen bir tiptir. metot tipinden

değişkenlerin tip kontrolleri, hata denetimleri program derleme zamanında ve çalışma zamanında gerçekleştirilir.

e) QDil yarı-statik tip kontrolü gerçekleştirmektedir. Metot gibi tiplerin kontrolleri çalışma zamanında gerçekleştirilmektedir.

f) Sınıflara ait metotlar içerisinde tanımlanan değişkenlerin tamamı yerel değişkenlerdir. Bu yerel değişkenler tanımlandıkları tiplere birer referanstır. Metot çağrımı sonlandırıldığında otomatik olarak silinirler. Referans oldukları nesneleri gösteren başka referans yoksa bu nesneler otomatik olarak QVM-OE (Quantum Virtual Machine – Object Eraser) nesne silicisi tarafından silinir. Referans tipindeki tüm değişkenler için yer ayrımı yığından çalışma zamanında gerçekleştirilir.

g) Metot içerisinde değişken birkez tanımlandığında aynı isimle ikinci bir değişkenin alt bloklarca farklı bir etki alanında tanımlanmasına izin verilmez. Yerel değişkenler birkez tanımlanabilirler.

h) Her değişken kullanılmadan tanımlanmalıdır.

i)Değişkenler kod içerisinde istenilen yerde tanımlanabilirler.

j)Global değişkenler yoktur. Tüm değişkenler sınıfların içerisinde ve metotlarda tanımlanırlar.

4.3. QDil Veri Tipleri

QDil programlama dilinde kullanılacak olan veri tipleri hakkında bilgi verilecektir. Tüm veri tipleri sınıftır. İlkel veri tipi yoktur. Tüm sınıfların atası **Object** sınıfıdır. Nesne ve sınıf değişkenlerine otomatik olarak varsayılan değer ataması gerçekleştirilmektedir**.** Sayısal tiplere referans olan değişkenler 0 değerli nesneleri otomatik olarak gösterirler, diğer tüm nesneleri gösteren değişkenlerin varsayılan değeri **null** dur. Mantıksal tipler için varsayılan değer **false** dır.

4.3.1. QDil klasik sayısal tipler

QDil programlama dilinde kullanılan sayısal tipler aşağıdaki gibidir. Sayısal değerler için varsayılan değer olarak 0 atanmaktadır.

a) byte: işaretli 8 bit tamsayıdır.

b) short: işaretli 16 bit tamsayıdır.

c) int: işaretli 32 bit tamsayıdır.

d) long: işaretli 64 bit tamsayıdır.

e) float: işaretli 32 bit tek-duyarlıklı kayan noktalı sayıdır. IEEE 754 ikili kayan-noktalı sayı standardına uygundur.
f) double: işaretli 64 bit çift-duyarlıklı kayan noktalı sayıdır. IEEE 754 standardına uygundur.
g) complex: Kompleks sayının gerçel ve sanal kısımları ardışık iki double sayı olarak tutulur. Kompleks sayılar için JScience (Dautelle J., 2011) kütüphanesi kullanılmıştır.
h) binary: İstenilen uzunluktaki bitten oluşan veriyi göstermede kullanılır. Tüm bitler birer byte olarak tutulmaktadır. Bu veri tipinin nasıl kullanıldığı hakkındaki bilgi daha sonra verilecektir.

4.3.1.1. QDil sayısal tipleri arasındaki işlem tabloları

QDil programlama dilinde kullanılan sayısal tiplerin arasında gerçekleştirilebilecek işleçler ve sonuçları çizelgeleri aşağıdaki gibi verilebilir.

Çizelge 4.2. "+" işlecine ait işleç-tip çizelgesi

+	byte	short	int	long	float	double	complex
byte	short	short	int	long	float	double	complex
short	short	short	int	long	float	double	complex
int	int	int	int	long	float	double	complex
long	long	long	long	long	float	double	complex
float	float	float	float	double	double	double	complex
double	double	double	double	double	double	double	complex
complex	complex	complex	complex	complex	complex	complex	complex

Çizelge 4.3. "-" işlecine ait işleç-tip çizelgesi

-	byte	short	int	long	float	double	complex
byte	byte	short	int	long	float	double	complex
short	short	short	int	long	float	double	complex
int	int	int	int	long	float	double	complex
long	long	long	long	long	float	double	complex
float	float	float	float	float	float	double	complex
double	double	double	double	double	double	double	complex
complex	complex	complex	complex	complex	complex	complex	complex

Çizelge 4.4. "*" işlecine ait işleç-tip çizelgesi

*	byte	short	int	long	float	double	complex
byte	short	short	int	long	float	double	complex
short	short	int	int	long	float	double	complex
int	int	int	int	long	float	double	complex
long	long	long	long	long	float	double	complex
float	float	float	float	float	double	double	complex
double	double	double	double	double	double	double	complex
complex	complex	complex	complex	complex	complex	complex	complex

Çizelge 4.5. "/" işlecine ait işleç-tip çizelgesi

/	byte	short	int	long	float	double	complex
byte	double	double	double	double	double	double	complex
short	double	double	double	double	double	double	complex
int	double	double	double	double	double	double	complex
long	double	double	double	double	double	double	complex
float	double	double	double	double	double	double	complex
double	double	double	double	double	double	double	complex
complex	complex	complex	complex	complex	complex	complex	complex

Çizelge 4.6. "%" işlecine ait işleç-tip çizelgesi

%	byte	short	int	long	float	double	complex
byte	byte	short	int	long	double	double	complex
short	short	short	int	long	double	double	complex
int	int	int	int	long	double	double	complex
long	long	long	long	long	double	double	complex
float	double	double	double	double	double	double	complex
double	double	double	double	double	double	double	complex
complex	complex	complex	complex	complex	complex	complex	complex

Çizelge 4.7. "^" işlecine ait işleç-tip çizelgesi

^	byte	short	int	long	float	double	complex
byte	long	long	long	long	double	double	complex
short	long	long	long	long	double	double	complex
int	long	long	long	long	double	double	complex
long	long	long	long	long	double	double	complex
float	double	double	double	double	double	double	complex
double	double	double	double	double	double	double	complex
complex	complex	complex	complex	complex	complex	complex	complex

Çizelge 4.8. "&" işlecine ait işleç-tip çizelgesi

&	byte	short	int	long	float	double	complex
byte	byte	short	int	long	-	-	-
short	long	long	long	long	-	-	-
int	long	long	long	long	-	-	-
long	long	long	long	long	-	-	-
float	-	-	-	-	-	-	-
double	-	-	-	-	-	-	-
complex	-	-	-	-	-	-	-

Çizelge 4.9. "|" işlecine ait işleç-tip çizelgesi

|	byte	short	int	long	float	double	complex
byte	byte	short	int	long	-	-	-
short	long	long	long	long	-	-	-
int	long	long	long	long	-	-	-
long	long	long	long	long	-	-	-
float	-	-	-	-	-	-	-
double	-	-	-	-	-	-	-
complex	-	-	-	-	-	-	-

Çizelge 4.10. "xor" işlecine ait işleç-tip çizelgesi

xor	byte	short	int	long	float	double	complex
byte	byte	short	int	long	-	-	-
short	long	long	long	long	-	-	-
int	long	long	long	long	-	-	-
long	long	long	long	long	-	-	-
float	-	-	-	-	-	-	-
double	-	-	-	-	-	-	-
complex	-	-	-	-	-	-	-

4.3.2. QDil klasik karakter tipleri

QDil içerisindeki tüm karakter tipleride sınıftır.

a) char: 16 bit işaretsiz tamsayıdır. UTF-16 Unicode şeklinde kodlanmıştır. **c**har tipine ait metotlar ve açıklamaları aşağıda verilmiştir.

Metot	**Açıklama**
+ *extdef getShortValue():short;*	Sayısal değerini short olarak geriye çeviri.
+ *extdef isLower():boolean;*	Karakterin küçük harf olup olmadığını belirtir.
+ *extdef isDigit():boolean;*	Karakterin rakam olup olmadığını belirtir.
+ *extdef isLetter():boolean;*	Karakterin harf olup olmadığını belirtir.
+ *extdef isLetterDigit():boolean;*	Karakterin rakam ya da sayı olup olmadığını belirtir.
+ *extdef isAlpha():boolean;*	Karakterin alfa nümerik olup olmadığını belirtir.
+ *extdef isSpace():boolean;*	Karakterin boşluk olup olmadığını belirtir.
+ *extdef isWhite():boolean;*	Beyaz karakter olup olmadığını belirtir.
+ *extdef toLower():char;*	Küçük harfe dönüştürülmüş halini geri çevirir.
+ *extdef toUpper():char;*	Büyük harfe çevrilmiş halini geriye çevirir.
+ *extdef eql(Object o):boolean;*	Verilen nesneyi el değer eşitliği kontrol edilir.

b) string**:** Bu sınıftan oluşturulan nesnelerin karakter uzunlukları sabittir. Bir string değişkene ekleme ya da çıkarma yapıldığında yeni bir string nesnesi oluşturulur. Unicode karakterlerden oluşurlar. String tipinde kullanılabilecek metotlar ve açıklamaları aşağıda verilmiştir.

Metot	Açıklama
+ extdef getBytes():byte[];	stringin karakterlerinin byte karşılığını geriye çevirir.
+ extdef getChars():char[];	stringin karakterlerini dizi olarak geriye çevirir.
+ extdef toUpper();	stringin tüm karakterlerini büyük harfe çevirir.
+ extdef toLower();	stringin tüm karakterlerini küçük harfe çevirir.
+extdef concat(string val);	stringe verilen diğer stringi sonuna ekler.
+ extdef slice(int start,int end):string raises IndexEception;	Verilen aralıktaki stringi geriye çevirir.
+ extdef reverse();	stringin karakterlerini tersine çevirir.
+ extdef @[](int index):char raises IndexException;	stringin verilen indisli karakterini geriye çevirir.
+ extdef @+(string val):string;	Bir stringin sonuna diğerini ekler.
+extdef @in(string val):boolean;	Bir string içinde diğer bir stringin var olup olmadığı bilgisini geriye çevirir.
+ extdef to_byte():byte raises ConvertException;	stringi byte a çevirir.
+ extdef to_short():short raises ConvertException;	stringi short a çevirir.
+ extdef to_int():int raises ConvertException;	stringi int e çevirir.
+ extdef to_long():long raises ConvertException;	stringi long a çevirir.
+ extdef to_float:float raises ConvertException;	stringi float a çevirir.
+extdef to_double():double raises ConvertException;	stringi double a çevirir.
+ extdef to_complex():complex raises ConvertException;	stringi complex e çevirir.
+ extdef to_binary():binary raises ConvertException	stringi binary e çevirir.

4.3.3. QDil klasik mantıksal tip

QDil içerisindeki mantıksal tipler boolean kelimesiyle tanımlanırlar ve sadece true ya da false değerinden birini alırlar. Boolean tipine ait kullanılabilecek metotlar ve açıklamaları aşağıda verilmiştir.

Metot	Açıklama
+ extdef str_to_boolean(string val):boolean raises ConvertException;	Verilen string i "true" ya da "false" ise booleana çevirir; aksi halde Exception oluşturur.
+ extdef to_byte():byte ;	booleanı byte a çevirir.
+ extdef to_short():short ;	booleanı short a çevirir.
+ extdef to_int():int ;	booleanı int e çevirir.
+ extdef to_long():long ;	booleanı long a çevirir.
+ extdef to_float:float ;	booleanı float a çevirir.
+extdef to_double():double ;	booleanı double a çevirir.
+ extdef to_complex():complex;	booleanı complex e çevirir.
+ extdef to_binary():binary;	booleanı binary e çevirir.
+ extdef @and(boolean val):boolean;	iki booleana ve işlemi uygular.
+ extdef @or(boolean val):boolean;	iki booleana veya işlemi uygular.
+ extdef @!(boolean val):boolean;	boolean değişkenin değilini alır.
+ extdef @xor(boolean val):boolean;	iki booleana dışlamalı ya da işlemi uygular.
+ extdef eql(Object o):boolean;	Verilen nesnenin değerinin eşitliği kontrol edilir.

4.3.4. QDil kuantum veri tipleri

QDil içerisindeki kullanılacak olan kuantum veri tipleri aşağıdaki gibidir.

4.3.4.1. qbit veri tipi

Kuantum programlama dilinin en temel veri yapısıdır. $|0\rangle,|1\rangle$ değerlerinden biri ya da bu hesaplama bazlarının belirli olasılık genlikleriyle süper pozisyonu durumunda olabilir. qbit veri tipinin kullanımıyla ilgili aşağıda bir örnek verilmiştir.

qbit q1 = new qbit(); *q1.set_zero();* *q1.H();*	Bir qbit oluşturur. Kuantum sistemden kullanılabilir bir fiziksel qbit q1 ile ilişkilendirilir. qbittin değeri $\|0\rangle$ yapılır.

int deger = q1.measure().to_int();	qbite H kapısı uygulanarak süper pozisyon durumuna getirilir. $\frac{\|0\rangle+\|1\rangle}{\sqrt{2}}$ Sonrada qbit üzerinde ölçme yapılır. Sonuç ya 0 ya da 1 tam sayısıdır.

Bu tipe uygulanabilecek olan temel işlemler aşağıda verilmiştir.

Metot	**Açıklama**
+ *extdef set_one():boolean*	Kübitin değerini $\|1\rangle$ yapar. Başarılı olunursa geriye true, olmazsa false çevirir.
+ *extdef set_zero():boolean*	Kübitin değerini $\|0\rangle$ yapar. Başarılı olunursa geriye true, olmazsa false çevirir.
+ *extdef measure():binary*	Kübitin değerini ölçer ve geriye 1 ya da 0 değerinden birini binary olarak çevirir.
+ *extdef rot_x(double theta):boolean*	Kübite x yönünde theta açısı kadar döndürme uygulanır. Başarılı olunursa geriye true, olmazsa false çevirir.
+ *extdef rot_y(double theta):boolean*	Kübite y yönünde theta açısı kadar döndürme uygulanır. Başarılı olunursa geriye true, olmazsa false çevirir.
+ *extdef rot_z(double theta):boolean*	Kübite z yönünde theta açısı kadar döndürme uygulanır. Başarılı olunursa geriye true, olmazsa false çevirir.
+ *extdef X():boolean*	Kübite X kapısı uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ *extdef X()':boolean*	Kübite X kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ *extdef Y():boolean*	Kübite Y kapısı uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ *extdef Y()':boolean*	Kübite Y kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.

+ extdef Z():boolean	Kübite Z kapısı uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef Z()':boolean	Kübite Z kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef H():boolean	Kübite Hadamard kapısı uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef H()':boolean	Kübite Hadamard kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef S():boolean	Kübite S kapısı uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef S()':boolean	Kübite S kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef T():boolean	Kübite T kapısı uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef T()':boolean	Kübite T kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef V():boolean	Kübite V kapısı uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef V()':boolean	Kübite V kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef send(qbitMachine destination):boolean;	Kübitin belirtilen kaynaktaki diğer kuantum sistemine gönderilmesini sağlar. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef receive(qbitMachine source):boolean;	Kübitin belirtilen kaynaktaki diğer kuantum sisteminden gelen kübit olmasını sağlar. Başarılı olunursa geriye true, olmazsa false çevirir.

4.3.4.2. qregister veri tipi

QDil programlama dilindeki kuantum veri tiplerinden biridir. Birden fazla kübitin oluşturduğu sistemleri temsil eder. QDil'in yüksek hesaplama gücünü sağlar. Aynı anda birden fazla değeri temsil edebildiği için bu değişkenlere uygulanabilecek işlemler aynı anda tüm değerlere uygulanmış olur. qregister'ın kullanımını gösteren basit bir örnek verilmiştir.

qregister ikiqubit = new qregister(2); *ikiqubit.set(0);* *ikiqubit.H();*	İki qbitten oluşan bir qregister oluşturulur. İki qbittin değeri $\|0\rangle$ yapılır. İki qbite H kapısı uygulanarak süper pozisyona geçmeleri sağlanır. Qregister ın temsil ettiği değer $\alpha_1 \|00\rangle + \alpha_2 \|01\rangle + \alpha_3 \|10\rangle + \alpha_4 \|11\rangle$ $\alpha_i \in \mathbb{C}$ şeklindedir.

Bu tipe uygulanabilecek olan temel işlemler aşağıda verilmiştir.

Metot	**Açıklama**
+ extdef setValue(binary value):boolean;	qregister içerisindeki kübitleri binary sayı içerisindeki bit değerlerini kullanarak ilkler. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef reset():boolean;	qregister içerisindeki tüm kübitlerin değerinin $\|0\rangle$ olmasını sağlar. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef reset(int[] qbits):boolean raises IndexException;	qregister içerisindeki qubits dizisi ile verilen kübitlerin değerinin $\|0\rangle$ olmasını sağlar. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef measure():binary;	qregister içerisindeki kübitlerde hesaplama bazları $(\|0\rangle,\|1\rangle)$ kullanılarak ölçme gerçekleştirilir. Sonuç binary olarak geriye çevrilir.
+ extdef measure(int[] qbits):binary raises IndexException;	qregister içerisindeki qubits dizisi ile verilen kübitlerde hesaplama bazları $(\|0\rangle,\|1\rangle)$ kullanılarak ölçme gerçekleştirilir. Sonuç binary olarak geriye çevrilir.
+ extdef swap (int qbit1, int qbit2):boolean raises IndexException;	qregister içerisindeki numaraları verilen iki kübit arasında yerdeğiştirme işlemi (SWAP) işlemi gerçekleştirir.

+ extdef cnot(int control, int qbit):boolean raises IndexException;	qregister içerisindeki kübitlerden birinci indis değerini kontrol biti olarak kullanır, bu bitin değeri $\|1\rangle$ ise verilen ikinci indisteki kübite NOT kapısı uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef cnot(int control, int qbit)':boolean raises IndexException;	qregister içerisindeki kübitlerden birinci indis değerini kontrol biti olarak kullanır, bu bitin değeri $\|1\rangle$ ise verilen ikinci indisteki kübite NOT kapısının tersini uygular. Başarı durumunu geriye boolean olarak döndürür.
+ extdef cnot(qregister control):boolean raises IndexException;	control isimli qregisterı kontrol kübitleri şeklinde kullanarak bu kübitlerin değerlerinin $\|1\rangle$ olması durumunda kübitlere NOT uygular.
+ extdef cnot(qregister control)':boolean raises IndexException;	control isimli qregisterı kontrol elemanı kullanarak içindeki kübitlere NOT kapısının tersini uygular.
+ extdef zero_controlled_not(int control, int qubit):boolean raises IndexException;	qregister içerisindeki kübitlerden birinci indis değerini kontrol biti olarak kullanır, bu bitin değeri $\|0\rangle$ ise verilen ikinci indisteki kübite NOT kapısı uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef zero_controlled_not(int control, int qubit)':boolean raises IndexException;	qregister içerisindeki kübitlerden birinci indis değerini kontrol biti olarak kullanır, bu bitin değeri $\|0\rangle$ ise verilen ikinci indisteki kübite NOT kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef toffoli (int control1,int control2,int qbit):boolean raises IndexException;	qregister içerisindeki kübitlerden birinci ve ikinci indis değerini kontrol bitleri olarak kullanır, bu bitlerin değeri $\|1\rangle$ ise verilen üçüncü indisteki kübite NOT kapısı uygular. Başarılı olunursa geriye true, olmazsa false çevirir.

+ extdef toffoli (int control1,int control2,int qbit)':boolean raises IndexException;	qregister içerisindeki kübitlerden birinci ve ikinci indis değerini kontrol bitleri olarak kullanır, bu bitlerin değeri $\|1\rangle$ ise verilen üçüncü indisteki kübite NOT kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef QFT():boolean;	Kübitlere kuantum fourier dönüşümünü uygular. Başarılı olursa geriye true, olmazsa false çevirir.
+ extdef QFT()':boolean;	Kübitlere kuantum fourier dönüşümünün tersini uygular. Başarılı olursa geriye true, olmazsa false çevirir.
+ extdef QFT(int[] qbits):boolean raises IndexException;	Verilen indis numaralı qbitlere QFT uygular. Başarılı olursa geriye true, olmazsa false çevirir.
+ extdef QFT(int[] qbits)':boolean raises IndexException;	Verilen indis numaralı qbitlere QFT nin tersini uygular. Başarılı olursa geriye true, olmazsa false çevirir.
+ extdef Phase():boolean;	Kübitlere phase dönüşümünü uygular. Başarılı olursa geriye true, olmazsa false çevirir.
+ extdef Phase()':boolean;	Kübitlere phase dönüşümünün tersini uygular. Başarılı olursa geriye true, olmazsa false çevirir.
+ extdef Phase(int[] qbits):boolean raises IndexException;	Verilen indis numaralı kübitlere Phase dönüşümü uygular. Başarılı olursa geriye true, olmazsa false çevirir.
+ extdef Phase(int[] qbits)':boolean raises IndexException;	Verilen indis numaralı kübitlere Phase dönüşümünün tersini uygular. Başarılı olursa geriye true, olmazsa false çevirir.
+ extdef flipQbits():boolean;	Kübitlerin sırasını tersine çevirir. Başarılı olursa true, olmazsa false çevirir.
+ extdef flipQubits(int[] qbits):boolean raises IndexException;	Verilen indis numaralı kübitlerin sırasını tersine çevirir. Başarılı olursa true, olmazsa false çevirir.

+ extdef qubitCount():int;	qregister içerisinde kaç adet kübit olduğunu geriye çevirir.
+ extdef slice(int start,int end):qregister raises IndexException;	qregister içerisindeki kübitlerin start ile end arasındakilerden yeni bir qregister oluşturur ve geriye çevirir.
+ extdef applyOperator (qoperator operator):boolean;	qregister içerisindeki kübitlere verilen qoperatoru uygular. Qregister içerisindeki kübit sayısı ile qoperatorun uygulanacağı kübit sayısı eşit olmalıdır. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef applyOperator (int[] qbits, qoperator operator):boolean raises IndexException;	qregister içerisinde verilen indis numaralı kübitlere verilen operatör uygulanır. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef controlledApplyOperator(int[] controls, int[] qbits, qoperator operator):boolean raises IndexException;	qregister içerisindeki control_qubits ile verilen kübitlerin $\|1\rangle$ olması durumunda, qubits ile verilen kübitlere, operatörü uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef H():boolean;	qregister içerisindeki tüm kübitlere H kapısını uygular. Böylece tüm kübitlerden oluşan bir süperpozisyon durumu oluşturur. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef H()':boolean;	qregister içerisindeki tüm kübitlere H kapısının tersini uygular. Böylece tüm kübitlerden oluşan bir süperpozisyon durumu oluşturur. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef H(int[] qbits):boolean raises IndexException;	qregister içerisinde verilen qubitlere H kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef H(int[] qbits)':boolean raises IndexException;	qregister içerisinde verilen qubitlere H kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.

+ extdef X():boolean;	qregister içerisindeki tüm kübitlere X kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef X()':boolean;	qregister içerisindeki tüm kübitlere X kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef X(int[] qbits):boolean raises IndexException;	qregister içerisinde verilen qubitlere X kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef X(int[] qbits)':boolean raises IndexException;	qregister içerisinde verilen qubitlere X kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef Y():boolean;	qregister içerisindeki tüm kübitlere Y kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef Y()':boolean;	qregister içerisindeki tüm kübitlere Y kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef Y(int[] qbits):boolean raises IndexException;	qregister içerisinde verilen qubitlere Y kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef Y(int[] qbits)':boolean raises IndexException;	qregister içerisinde verilen qubitlere Y kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef Z():boolean;	qregister içerisindeki tüm kübitlere Z kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef Z()':boolean;	qregister içerisindeki tüm kübitlere Z kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef Z(int[] qbits):boolean raises IndexException;	qregister içerisinde verilen kübitlere Z kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.

+ extdef Z(int[] qbits)':boolean raises IndexException;	qregister içerisinde verilen kübitlere Z kapısının tersini uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef controledRotX(double theta):boolean;	qregister içerisindeki tüm kübitlere $RotX$ kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef controledRotX(int[] qbits,double theta):boolean raises IndexException;	qregister içerisindeki belirtilen kübitlere $RotX$ kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef controledRotY(double theta):boolean;	qregister içerisindeki tüm kübitlere $RotY$ kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef controledRotY(int[] qbits,double theta):boolean raises IndexException;	qregister içerisindeki belirtilen kübitlere $RotY$ kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef controledRotZ(double theta):boolean;	qregister içerisindeki tüm kübitlere $RotZ$ kapısını uygular. Başarılı olunursa geriye true, olmazsa false çevirir.
+ extdef controledRotZ(int[] qbits,double theta):boolean raises IndexException;	qregister içerisindeki belirtilen kübitlere $RotZ$ kapısını uygular.
+ const def @[](int index):qbit raises IndexException;	qregister içerisindeki belirli bir indisteki kübiti geriye çevirir.
+ extdef makeEntangled():boolean;	qregister içerisindeki kübitlerin tam dolanık olmasını sağlar.
+ extdef makeEntangled(int[] qbits):boolean raises IndexException;	qregister içerisinde verilen kübitlerin tam dolanık olmasını sağlar.

4.3.4.3. qoperator veri tipi

QDil programlama dilindeki kuantum veri tiplerinin zaman içerisindeki evrimlerini gerçekleştiren veri tipidir. Kuantum fiziğindeki birimsel dönüşümlere karşılık gelmektedir. QDil ile yazılan kod içerisinde string veri tipi şeklinde tutulurlar. Kendisine ait bir yazım biçimi vardır. Çalışma zamanında QVM 'nin QOPINT (qoperator yorumlayıcısı) tarafından yorumlanmakta ve kuantum kapılarına dönüştürülerek (Anderson ve ark., 1999) kuantum sistemlerde çalıştırılması sağlanmaktadır. QDil'in kuantum hesaplamalardaki

dinamikliğini sağlamaktadır. Yazılan qoperator kodunun yorumlanması çalışma zamanında olduğu için tüm hata denetimleri de çalışma zamanında gerçekleştirilmektedir. Yazımsal hatalar QDil kodunun derlenmesinde belirtilmektedir. qoperator tipinde kullanılabilecek olan ayrılmış kelimeler aşağıdaki gibidir.

I, X, Y, Z, H, S, T, R, V, ROTX, ROTY, ROTZ, QFT, PHASE, ENTANGLE, FLIP, CNOT12, CNOT21, SWAP, TOFFOLI, SIN, COS, SQRT, EXP, SUM

Bu tipin sözdizim kuralları aşağıdaki çizelgede verilmektedir.

Çizelge 4.11. qoperator'un sözdizim kuralları

Kural no	Yazım kuralı
1	goal ::= expr;
2	expr ::= expr_add_sub
3	expr_add_sub ::= expr_mult_div | expr_add_sub "+" expr_mult_div | expr_add_sub "-" expr_mult_div
4	expr_mult_div ::= expr_pow | expr_mult_div "*" expr_pow | expr_mult_div "/" expr_pow
5	expr_pow ::= expr_un | expr_un "^" expr_pow
6	expr_un ::= "-" expr_un | "!" expr_un | expr_ten
7	expr_ten ::= expr_ten_pow | expr_ten "tp" expr_ten_pow
8	expr_ten_pow ::= expr_mtrx | expr_mtrx "tp^" expr_ten_pow
9	expr_mtrx ::= "<" mtrx_rows ">" | expr_lit
10	mtrx_rows ::= mtrx_row | mtrx_rows ";" mtrx_row
11	mtrx_row ::= cols
12	cols ::= expr | cols "," expr
13	expr_lit ::=lit | com_num | expr_cross
14	lit ::= INT_LIT | FL_LIT
15	com_num ::= "{" lit "," lit"}" | "{" lit "," "?""}" | "{" "?" "," lit"}" | "{" "?" "," "?""}"
16	expr_cross ::= "|" expr "><" expr "|" | expr_id_name
17	expr_id_name ::= ID_NAME | expr_cons
18	expr_cons ::= "pi" | "i" | "?" | expr_ops
19	expr_ops ::= "I" | "X" | "Y" | "Z" | "S" | "T" | "H" | "V" | "CNOT12" | "CNOT21" | "SWAP" | "TOFFOLI" | expr_funs

Çizelge 4.11.'in devamı

20	expr_funs ::= "SIN" "[" expr "]" | "COS" "[" expr "]" | "EXP" "[" expr "]" | "SQRT" "[" expr "]" | "QFT" "[" expr "]" | "PHASE" "[" expr "]" | "ENTANGLE" "[" expr "]" | "R" "[" expr "]" | "FLIP" "[" expr "]" | "ROTX" "[" expr "]" | "ROTY" "[" expr "]" | "ROTZ" "[" expr "]" | "SUM" "[" IDENTIFIER "=" expr "," expr "," expr "]" | expr_end
21	expr_end ::= "(" expr ")"

Bu tipe ait kullanılabilecek metotlar aşağıda verilmektedir.

Metot	Açıklama
+ extdef checkErrors():boolean raises QopException;	qoperatorun yazımın bir problem olup olmadığını test eder.
+ extdef isCheckedErrors():boolean;	Hata denetimi yapılıp yapılmadığını belirtir.
+ extdef interpret():boolean raises QOpException;	qoperatorü yorumlar ve qregistera uygulanacak hale getirir.
+ extdef isInterpreted():boolean	Yorumlama işleminin gerçekleşip gerçekleşmediğini belirtir.
+ extdef applyTo(qregister qreg):boolean raises QOpException, NoSuitableSizeException;	Operatörü verilen qregistera uygular. Eğer qregister içindeki qbit sayısı qoperatörün qbit sayısından az ya da fazla ise bir Exception oluşturur.
+ extdef setParam(int param_num, int value):boolean raises IndexException;	Verilen parametre numarasına, verilmiş int değeri yazar.
+ extdef setParam(int param_num, double value):boolean raises IndexException;	Verilen parametre numarasına, verilmiş double değeri yazar.
+ extdef setParam(int param_num, qoperator op):boolean raises IndexException;	Verilen parametre numarasına, verilmiş operatörün yorumlanmamış halini yazar.

qoperator tipi kullanılmadan da QDil programlama dilindeki ifadeler ve kontrol deyimleri aynı kodları yazmamıza olanak sağlamaktadır. Bununla birlikte, kuantum fiziğinin kurallarını bilen bir araştırmacı için bilinen bir yazım şekline sahiptir ve hızlı kod geliştirmeye olanak sağlamaktadır. qoperatorun nasıl yazıldığı ve kullanıldığıyla ilgili aşağıda bazı örnekler verilmiştir.

Operatör	**Dirac Gösterimi**	**Matris Gösterimi**
$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	*qoperator X = "X";* *ve ya* *qoperator X ="|0><1| + |1><0|";*	*qoperator X = "< 0,1 ; 1,0>"*
$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	*qoperator Y = "Y";* *ve ya* *qoperator Y= "-i*|0><1| + i*|1><0|";*	*qoperator Y = "<0,{0,-1};* *{0,1},0>";*
$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	*qoperator Z="Z";* *ve ya* *qoperator Z="|0><0| - |1><1|";*	*qoperator Z="< 1,0 ; 0,-1>" ;*
$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}$	*qoperator T="T";* *ve ya* *qoperator T="|0><0| + EXP[pi*i*1/4]*|1><1|";*	*qoperator T= "<1,0 ; 0, EXP[pi*i*1/4]>";*
$X \otimes I$	*qoperator XI = "X tp 1";*	*qoperator XI = "<0,0,1,0;0,0,0,1; 1,0,0,0;0,1,0,0>";*
$X \otimes X = X^{\otimes 2}$	*qoperator X2="X tp^2";*	*qoperator X2= "<0,0,0,1;0,0,1,0; 0,1,0,0;1,0,0,0>";*
$R_x(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}$	*qoperator Rx="ROTX[?]";* *ve ya* *qoperator Rx = "COS[?/2]*|0><0| - i*SIN[?/2]*|0><1| -*	*qoperator Rx = "<COS[?/2] ,-i*SIN[?/2] ; -i*SIN[?/2] , COS[?/2] >";*

	*i*SIN[?/2]*|1><0| +* *COS[?/2]*|1><1|";*	
$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ $CNOT = \|1\rangle\langle 1\| \otimes X + \|0\rangle\langle 0\| \otimes I$	*qoperator cnot =* *"CNOT12";* *ve ya* *qoperator cnot =* *"|1><1| tp X + |0><0|* *tp I";*	*qoperator cnot=* *"<1,0,0,0;* *0,1,0,0;* *0,0,0,1;* *0,0,1,0>";*
ENTANGLE2= $\|\psi\rangle_1 = H \otimes \|1\rangle\langle 1\| \|\psi\rangle_0$ $\|\psi\rangle_2 = (\|1\rangle\langle 1\| \otimes X + \|0\rangle\langle 0\| \otimes I)\|\psi\rangle_0$	*qoperator entangle =* *"ENTANGLE[2]";*	
$QFT_n : \|x\rangle \rightarrow \frac{1}{\sqrt{n}} \sum_{y=0}^{n-1} e^{2\pi i \frac{x}{n} y} \|y\rangle$	*qoperator qft =* *"QFT[?]";* *qft.setParam(1,n);*	
$QFT_n^{-1} : \|y\rangle \rightarrow \frac{1}{\sqrt{n}} \sum_{x=0}^{n-1} e^{-2\pi i \frac{y}{n} x} \|x\rangle$	*qoperator inv_qft =* *"!QFT[?]";* *inv_qft.setParam(1,n);*	

Yukarıda gösterilen operatörler Nielsen ve Chuang (2000) de bulunmaktadır.

4.3.5. QDil klasik referans veri tipi

QDil içerisindeki metotları gösterebilen **method** referans tipi de bulunmaktadır. Bu tipteki değişken, tanımlı bir metodu gösterebilir ve gösterdiği metodu çağırmada kullanılabilir. Aşağıda tipin kullanımı gösterilmiştir.

package qdil.a; *use qdil.lang.Object;* *use qdil.lang.method;* *class A:Object{* *method yy=self.yaz{+get;+set;}* *+ def yaz(string str){* *}* *+ def A.main(string[] args){* *A aa = new A(); aa.yy("deneme");* *}* *}*	Burada tanımlanan method tipindeki yy değişkeni yaz metodunu göstermektedir ve onun yerine kullanılabilir. Method tipi takma ad gibi kullanılmaktadır.

4.4. QDil Temel Sınıfları

QDil programlama dilinde var olan temel sınıflar aşağıdaki şekilde açıklanmıştır.

4.4.1. QDil Object sınıfı

QDil programlama dilindeki tüm sınıflar bu sınıfın alt sınıfıdır. Object sınıfına ait temel metotlar ve açıklamaları aşağıda verilmektedir.

Metot	Açıklama
+ const extdef getClass():Class;	Geriye Object'in ait olduğu sınıfı çevirir.
# extdef clone():Object;	Bir nesnenin kopyasını oluşturur. Sayısal değişkenlere ait aynı değerdeki başka nesneler oluştururken, diğer nesnelere ait referansların sadece kopyası oluşturulur, nesnelerin kopyaları oluşturulmaz.
+ extdef eql(Object o):boolean;	Bir object'in başka bir object ile içeriklerinin aynı olup olmadığını test eder.
+ extdef hash():int;	Sadece bu nesneye ait bir sayısal kod geri çevirir. eql() metodunda iki nesnenin eşitliği için kullanılır.
+ extdef to_str():string	Geriye nesneyi açıklayan bir string çevirir.

4.4.2. QDil Number sınıfı

QDil programlama dilindeki tüm sayısal tipler bu sınıftan türetilmiştir. Soyut bir sınıftır. Sahip olduğu metotlar ve açıklamaları aşağıdaki gibidir.

Metot	Açıklama
+ abstract def to_byte():byte;	Geriye byte sayı çevirir.
+ abstract def to_short():short;	Geriye short sayı çevirir.
+ abstract def to_int():int;	Geriye int sayı çevirir.
+ abstract def to_long():long;	Geriye long sayı çevirir.
+ abstract def to_float():float;	Geriye float sayı çevirir.
+ abstract def to_double():double;	Geriye double sayı çevirir.
+ abstract def to_complex():complex;	Geriye gerçel kısmının referansın gösterdiği sayı olan bir complex sayı çevirir.
+ abstract def to_binary():binary;	Verilen sayının bitlerinden oluşan bir binary'i geriye çevirir.
+ extdef to_str():string	Geriye sayının string halini çevirir.

4.4.3. QDil binary sınıfı

Bu sınıf sayıları ikili olarak simgeleyen boolean dizisidir. Her boolean değer, sayıdaki bir biti temsil etmektedir. Bu sınıf QDil içerisindeki kahin (oracle) fonksiyonlarının yazılmasında kullanılmaktadır. Kahin fonksiyonları $f:\{0,1\}^n \to \{0,1\}^m$ şeklindeki fonksiyonlardır. Bu fonksiyonlar kuantum algoritmalarında kullanılmaktadır. Klasik kahin fonksiyonu otomatik olarak QDil derleyicisi tarafından kuantum kahin fonksiyonuna çevrilir. Bu amaçla fonksiyonun doğruluk tablosu dışlamalı-ya da (XOR) toplamları şeklinde ifade edilerek sadeleştirilmektedir (Fazel K., Thornton M. ve Rice J.E., 2007; Mischenko A. ve Perkowski M., 2001; Sanee Y. ve Dueck G.W., 2009). Sadeleştirilen ifadeler Toffoli kapıları şeklinde ifade edilmektedir. Kuantum kahin fonksiyonu $f:\mathbb{C}^{n+m} \to \mathbb{C}^{n+m}$ şeklinde terslenebilir bir fonksiyondur. $f:(x,y) \to (x, y \oplus f(x))$ şeklinde gerçekleştirilir. Gerçekleştirilen kuantum kahin fonksiyonu, klasik kahin fonksiyonunun kuantum devre kapılarıyla terlenebilir şekilde ifadesidir. Binary sınıfına ait metotlar ve kullanımları aşağıda verilmektedir.

Metot	Açıklama
+ extdef to_byte():byte;	İkili gösterimin byte sayı karşılığını çevirir. İkili gösterim 8 bitten daha büyük ise en sağdaki ilk 8 biti kullanır. Daha az ise, en sola sıfır koyarak çevirir.
+ extdef to_short():short;	İkili gösterimin short sayı karşılığını çevirir. İkili gösterim 16 bitten daha büyük ise en sağdaki ilk 16 biti kullanır. Daha az ise, en sola sıfır koyarak çevirir.
+ extdef to_int():int;	İkili gösterimin int sayı karşılığını çevirir. İkili gösterim 32 bitten daha büyük ise en sağdaki ilk 32 biti kullanır. Daha az ise, en sola sıfır koyarak çevirir.
+ extdef to_long():long;	İkili gösterimin long sayı karşılığını çevirir. İkili gösterim 64 bitten daha büyük ise en sağdaki ilk 64 biti kullanır. Daha az ise, en sola sıfır koyarak çevirir.
+ extdef to_float():float;	İkili gösterimin float sayı karşılığını çevirir. İkili gösterim 32 bitten daha büyük ise en sağdaki ilk 32 biti kullanır. Daha az ise, en sola sıfır koyarak çevirir.
+ extdef to_double():double;	İkili gösterimin double sayı karşılığını çevirir. İkili gösterim 64 bitten daha büyük ise en sağdaki ilk 64

	biti kullanır. Daha az ise, en sola sıfır koyarak çevirir.
+ extdef to_complex():complex;	İkili gösterimi float sayı karşılığını çevirir ve bunu karmaşık sayının gerçel kısmı için kullanır. İkili gösterim 32 bitten daha büyük ise en sağdaki ilk 32 biti kullanır. Daha az ise, en sola sıfır koyarak çevirir.
+ extdef getBitsCount():int	Binary sayının kaç bitten oluştuğunu geriye çevirir.
+ extdef @[](int index):boolean;	Binary sayının index ile belirtilen byte değeri geriye çevirir.
+ extdef @+(binary val):binary;	Modül 2 ye göre toplama işlemini gerçekleştirir.
+ extdef @and(binary val):binary;	İki binary değişkeni arasında ve işlemi uygular.
+extdef @or(binary):binary;	İki binary değişken arasında veya işlemi uygular.
+ extdef @!():binary;	Verilen binary değişkenin değilini alır.
+ extdef @xor(binary):binary;	İki binary değiken arasında dışlamalı ya da işlemini uygular.

4.5. QDil İfadeler Ve Atama Deyimleri

QDil içerisindeki ifadelerin işleçlerinin nasıl işleneceği ile ilgili bilgi verilecektir.

4.5.1. QDil aritmetik ifadeler

Aritmetik ifadelerin nasıl işleneceği, işleçlerin öncelik kurallarına ve parantezlere bağlıdır.

4.5.1.1. QDil aritmetik işleçlerin hesaplanma sırası

İşleçlerin öncelik ve birleşme kuralları hesaplanma sırasını belirler. Aşağıdaki çizelgede öncelik sıraları yukarıdan aşağıya azalan bir şekilde verilmiştir.

İşleç	**Açıklama**
( )	Gruplama
^	Üs Alma
+ , -	Tek işlenenli işleçler, pozitif ve negatif yapma

* , / , %	Çarpma, bölme ve modül alma işleçleri
+ , -	Toplama ve çıkarma işleci
<< , >>, <<<	Sola öteleme, sağa öteleme, sola işaretsi öteleme

Eşit önceliğe sahip işleçler, üs alma işlemi hariç, soldan sağa doğru çalıştırılırlar. Üs alma işlemi sağdan sola doğru çalıştırılır.

4.5.1.2. QDil işleçlerin aşırı yüklenmesi

QDil programlama dilinde yukarıda tanımlı olan işleçler sadece kullanıcı tanımlı sınıflar için aşırı yüklenebilir. Ön tanımlı veri tipleri için aşırı yükleme yapılamaz.

4.5.2. QDil ilişkisel ve mantıksal işleçler

4.5.2.1. QDil ilişkisel işleçler

İlişkisel işleçler iki değişkenin değerinin karşılaştırır ve karşılaştırma sonucunu boolean olarak geriye döndürür. QDil programlama dilindeki ilişkisel işleçler aşağıda verilmiştir.

İşleç	**Açıklama**
<	a < b şeklinde kullanılır. a nın b den küçük olup olmadığını karşılaştır. Küçükse true, değilse false değeri geriye çevirir.
>	a > b şeklinde kullanılır. a nın b den büyük olup olmadığını karşılaştır. Büyükse true, değilse false değeri geriye çevirir.
<=	a <= b şeklinde kullanılır. a nın b den küçük ve ya eşit olup olmadığını karşılaştır. Küçük ve ya eşitse true, değilse false değeri geriye çevirir.
>=	a >= b şeklinde kullanılır. a nın b den büyük ve ya eşit olup olmadığını karşılaştır. Büyük ve ya eşitse true, değilse false değeri geriye çevirir.
in	a in b şeklinde kullanılır. a nın b içerisinde olup olmadığını belirler. İçerisindeyse true, değilse false değeri geriye çevirir.

isa	a isa b şeklinde kullanılır. a nın b sınıfından olup olmadığını belirler. a, b sınıfından ya da ondan türetilen bir sınıfsa geriye true, değilse false değeri geriye çevirir.
?=	a ?= b şeklinde kullanılır. a nın b ye eşit olduğunu belirler. Eşitse true, değilse false değeri geriye çevirir. Burada nesnelerin hashcode ları karşılaştırılır.
!=	a != b şeklinde kullanılır. a nın b ye eşit olmadığını belirler. Eşit değilse true, eşitse false değeri geriye çevirir. Burada nesnelerin hashcode ları karşılaştırılır.

4.5.2.2. QDil mantıksal işleçler

İlişkisel ifadeler arasında kullanılırlar. Birden fazla ilişkisel ifadenin bir arada doğruluklarını belirlemek için kullanılır. QDil programlama dilindeki mantıksal işleçler aşağıda verilmiştir.

İşleç	**Açıklama**
and	a and b şeklinde kullanılır. a ve b boolean tipinden olmalıdır. Mantıksal ve işlemini bu iki değişkene uygular ve sonucu mantıksal değişken olarak çevirir.
or	a or b şeklinde kullanılır. a ve b boolean tipinden olmalıdır. Mantıksal ve ya işlemini bu iki değişkene uygular ve sonucu mantıksal değişken olarak çevirir.
xor	a xor b şeklinde kullanılır. a ve b boolean tipinden olmalıdır. Mantıksal dışlamalı ve ya işlemini bu iki değişkene uygular ve sonucu mantıksal değişken olarak çevirir.
!	! a şeklinde kullanılır. Bir boolean değişkenin tersini alarak geriye çevirir.

Yukarıda verilen işleçler de kısa devre hesaplama gerçekleşmektedir. Birden fazla mantıksal işleç içeren ifadelerde herhangi birinin hesaplanmasıyla sonuç biliniyorsa böylece diğerleri hesaplanmaz. Örneğin:

(A ?=0) and (B!= -1)

Mantıksal ifadesinde A sayısı 0 a eşit değilse B nin -1 e eşit olup olmadığı kısa devre hesaplamada kontrol edilmez. Bununla birlikte QDil içerisinde kısa devre hesaplamanın

yapılmaması istenilen durumlar için yukarıdaki işleçlerin kısa devre yapmayan şekilleri bulunmaktadır. Bu işleçler aşağıda gösterilmiştir.

İşleç	**Açıklama**
AND	a AND b şeklinde kullanılır. a ve b boolean tipinden olmalıdır. Mantıksal ve işlemini bu iki değişkene uygular ve sonucu mantıksal değişken olarak çevirir. a ve b nin her ikisininde değeri hesaplanır.
OR	a OR b şeklinde kullanılır. a ve b boolean tipinden olmalıdır. Mantıksal veya işlemini bu iki değişkene uygular ve sonucu mantıksal değişken olarak çevirir. a ve b nin her ikisininde değeri hesaplanır.
XOR	a XOR b şeklinde kullanılır. a ve b boolean tipinden olmalıdır. Mantıksal dışlamalı ve ya işlemini bu iki değişkene uygular ve sonucu mantıksal değişken olarak çevirir. a ve b nin her ikisininde değeri hesaplanır.

4.5.3. QDil kontrol deyimleri

4.5.3.1. QDil seçim kontrolleri

Yazılan deyimler arasında kontrol deyimlerinin durumuna göre yönlenme sağlar.

4.5.3.1.1. QDil iki yönlü seçim

Kontrol deyimi olarak sadece mantıksal ifadeler kullanılabilir. Kontrol deyiminin doğru olduğu durumda birinci blok, yanlış olduğuna ikinci blok çalışır. Bir kuantum tip aynı boyutlu başka bir kuantum tiple ya da bir tamsayıyla karşılaştırılabilir. Kontrolün doğru olduğu durum için blok kontrollü kapılara dönüştürülür. Kontrol bitleri, kontrol deyimindeki kuantum tipin kübitlerini kontrol amacıyla kullanılır. Aynı anda klasik kontrol ve kuantum kontrol bir arada yapılıyorsa öncelikli olarak klasik kontrol ifadesinin doğruluğu sınanır daha sonra bu kontrolün doğruluğu durumunda kuantum ifadeler çalıştırılır.

İki yönlü seçimin tanımlanması aşağıdaki gibidir.

if (kontrol_deyimi){ //blok }	kontrol deyiminin sonucunun doğru olduğunda blok kısmındaki diğer kodlar çalıştırılır.
if (kontrol_deyimi){ // doğru bloğu }else{ //kontrol yanlış bloğu } }	kontrol deyimi doğru olduğunda kontrol doğru bloğu, aksi durumda yankış bloğu çalıştırılır.

İki yönlü seçimin kullanımı aşağıdaki örnekle verilmiştir.

if (a?=1 and b<5) { *}*	Burada a değişkeninin 1 olması ve b değişkeninin 5 den küçük olması durumunda blok içeriğindeki kod çalıştırılır.
if (a?=1) { *b.X();* *}else{* *b.Y();* *}*	Burada a değişkeni klasik bir değişken ise kontrolü klasik olarak yapılır ve kontrol sonucuna göre b kuantum tipine X ya da Y kapıları uygulanır. Bununla birlikte a nın kuantum tipi olması durumunda a değişkeninin kübitlerinin tamamı kontrol kübiti olarak kullanılarak ard arda kontrollü X ve Y kapıları eklenir. Kontrollü-X kapısında a nın kübitlerinin değerinin 1 olduğu kontrol edilirken, kontrollü-Y kapısında a nın kübitlerinin değerinin 1 olmadığı kontrol edilir.

4.5.3.1.2. QDil çok yönlü seçim

Kontrol deyimi olarak sadece mantıksal ifadeler kullanılabilir. Birden fazla if şeklinde gerçekleştirilmektedir. Çok yönlü seçim ifadesi ve kullanımı aşağıda verilmektedir.

case (ifade){ *when( ){* *}* *when(){* *}* *...* *others{* *}* *}*	İfade kısmında bir değişken yazılır. Bu değişkenin değeri when bloklarında eşitlik için kontrol edilir. When bloklarıından birinde eşitlik sağlanırsa bu blok içindeki komutlar çalıştırılır. Hiçbir when bloğunda eşitlik sağlanmazsa others bloğu yazılmış ise çalıştırılır.

Çok yönlü seçimin kullanım örneği aşağıdaki gibidir.

int[] c ={2,5,7}; *case(a){* *when(1){* *b.H();* *}* *when(2){ b.X(); }* *when(in c){*	a değişkenini değerinin kontrol yapıldığı bir ifadedir. Her when bloğunda a nın belirli bir değerinin olup olmadığı kontrol edilir.

b.Z(); *}* *others{* *b.RotX(Math.Pi/2.0);* *}* *}*	

4.5.4. QDil tekrarlama deyimleri

4.5.4.1. QDil sayaç kontrollü döngü

Sayaç kontrollü döngülerde bir sayaç değişkeni bulunur. Bu değişken ilk değerden başlar ve belirli bir sayıya ulaşana ya da geçene kadar belirli bir adım miktarıyla arttırılır. Her sayaç arttırımında döngü içindeki deyimler tekrar tekrar çalıştırılır. QDil deki sayaç kontrollü döngü yapısı aşağıdaki gibi gösterilir.

```
for (değişken = ilk_değer, son_değer, adım_miktarı){
}
for (değişken = ilk_değer, son_değer){
}
```

Buradaki döngü değişkeni ilk değerden başlayarak adım miktarı kadar arttırılarak son değere ulaşır ya da geçer. Buradaki adım miktarı negatif bir sayı olabilir. Buradaki değişkenin tipi tamsayı olabilir. Adım miktarı yazılmazsa (+1) kabul edilir. İlk değer, son değer ve adım miktarları birer ifade olabilir. Bu ifadeler döngü çalışmaya başlamadan önce bir kez hesaplanırlar. Bu değişkenlerin döngü içerisinde değiştirilmesi döngüyü etkilemez. Döngü değişkeni for bloğu içerisinde sadece-okunabilir bir değişken olarak değerlendirilir. Aşağıda kullanıma bir örnek verilmiştir.

for (i = 1 , 5) { *if (a[i] ?= 1) {* *b[i].X();* *}* *}*	Burada i değişkeni klasik bir değişkendir. a qregister tipinde verilen i indisli kübitler kontrol biti olarak kullanılırlar. Kontrol bitlerinin 1 olduğu durumlarda b qregister tipinin indis ile belirtilen kübitine X kapısı uygulanır.

4.5.4.2. QDil mantıksal kontrollü döngüler

Mantıksal kontrollü döngülerde kontrol ifadesi olarak sadece true ya da false üreten mantıksal ifadeler kullanılabilir. Mantıksal kontrollü döngüler ön kontrollü ve son

kontrollü olmak üzere ikiye ayrılır. QDil de bir adet ön kontrollü ve bir adet son kontrollü döngü bulunmaktadır. Bu döngüler aşağıda gösterilmiştir.

while (mantıksal_kontrol){ *}*	Mantıksal kontrol ifadesi true olduğu sürece blok içerisindeki kodlar gerçekleştirilir.
do{ *}until(mantıksal_kontrol);*	Bu döngü içerisindeki kodlar en az birkez işlenir, daha sonra mantıksal kontrol ifadesi true oluncaya kadar döngü içerisindeki kodlar tekrarlanır.

Bu döngülerin kullanımıyla ilgili örnekler aşağıda verilmektedir.

int i=1; *while (i<=5){* *if (a[i] ?= 1) {* *b[i].X();* *}* *i = i + 1;* *}*	Burada i klasik değişkeni 1 ile 5 arasındaki değerleri sırasıyla alır. i değişkeni indis değeri olarak kullanılır. A qregisterının i indisli kübiti kontrol biti olarak kullanılır. Kontrolün 1 olduğu durumlarda b qregisterının i indisli kübitine X kapısı uygulanır.
int i=1; *do {* *if (a[i] ?= 1) {* *b[i].X();* *}* *i = i + 1;* *} until (i>5);*	Buradaki döngü yukarıdaki while örneği ile aynıdır. İ değişkeninin aldığı değer kontrolü while a göre farklı olarak döngü çalıştıktan sonra yapılır.

4.5.4.3. QDil veri yapılarına bağlı döngü

QDil içerisinde diziler ve bağlı listeler gibi çeşitli veri yapıları vardır. Ayrıca kullanıcı kendi veri yapılarını da geliştirebilir. Bu veri yapıları üzerinde hareket ederek elemanlarını gezmeye yarayan bir döngüdür. QDil içerisinde tanımlı veri yapıları Iterator arayüzünün metotlarını gerçekleştirmiştir. Aşağıdaki gösterilen döngü bu arayüzün metotlarını veri yapısının elemanlarını gezmede kullanmaktadır. Bu arayüzü gerçeklemeyen bir sınıfta bu döngü kullanılamaz.

each (VeriTipi değişken : veri_yapısı){ *}*	Veri yapısı içerisindeki nesnelerin tek tek değişken tarafından gösterilmesini sağlar.

Bu döngünün kullanım örneği aşağıda verilmektedir.

int[] dizi = {1,2,3,4,5}; *each (int i : dizi) {* *if (a[i] ?= 1) {* *b[i].X();* *}* *}*	i değişkeni sırasıyla dizi içerisindeki elemanların değerlerini alır. i değişkeni indis değişkeni olarak kullanılır. a qregisterının i indisli kübiti kontrol olarak kullanılarak b qregisterının i indisli kübitine X kapısı uygulanır.

4.5.4.4. QDil döngü deyimlerinden çıkış

Bazen döngü deyimleri içerisinden çıkış yapmak gerekebilir. Bu durumlarda kullanılabilecek olan deyimler aşağıda verilmiştir.

break;	Yazıldığı yere en yakın konumdaki döngüden dışarıya çıkar.
breakall;	Yazıldığı yerdeki iç içe tüm döngülerden dışarıya çıkar.
continue;	Yazıldığı yere en yakın döngünün kontrol yapılan noktasına gider.

4.5.5. QDil hata kontrol deyimi

QDil programlama dilinde derleme ve çalıştırma zamanında hatalar oluşabilir. Bu hatalar Exception sınıfından nesneler şeklinde QVM tarafından oluşturulur. Program geliştiricilerde isterlerse kendi hatalarını oluşturabilir ve **raise** özel kelimesiyle bu hataların değerlendirilmesi için fırlatabilirler. QDil de hatalar aşağıdaki çizelgede verilen **try** deyimiyle yakalanırlar ve değerlendirilirler. Hata kontrol deyimleri aşağıda gösterilmiştir.

try { *// hata oluşma olasılığı bulunan kod* *}catch (Exception e){* *}*	Try bloğu içerisinde yazılan kodlarda oluşacak olan hatalar catch bloğunda yakalanarak işlenirler.

try { *// hata oluşma olasılığı bulunan kod* *}catch (Exception e){* *}finally{* *// her durumda çalışacak kod }*	Finally bloğunun çalışmasını hatanın olması ya da olmaması etkilemez. Her durumda bu blokta yazılan kodlar çalıştırılır.

Hata kontrol ifadelerinin kullanımı ile ilgili örnek aşağıda verilmektedir.

try{ *qbit q1 = new qbit();* *}catch(QbitNotAvailableEx e){* *// bir kübit QMM den istenildiğinde* *//kullanılabilir qbit olmadığında oluşur.* *}*	Burada yeni bir kübitin oluşturulması amaçlanmaktadır. Kuantum Bellek Yöneticisi (QMM) nin kullanabileceği boşta ve uygun bir kübit yoksa bir hata oluşturulur. Bu hata catch tarafından yakalanır ve kullanıcı tarafından işlenir.

4.6. QDil Nesneye Yönelik Programlama Kavramları

QDil nesneye yönelik programlama dilidir. Birden fazla sınıftan türetilmeye izin verilmez. Çoklu mirasın sağlanması için arayüz kavramı kullanılmaktadır. Sınıfların ve arayüzlerin tanımlanması aşağıda açıklanmaktadır. Aşağıdaki açıklamalarda "[ ]" arasında yazılan ifadelerden bir tanesi yazılacaktır. $\mathcal{E}$ ifadesi hiçbir şeyin yazılmadığı durumu göstermektedir.

4.6.1. QDil sınıflarının tanımlanması

QDil içerisinde, her **.qdil** dosyası içerisinde sadece tek bir sınıf tanımlanabilir. Sınıf ismi ile dosya ismi aynı olmalıdır. Aşağıda bir sınıfın genel yapısı örnek verilmektedir.

Sınıf terimi	Açıklama
package paket_adı; *package A;*	Sınıflar bir paket tanımıyla başlar. Her sınıfın bir pakete ait olması zorunludur.
use paket_adı.sınıf_adı; *use B.C;* *use B.D;*	use deyimi bir paket içerisindeki sınıfın sizin geliştirdiğiniz sınıfta nesnesi oluşturularak kullanılacağını gösterir. Kullanılan her sınıf bu şekilde package deyiminden sonra tanımlanmalıdır.
class sınıf_adı : üst_sınıf, arayüzler{ *//sınıf elemanları }*	Bir sınıf sadece tek bir üst sınıftan türeyebilir. Birden fazla arayüzü

class A : B,IC,ID{ *}*	gerçekleyebilir. Arayüzler virgül ile ayrılarak yazılırlar.
abstract class sınıf_adı : üst_sınıf,arayüzler{ *//sınıf elemanları* *}*	Soyut sınıf tanımlanmasıdır. Soyut sınıf içerisinde en az bir tane soyut metot olmalıdır. Soyut sınıflardan nesne oluşturulamaz.
const class sınıf_adı : üst_sınıf, arayüzler{ *//sınıf elemanları* *}*	const deyimiyle yazılan sınıftan alt sınıflar geliştirilemez.

4.6.2. QDil sınıflarının iç yapısı

QDil sınıflarının iç yapısında aşağıdaki elemanlar bulunmaktadır.

4.6.2.1. QDil sınıf elemanlarının erişim denetimcileri

QDil sınıflarında tanımlanan her elemanın bir erişim denetleyicisi bulunmalıdır. Erişim denetleyicileri, sınıf elemanlarına sınıfın içinden, dışından ve paketin içinden dışından nasıl erişilebileceğini belirtirler. QDil de kullanılan erişim denetleyicileri aşağıda gösterilmiştir.

+	Bu şekilde tanımlanan elemanlar genel elemanlardır ve tüm paketlerdeki tüm sınıflarca erişilebilirler.
-	Bu şekilde tanımlanan elemanlar özel elemanlardır. Sadece tanımlandıkları sınıfın elemanlarınca erişilebilirler.
#	Bu şekilde tanımlanan elemanlar korunmuş elemanlardır. Sadece tanımlandıkları sınıf ve bu sınıftan türetilmiş alt sınıflarca erişilebilirdirler.
ε	Bir şey yazılmaması durumunda bu elemanlar paket genel olarak adlandırılırlar. Sınıfın üyesi olduğu paket içerisindeki tüm sınıflarca erişilebilirdirler.

4.6.2.2. QDil alan tanımlanması

QDil sınıflarında iki tip alan vardır. Bunlardan birincisi nesne değişkenleri, diğeri sınıf değişkenleridir. Nesne değişkenleri her nesne için kendine ait olarak bulunurken, sınıf değişkenleri bu sınıftan türetilen tüm nesnelerce ortak olarak kullanılırlar. Aşağıda bu değişkenlerin nasıl yazıldıkları görülmektedir.

tip değişken_adı { *[+ | - | # | Ɛ] get{ }* *[+ | - | # | Ɛ] set{ }* *}*	Bu tanımlama nesneye ait alanlardır. Her nesne oluşturulmasında bu tip alandan bir tane nesneye ait olarak oluşturulur. Bu tip değişkene atamalarda set metodu, okumalarda get metodu işlemlerden önce çalıştırılır.
tip değişken_adı { *[+ | - | # | Ɛ] get;* *[+ | - | # | Ɛ] set;* *}*	get ve set metotlarında herhangi bir işlem yapılmak istenmezse bu şekilde yazılabilir.
tip SınıfAdı.değişken_adı { *[+ | - | # | Ɛ] get{ }* *[+ | - | # | Ɛ] set{ }* *}*	Değişkenler bu şekilde tanımlanırsa sınıf değişkeni adlandırılır ve sınıftan türetilen tüm nesnelerce ortak olarak kullanılırlar.

4.6.2.3. QDil metotların tanımlanması

QDil sınıflarında iki tip metot vardır. Bunlardan birincisi nesne metotları, diğeri sınıf metotlarıdır. Sınıf metotları sınıftan bir nesne olmasa da kullanılabilirler ve kodları içerisinde sadece sınıf değişkenlerinin kullanımına izin verilir. Nesne metotları, nesne değişkenlerini ve sınıf değişkenlerini kullanabilirler. Bu metotların nasıl tanımlandıkları aşağıda gösterilmektedir.

```
[+ | - | # | Ɛ ] [const | Ɛ ] def metot_adı (formal_parametreler)
[ :geri_dönüş_tipi | Ɛ ] [raises fırlatılan_exception_listesi | Ɛ ]
{   }
```

Yukarıda gösterilen metot nesne metodudur. Metot **const** ifadesiyle tanımlanırsa bu metodun alt sınıflarca üzerine yazılarak değiştirilmesi engellenir. Sınıf metotları aşağıda gösterildiği gibi tanımlanırlar.

```
[+ | - | # | Ɛ ] [const | Ɛ ] def SınıfAdı.metot_adı (formal_parametreler)
[ :geri_dönüş_tipi | Ɛ ] [raises fırlatılan_exception_listesi | Ɛ ]
{
}
```

Soyut metotlar aşağıdaki çizelgedeki gibi tanımlanırlar. Soyut metotların kodları bulunmaz. Soyut metodun tanımlandığı sınıfın alt sınıflarınca kodları yazılmak zorundadır. Bu metotlar aşağıdaki gibi tanımlanırlar.

[+ | - | # | Ɛ] abstract def metot_adı (formal_parametreler) [:geri_dönüş_tipi | Ɛ] [raises fırlatılan_exception_listesi | Ɛ] ;

QDil java ile geliştirildiği için şu anda java dili harici dil olarak kulanılmaktadır. Harici metotlar aşağıda gösterildiği gibi tanımlanırlar.

[+ | - | # | Ɛ] [const | Ɛ]extdef metot_adı (formal_parametreler) [:geri_dönüş_tipi | Ɛ] [raises fırlatılan_exception_listesi | Ɛ] ;

Harici metotların çalıştırılması QVM tarafından çalışma zamanında dinamik olarak gerçekleştirilir. Bu metotlar tanımlandıkları sınıftan türetilen alt sınıflarca üzerlerine yazılarak değiştirilemezler.

4.6.2.4. QDil işleç metotlarının tanımlanması

QDil belirli işleçlerin aşırı yüklenmesine kullanıcı sınıfları için izin vermiştir. İşleç metotlarının nasıl tanımlandığı aşağıda gösterilmektedir.

[+ | - | # | Ɛ] [const | Ɛ] [def | extdef] işleç_adı (formal_parametreler) [:geri_dönüş_tipi | Ɛ] [raises fırlatılan_exception_listesi | Ɛ] { }

Aşağıda QDil'de kullanılabilecek işleç adları verilmektedir.

İşleç Adı	**Açıklama**
@+	**+ ,** işleminin üzerine yazmak için kullanılır. a + b işlemi, a.@+(b) şeklinde QVM tarafından çalıştırılmaktadır. Bu işleçin tek bir formal parametre alması gerekmektedir.
@-	**- ,** işleminin üzerine yazmak için kullanılır. a - b işlemi, a.@-(b) şeklinde QVM tarafından çalıştırılmaktadır.
@/	**/ ,** işleminin üzerine yazmak için kullanılır. a / b işlemi, a.@/(b) şeklinde QVM tarafından çalıştırılmaktadır.
@%	**% ,** işleminin üzerine yazmak için kullanılır. a % b işlemi, a.@%(b) şeklinde QVM tarafından çalıştırılmaktadır.

*@**	*** ,** işleminin üzerine yazmak için kullanılır. a * b işlemi, a.@*(b) şeklinde QVM tarafından çalıştırılmaktadır.
@&	**& ,** işleminin üzerine yazmak için kullanılır. a & b işlemi, a.@&(b) şeklinde QVM tarafından çalıştırılmaktadır.
@|	**| ,** işleminin üzerine yazmak için kullanılır. a | b işlemi, a.@|(b) şeklinde QVM tarafından çalıştırılmaktadır.
*@^*	**^ ,** işleminin üzerine yazmak için kullanılır. a ^ b işlemi, a.@^(b) şeklinde QVM tarafından çalıştırılmaktadır. Bu işleçin tek bir formal parametre alması gerekmektedir.
@>>	**>>,** işleminin üzerine yazmak için kullanılır. a >> b işlemi, a.@>>(b) şeklinde QVM tarafından çalıştırılmaktadır. Bu işleçin tek bir formal parametre alması gerekmektedir.
@<<	**<<,** işleminin üzerine yazmak için kullanılır. a << b işlemi, a.@<<(b) şeklinde QVM tarafından çalıştırılmaktadır. Bu işleçin tek bir formal parametre alması gerekmektedir.
@in	**in,** işleminin üzerine yazmak için kullanılır. a in b işlemi, a.@in(b) şeklinde QVM tarafından çalıştırılmaktadır. Bu işleçin tek bir formal parametre alması gerekmektedir.
@!	**!,** işleminin üzerine yazmak için kullanılır. !a işlemi, a.@!() şeklinde QVM tarafından çalıştırılmaktadır. Bu işleç parametre almaz.
@[]	**[],** işleminin üzerine yazmak için kullanılır. a[değişken] işlemi, a.@[](değişken) şeklinde QVM tarafından çalıştırılmaktadır. Bu işleçin tek bir formal parametre alması gerekmektedir.
@and	**and ,** işleminin üzerine yazmak için kullanılır. a and b işlemi, a.@and(b) şeklinde QVM tarafından çalıştırılmaktadır. Bu işleçin tek bir formal parametre alması gerekmektedir.
@or	**or ,** işleminin üzerine yazmak için kullanılır. a or b işlemi, a.@or(b) şeklinde QVM tarafından çalıştırılmaktadır.
@?=	**?= ,** işleminin üzerine yazmak için kullanılır. a ?= b işlemi, a.@?=(b) şeklinde QVM tarafından çalıştırılmaktadır.
@!=	**!= ,** işleminin üzerine yazmak için kullanılır. a != b işlemi, a.@!=(b) şeklinde QVM tarafından çalıştırılmaktadır. Bu işleçin tek bir formal parametre alması gerekmektedir.

4.6.3. QDil arayüzlerinin tanımlanması

QDil çoklu kalıtıma izin vermez. Bu nedenle çoklu kalıtımı sağlamak ve sınıfların ortak metot arayüzüne sahip olmasını sağlamak için arayüz kavramı kullanılmıştır. Arayüzler sadece sabit değişkenler ve soyut metotlar içerirler. Bir sınıf birden fazla arayüzün metotlarını gerçekleyebilir. Arayüzlerde .qdil dosyası içerisinde yazılırlar. Bir dosya içeçrisinde sadece tek bir arayüz tanımı olabilir. Arayüzün adı ile dosyanın adı aynı olmalıdır. QDil içerisindeki arayüz tanımlaması aşağıda gösterilmiştir.

package paket_adı; *package A;*	Arayüzler bir paket tanımıyla başlar. Her sınıfın bir pakete ait olması zorunludur.
use paket_adı.sınıf_adı; *use B.IC;* *use B.ID;*	use deyimi bir paket içerisindeki sınıfın sizin geliştirdiğiniz arayüzde kullanılacağını gösterir. Kullanmak istediğiniz her sınıfı bu şekilde package deyiminden sonra tanımlamalısınız.
interface interface_adı: arayüzler{ *}* *interface IA : IB,IC,ID{* *}*	Bir arayüz birden fazla arayüzden türetilebilir.

4.6.3.1. QDil arayüzlerinin alanlarının tanımlanması

QDil içindeki arayüzlerde sadece sabit alanlar tanımlanabilir. Bu alanlara ilk değer ataması yapılması zorunludur. Arayüz içerisindeki tüm alanlara genel olarak herkes tarafından erişilebilir. Bu alanlar sabittir ve içerikleri değiştirilemez. Arayüz alanlarının nasıl tanımlandığı aşağıda verilmektedir.

tip değişkenin_adı = ilkdeğer;	Arayüz içerisinde bulunan tüm alanlar bu şekilde tanımlanırlar. İçerikleri değiştirilemez.

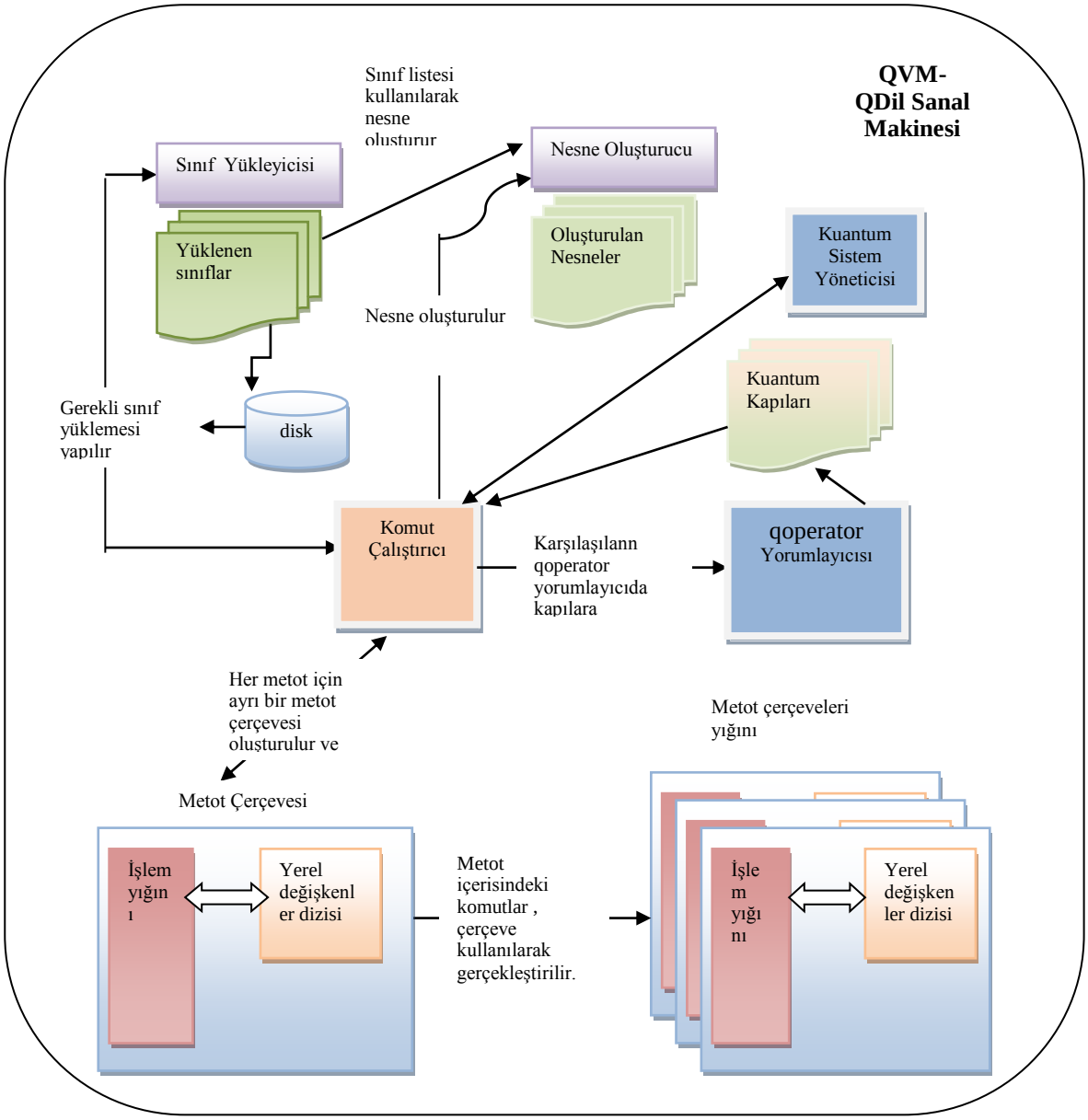
4.6.3.2. QDil arayüz metotlarının tanımlanması

QDil içindeki arayüz metotlarının tamamı geneldir, tüm sınıflarca erişilebilirler ve soyutturlar. Bu metotların kodları bu arayüzü gerçekleyen sınıflarca yazılmak zorundadır. Bu metotların nasıl tanımlandıkları aşağıda gösterilmektedir.

[+ | - | # | ε] abstract def metot_adı (formal_parametreler) [:geri_dönüş_tipi | ε] *[raises fırlatılan_exception_listesi | ε] ;*

4.7. QDil Sanal Makinesi Yapısı

QDil sanal makinesi bir yığın makinesi gibi çalışmaktadır. Komutlar tüm işlemleri bu yığını kullanarak gerçekleştirir. Sanal makine byte veri tipini temel olarak kullanır. QDil sanal makinesinin temel yapısını aşağıdaki resim ile gösterebiliriz.



Şekil 4.1. Kuantum sanal makinesi yapısı.

QDil sanal makinesi içindeki yukarıdaki şekilde tanımlanan bazı birimleri açıklayalım:

a) Klasik Komut Çalıştırıcısı: Bu birim metot içerisindeki klasik kodları çalıştırmadan sorumludur. Bu çalıştırma esnasında bazı sınıfların belleğe yüklenmesi, yüklenen sınıflardan nesneler oluşturulması gerekebilir. Bu durumda sınıf yükleyicisini kullanarak sınıfları QVM nin sınıf listesine ekler. Nesne oluşturucusunu kullanarak bu sınıftan nesneler oluşturur. Oluşan nesneler çalıştırılan metodun yerel değişkenler listesine referans olarak eklenirler. Yeni metot çağrımında, bu metoda ait yeni bir metot çeçevesi oluşturulur. Bu metoda gönderilecek olan parametreler metodun yerel değişkenler listesine eklenir. Her metoda ait bir işlem yığını bulunmaktadır. Klasik komut çalıştırıcı metoda ait bu işlem yığınını kullanır. Metot sonlandığında, sonlanana metoda ait metot çerçevesi QVM nin metot çerçeveleri yığınından çıkarılır ve bir önceki metot çerçevesine geri dönülür.
b) Kuantum Komut Çalıştırıcısı: Bu birim metot içerisindeki kuantum kodlarını çalıştırmadan sorumludur. Karşılaşılan yeni kuantum veri tiplerinin oluşturulmasında QMM (kuantum bellek yöneticisi) kullanılır ve fiziksel kuantum sisteminden gerekli kübitler kullanılmak için ayrılırlar. Kullanılan kübitlerin geri verilmeside QMM aracılığıyla kuantum komut çalıştırıcı tarafından yapılır. Kuantum komut çalıştırıcısı karşılaştığı qoperator işlemlerinde qoperator yorumlayıcısını çalıştırır.
c) qoperator Yorumlayıcısı: Çalışma zamanında string şeklinde kendisine verilen operatörü yorumlayarak uygun kapılarak çevirmeden sorumludur. Bu kapıları oluşturur ve QVM içerisindeki çalıştırılacak kapılar listesine ekler. Kuantum komut çalıştırıcı bu kapıları fiziksel kuantum sistemde çalıştırır.
d) Sınıf Yükleyici: Bu birim sınıfların diskten belleğe yüklenmesinden sorumludur. Yüklenen tüm sınıflar, sınıf yükleyicisinin içerisindeki sınıf tablosunda tutulur.
e) Nesne Oluşturucu: Sınıf yükleyicisi bulunan sınıflardan nesne oluşturur. Nesnesi oluşturulacak nesne sınıf listesinde yoksa sınıf yükleyicisinin bunu yüklemesini sağlar.
f) Metot Çerçevesi: Her metoda ait bir metot çerçevesi vardır. Metodun her çağrımında yeni bir tane çerçeve oluşturulur ve QVM nin metot çerçeveleri yığınının en üstüne konulur. Metot çerçevesi komutları çalıştırmada kullanılan bir yığın ve metot parametreleri ile yerel değişkenleri tutan yerel değişken dizisinden oluşur. Bu dizinin ilk elemanı her zaman metodun içinde tanımlandığı sınıfa ait nesneyi gösterir. Eğer metot sınıf metoduysa bu değer null olur.

g) Sabit Dizisi: Sınıf içerisinde tanımlanan her sabit bilgi (paket adı, sınıf adı, arayüz adları, süper sınıf adı, kullanılan diğer sınıf adları, alan adları, tipleri, metot adları, parametreleri, geri dönüş değerleri, kodları, vb.) bu listenin içerisine byte dizisi olarak eklenmiştir. Yazılan QDil kodları aradile çevrilirken bu sabit dizisindeki indis değerlerini parametre olarak kullanırlar.

4.7.1. QDil sanal makinesi komut kümesi

QDil sanal makinesi sınıf içerisinde bulunan sanal makine komutlarını çalıştırır. Sanal makinenin komut kümesi Çizelge 4.12 de gösterilmiştir.

Çizelge 4.12. QDil sanal makinesi komut kümesi

Komut kodu	Komut	Açıklama
1	Load <local_index>	Metot çerçevesindeki yerel değişken dizisinden verilen indisteki referansı yığına koyar. Komut 3 byte dır. Load 1 byte , local_index 2 byte dır.
2	Load_0	Metot çerçevesindeki yerel değişken dizisinden 0 numaralı referansı yığına koyar. Komut 1 byte dır.
3	Load_1	Metot çerçevesindeki yerel değişken dizisinden 1 numaralı referansı yığına koyar. Komut 1 byte dır.
4	Load_2	Metot çerçevesindeki yerel değişken dizisinden 2 numaralı referansı yığına koyar. Komut 1 byte dır.
5	Load_3	Metot çerçevesindeki yerel değişken dizisinden 3 numaralı referansı yığına koyar. Komut 1 byte dır.
6	Puss_null	Metot çerçevesindeki işlem yığını üzerine null referansı koyar. Komut 1 byte dır.
7	Store <local_index>	Metot çerçevesindeki işlem yığını üzerindeki referansı yerel değişken dizisinden verilen indis numaralı yere koyar. Komut 3 byte yer kaplar. Store 1 byte, local_index 2 byte dır.

Çizelge 4.12.'in devamı

8	Store_0	Metot çerçevesindeki işlem yığını üzerindeki referansı yerel değişken dizisinin 0 indis numaralı yere koyar. Komut 1 byte dır.
9	Store_1	Metot çerçevesindeki işlem yığını üzerindeki referansı yerel değişken dizisinin 1 indis numaralı yere koyar. Komut 1 byte dır.
10	Store_2	Metot çerçevesindeki işlem yığını üzerindeki referansı yerel değişken dizisinin 2 indis numaralı yere koyar. Komut 1 byte dır.
11	Store_3	Metot çerçevesindeki işlem yığını üzerindeki referansı yerel değişken dizisinin 3 indis numaralı yere koyar. Komut 1 byte dır.
12	NewArray <class_index> <dim_index>	Kullanılan sınıflar dizisinde verilen tip indis numaralı sınıftan bir dizi oluşturur. Dizi çok boyutlu olabilir. Dizi sayısal tiplerden oluşuyorsa dizinin elemanları 0 değeri, diğer tiplerden oluşuyorsa null değerine sahiptir. Dizinin boyut sayısını boyut_indis değeri gösterir. Boyutlardaki dizi eleman sayıları işlem yığını üzerinden alınır. Komut 3 byte yer kaplar. NewArray 1 byte, class_index 1 byte, dim_index 1 byte dır. Örnek: int[][] a = new int[3][4](); <sınıf indis> : değeri int tipini gösterir. <boyut_indis>: dizi iki boyutlu olduğu için 2 değerini içerir. İşlem yığını üzerinde sırasıyla 3 ve 4 sayıları vardır.
13	GetArrayDimLength <dim_index>	İşlem yığını üzerindeki dizi referansının gösterdiği dizinin verilen indis değerindeki boyutunda kaç eleman olduğu işlem yığını üzerine konulur. Komut 2 byte dır.
14	ReturnValue	Bir metodun geriye dönüşünü bir önceki çağıran metoda yapar. Metodun işlem yığını üzerindeki referansı alır ve geri dönülecek olan metodun işlem yığını üzerine koyar. Komut 1 byte dır.

Çizelge 4.12.'in devamı

15	Return	Bir metodun geriye dönüşünü bir önceki çağıran metoda yapar. Çağıran metoda herhangi bir değer çevirmez. Komut 1 byte dır.
16	Raise	İşlem yığını üzerindeki Exception sınıfından bir hata nesnenin fırlatılmasını sağlar. Mevcut metot içerisinde bu exception ile ilgilinecek komut aranır. Varsa bu komut satırına geçilir, işlem yığını temizlenir, fırlatılan Exception nesnesi işlem yığını üzerine konulur. Yoksa, bir önceki metoda geçiş yapılır ve arama bu metod içerisinde devam eder. Komut 1 byte dır.
17	ArrayLoad	İşlem yığınından dizi referansı, boyut sayısı ve elemanın indis değerlerini alır; bu elemanın referansını işlem yığını üzerine koyar.
18	ArrayStore	İşlem yığınından dizi referansı, boyut sayısı, elemanın indis değerlerini alır ve değiştirilecek eleman referansını alır ve dizinin belirtilen indisteki elemanını değiştirir.
19	PushConstValue <const_index>	Sınıfın sabit listesi içerisinden verilen indisteki sabitten bir nesne oluşturarak referansını metot yığını üzerine koyar. Komut 5 bytedır. PushConstValue 1 byte, sabit_indis 4 byte
20	PushClassRef <class_index>	Kullanılan sınıflar dizisinde verilen indis değerli sınıfın referansını işlem yığını üzerine koyar. Komut 2 byte dır.
21	Isa <const_index>	İşlem yığını üzerinde bulunan referans ile sabit listesindeki sabit indekste bulunan sınıfın aynı olup olmadığını test eder. Aynı ise işlem yığını üzerine 1, değilse 0 sayısını yazar. Komut 5 byte dır. Isa 1 byte, const_index 4 byte dır.
22	NumberConvert <first_type> <second_type>	İşlem yığını üzerinde bulunan <first_type> tipindeki sayıyı <second_type> a çevirir ve yeni sayı referansını işlem yığını üzerine koyar. Komut 3 byte dır.

Çizelge 4.12.'in devamı

23	NumberCompare	İşlem yığını üzerinde bulunan iki referansın gösterdiği sayıları karşılaştırır. Birinci sayı ikinciden büyükse işlem yığınına 1, eşitse 0, küçükse -1 yazılır. Komut 1 byte dır.
24	DublicateInsert	İşlem yığını üzerindeki değerin kopyasını alır ve işlem yığını üzerine koyar. Komut 1 byte dır.
25	DublicateInsert2	İşlem yığını üzerindeki değerin kopyasını alır ve işlem yığını üzerindeki iki elemanın altına koyar. Komut 1 byte dır.
26	DublicateInsert3	İşlem yığını üzerindeki değerin kopyasını alır ve işlem yığını üzerindeki üç elemanın altına koyar. Komut 1 byte dır.
27	Dublicate2Insert	İşlem yığını üzerindeki ilk iki değerin kopyasını alır ve bu elemanların altına ekler.
28	Goto <branch_offset>	Çalıştırılan metot alanında belirtilen bytedaki komuta gidilir. Branch_offset metodun başlangıç adresinden kaç byte sonrasına atlanacağını gösterir. Komut 5 byte dır. Goto 1 byte, branch_offset 4 byte dır.
29	GetField <field_index>	Mevcut nesnenin indeks ile belirtilen alanının değeri alınır ve işlem yığını üzerine konulur. Bu işlemden önce alana ait get metodu çalıştırılır. Komut 2 bytedır.
30	GetRefField <field_index>	İşlem yığını üzerinde bulunan nesnenin indeks ile belirtilen alanının değeri alınır ve işlem yığını üzerine konulur. Bu işlemden önce alana ait get metodu çalıştırılır. Komut 2 bytedır.
31	GetClassField <field_index>	Mevcut nesnenin sınıfındaki indeks ile belirtilen sınıf alanı değeri alınır ve işlem yığını üzerine konulur. Komut 2 bytedır.
32	GetClassRefField <field_index>	İşlem yığını üzerinde bulunan nesnenin ait olduğu sınıfın indeks ile belirtilen alanının değeri alınır ve işlem yığını üzerine konulur.

Çizelge 4.12.'in devamı

33	CallOperator <method_index >	İşlem yığını üzerinde bulunan nesnenin method_index değeri ile verilen işleci çağrılır. İşlecin kullandığı diğer nesnelerin referansı işlem yığınındadır. Komut 2 bytedır. CallOperator 1 byte, op_index 1 byte dır.
34	CallMethod <method_index>	Mevcut sınıf ya da nesne içindeki method_index ile verilen metodu çağırır. Yoksa üst sınıflarında arama yapılır. Varsa metodun parametreleri işlem yığını üzerindedir. Geri dönüş değeri varsa işlem yığını üzerine konulur. Komut 2 byte dır.
35	CallRefMethod <method_index>	İşlem yığını üzerinde referansı bulunan nesnenin method_index ile gösterilen metodu çağrılır. Metot bu nesne içerinde yoksa bir üst sınıf içerisinde aranır. Varsa metodun parametreleri işlem yığını üzerindedir. Geri dönüş değeri varsa işlem yığını üzerine konulur. Komut iki byte dır.
36	CallSpecialMethod <method_index>	İşlem yığını üzerinde referansı bulunan nesnenin method_index ile gösterilen özel metodu çalıştırılır. Bu metotlar **@class_init** ve **@object_init** metotlarıdır. Sınıfın ve nesnelerin ilklenmesi için kullanılan metotlardır. Derleyici tarafından oluşturulurlar. Komutun ihtiyaç duyduğu parametreler işlem yığını üzerinde olmalıdır. Komut 5 byte dır. CallSpecialMethod 1 byte, method_index 4 bytedır.
37	CallClassMethod <method_index>	Mevcut sınıfın ya da nesnenin ait olduğu sınıfın belirtilen metod indisli sınıf metodunu çağırır. Metodun ihtiyaç duyduğu parametreleri işlem yığını üzerindedir. Geriye dönüş değeri varsa işlem yığını üzerine koyar. Komut 2 bytedır.
38	CallClassRefMethod <method_index>	İşlem yığını üzerinde bulunan sınıfın ya da nesnenin ait olduğu sınıfın method indisi ile belirtilen sınıf metodu çağrılır. Metot yoksa üst sınıflar araştırılır. Metodun ihtiyaç duyduğu parametreler işlem yığını üzerindedir. Geriye dönüş değeri varsa işlem yığını üzerine konulur.

Çizelge 4.12.'in devamı

39	CallExtMethod <const_index>	İşlem yığını üzerindeki nesneye ait sabit listesinde const_index de belirtilen harici bir metodun çağrılmasını sağlar. Metodun geri dönüş değeri işlem yığını üzerine konulur. Komut 5 byte dır. CallExtMethod 1 byte, const_index 4 byte dır.
40	SetField <const_index>	İşlem yığını üzerinde referansı bulunana nesneye ait sabit listesinde const_index ile belirtilen alanının değeri işlem yığınındaki ikinci değişken yapılır. Komut 5 byte dır. SetField 1 byte, const_index 4 byte dır.
41	SetRefField <field_index>	İşlem yığını üzerinde bulunan değerin, işlem yığınında değerin hemen altındaki referansı gösterilen nesnenin belirtilen indisteki değerine ataması gerçekleştirilir. Komut iki byte dır.
42	SetClassField <field_index>	İşlem yığını üzerinde bulunan nesnenin metodun çalışan sınıf içerisindeki belirtilen indis değerli alana atanması sağlanır. Komut iki byte dır.
43	SetClassRefField <field_index>	İşlem yığını üzerinde bulunan değerin işlem yığınında değerin hemen altındaki referansın gösterdiği sınıfın belirtilen indisteki sınıf alanına ataması yapılır. Komut iki bytedır.
44	Jsr <branch_offset>	Kendinden hemen sonra gelen komutun adresini işlem yığını üzerine koyar ve çalışmanın branch_offset adresinden devam etmesini sağlar. Komut 5 byte dır. Jsr 1 byte, branch_offset 4 byte dır.
45	New <class_index>	Kullanılan sınıflar listesinde belirtilen class_index de bulunan sınıftan bir nesne oluşturur. Bu nesnenin referansını işlem yığını üzerine koyar. Yeni nesne oluşturmada ihtiyaç duyduğu parametreler işlem yığınında olmalıdır. Komut 5 byte dır. New 1 byte, const_index 4 byte dır.

Çizelge 4.12.'in devamı

46	Pop	İşlem yığını üzerinden bir eleman çıkarırı. Komut 1 byte dır.
47	Pop2	İşlem yığını üzerindne 2 eleman çıkarırı.
48	Ret <local_index>	Yerel değişkenler listesinde local_index ile verilen adrese geri dönülür. Programın çalışması buradan devam eder. Komut 1 byte yer kaplar. Ret 1 byte, local_index 1 byte dır.
49	IfCmpEq <branch_offset>	İşlem yığını üzerindeki iki referansın değerleri birbirlerine eşit ise branch_offset e atlanır. Çalışma atlanan yerden devam eder. Komut 5 byte dır. IfCmpEq 1 byte, brach_offset 4 byte dır.
50	IfCmpNeq <branch_offset>	İşlem yığını üzerindeki iki referansın değerleri birbirlerine eşit değilse branch_offset e atlanır. Çalışma atlanan yerden devam eder. Komut 5 byte dır. IfCmpNeq 1 byte, brach_offset 4 byte dır.
51	IfNumCmpEq <branch_offset>	İşlem yığını üzerindeki iki referansın gösterdiği sayıların eşit olması durumunda brach_offset e atlanır. Çalışma atlanan yerden devam eder. Komut 5 byte dır. IfNumCmpEq 1 byte, branch_offset 4 byte dır.
52	IfNumCmpNeq	İşlem yığını üzerindeki iki referansın gösterdiği sayıların eşit olmaması durumunda brach_offset e atlanır. Çalışma atlanan yerden devam eder. Komut 5 byte dır. IfNumCmpEq 1 byte, branch_offset 4 byte dır.
53	IfNumCmpLt <branch_offset>	İşlem yığını üzerindeki iki referansın gösterdiği sayılardan birincisinin ikincisinden küçük olması durumunda brach_offset e atlanır. Çalışma atlanan yerden devam eder. Komut 5 byte dır. IfNumCmpLt 1 byte, branch_offset 4 byte dır.

Çizelge 4.12.'in devamı

54	IfNumCmpLe <branch_offset>	İşlem yığını üzerindeki iki referansın gösterdiği sayılardan birincisinin ikincisinden küçük ya da eşit olması durumunda brach_offset e atlanır. Çalışma atlanan yerden devam eder. Komut 5 byte dır. IfNumCmpEqLe 1 byte, branch_offset 4 byte dır.
55	IfNumCmpGt <branch_offset>	İşlem yığını üzerindeki iki referansın gösterdiği sayılardan birincisinin ikincisinden büyük olması durumunda brach_offset e atlanır. Çalışma atlanan yerden devam eder. Komut 5 byte dır. IfNumCmpGt 1 byte, branch_offset 4 byte dır.
56	IfNumCmpGe <branch_offset>	İşlem yığını üzerindeki iki referansın gösterdiği sayılardan birincisinin ikincisinden büyük ve ya eşit olması durumunda brach_offset e atlanır. Çalışma atlanan yerden devam eder. Komut 5 byte dır. IfNumCmpGe 1 byte, branch_offset 4 byte dır.
57	IfStackValEqZero <branch_offset>	İşlem yığını üzerinde bulunan sayı sıfır ise belirtilen offset e gidilir. Komut 5 byte dır. IfStackValEqZero 1 byte, branch_offset 4 byte dır.
58	IfStackValNeqZero <branch_offset>	İşlem yığını üzerinde bulunan sayı sıfır değilse belirtilen offset e gidilir. Komut 5 byte dır. IfStackValEqZero 1 byte, branch_offset 4 byte dır.
59	IfStackValLtZero <branch_offset>	İşlem yığını üzerinde bulunan sayı sıfırdan küçük ise belirtilen offset e gidilir. Komut 5 byte dır. IfStackValEqZero 1 byte, branch_offset 4 byte dır.
60	IfStackValLeZero <branch_offset>	İşlem yığını üzerinde bulunan sayı sıfır ya da ise belirtilen offset e gidilir. Komut 5 byte dır. IfStackValEqZero 1 byte, branch_offset 4 byte dır.

Çizelge 4.12.'in devamı

61	IfStackValGtZero <branch_offset>	İşlem yığını üzerinde bulunan sayı sıfırdan büyük ise belirtilen offset e gidilir. Komut 5 byte dır. IfStackValEqZero 1 byte, branch_offset 4 byte dır.
62	IfStackValGeZero <branch_offset>	İşlem yığını üzerinde bulunan sayı sıfırdan büyük eşit ise belirtilen offset e gidilir. Komut 5 byte dır. IfStackValEqZero 1 byte, branch_offset 4 byte dır.
63	IfNotNull <branch_offset>	İşlem yığını üzerinde bulunan referans null değilse belirtilen offset e gidilir. Komut 5 byte dır. IfStackValEqZero 1 byte, branch_offset 4 byte dır.
64	QNewQubit	Yeni bir kübitin kuantum sisteminden alınmasını sağlar. Bu kübiti gösteren bir qbit nesnesi oluşturarak işlem yığını üzerine referansını koyar.
65	QFinalizeQubit	İşlem yığını üzerindeki kübiti kuantum sistemine geri verir ve nesneyi yok eder.
66	QNewQregister	İşlem yığını üzerindeki sayı kadar kübitten oluşan bir qregister nesnesi oluşturur. Bu nesnenin referansını işlem yığını üzerine koyar.
67	QFinalizeQregister	İşlem yığını üzerindeki qregisterın içindeki kübitleri kuantum sistemine geri verir ve nesneyi yok eder.
68	QSetQubit	İşlem yığını üzerindeki kübitin değerini yine işlem yığını üzerindeki sayı yapar.
69	QSetQregister	İşlem yığını üzerindeki qregisterın değerini işlem yığını üzerindeki sayı yapar.

Çizelge 4.12.'in devamı

70	QMeasure	İşlem yığını üzerindeki kübit ya da qregisterda hesaplamaa bazları kullanılarak ölçme yapılır. Sonuç binary nesne olarak işlem yığınının üzerine konulur.
71	QSwapQubits	İşlem yığını üzerindeki iki kübitin yerlerinin değiştirir.
72	QRotX	İşlem yığını üzerindeki kübite işlem yığını üzerindeki sayı kadar X döndürmesi gerçekleştirir.
73	QRotY	İşlem yığını üzerindeki kübite işlem yığını üzerindeki sayı kadar Y döndürmesi gerçekleştirir.
74	QRotZ	İşlem yığını üzerindeki kübite işlem yığını üzerindeki sayı kadar Z döndürmesi gerçekleştirir.
75	QX	İşlem yığını üzerindeki kübite X kapısını uygular.
76	QinvX	İşlem yığını üzerindeki kübite X kapısının tersini uygular.
77	QY	İşlem yığını üzerindeki kübite Y kapısını uygular.
78	QinvY	İşlem yığını üzerindeki kübite Y kapısının tersini uygular.
79	QZ	İşlem yığını üzerindeki kübite Z kapısını uygular.
80	QinvZ	İşlem yığını üzerindeki kübite Z kapısının tersini uygular.
81	QCNOT	İşlem yığını üzerindeki iki kübitten birinciyi kontrol kübiti olarak kullanır ve bu kübitin durumuna göre ikinci kübite NOT kapısı uygular.
82	QH	İşlem yığını üzerindeki kübite H kapısı uygular.

Çizelge 4.12.'in devamı

83	QToffoli <control_num>	İşlem yığını üzerindeki belirtilen sayıdaki kübiti kontrol kübiti olarak kullanır ve bu kübitlerin durumuna göre verilen son kübite NOT kapısı uygular.
84	QS	İşlem yığını üzerindeki kübite S kapısını uygular.
85	QinvS	İşlem yığını üzerindeki kübite S kapısının tersini uygular.
86	QT	İşlem yığını üzerindeki kübite T kapısını uygular.
87	QinvT	İşlem yığını üzerindeki kübite T kapısının tersini uygular.
88	QV	İşlem yığını üzerindeki kübite V kapısını uygular.
89	QinvV	İşlem yığını üzerindeki kübite V kapısının tersini uygular.
90	QRegEntangle	İşlem yığını üzerindeki qregister ın elemanlarını tam dolanık hale getirir.
91	QInterpretQoperator	İşlem yığın üzerinde bulunan qoperator un qoperator yorumlayıcı tarafından yorumlanmasını sağlar.
92	QRunQoperator	İşlem yığını üzerindeki yorumlanmış olan qoperatorun qregisterın kübitlerine uygulanmasını sağlar.
93	QApplyOracle	İlem yığınındaki Oracle ın qregisterın kübitlerine uygulanmasını sağlar

4.7.2. QDil sınıflarının fiziksel gerçekleştirim yapıları

QDil sanal makinesinin derlenerek oluşturulan sınıfları belleğe yüklemesi ve kodlarını çalıştırması gereklidir. QDil kullanıdığı sınıf yapıları aşağıdaki çizelgelerde açıklanmıştır.

Çizelge 4.13. QDil sınıflarının fiziksel gerçekleştirimlerinin (QdClass) içyapısı

	Tip	**Alan**	**Açıklama**
1	byte	version_maj	Sınıfların derlendiği derleyicinin üst sürüm numarası
2	byte	version_min	Sınıfların derlendiği derleyicinin alt sürüm numarası

Çizelge 4.13.'in devamı

3	byte	class_qvm_state	Sınıfın QVM içine yüklenmesinden sonraki durumlarının bilgisini tutan alan
4	String	package_name	Sınıfın ait olduğu paket bilgisini tutan alan
5	String	class_name	Sınıfın adını tutan alan
6	byte	class_type	CLASS, INTERFACE, CONSTCLASS, ABSTRACTCLASS değerlerinden birini tutan alandır
7	String	super_name	Sınıfın üst sınıfın adını tutar. Yoksa qdil.lang.Object sınıfı üst sınıftır.
8	byte	interface_count	Bu sınıfın kullandığı interface sayısını tutar.
9	int	field_count	Bu sınıfın içerisinde tanımlanan toplam alan sayısını tutar.
10	int	method_count	Bu sınınf içerisinde tanımlanan toplam metod sayısını tutar.
11	int	used_class_count	Bu sınıf içerisinde kullanılan tüm sınıfların, arayüzlerin sayısını tutar.
12	String[]	used_class_names	Bu sınıf içerisinde kullanılan tüm sınıfların, arayüzlerin adlarını paket adlarıyla birlikte tutar.
13	int[]	interface_indexes	Kullanılan arayüzlerin used_class_names dizisi içerisindeki indekslerini tutar.

Çizelge 4.14. QDil sınıflarının fiziksel gerçekleştirimlerindeki metotların (QdMethod) içyapısı

	Tip	**Alan**	**Açıklama**
1	QdClass	class_of_method	Metodun ait olduğu sınıfa bir referanstır. Sınıf belleğe yüklendiğinde değer atanır.
2	byte	class_index	Bu metodun ait olduğu sınıfın used_class_names içindeki indeks numarasını tutan alandır.
3	String	method_name	Metodun adını tutan alandır. String tipindedir.

Çizelge 4.14.'in devamı

4	byte	method_type	TYPE_DEF, TYPE_EXTDEF, TYPE_ORADEF, TYPE_OPERATOR değerlerinden birine sahiptir. Metodun tipini tutar.
5	byte	access_type	ACCESS_PLUS, ACCESS_MINUS, ACCESS_SHARP, ACCESS_PACKAGE değerlerinden birini tutar. Metodun diğer metotlar ve sınıflarca nasıl erişilebileceğini belirtir.
6	byte	method_modifier	CONST, ABSTRACT, CLASS_MTD, FLD_MTD değerlerinden birini tutar.Metodun nasıl bir metod olduğunu belirtir.
7	byte	raise_count	Bu metot tarafından çıkartılabilecek toplam Exception sayısını tutar.
8	byte[]	raises_indexes	Bu metodun çıkartabileceği Exceptionların adlarının used_class_names içindeki indeks numaralarıdır.
9	QdCatchTable	catch_table	Metod içindeki try-catch ifadelerinin bilgilerini tutan bir tablodur. Bu tablo kullanılarak QVM tarafından Exceptionlar değerlendirilir.
10	byte	param_number	Metodun aldığı toplam parametre sayısını tutar.

Çizelge 4.15. QDil sınıflarının fiziksel gerçekleştirimlerindeki alanların (QdField) içyapısı

	Tip	**Alan**	**Açıklama**
1	QdClass	class_of_field	Alanın ait olduğu sınıfa bir referanstır. Sınıf belleğe yüklendiğinde değer atanır.
2	byte	class_index	Bu alanın ait olduğu sınıfnıı used_class_names içindeki indeksini verir.
3	byte	field_type_index	Alanın tipinin used_class_names içindeki indeks numarasını verir.

Çizelge 4.15.'in devamı

4	byte	access_type_get	Değer okunmada diğer sınıflara uygulanacak olan erişim denetimi bilgisini tutar. ACCESS_PLUS, ACCESS_MINUS, ACCESS_SHARP, ACCESS_PACKAGE değerlerinden birine sahiptir.
5	byte	access_type_set	Bu alana değer akatarılması yapılırken diğer sınıflara uygulacak olan erişim denetimi bilgisini tutar. ACCESS_PLUS, ACCESS_MINUS, ACCESS_SHARP, ACCESS_PACKAGE değerlerinden birine sahiptir.
6	byte	kind_of_field	OBJECT ya da CLASS tiplerinden birini alır. Alanın nesneye ya da sınıfa ait olup olmadığını gösterir.
7	int	field_init_cons_index	Bu alanın ilklemesiyle ilgili kodların sınıf içerisindeki constant_list içerisinde nerede olduğunu gösteren indeks değerini tutar.
8	String	filed_name	Alanın adını tutan değişkendir.

Çizelge 4.16. QDil sınıflarının fiziksel gerçekleştirimlerindeki sabitlerin (QdConstant) içyapısı

	Tip	**Alan**	**Açıklama**
1	byte	constant_type	Sabitin tipini tutar. TYPE_BYTE, TYPE_SHORT, TYPE_INT, TYPE_LONG, TYPE_FLOAT, TYPE_DOUBLE, TYPE_COMPLEX, TYPE_BINARY, TYPE_CHAR, TYPE_STRING, TYPE_ARRAY, TYPE_CODE değerlerinden birisine sahiptir.
2	int	size	Sabitin toplam byte büyüklüğünü tutar.
3	byte[]	value	Sabitin değerini tutar.

Çizelge 4.17. QDil sınıflarının fiziksel gerçekleştirimlerindeki metotlardaki try-catch yapısını tutan (QdCatchTable) ın içyapısı

	Tip	Alan	Açıklama
1	int	catch_table_entry_count	Catch tablosunda kaç tane girdi olduğunu tutar.
2	QdCatchTableEntry[]	entries	Catch tablosundaki girdileri tutar.

Çizelge 4.18. QDil sınıflarının fiziksel gerçekleştirimlerindeki metotlardaki try-catch yapısını tutan (QdCatchTable) ın girdidlerini tutan (QdCatchTableEntry) içyapısı

	Tip	Alan	Açıklama
1	int	start_ins_number	try bloğunun başladığı komutun offset değeri
2	İnt	end_ins_number	try bloğunun bittiği komutun offset değeri
3	int	exp_type_index	Çıkartılacak olan Exceptionın used_class_names içindeki indis numarasıdır. Bu indisteki exception çıkartıldığında ilgili ofsetteki kod çalıştırılır.
4	int	jump_code_offset	Exception çıkartıldığında kodda hangi offsete atlanacağını tutar.

4.8. QDil İle Bazı Kuantum Algoritmalarının Gerçekleştirimi

QDil kullanılarak Deutsch ve Shor kuantum algoritmalarının gerçekleştirimi aşağıda sırasıyla verilmiştir.

```
package qdil.algorithms;

use qdil.lang.Object;
use qdil.lang.boolean;
use qdil.lang.Sys;
use qdil.lang.binary;
use qdil.lang.qregister;

class deutsch:Object {
    + def deutsch.UF(qregister x){      x.cnot(0,1);       }
    + def deutsch.main(string[] args){
      qregister x = new qregister(2);
      x.reset();
```

```
        x[0].set_zero();
          x[1].set_one();
         x.H();
         UF(x);
          x.H();
          int sonuc = x[0].measure().to_int();
          if (sonuc?=1){
            Sys.out.println("dengeli");
          }else{
            Sys.out.println("sabit");
          }
    }
}
```

```
package qdil.algorithms;

use qdil.lang.Object;
use qdil.lang.boolean;
use qdil.lang.Sys;
use qdil.lang.binary;
use qdil.lang.qregister;
use qdil.lang.int;

class shor:Object {
    + oradef shor.U(binary x,int a,int N):binary{
       return (a ^ x) % N;
    }
    + def shor.main(string[] args){
      qregister x = new qregister(4);
      qregister y = new qregister(4);
      x.reset();
      int i=3;
      x.QFT();
         N=16;
```

```
        y = U(x,i,N);
        x. QFT().inv();
        y.H();
        int sonuc = y.measure().to_int();
        int xsonuc;
        if (sonuc?=1){
          xsonuc = x.measure().to_int();
          Sys.out.println(x);
        }
    }
}
```

BÖLÜM 5
SONUÇ VE ÖNERİLER

Bu tezde hem klasik hem de kuantum bilgisayarlarda kullanılabilecek QDil(Quantum Dynamicallay Interpreted Language) adlı özgün bir kuantum programlama dili geliştirilmiştir. Kuantum bilgisayarları için literatürde geliştirilmiş olan dillerin çoğu mevcut klasik bir programlama diline kütüphane olarak geliştirilmiştir. Bununla birlikte ayrı bir kuantum programlama dili olarak geliştirilen birkaç dil de mevcuttur. Bu tez kapsamında geliştirilen QDil, çalışacağı klasik ve kuantum sistemlerden bağımsız olması, klasik programlama kısmının nesneye yönelik olması, kuantum operatörlerin dinamik olarak yorumlanması ve uygulanması, ikili kahin fonksiyonlarının terslenebilir kübit kapı ve devrelerine otomatik olarak çevrilmesi, qoperatörlerin bilinen matris biçimlerinin yanında Dirac gösterimi ile de yazılabilmesi bağlamında mevcut dillerden farklıdır.

Bu tezde hem klasik hem de kuantum bilgisayarlarda kullanılabilecek QDil(Quantum Dynamicallay Interpreted Language) adlı özgün bir kuantum programlama dili geliştirilmiştir.

Çizelge 5.1. Bazı kuantum programlama dilleri ve özellikleri

Programlama Dili	Artıları ve özellikleri	Eksileri
Psödo-code (Knill,1996)	Matematiksel ifade edilen kuantum programlamayı psödo-code şeklinde ilk kez ifade etti. Kuantum ve klasik registerlar üzerinde işlem yapar.	Gerçek bir programlama dili yoktur.
QCL (Ömer,2003)	İlk geliştirilen gerçek kuantum programlama dilidir. Emirsel bir dildir. C diline benzeyen yazım kurallar vardır. Klasik bilgisayarda simülasyon yapmaya imkan verir. Kaynak kodu verilmektedir.	Simülasyon kullanılacak kübit sayısını sınırlandırır. Klasik bilgisayarın programlanması ile ilgili birimler içermez. Yorumlayıcısı klasik bilgisayarda çalışmak için yazılan kütüphaneyi kullanır. Bellek yönetimini farklı veri tipleriyle yapmaktadır.

Çizelge 5.1.'in devamı

qlang (Betteli ve ark., 2003)	Kuantum operatörler sınıflar şeklinde tasarlanmıştır. C++ dilinde bir kütüphane olarak geliştirilmiştir.	C++ dilinin eksikliklerini barındırır. Yeni operatörler C++ dilinin class mekanizması ile geliştirilebilir. Kuantum hesaplamalarda bilgisayar cebir sistemlerinin kullanan paket gibidir.
LanQ (Mlnarik,2007)	C yazımına benzer bir yazımı vardır. Kuantum iletişim protokollerinin programlanması için geliştirilmiştir. Süreçleri oluşturmak için fork() komutunu kullanır. Simülasyon içerir.	Tüm kuantum operatörler channel şeklinde tanımlanır. Tipler klasik ve kuantumu ayırmada kullanılır. Kopyalama engellemede kullanılır. Basit aritmetik işlerde tek tek kuantum kapılarıyla yapılmalıdır.
QPL (Selinger,2004)	Fonksiyonel programlama paradigmasını kullanır. Statik tipli programlama dilidir. İlk fonksiyonel kuantum programlama dilidir. Derleme zamanında hataların tespitine izin verir. Haskell diline kütüphane şeklindedir.	Programlar akış diyagramları şeklinde karmaşık bir yapıda tanımlanır. Anlaşılması zordur.
cQPL (Mauerer,2005)	Ömer in geliştirdiği QCL arka tarafta kullanılmaktadır. QPL temellidir. Kuantum protokolleri geliştirmek için tasarlanmıştır. Haskell diline kütüphane şeklindedir.	Düşük seviyeli veri yapıları bulunmaktadır. En fazla 16 qubitten oluşan qint veri tipi bulunmaktadır. Arka tarafta C++ koduna çevirme işlemi gerçekleşmektedir. QCL simülasyonunu kullanır.

Çizelge 5.1.'in devamı

QML (Altenkirch ve Gratte,2005)	ML dili temelinde geliştirilen bir fonksiyonel programlama dilidir.	Matris biçiminde operatör tanımlanamaz. Yazım şekli zordur.
quipper (Green ve ark.,2013)	Haskell programlama dili kullanılarak geliştirilmiş fonksiyonel programlama dilidir. Devre diyagramı oluşturur ve parametreli devre diyagramlarına izin verir.	Kahin fonksiyonları sadece kuantum kapıları ile ifade edilmektedir. Klasik koddan kuantum kapılarına dönüşüm yoktur.
LIQUi|> (Wecker ve Svore,2014)	Microsoft F# dili kullanılarak geliştirilmiş fonksiyonel programlama dilidir. Kuantum hata ve hata düzeltme simülasyonu yapabilir.	Microsoft sistemlerine bağımlıdır.
QDil	Yeni ve genişletilebilir bir dildir. qoperator çalışma zamanında yorumlanıyor ve bu operatörlerin çalışma zamanında değişebilmesine imkan sağlıyor. Qoperator yorumlayıcısı kuantum bilgisayar implementasyonuna göre farklı yorumlamalar gerçekleştirebilir.	Simülatör yoktur. Java ile geliştirilen yorumlayıcıda en iyilemeler yapılmalı ve hızlanma sağlanmalıdır.

Çizelge 5.2. Bazı kuantum programlama dilleriyle QDil'in karşılaştırılması

	Psedö-kod	**QCL**	**qlang**	**LanQ**	**QPL**	**c-QPL**	**QML**	**quipper**	**LIQUi|>**	**QDil**
Referans	(Knill, 1996)	(Ömer, 2003)	(Betteli ve ark., 2003)	(Mlnarik, 2007)	(Selinger , 2004)	(Mauerer, 2005)	(Altenkirch ve Grette, 2005)	(Green ve ark, 2013)	(Wecker ve Svore, 2014)	
Yeni programlama dili yazılmış	-	+	-	+	+	-	-	-	-	+
Bir programlama diline kütüphanedir	-	-	C++	-	-	QCL	ML	Haskell	F#	-
Klasik bilgisayar programlama birimleri vardır	-	-	+	-	+	+	+	+	+	+
Oracle dinamik yorumlanıyor	-	-	-	-	-	-	-	+	+	+
Operatör dinamik yorumlanıyor.	-	-	-	-	-	-	-	+	+	+
Devre Diyagramı Çıktı Veriliyor	-	-	-	-	-	-	-	+	+	-
Oracle otomatik derleniyor	-	-	-	-	-	-	-	+	+	+
Kuantum İletişim Protokollerine Destek	-	-	-	+	+	+	-	-	+	+
Dirac notasyonuyla operatör tanımlama	-	-	-	-	-	-	-	-	-	+
Simülatör Var	-	+	-	-	+	+	-	+	+	-
Kuantum Bellek Yönetimi	-	+	-	-	-		-	+	+	+
Kuantum Donanım Bağımsızlığı	-	-	+	-	-	-	-	+	+	+

KAYNAKLAR

Altenkrich T., Grattate J., 2005. A Functional Quantum Programming Language. *In Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science.* IEEE Computer Society.

Ambainis A., Childs A., Reichardt B., 2007. Any And-Or Formula of Size n can be Evaluated in Time $n^{1/2+o(1)}$ On A Quantum Computer. *SIAM J. COMPUT.* 39(6):2513-2530.

Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., Sorensen D., 1999. *LAPACK Users' Guide.* Society for Industrial and Applied Mathematics. PA.

Baker G.D., 1996. Ogol, A System for Simulating Quantum Computations: Theory, Implementations and Insights. Master Thesis (Yüksek Lisans Tezi). Department of Computing. Macquire Universty. :62.

Benioff P., 1980. The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines. *Journal of Statistical Physics* 22:563-591.

Bennett C.H., Brassard G., 1984. Quantum Cryptography: Public Key Distribution and Coin Tossing. *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing* 1:175-179.

Bennett C.H. ve Wiesner S.J., 1992. Communication via One- and TwoPparticle Operators on Einstein-Podolsky-Rosen States. *Phys. Rev.Let.* 69:2881-2884.

Bettelli S., Serafini L. ve Calarco T., 2003. Toward An Architecture for Quantum Programming. *Eur. Phys. J.* D 25(2):181-200.

Childs A., van Dom W., 2010. Quantum Algorithms for Algebric Problems. *Rev. Mod. Phys.* 82(1):1-52.

Church A., 1936. An Unsolvable Problem of Elementary Number Theory. *American J. Mathematics* 58:345-363.

Cook S.A., Reckhow R.A., 1973. Time-Bounded Random Access Machines. *Proceedings Forth Annual ACM Symposium on Theory of Computing* 1:73-80.

Dautelle J., 2011. JScience, Java Tools and Libraries for the Advancement of Sciences . web:http://www.jscience.org.

Deutsch D., 1985. Quantum Theory, the Church-Turing Principle, and the Universal Quantum Computer. *Proceedings Royal Society* London A400:97-117.

Deutsch D., 1989. Quantum Computational Networks. *Proceedings of Royal Society London* A 425:73.

Deutsch D., Jozsa R., 1992. Rapid Solutions of Problems by Quantum Computation. *Proceedings Royal Society* London 439A:553-558.

Divincenzo D.P., 1995. Two-Bit Gates are Universal for Quantum Computation. *Phys.Rev. A* 51(2):1015-1022.

Einstein A., Podolsky B., Rosen N., 1935. Can Quantum-Mechanical Description of Physical Reality be Considered Complete?. *Phys. Rev.* 47,777.

Ekert A., 1991. Quantum Cryptography Based on Bell's Theorem. *Physical Review Letters* 67:661-663.

Farhi E., Goldstone J., Gutmann S., 2007. A Quantum Algorithm for the Hamiltonian nand Tree. *Theory of Computing* 4(1):169-190.

Fazel K., Thornton M., Rice J. E., 2007. ESOP-Based Toffoli Gate Cascade Generation. *In Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Proceeding* (PACRIM). 206-209.

Feynman R.P., 1982. Simulating Physics with Computers. *International Journal of Theorical Physics*:21:467-488.

Green A., Lumsdaine P.L., Ross N., Selinger P., Valiron B., 2013. Quipper: A Scalable Quantum Programming Language. *In Proceedings of PLDI'13.*

Griffiths D.J., 2004. *Introduction to Quantum Mechanics.* Pearson Prentice Hall.

Grover L.K., 1997. Quantum Mechanics Helps in Searching for a Needle in Haystack. *Physical Review Letters* 79: 325-328.

Hudson S.E., Flannery F., Ananian C.S., 1999. CUP, LALR Parser Generator For Java. web:http://www.cs.princeton.edu/~appel/modern/java/CUP.

Jouguet P., Kunz-Jacques S., Leverrier A., Grangier P., Diamanti E., 2013. Experimental Demonstration of Long-Distance Continuous-Variable Quantum Key Distribution. *Nature Photonics* 7:378-381.

Kaye P., Laflamme R., Mosca M. Çevirenler: Köymen K., Türkyılmaz İ., Şahin M., Yılmaz İ., 2011. *Kuantum Bilgisayarlarda Hesaplamaya Giriş.* Maltepe Yayınları No:45.

Kempe J., 2003. Quantum Random Walks: An Introductory Overview. *Contemp. Phys. 44(4)*:307-327.

Kitaev A.Yu., Shen A.H., Vyalyi M.N., 2002. Classical and Quantum Computation. *American Mathematical Society.* Graduate Studies in Mathematics vol.47.

Klein G., Rowe S., Decamps R., 1998. The Fast Lexical Anayser Generator. web : http://jflex.de.

Knill E., 1996. Conventions for Quantum Pseudocode. *LANL Tech.Rep.* LAUR-96-2724.

Koŝik J., 2003. Two Models of Quantum Random Walk. *Cent.Eur.J.Phys.* 4:556-573.

Mauerer W., 2005. Semantics and Simulation of Communication in Quantum Programming. Master Thesis (Yüksek Lisans Tezi). Universty of Erlangen-Nuremberg, Nuremberg.

Maymin P., 1996. Extending the Lambda Calculus to Express Randomized and Quantumized Algorithms. arXiv:quanth-ph/9612052.

Mishchenko A., Perkowski M., 2001. Fast Heuristic Minimization of Exclusive-Sums-of-Products. *Proc. Reed-Muller Workshop.* 242-250.

Mlnarik. H. , 2007. Operational Semantics and Type Soundness for Quantum Programming Language LanQ. PhD Dissertation (Doktora Tezi). Masaryk Universty.Czech Republic.

Moore G.E., 1965. Cramming More Components onto Integrated Circuits. *Electronics 38(8)*:114-117.

Mosca M., Smith J., 2011. Algorihms for Quantum Computers. *In Handbook of Natural Computing*, Springer Verlag, Berlin.

Möttönen M., Vartiainen J.J., Bergholm V. ve Salomaa M.M., 2004. Quantum Circuits for General Multiqubit Gates. *Phys. Rev. Lett.* 93 (13):130502.

Nielsen M.A. ve Chuang I.L., 2000. *Quantum Computation and Quantum Information*. Cambridge Universty Press.

Ömer B., 2003. Structured Quantum Programming. PhD Dissertation (Doktora Tezi). Institute for Theorical Physics. Viena Universty of Technology, Viena.

Rosenberg D., Peterson C. G., Harrington J.W., Rice P.R., Dallmann N., Tyagi K.T., McCabe K.P., Nam S., Baek B., Hadfield R.H., Hughes R.J, Nordholt J.E. 2008. Practical Long-Distance Quantum Key Distribution System Using Decoy Levels. *New J. Phys.* 11.

Sabry A., 2003. Modeling Quantum Computing in Haskell. *In Proceedings of the 2003 ACM SIGPLAN Workshop on Haskell.* Upsala, Sweden. ACM Press.

Sanaee Y., Dueck G.W., 2009. Generating Toffoli Networks From ESOP Expressions. *In Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing* (PACRIM). 715-719.

Savage J.E., 2008. *Models of Computation Exploring the Power of Computing.* Brown Universty.

Sebesta R.W., 2008. *Concepts of Programming Languages.* Addison Wesley.

Selinger P., 2004. Towards A Quantum Programming Language. *Math. Struct. Comput. Science* 14(4):527-586.

Shende V.V., Markov I.L. ve Bullock S.S., 2004. Minimal Universal Two-Qubit Controlled-NOT-Based Circuits. *Phys.Rev.A* (69):062321.

Shor P., 1994. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science* :124-134.

Shor P. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computers, *SIAM J. Computing* 26:1484-1509.

Simon D., 1994. On the Power of Quantum Computation. *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science* :116-123.

Toffoli T., 1981. Bicontinuous Extension of Reversible Combinatorial Functions. *Math. Syst. Theory.* 14:13-23.

Turing A., 1936. On Computable Numbers with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society* 42:230-265.

VanTonder A., 2004. A Lambda Calculus For Quantum Computation. *SIAM J. Comput.* 33 (5): 1109-1135.

Vartiainen J.J., Möttönen M. ve Salomaa M.M., 2004. Efficient Decomposition of Quantum Gates. *Phys.Rev.Lett.* 92:177902.

Wecker D., Svore K.M. 2014., LIQIi|>: A Software Design Architecture and Domain-Specific Language for Quantum Computing. arXiv: quant-ph /1402.4467v1.

EKLER

EK 1- QDil JCup Dosyası

```
package qdil.parser;
import java_cup.runtime.*;
import qdil.*;


parser code  {:

  public static void printError(String msg,int line,int col){
     System.out.println(line+". satir "+ col+". sutunda hata:"+msg);
  }
  public void report_error(String msg, Object sym) {
    StringBuffer m = new StringBuffer("HATA(ERROR): ");
                java_cup.runtime.Symbol s;
    if (sym instanceof java_cup.runtime.Symbol) {
      s = (java_cup.runtime.Symbol)sym;
      m.append(findSymbolText(s) );
      m.append(" ,satir:"+s.left);
      m.append(" : "+msg);
    }
     System.out.println(m);
  }
   private static String findSymbolText(Symbol sym){
       String symText = null;
       if (sym.value==null){
          symText="";
       }else{
          symText=" (\""+sym.value.toString()+"\")";
       }
       java.lang.reflect.Field [] flds = sym.class.getFields();
       for (int i = 0; i < flds.length ; i++){
          if (!java.lang.reflect.Modifier.isPublic(flds[i].getModifiers())) continue;
          try {
             if (flds[i].getInt(null) == sym.sym) return flds[i].getName()+symText;
          }catch (Exception ex) {
              return "";
          }
       }
       return sym.toString()+symText;
    }
  public void report_fatal_error(String msg, Object sym) {
    report_error(msg, sym);
    throw new RuntimeException("!...Yazım Hatası...");
  }
:};

scan with {: return getScanner().next_token(); :}

terminal BOOLEAN;
terminal BYTE, SHORT, INT, LONG, CHAR;
terminal FLOAT, DOUBLE;
terminal LBRACK, RBRACK;
terminal DOT;
terminal SEMICOLON, MULT, COMMA, LBRACE, RBRACE, EQ, LPAREN, RPAREN, COLON;
terminal PACKAGE;
terminal USE;
terminal ABSTRACT, CONST;
terminal OP_INDEX;
terminal QDILC;
terminal CLASS;
terminal COMPLEX;
terminal EACH;
terminal GET,SET;
terminal METHOD;
terminal SELF, SUPER;
terminal STRING;
terminal INTERFACE;
terminal IF, ELSE;
terminal CASE, OTHERS, WHEN;
terminal DO, UNTIL, WHILE;
terminal IN;
```

```
terminal ISA;
terminal FOR;
terminal BREAK;
terminal BREAKALL;
terminal CONTINUE;
terminal BINARY;
terminal RETURN;
terminal RAISE,RAISES,TRY,FINALLY,CATCH;
terminal NEW;
terminal QBIT, QREGISTER, QOPERATOR;
terminal PLUS, MINUS, SHARP,COMP, NOT, DIV, MOD;
terminal LSHIFT, RSHIFT, URSHIFT;
terminal LT, GT, LTEQ, GTEQ;
terminal ISEQ, NOTEQ;
terminal AND,OR,BIGAND,BITAND,BITXOR,BIGOR,BITOR;
terminal POWER;
terminal OP_PLUS,OP_MINUS,OP_DIV,OP_MULT,OP_MOD,OP_BITAND,OP_BITOR,OP_POWER;
terminal OP_RSHIFT,OP_LSHIFT,OP_URSHIFT,OP_NOT,OP_IN,OP_BITCOMP,OP_BITXOR;
terminal OP_AND, OP_OR, OP_ISEQ,OP_ISNEQ;
terminal DEF;
terminal EXTDEF;
terminal ORADEF;

terminal java.lang.Number INT_LIT;
terminal java.lang.Number FL_LIT;
terminal java.lang.Number DBL_LIT;
terminal java.lang.Boolean BOOL_LIT;
terminal java.lang.Character CHR_LIT;
terminal java.lang.String STR_LIT;
terminal java.lang.String ID_NAME;
terminal NULL_LIT;


non terminal Qd_Com_Unit goal;

non terminal Qd_Lit lit;
non terminal Qd_Type var_type;
non terminal Qd_Cl_Type cl_type;
non terminal Qd_Ary_Type ary_type;
non terminal Qd_Id_Name id_name;
non terminal Qd_Id_Name short_id_name;
non terminal Qd_Id_Name long_id_name;
non terminal Qd_Com_Unit compile_unit;
non terminal Qd_Use use_dec_o;
non terminal Qd_Use use_decs;
non terminal Qd_Use use_dec;
non terminal Qd_Acc_Type access_type;
non terminal Qd_Acc_Type access_types_opt;
non terminal Qd_Acc_Type access_types;
non terminal Qd_Pac pac_dec;
non terminal Qd_Cl_Or_Int_Dec cl_or_int_dec;
non terminal Qd_Cl_Dec cl_dec;
non terminal Qd_Cl_Type sup_and_int_opt;
non terminal Qd_Cl_Type sup_and_ints;
non terminal Qd_Cl_Type sup_and_ints_list;
non terminal Qd_Cl_Bdy cl_bdy;
non terminal Qd_Cl_Mem_Dec cl_mem_decs;
non terminal Qd_Cl_Mem_Dec cl_mem_dec;
non terminal Qd_Fld_Dec fld_dec;
non terminal Qd_Fld_Bdy fld_bdy;
non terminal Qd_Get_Set_Dec get_dec;
non terminal Qd_Get_Set_Dec set_dec;
non terminal Qd_Var_Dec var_decs;
non terminal Qd_Var_Dec var_dec;
non terminal Qd_Var_Dec_ID var_dec_id;
non terminal Qd_Expr_Var_Init var_init;
non terminal Qd_Mtd_Dec mtd_dec;
non terminal Qd_Mtd_Head mtd_head;
non terminal Qd_Mtd_Name mtd_name;
non terminal Qd_Formal_Param formal_param_lst_opt;
non terminal Qd_Formal_Param formal_param_lst;
non terminal Qd_Formal_Param formal_param;
non terminal Qd_Mtd_Back mtd_back;
non terminal Qd_Type mtd_ret_type;
non terminal Qd_Stmt_Raises raise;
non terminal Qd_Cl_Type cl_type_lst;
non terminal Qd_Stmt_Mtd_Bdy mtd_bdy;
non terminal Qd_Id_Name opr;
```

```
non terminal Qd_Stmt_Cons_Bdy cons_bdy;
non terminal Qd_Stmt_Call_Cons call_cons;
non terminal Qd_Int_Dec int_dec;
non terminal Qd_Int_Bdy int_bdy;
non terminal Qd_Int_Mem_Dec int_mem_decs_opt;
non terminal Qd_Int_Mem_Dec int_mem_decs;
non terminal Qd_Int_Mem_Dec int_mem_dec;
non terminal Qd_Int_Fld_Dec int_fld_dec;
non terminal Qd_Expr_Ary_Init ary_init;
non terminal Qd_Expr_Var_Init var_inits;
non terminal Qd_Blk blk;
non terminal Qd_Stmt_Blk blk_stmts_opt;
non terminal Qd_Stmt_Blk blk_stmts;
non terminal Qd_Stmt_Blk blk_stmt;
non terminal Qd_Stmt_Loc_Var_Dec loc_var_dec;
non terminal Qd_Stmt stmt;
non terminal Qd_Stmt stmt_no_sub_stmt;
non terminal Qd_Stmt_No no_stmt;
non terminal Qd_Stmt_With_Lbl stmt_with_lbl;
non terminal Qd_Stmt_Expr_Stmt expr_stmt;
non terminal Qd_Stmt_Expr stmt_expr;
non terminal Qd_Stmt_If if_stmt;
non terminal Qd_Stmt_If if_else_stmt;
non terminal Qd_Stmt_Case case_stmt;
non terminal Qd_Case_Blk case_blk;
non terminal Qd_Stmt_When_Blk_Stmt_Grp when_blk_stmt_groups;
non terminal Qd_Stmt_When_Blk_Stmt_Grp when_blk_stmt_group;
non terminal Qd_Expr when_cnt_expr;
non terminal Qd_Expr_In in_expr;
non terminal Qd_Stmt_Others_Blk others_blk;
non terminal Qd_Stmt_While while_stmt;
non terminal Qd_Stmt_Until until_stmt;
non terminal Qd_Stmt_Each each_stmt;
non terminal Qd_Stmt_For for_stmt;
non terminal Qd_Expr for_init_opt;
non terminal Qd_Expr for_last_opt;
non terminal Qd_Expr for_step_opt;
non terminal Qd_Id_Name id_opt;
non terminal Qd_Stmt_Break break_stmt;
non terminal Qd_Stmt_Break breakall_stmt;
non terminal Qd_Stmt_Continue continue_stmt;
non terminal Qd_Stmt_Ret ret_stmt;
non terminal Qd_Stmt_Raise raise_stmt;
non terminal Qd_Stmt_Try try_stmt;
non terminal Qd_Try_Catch catches_opt;
non terminal Qd_Try_Catch catches;
non terminal Qd_Try_Catch catch;
non terminal Qd_Stmt_Finally finally;
non terminal Qd_Expr first_expr;
non terminal Qd_Expr first_expr_no_ary;
non terminal Qd_New_Expr new_expr;
non terminal Qd_Expr_Arg arg_lst_opt;
non terminal Qd_Expr_Arg arg_list;
non terminal Qd_Expr_New_Ary new_ary_no_init;
non terminal Qd_Expr_New_Ary new_ary_init;
non terminal Qd_Expr_D d_exprs;
non terminal Qd_Expr_D d_expr;
non terminal Integer dms_opt;
non terminal Integer dms;
non terminal Qd_Expr_Fld_Var fld_var;
non terminal Qd_Expr_Mtd_Call mtd_call;
non terminal Qd_Expr_Ary_Elmnt ary_elmnt;
non terminal Qd_Expr post_expr;
non terminal Qd_Expr sgn_expr;
non terminal Qd_Expr un_exprs;
non terminal Qd_Expr_Type_Ch type_ch_expr;
non terminal Qd_Expr pow_expr;
non terminal Qd_Expr mul_div_mod_expr;
non terminal Qd_Expr add_sub_expr;
non terminal Qd_Expr shft_expr;
non terminal Qd_Expr rel_expr;
non terminal Qd_Expr eq_or_noteq_expr;
non terminal Qd_Expr bitand_expr;
non terminal Qd_Expr xor_expr;
non terminal Qd_Expr bitor_expr;
non terminal Qd_Expr and_expr;
non terminal Qd_Expr or_expr;
non terminal Qd_Expr asgn_expr;
```

```
non terminal Qd_Expr asgn;
non terminal Qd_Expr l_expr;
non terminal Qd_Expr expr_opt;
non terminal Qd_Expr expr;
non terminal Qd_Expr cnst_expr;

//--------------------------------------------------------------------------
// PARSER KODU
//--------------------------------------------------------------------------
start with goal;

//Qd_Com_Unit
goal ::=
        compile_unit:cu
        {:
                cu.setUsedClasses(Qd_Cen_Cntrl.getUsedClasses());
                cu.setConstantList(Qd_Cen_Cntrl.getConstantList());
           RESULT=cu;
        :}
        ;
//Qd_Lit
lit ::=
    INT_LIT:il
                {:
                        Qd_Int_Lit re = new Qd_Int_Lit(illeft,ilright,il);
                  Qd_Cen_Cntrl.addConstant(QdConstant.getConstant(re));
                  RESULT= re;
                :}
        |       FL_LIT:fl
    {:
       Qd_Fl_Lit re = new Qd_Fl_Lit(flleft,flright,fl);
       Qd_Cen_Cntrl.addConstant(QdConstant.getConstant(re));
       RESULT=re;
                :}
        |       DBL_LIT:fl
    {:
            Qd_Dbl_Lit re = new Qd_Dbl_Lit(flleft,flright,fl);
      Qd_Cen_Cntrl.addConstant(QdConstant.getConstant(re));
      RESULT=re;
                :}
        |       BOOL_LIT:bl
                {:
                        Qd_Bool_Lit re = new Qd_Bool_Lit(blleft,blright,bl);
      Qd_Cen_Cntrl.addConstant(QdConstant.getConstant(re));
                        RESULT= re;
                :}
        |       CHR_LIT:cl
                {:
                        Qd_Char_Lit re = new Qd_Char_Lit(clleft,clright,cl);
      Qd_Cen_Cntrl.addConstant(QdConstant.getConstant(re));
                        RESULT = re;
                :}
        |       STR_LIT:sl
                {:
                        Qd_Str_Lit re = new Qd_Str_Lit(slleft,slright,sl);
      Qd_Cen_Cntrl.addConstant(QdConstant.getConstant(re));
                        RESULT = re;
                :}
        |       NULL_LIT:nl
                {:
                        Qd_Null_Lit re = new Qd_Null_Lit(nlleft,nlright,nl);
      Qd_Cen_Cntrl.addConstant(QdConstant.getConstant(re));
                        RESULT = re;
                :}
  ;

//Qd_Type
var_type ::=
                cl_type:c
                {:
                        RESULT = c;
                :}
        |       ary_type:a
                {:
                        RESULT =a;
                :}
        ;
```

```
//Qd_Cl_Type:Qd_Type
cl_type ::=
                    id_name:n
                    {:
                              RESULT = new Qd_Cl_Type(nleft,nright,Qd_Type.CLASS_OR_INTERFACE,n,null);
                    :}
          ;

//Qd_Ary_Type:Qd_Type
ary_type ::=
          id_name:n dms:d
                    {:
                              Qd_Cl_Type name_to_coi = new
Qd_Cl_Type(nleft,nright,Qd_Type.CLASS_OR_INTERFACE,n,null);
                              RESULT = new Qd_Ary_Type(nleft,nright,Qd_Type.ARRAY,name_to_coi,d.intValue());
                    :}
          ;

//Qd_Id_Name
id_name    ::=
                    short_id_name:m
                    {:
                              RESULT = m;
                    :}
          |         long_id_name:m
                    {:
                              RESULT = m;
                    :}
          |  opr:m
             {:
                RESULT = m;
             :}
          ;

//Qd_Id_Name
short_id_name ::=
                    ID_NAME:id
                    {:
                              RESULT = new Qd_Id_Name(idleft,idright,id,null,Qd_Id_Name.IDNAME);
                    :}
          |  METHOD:m
                    {:
                       Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"method",null,Qd_Id_Name.METHOD);
                       RESULT = id;
                    :}
          |         BYTE:m
                    {:
                              Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"byte",null,Qd_Id_Name.BYTE);
                       RESULT = id;
                    :}
          |         SHORT:m
                    {:
                              Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"short",null,Qd_Id_Name.SHORT);
                       RESULT = id;
                    :}
          |         INT:m
                    {:
                              Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"int",null,Qd_Id_Name.INT);
                       RESULT = id;
                    :}
          |         LONG:m
                    {:
                              Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"long",null,Qd_Id_Name.LONG);
                       RESULT = id;
                    :}
          |         CHAR:m
                    {:
                              Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"char",null,Qd_Id_Name.CHAR);
                       RESULT = id;
                    :}
          |         STRING:m
                    {:
                              Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"string",null,Qd_Id_Name.STRING);
                       RESULT = id;
                    :}
          |         FLOAT:m
                    {:
                              Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"float",null,Qd_Id_Name.FLOAT);
```

```
                        RESULT = id;
                    :}
            |       DOUBLE:m
                    {:
                            Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"double",null,Qd_Id_Name.DOUBLE);
                        RESULT = id;
                    :}
    |  COMPLEX:m
                    {:
                            Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"complex",null,Qd_Id_Name.COMPLEX);
                        RESULT = id;
                    :}
    |       BINARY:m
                    {:
                            Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"binary",null,Qd_Id_Name.BINARY);
                        RESULT = id;
                    :}
            |       BOOLEAN:m
                    {:
                            Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"boolean",null,Qd_Id_Name.BOOLEAN);
                        RESULT = id;
                    :}
            |       QBIT:m
                    {:
                            Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"qbit",null,Qd_Id_Name.QBIT);
                        RESULT = id;
                    :}
            |       QREGISTER:m
                    {:
                            Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"qregister",null,Qd_Id_Name.QREGISTER);
                        RESULT = id;
                    :}
            |       QOPERATOR:m
                    {:
                            Qd_Id_Name id= new Qd_Id_Name(mleft,mright,"qoperator",null,Qd_Id_Name.QOPERATOR);
                        RESULT = id;
                    :}
            ;
//Qd_Id_Name
long_id_name ::=
                    id_name:nm DOT short_id_name:id
                    {:
                            //Qd_Id_Name new_id = new Qd_Id_Name(idleft,idright,id,null);
                            Qd_Id_Name root = nm;
                            while(root.getNext()!=null){
                                    root = root.getNext();
                            }
                            root.setNext(id);
                            RESULT = nm;
            :}
            ;
//Qd_Id_Name
opr ::=

                    OP_PLUS:o //@+
                    {:
                            RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.PLUS);
                    :}
            |       OP_MINUS:o //@-
                    {:
                            RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.MINUS);
                    :}
            |       OP_DIV:o //@/
                    {:
                            RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.DIV);
                    :}
            |       OP_MULT:o //@*
                    {:
                            RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.MULT);
                    :}
            |       OP_MOD:o //@%
                    {:
                            RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.MOD);
                    :}
            |       OP_BITAND:o //@&
                    {:
                            RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.BITAND);
                    :}
```

```
        |       OP_BITOR:o //@|
                {:
                        RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.BITOR);
                :}
        |       OP_POWER:o //@^
                {:
                        RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.POWER);
                :}
        |       OP_RSHIFT:o //@>>
                {:
                        RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.RSHIFT);
                :}
        |       OP_LSHIFT:o //@<<
                {:
                        RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.LSHIFT);
                :}
        |       OP_URSHIFT:o //@>>>
                {:
                        RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.URSHIFT);
                :}
        | OP_NOT:o // @!
                {:
                RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.NOT);
                :}
        | OP_IN:o // @in
                {:
                RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.IN);
                :}
        | OP_INDEX:o // @[]
                {:
                RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.INDEX);
                :}
        | OP_BITCOMP:o // @~
                {:
                RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.BITCOMP);
                :}
        | OP_BITXOR:o // @xor
                {:
                RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.BITXOR);
                :}
        | OP_AND:o // @and
                {:
                RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.AND);
                :}
        | OP_OR:o // @or
                {:
                RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.OR);
                :}
        | OP_ISEQ:o // @?=
                {:
                RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.ISEQ);
                :}
        | OP_ISNEQ:o // @!=
                {:
                RESULT = new Qd_Id_Name(oleft,oright,null,null,Qd_Id_Name.ISNEQ);
                :}
        ;
//Qd_Com_Unit
compile_unit ::=
                pac_dec:pacdec
                use_dec_o:usedec
                cl_or_int_dec:t
                {:
                        RESULT = new Qd_Com_Unit(pacdecleft,pacdecright,pacdec,usedec,t);
                :}
                ;
//Qd_Use
use_dec_o ::=
                use_decs:usedec
                {:
                        RESULT = usedec;
                :}
        |
                {:
                        RESULT = null;
                :}
        ;
//Qd_Use
```

```
use_decs ::=
                    use_dec:usedec
                    {:
                            RESULT = usedec;
                    :}
          |
                    use_decs:u1 use_dec:u2
                    {:
                            Qd_Use g = u1;
                            while (g.getNext()!=null){
                                    g = g.getNext();
                            }
                            g.setNext(u2);
                            RESULT = u1;
                    :}
          ;

//Qd_Use
use_dec ::=
                    USE:u id_name:nm SEMICOLON
                    {:
                            Qd_Cen_Cntrl.addUses(nm);
                            RESULT = new Qd_Use(uleft,uright,nm,null);
                    :}
          ;

//Qd_Pac
pac_dec ::=
                    PACKAGE:p id_name:nm SEMICOLON
                    {:
                       Qd_Pac pac = new Qd_Pac(pleft,pright,nm);

                            RESULT =pac;
                    :}
          ;

//Qd_Cl_Or_Int_Dec
cl_or_int_dec ::=
                    cl_dec:c
                    {:
                            RESULT = c;
                    :}
          |         int_dec:i
                    {:
                            RESULT =i;
                    :}
          ;

//Qd_Acc_Type
access_types_opt ::=
                    {:
                        RESULT = null;
                    :}
          |         access_types:m
                    {:
                      if (m==null){
                            m=new Qd_Acc_Type(mleft,mright,Qd_Acc_Type.PACKAGE);
                      }

                            RESULT = m;
                    :}
          ;
//Qd_Acc_Type
access_types ::=
                    access_type:m
                    {:
                       RESULT = m;
                    :}
          |         access_types:ms access_type:m
                    {:
                       Qd_Acc_Type root = ms;
                            while(root.getNext()!=null){
                                    root = root.getNext();
                            }
                            root.setNext(m);
                            RESULT = ms;

                    :}
```

```
        ;
//Qd_Acc_Type
access_type ::=
                PLUS:m
                {:
                        RESULT = new Qd_Acc_Type(mleft,mright,Qd_Acc_Type.PLUS);
                :}
        |       MINUS:m
                {:

                        RESULT = new Qd_Acc_Type(mleft,mright,Qd_Acc_Type.MINUS);
                :}
        |       SHARP:m
                {:

                        RESULT = new Qd_Acc_Type(mleft,mright,Qd_Acc_Type.SHARP);
                :}
        |       CONST:m
                {:

                        RESULT = new Qd_Acc_Type(mleft,mright,Qd_Acc_Type.CONST);
                :}
        |       ABSTRACT:m
                {:

                        RESULT = new Qd_Acc_Type(mleft,mright,Qd_Acc_Type.ABSTRACT);
                :}
        ;

//Qd_Cl_Dec
cl_dec ::=
        access_types_opt:ato CLASS:c id_name:cn sup_and_int_opt:sio cl_bdy:cb
        {:
          if (ato!=null){
                if (ato.hasAccessType(Qd_Acc_Type.CONST) && ato.hasAccessType(Qd_Acc_Type.ABSTRACT)){
                  printError("Bir sınıf hem const hem de abstract olamaz!",atoleft,atoright);
                  Qd_Cen_Cntrl.incErrorCount();
                }
          }else{
            ato = new Qd_Acc_Type(atoleft,atoright,Qd_Acc_Type.NONE);
          }
                if (cn.getNext()!=null){
                  printError("Sınıf adı tek isim olmalıdır . kullanılamaz!",cnleft,cnright);
                  Qd_Cen_Cntrl.incErrorCount();
                }
                if (cb.hasAbstractMethod() && !ato.hasAccessType(Qd_Acc_Type.ABSTRACT)){
                  printError("Sınıf abstract metot içeriyor abstract olmalıdır!",cbleft,cbright);
                  Qd_Cen_Cntrl.incErrorCount();
                }
                if (cn.getNext()!=null){
                  printError("Sınıf adı tek isim olmalıdır!",cnleft,cnright);
                  Qd_Cen_Cntrl.incErrorCount();
                }
                cb.setClass_name(cn);
                List<Integer> list = Qd_Cen_Cntrl.checkClassFields(cn.getName());
                if (list!=null){
                   for (Integer i:list){
                     printError("Static Alan ve metotlar sınıf adı ile tanımlanır!!! satir:",i);
                     Qd_Cen_Cntrl.incErrorCount();
                   }
                }
                RESULT = new Qd_Cl_Dec(cleft,cright,ato,cn,sio,cb);
        :}
        ;

//Qd_Cl_Type
sup_and_int_opt ::=
                {:
                        RESULT = null;
                :}
        |       sup_and_ints:si
                {:
                        RESULT = si;
                :}
        ;
//Qd_Cl_Type
sup_and_ints ::=
                COLON sup_and_ints_list:sil
```

```
                {:
                        RESULT = sil;
                :}
        ;
//Qd_Cl_Type
sup_and_ints_list ::=
                cl_type:coi
                {:
                        if (!Qd_Cen_Cntrl.addUsedClass(coi)){
                           printError("Bu arayüz use da eklenmeli!",coileft,coiright);
                           Qd_Cen_Cntrl.incErrorCount();
                        }
                        RESULT = coi;
                :}
        |       sup_and_ints_list:s COMMA cl_type:coi
                {:
                        Qd_Cl_Type root = s;
                        while (root.getNext()!=null){
                                root = root.getNext();
                        }
                        root.setNext(coi);
                        if (!Qd_Cen_Cntrl.addUsedClass(coi)){
                           printError("Bu arayüz use da eklenmeli!",coileft,coiright);
                           Qd_Cen_Cntrl.incErrorCount();
                        }
                        RESULT = s;
                :}
        ;
//Qd_Cl_Bdy
cl_bdy ::=
                LBRACE:l cl_mem_decs:cbdo RBRACE
                {:
                        RESULT = new Qd_Cl_Bdy(lleft,lright,cbdo);
                :}
        |       LBRACE:l  RBRACE
                {:
                        RESULT = new Qd_Cl_Bdy(lleft,lright,null);
                :}
        ;
//Qd_Cl_Mem_Dec
cl_mem_decs ::=
                cl_mem_dec:cbd
                {:
                        RESULT = cbd;
                :}
        |       cl_mem_decs:cbdo cl_mem_dec:cbd
                {:
                        Qd_Cl_Mem_Dec root = cbdo;
                        while(root.getNext()!=null){
                                root = root.getNext();
                        }
                        root.setNext(cbd);
                        RESULT = cbdo;
                :}
        ;

//Qd_Cl_Mem_Dec
cl_mem_dec ::=
                fld_dec:f
                {:
                   //ayni adli bir alan var demektir.
                        if (!Qd_Cen_Cntrl.addField(f)){
                                printError("Aynı adli bir alan tanimli",f.getLine(),f.getCol());
                                 Qd_Cen_Cntrl.incErrorCount();
                        }
                        RESULT = f;
                :}
        |       mtd_dec:m
                {:
                        //ayni adli ve parametreli bir metot varsa
                        if (!Qd_Cen_Cntrl.addMethod(m)){
                                printError("aynı imzali method ",m.getLine(),m.getCol());
                                 Qd_Cen_Cntrl.incErrorCount();
                        }
                        RESULT = m;
                :}
        ;
```

```
//Qd_Fld_Dec
fld_dec ::=
                    var_type:t var_dec:vd LBRACE fld_bdy:fb RBRACE
                    {:
                              if (vd.getVar_id_name().getNext()!=null){
                                Qd_Cen_Cntrl.getclassNameLineList().put(vd.getVar_id_name().getValue(),vdleft);
                              }
                              Qd_Cen_Cntrl.addUsedClass(t);
                              RESULT = new Qd_Fld_Dec(tleft,tright,t,vd,fb);
                    :}
          ;
//Qd_Fld_Bdy
fld_bdy ::=
                    get_dec:gd set_dec:sd
                    {:
                              RESULT = new Qd_Fld_Bdy(gdleft,gdright,gd,sd);
                    :}
          |         set_dec:sd get_dec:gd
                    {:
                              RESULT = new Qd_Fld_Bdy(gdleft,gdright,gd,sd);
                    :}
          ;
//Qd_Get_Set_Dec
get_dec ::=
                    access_types_opt:mo GET mtd_bdy:mb
                    {:
                              if (mo==null){
                                mo = new Qd_Acc_Type(Qd_Acc_Type.PACKAGE);
                              }
                              RESULT = new Qd_Get_Set_Dec(moleft,moright,mo,mb,Qd_Get_Set_Dec.TYPE_GET);

                    :}
          ;
//Qd_Get_Set_Dec
set_dec ::=
                    access_types_opt:mo SET mtd_bdy:mb
                    {:
                              if (mo==null){
                                mo = new Qd_Acc_Type(Qd_Acc_Type.PACKAGE);
                              }
                              RESULT = new Qd_Get_Set_Dec(moleft,moright,mo,mb,Qd_Get_Set_Dec.TYPE_SET);
                    :}
          ;
//Qd_Var_Dec
var_decs ::=
                    var_dec:d
                    {:
                              RESULT = d;
                    :}
          |         var_decs:a COMMA var_dec:b
                    {:
                              Qd_Var_Dec root=a;
                              while(root.getNext()!=null){
                                        root = root.getNext();
                              }
                              root.setNext(b);
                              RESULT = a;
                    :}
          ;
//Qd_Var_Dec
var_dec ::=
              id_name:id
                    {:
                              RESULT = new Qd_Var_Dec(idleft,idright,id,null,null);
                    :}
          |         id_name:id EQ var_init:in
                    {:
                              RESULT = new Qd_Var_Dec(idleft,idright,id,in,null);
                    :}
          ;
//Qd_Var_Dec_ID
var_dec_id ::=
                    id_name:id
                    {:
                              RESULT=new Qd_Var_Dec_ID(idleft,idright,id,0);
                    :}

          ;
```

```
//Qd_Expr_Var_Init
var_init ::=
                        expr:e
                        {:
                                RESULT = new Qd_Expr_Var_Init(eleft,eright,e,null);
                        :}
                |       ary_init:a
                        {:
                                RESULT = new Qd_Expr_Var_Init(aleft,aright,a,null);
                        :}
                ;

//Qd_Mtd_Dec
mtd_dec ::=
                        mtd_head:h mtd_bdy:b
                        {:

                                RESULT = new Qd_Mtd_Dec(hleft,hright,h,b);
                        :}
                ;
//Qd_Mtd_Head
mtd_head ::=
                        access_types_opt:mo mtd_name:c mtd_back:mb
                        {:
                          if (mo==null){
                            mo = new Qd_Acc_Type(Qd_Acc_Type.PACKAGE);
                          }
                                if (c.getMethod_name().getNext()!=null){
                                   Qd_Cen_Cntrl.getclassNameLineList().put(c.getMethod_name().getValue(),cleft);
                                }
                          RESULT = new Qd_Mtd_Head(moleft,moright,mo,c,mb);
                        :}
                ;
//Qd_Mtd_Name
mtd_name ::=
                        DEF:d id_name:s LPAREN formal_param_lst_opt:f RPAREN
                        {:
                                RESULT = new Qd_Mtd_Name(dleft,dright,s,f,Qd_Mtd_Name.DEF);
                        :}
                |
                        EXTDEF:d id_name:s LPAREN formal_param_lst_opt:f RPAREN
                        {:
                                RESULT = new Qd_Mtd_Name(dleft,dright,s,f,Qd_Mtd_Name.EXTDEF);
                        :}
                |       ORADEF:d id_name:s LPAREN formal_param_lst_opt:f RPAREN
                        {:
                                RESULT = new Qd_Mtd_Name(dleft,dright,s,f,Qd_Mtd_Name.ORADEF);
                        :}
                ;
//Qd_Formal_Param
formal_param_lst_opt ::=
                        {:
                                RESULT = null;
                        :}
                |       formal_param_lst:f
                        {:
                                RESULT = f;
                        :}
                ;
//Qd_Formal_Param
formal_param_lst ::=
                        formal_param:f
                        {:
                                RESULT = f;
                        :}
                |       formal_param_lst:a COMMA formal_param:b
                        {:
                                Qd_Formal_Param root = a;
                                while (root.getNext()!=null){
                                        root = root.getNext();
                                }
                                root.setNext(b);
                                RESULT = a;
                        :}
                ;
//Qd_Formal_Param
formal_param ::=
                        var_type:t var_dec_id:i
```

```
                        {:
                                Qd_Cen_Cntrl.addUsedClass(t);
                                if (!Qd_Cen_Cntrl.addVar(i)){
                                   printError("Aynı adlı baska yerel degisken tanimli",ileft,iright);
                                    Qd_Cen_Cntrl.incErrorCount();
                                }
                                RESULT = new Qd_Formal_Param(tleft,tright,t,i,false,null);
                        :}
        |               CONST:c var_type:t var_dec_id:i
                        {:
                                Qd_Cen_Cntrl.addUsedClass(t);
                                if (!Qd_Cen_Cntrl.addVar(i)){
                                  printError("Aynı adlı baska yerel degisken tanimli",ileft,iright);
                                  Qd_Cen_Cntrl.incErrorCount();
                                }
                                RESULT = new Qd_Formal_Param(tleft,tright,t,i,true,null);
                        :}
        ;
//Qd_Mtd_Back
mtd_back ::=
                        {:
                          RESULT = null;
                        :}
        |               mtd_ret_type:mrt
                        {:
                                RESULT = new Qd_Mtd_Back(mrtleft,mrtright,mrt,null);
                        :}
        |               raise:to
                        {:
                                RESULT = new Qd_Mtd_Back(toleft,toright,null,to);
                        :}
        |  mtd_ret_type:mrt raise:to
                        {:
                                RESULT = new Qd_Mtd_Back(mrtleft,mrtright,mrt,to);
                        :}
        ;
//Qd_Type
mtd_ret_type ::=
                        COLON var_type:t
                        {:
                                Qd_Cen_Cntrl.addUsedClass(t);
                                RESULT = t;
                        :}
        ;

//Qd_Stmt_Raises
raise ::=
                        RAISES:t cl_type_lst:l
                        {:

                                RESULT = new Qd_Stmt_Raises(tleft,tright,l);
                        :}
        ;
//Qd_Cl_Type
cl_type_lst ::=
                        cl_type:c
                        {:
                                Qd_Cen_Cntrl.addUsedClass(c);
                                RESULT = c;
                        :}
        |               cl_type_lst:a COMMA cl_type:b
                        {:
                                Qd_Cl_Type root= a;
                                while(root.getNext()!=null){
                                        root = root.getNext();
                                }
                                root.setNext(b);
                                Qd_Cen_Cntrl.addUsedClass(b);
                                RESULT = a;
                        :}
        ;

//Qd_Stmt_Mtd_Bdy
mtd_bdy ::=
                        blk:b
                        {:
                          Qd_Cen_Cntrl.getCC().clearVars();
                          RESULT = new Qd_Stmt_Mtd_Bdy(bleft,bright,b,null);
```

```
                    :}
        |           SEMICOLON:s    //abstract method
                    {:

                      RESULT = new Qd_Stmt_Mtd_Bdy(sleft,sright,null,null);
                    :}
        |  cons_bdy:cb
                    {:
                     Qd_Cen_Cntrl.getCC().clearVars();
                      RESULT = new Qd_Stmt_Mtd_Bdy(cbleft,cbright,null,cb);
                    :}
        ;


//Qd_Stmt_Cons_Bdy
cons_bdy ::=
                    LBRACE:l call_cons:e
                            blk_stmts:b RBRACE
                    {:
                            RESULT = new Qd_Stmt_Cons_Bdy(lleft,lright,e,b);
                    :}
        |           LBRACE:l call_cons:e RBRACE
                    {:
                            RESULT = new Qd_Stmt_Cons_Bdy(lleft,lright,e,null);
                    :}

        ;
//Qd_Stmt_Call_Cons
call_cons ::=
                    SELF:t LPAREN arg_lst_opt:a RPAREN SEMICOLON
                    {:
                            RESULT = new Qd_Stmt_Call_Cons(
                            tleft,tright,Qd_Stmt_Call_Cons.SELF,null,a);
                    :}
        |           SUPER:t LPAREN arg_lst_opt:a RPAREN SEMICOLON
                    {:
                            RESULT = new Qd_Stmt_Call_Cons(
                            tleft,tright,Qd_Stmt_Call_Cons.SUPER,null,a);
                    :}
        |           first_expr:p DOT SELF LPAREN arg_lst_opt:a RPAREN SEMICOLON
                    {:
                            RESULT = new Qd_Stmt_Call_Cons(
                            pleft,pright,Qd_Stmt_Call_Cons.SELF,p,a);
                    :}
        |           first_expr:p DOT SUPER LPAREN arg_lst_opt:a RPAREN SEMICOLON
                    {:
                            RESULT = new Qd_Stmt_Call_Cons(
                            pleft,pright,Qd_Stmt_Call_Cons.SUPER,p,a);
                    :}
        ;

//Qd_Int_Dec
int_dec ::=
                    INTERFACE:i id_name:b sup_and_int_opt:sio int_bdy:d
                    {:
                            //Qd_Id_Name interface_id_name = new Qd_Id_Name(bleft,bright,b,null);
                            RESULT = new Qd_Int_Dec(ileft,iright,b,sio,d);
                    :}
        ;
//Qd_Int_Bdy
int_bdy ::=
                    LBRACE:lb int_mem_decs_opt:i RBRACE
                    {:
                            RESULT = new Qd_Int_Bdy(lbleft,lbright,i);
                    :}
        ;
//QdInt_Mem_Dec
int_mem_decs_opt ::=
                    {:
                            RESULT = null;
                    :}
        |           int_mem_decs:i
                    {:
                            RESULT = i;
                    :}
        ;
int_mem_decs ::=
                    int_mem_dec:i
```

```
                {:
                        RESULT = i;
                :}
        |       int_mem_decs:a int_mem_dec:b
                {:
                        Qd_Int_Mem_Dec root = a;
                        while (root.getNext()!=null){
                                root = root.getNext();
                        }
                        root.setNext(b);
                        RESULT = a;
                :}
        ;
//Qd_Int_Mem_Dec
int_mem_dec ::=
                int_fld_dec:c
                {:
                        RESULT = c;
                :}
        |       mtd_dec:a
                {:
                        if (a.getMethod_header().getMethod_dec().getMethod_name().getNext()!=null){
                           printError("Arayüz metot isimleri sınıfa ait olarak tanımlanamaz",aleft,aright);
                        }
                        RESULT = new Qd_Int_Mtd_Dec(aleft,aright,a);
                :}
        ;


//Qd_Int_Fld_Dec
int_fld_dec ::=
                var_type:t var_dec:vd SEMICOLON
                {:
                        Qd_Cen_Cntrl.addUsedClass(t);
                        RESULT = new Qd_Int_Fld_Dec(tleft,tright,t,vd);
                :}
        ;
//Qd_Expr_Ary_Init
ary_init ::=
                LBRACE:l var_inits:var COMMA RBRACE
                {:
                        RESULT = new Qd_Expr_Ary_Init(lleft,lright,var,true);
                :}
        |       LBRACE:l var_inits:var RBRACE
                {:
                        RESULT = new Qd_Expr_Ary_Init(lleft,lright,var,false);
                :}
        |       LBRACE:l COMMA RBRACE
                {:
                        RESULT = new Qd_Expr_Ary_Init(lleft,lright,null,true);
                :}
        |       LBRACE:l RBRACE
                {:
                        RESULT = new Qd_Expr_Ary_Init(lleft,lright,null,false);
                :}
        ;
//Qd_Expr_Var_Init
var_inits ::=
                var_init:in
                {:
                        RESULT = in;
                :}
        |       var_inits:a COMMA var_init:b
                {:
                        Qd_Expr_Var_Init root = a;
                        while(root.getNext()!=null){
                                root = root.getNext();
                        }
                        root.setNext(b);
                        RESULT = a;
                :}
        ;

//Qd_Blk
blk ::=
                LBRACE:l blk_stmts_opt:s RBRACE
                {:
                        RESULT = new Qd_Blk(lleft,lright,s);
```

```
                        :}
                ;
//Qd_Stmt_Blk
blk_stmts_opt ::=
                        {:
                                RESULT = null;
                        :}
                |       blk_stmts:b
                        {:
                                RESULT = b;
                        :}
                ;
//Qd_Stmt_Blk
blk_stmts ::=
                        blk_stmt:b
                        {:
                                RESULT = b;
                        :}
                |       blk_stmts:b blk_stmt:s
                        {:
                                Qd_Stmt_Blk root = b;
                                while(root.getNext()!=null){
                                        root = root.getNext();
                                }
                                root.setNext(s);
                                RESULT = b;
        :}
                ;
//Qd_Stmt_Blk
blk_stmt ::=
                        loc_var_dec:l SEMICOLON
                        {:
                                if (!Qd_Cen_Cntrl.addVar(l)){
                                  printError("Aynı adli yerel değişken tanımlama hatası:",lleft,lright);
                                  Qd_Cen_Cntrl.incErrorCount();
                                }
                                RESULT = new Qd_Stmt_Blk(lleft,lright,l,null);
                        :}
                |       stmt:s
                        {:
                                RESULT = new Qd_Stmt_Blk(sleft,sright,s,null);
                        :}
                ;

//Qd_Stmt_Loc_Var_Dec
loc_var_dec ::=
                        var_type:t var_decs:v
                        {:
                                if (!Qd_Cen_Cntrl.addUsedClass(t)){
                                        printError("Bu sınıfın use da tanımlanması gereklidir!!",tleft,tright);
                                  Qd_Cen_Cntrl.incErrorCount();
                                }
                                RESULT = new Qd_Stmt_Loc_Var_Dec(tleft,tright,t,v,false);
                        :}
                |       CONST var_type:t var_decs:v
                        {:
                                if (!Qd_Cen_Cntrl.addUsedClass(t)){
                                        printError("Bu sınıfın use da tanımlanması gereklidir!!",tleft,tright);
                                  Qd_Cen_Cntrl.incErrorCount();
                                }
                                RESULT = new Qd_Stmt_Loc_Var_Dec(tleft,tright,t,v,true);
                        :}
                ;
//Qd_Stmt
stmt ::=
                        stmt_no_sub_stmt:s
                        {:
                                RESULT= s;
                        :}
                |       stmt_with_lbl:s
                        {:
                                RESULT= s;
                        :}
                |       if_stmt:s
                        {:
                                RESULT= s;
                        :}
                |       if_else_stmt:s
```

```
                        {:
                                RESULT= s;
                        :}
                |       while_stmt:s
                        {:
                                RESULT= s;
                        :}
                |       for_stmt:s
                        {:
                                RESULT= s;
                        :}
                |       each_stmt:s
                        {:
                                RESULT= s;
                        :}
                ;
//Qd_Stmt
stmt_no_sub_stmt ::=
                        blk:a
                        {:
                                RESULT= a;
                        :}
                |       no_stmt:a
                        {:
                                RESULT= a;
                        :}
                |       expr_stmt:a
                        {:
                                RESULT= a;
                        :}
                |       case_stmt:a
                        {:
                                RESULT= a;
                        :}
                |       until_stmt:a
                        {:
                                RESULT= a;
                        :}
                |       break_stmt:a
                        {:
                                RESULT= a;
                        :}
                |       breakall_stmt:a
                        {:
                                RESULT= a;
                        :}
                |       continue_stmt:a
                        {:
                                RESULT= a;
                        :}
                |       ret_stmt:a
                        {:
                                RESULT= a;
                        :}
                |       raise_stmt:a
                        {:
                                RESULT= a;
                        :}
                |       try_stmt:a
                        {:
                                RESULT= a;
                        :}
                ;
//Qd_Stmt_No
no_stmt ::=
                        SEMICOLON:s
                        {:
                                RESULT = new Qd_Stmt_No(sleft,sright);
                        :}
                ;
//Qd_Stmt_With_Lbl
stmt_with_lbl ::=
                        ID_NAME:id COLON stmt:st
                        {:
                                Qd_Id_Name label = new Qd_Id_Name(idleft,idright,id,null);
                                RESULT = new Qd_Stmt_With_Lbl(idleft,idright,label,st);
                        :}
                ;
```

```
//Qd_Stmt_Expr_Stmt
expr_stmt ::=
                    stmt_expr:s SEMICOLON
                    {:
                              RESULT = new Qd_Stmt_Expr_Stmt(sleft,sright,s);
                    :}
          ;
//Qd_Stmt_Expr
stmt_expr ::=
                    asgn:a
                    {:
                              RESULT= new Qd_Stmt_Expr(aleft,aright,a);
                    :}
          |         mtd_call:p
                    {:
                              RESULT= new Qd_Stmt_Expr(pleft,pright,p);
                    :}
          |         new_expr:p
                    {:
                              RESULT= new Qd_Stmt_Expr(pleft,pright,p);
                    :}
          ;
//Qd_Stmt_If
if_stmt ::=
                    IF:i LPAREN expr:e RPAREN LBRACE blk_stmts_opt:a RBRACE
                    {:
                              RESULT = new Qd_Stmt_If(ileft,iright,e,a,null);
                    :}
          ;
//Qd_Stmt_If
if_else_stmt ::=
                    IF:i LPAREN expr:e RPAREN LBRACE blk_stmts_opt:ts RBRACE
                              ELSE LBRACE blk_stmts_opt:fs RBRACE
                    {:
                              RESULT = new Qd_Stmt_If(ileft,iright,e,ts,fs);
                    :}
          ;
//Qd_Stmt_Case
case_stmt ::=
                    CASE:c LPAREN id_name:e RPAREN case_blk:b
                    {:
                              //buradaki id_name bir degisken adi olmalidir
                              RESULT = new Qd_Stmt_Case(cleft,cright,e,b);
                    :}
          ;
//Qd_Case_Blk
case_blk ::=
                    LBRACE:l when_blk_stmt_groups:g others_blk:o RBRACE
                    {:
                              RESULT = new Qd_Case_Blk(lleft,lright,g,o);
                    :}
          |         LBRACE:l when_blk_stmt_groups:g RBRACE
                    {:
                              RESULT = new Qd_Case_Blk(lleft,lright,g,null);
                    :}
          ;
//Qd_Stmt_When_Blk_Stmt_Grp
when_blk_stmt_groups ::=
                    when_blk_stmt_group:s
                    {:
                              RESULT = s;
                    :}
          |         when_blk_stmt_groups:a when_blk_stmt_group:b
                    {:
                              Qd_Stmt_When_Blk_Stmt_Grp root = a;
                              while(root.getNext()!=null){
                                        root = root.getNext();
                              }
                              root.setNext(b);
                              RESULT = a;
                    :}
          ;
//Qd_Stmt_When_Blk_Stmt_Grp
when_blk_stmt_group ::=
                    WHEN:w LPAREN when_cnt_expr:wce RPAREN blk:b
                    {:
                              RESULT  = new Qd_Stmt_When_Blk_Stmt_Grp(wleft,wright,wce,b,null);
                    :}
```

```
		;
//Qd_Expr
when_cnt_expr ::=
			cnst_expr:c
			{:
				RESULT = c;
			:}
		|	in_expr:i
			{:
				RESULT = i;
			:}
		;
//Qd_Expr_In
in_expr ::=
			IN:i var_dec_id:v
			{:
				RESULT = new Qd_Expr_In(ileft,iright,v);
			:}
		;
 //Qd_Stmt_Others_Blk
others_blk ::=
			OTHERS:o blk:b
			{:
				RESULT = new Qd_Stmt_Others_Blk(oleft,oright,b);
			:}
		;
//Qd_Stmt_While
while_stmt ::=
			//WHILE:w LPAREN expr:e RPAREN stmt:s
			WHILE:w LPAREN expr:e RPAREN blk:b
			{:
				RESULT = new Qd_Stmt_While(wleft,wright,e,b);
			:}
		;
//Qd_Stmt_Until
until_stmt ::=
			DO:d blk:b UNTIL LPAREN expr:e RPAREN SEMICOLON
			{:
				RESULT = new Qd_Stmt_Until(dleft,dright,e,b);
			:}
		;
//Qd_Stmt_Each
each_stmt ::=
			EACH:ec LPAREN var_type:t var_dec_id:v COLON expr:e RPAREN blk:b
			{:
				if (!Qd_Cen_Cntrl.addUsedClass(t)){
					printError("Bu sınıfın use da tanımlanması gereklidir!!",tleft,tright);
					 Qd_Cen_Cntrl.incErrorCount();
				}
				RESULT = new Qd_Stmt_Each(ecleft,ecright,t,v,e,b);
			:}
		;
//Qd_Stmt_For
for_stmt ::=
			FOR:a LPAREN for_init_opt:fio COMMA for_last_opt:flo for_step_opt:fso RPAREN blk:b
			{:
				RESULT = new Qd_Stmt_For(aleft,aright,fio,flo,fso,b);
			:}
		;
//Qd_Expr_Two_Op
for_init_opt ::=
			asgn:a
			{:
				RESULT = a;
			:}
		;
//Qd_Expr
for_last_opt ::=
			expr:e
			{:
				RESULT = e;
			:}
		;
//Qd_Expr
for_step_opt ::=
			{:
				RESULT = null;
			:}
```

```
        |       COMMA expr:e
                {:
                        RESULT = e;
                :}
        ;
//Qd_Id_Name
id_opt ::=
                {:
                        RESULT = null;
                :}
        |       ID_NAME:id
                {:
                        RESULT = new Qd_Id_Name(idleft,idright,id,null);
                :}
        ;
//Qd_Stmt_Break
break_stmt ::=
                BREAK:b id_opt:o SEMICOLON
                {:
                        RESULT = new Qd_Stmt_Break(bleft,bright,o,false);
                :}
        ;
//Qd_Stmt_Break
breakall_stmt ::=
                BREAKALL:b SEMICOLON
                {:
                        RESULT = new Qd_Stmt_Break(bleft,bright,null,true);
                :}
        ;
//Qd_Stmt_Continue
continue_stmt ::=
                CONTINUE:c id_opt:o SEMICOLON
                {:
                        RESULT = new Qd_Stmt_Continue(cleft,cright,o);
                :}
        ;
//Qd_Stmt_Ret
ret_stmt ::=
                RETURN:r expr_opt:e SEMICOLON
                {:
                        RESULT = new Qd_Stmt_Ret(rleft,rright,e);
                :}
        ;
//Qd_Stmt_Raise
raise_stmt ::=
                RAISE:b expr:e SEMICOLON
                {:
                        RESULT = new Qd_Stmt_Raise(bleft,bright,e);
                :}
        ;

//Qd_Stmt_Try
try_stmt ::=
                TRY:t blk:b catches:c
                {:
                        RESULT = new Qd_Stmt_Try(tleft,tright,b,c,null);
                :}
        |       TRY:t blk:b catches_opt:c finally:f
                {:
                        RESULT = new Qd_Stmt_Try(tleft,tright,b,c,f);
                :}
        ;
//Qd_Try_Catch
catches_opt ::=
                {:
                        RESULT = null;
                :}
        |       catches:c
                {:
                        RESULT = c;
                :}
        ;
//Qd_Try_Catch
catches ::=
                catch:c
                {:
                        RESULT = c;
                :}
```

```
        |       catches:a catch:b
                {:
                        Qd_Try_Catch root = a;
                        while(root.getNext()!=null){
                                root = root.getNext();
                        }
                        root.setNext(b);
                        RESULT = a;
    :}
        ;
//Qd_Try_Catch
catch ::=
                CATCH:a LPAREN formal_param:b RPAREN blk:c
                {:
                        RESULT = new Qd_Try_Catch(aleft,aright,b,c,null);
                :}
        ;
//Qd_Stmt_Finally
finally ::=
                FINALLY:f blk:b
                {:
                        RESULT = new Qd_Stmt_Finally(fleft,fright,b);
                :}
        ;

//Qd_Expr
first_expr ::=
                first_expr_no_ary:p
                {:
                        RESULT = p;
                :}
        |       new_ary_init:a
                {:
                        RESULT =a;
                :}
        |       new_ary_no_init:a
                {:
                        RESULT = a;
                :}
        ;
//Qd_Expr
first_expr_no_ary ::=
                lit:l
                {:
                        RESULT = l;
                :}
        |       SELF:t
                {:
                        RESULT = new Qd_Expr_Self(tleft,tright);
                :}
        |       LPAREN:l expr:e RPAREN
                {:
                        RESULT = new Qd_Expr_Pa(lleft,lright,e);
                :}
        |       new_expr:e
                {:
                        RESULT = e;
                :}
        |       fld_var:f
                {:
                        RESULT = f;
                :}
        |       mtd_call:m
                {:
                        RESULT = m;
                :}
        |       ary_elmnt:a
                {:
                        RESULT = a;
                :}
        |       id_name:n DOT SELF
                {:
                        Qd_Type_Name nt = new Qd_Type_Name(nleft,nright,n);
                        RESULT = new Qd_Expr_First_No_Ary(nleft,nright,nt,Qd_Expr_First_No_Ary.SELF);
                :}
        |       id_name:n DOT QDILC
                {:
                   Qd_Type_Name nt = new Qd_Type_Name(nleft,nright,n);
```

```
                                RESULT = new Qd_Expr_First_No_Ary(nleft,nright,nt,Qd_Expr_First_No_Ary.QDILC);
                        :}
        | ary_type:a DOT QDILC
                        {:
                                RESULT = new Qd_Expr_First_No_Ary(aleft,aright,a,Qd_Expr_First_No_Ary.QDILC);
                        :}
        ;

//Qd_New_Expr
new_expr ::=
                        NEW:n cl_type:t LPAREN arg_lst_opt:a RPAREN
                        {:
                                if (!Qd_Cen_Cntrl.addUsedClass(t)){
                                        printError("Bu sınıfın use da tanımlanması gereklidir!!",tleft,tright);
                                   Qd_Cen_Cntrl.incErrorCount();
                                }
                                RESULT = new Qd_New_Expr(nleft,nright,t,a);
                        :}
        ;
//Qd_Expr_Arg
arg_lst_opt ::=
                        {:
                                RESULT = null;
                        :}
        |               arg_list:a
                        {:
                                RESULT = a;
                        :}
        ;
//Qd_Expr_Arg
arg_list ::=
                        expr:e
                        {:
                                RESULT = new Qd_Expr_Arg(eleft,eright,e,null);
                        :}
        |               arg_list:a COMMA expr:b
                        {:
                                Qd_Expr_Arg root = a;
                                Qd_Expr_Arg newarg = new Qd_Expr_Arg(bleft,bright,b,null);
                                while(root.getNext()!=null){
                                        root = root.getNext();
                                }
                                root.setNext(newarg);
                                RESULT = a;
                        :}
        ;
//Qd_Expr_New_Ary
new_ary_no_init ::=
                        NEW:n cl_type:t d_exprs:e dms_opt:o
                        {:
                                if (!Qd_Cen_Cntrl.addUsedClass(t)){
                                        printError("Bu sınıfın use da tanımlanması gereklidir!!",tleft,tright);
                                         Qd_Cen_Cntrl.incErrorCount();
                                }
                                RESULT = new Qd_Expr_New_Ary(nleft,nright,t,e,o.intValue(),null);
                        :}
        ;
//Qd_Expr_New_Ary
new_ary_init ::=
                        NEW:n cl_type:t dms:d ary_init:a
                        {:
                                if (!Qd_Cen_Cntrl.addUsedClass(t)){
                                        printError("Bu sınıfın use da tanımlanması gereklidir!!",tleft,tright);
                                         Qd_Cen_Cntrl.incErrorCount();
                                }
                                RESULT = new Qd_Expr_New_Ary(nleft,nright,t,null,d.intValue(),a);
                        :}

        ;
//Qd_Expr_D
d_exprs ::=
                        d_expr:e
                        {:
                                RESULT = e;
                        :}
        |               d_exprs:d d_expr:e
                        {:
                                Qd_Expr_D root = d;
```

```
                                        while(root.getNext()!=null){
                                                root = root.getNext();
                                        }
                                        root.setNext(e);
                                        RESULT = d;
                                :}
        ;
//Qd_Expr_D
d_expr ::=
                                LBRACK:l expr:e RBRACK
                                {:
                                        RESULT = new Qd_Expr_D(lleft,lright,e,null);
                                :}
        ;
//Integer
dms_opt ::=
                                {:
                                        RESULT = new Integer(0);
                                :}
        |               dms:d
                                {:
                                        RESULT = d;
                                :}
        ;
//Integer
dms ::=
                                LBRACK RBRACK
                                {:
                                        RESULT = new Integer(1);
                                :}
        |               dms:d LBRACK RBRACK
                                {:
                                        RESULT = new Integer(d.intValue()+1);
                                :}
        ;
//Qd_Expr_Fld_Var
fld_var ::=
                                first_expr:p DOT ID_NAME:id
                                {:
                                        Qd_Id_Name i = new Qd_Id_Name(idleft,idright,id,null);
                                        RESULT = new Qd_Expr_Fld_Var(pleft,pright,p,Qd_Expr_Fld_Var.FIRST,i);
                                :}
        |               SUPER:s DOT ID_NAME:id
                                {:
                                        Qd_Id_Name i = new Qd_Id_Name(idleft,idright,id,null);
                                        RESULT = new Qd_Expr_Fld_Var(sleft,sright,null,Qd_Expr_Fld_Var.SUPER,i);
                                :}
        |               id_name:n DOT SUPER DOT ID_NAME:id
                                {:
                                        Qd_Id_Name i = new Qd_Id_Name(idleft,idright,id,null);
                                        Qd_Expr_Id ei = new Qd_Expr_Id(nleft,nright,n);
                                        RESULT = new Qd_Expr_Fld_Var(nleft,nright,ei,Qd_Expr_Fld_Var.SUPER,i);
                                :}
        ;
//Qd_Expr_Mtd_Call
mtd_call ::=
                                id_name:n LPAREN arg_lst_opt:a RPAREN
                                {:
                                        RESULT = new Qd_Expr_Mtd_Call(nleft,nright,null,n,a,Qd_Expr_Mtd_Call.NAME);
                                :}
        |               first_expr:p DOT ID_NAME:id LPAREN arg_lst_opt:a RPAREN
                                {:
                                        Qd_Id_Name i = new Qd_Id_Name(idleft,idright,id,null);
                                        RESULT = new Qd_Expr_Mtd_Call(pleft,pright,p,i,a,Qd_Expr_Mtd_Call.FIRST);
                                :}
        |               SUPER:s DOT ID_NAME:id LPAREN arg_lst_opt:a RPAREN
                                {:
                                        Qd_Id_Name i = new Qd_Id_Name(idleft,idright,id,null);
                                        RESULT = new Qd_Expr_Mtd_Call(sleft,sright,null,i,a,Qd_Expr_Mtd_Call.SUPER);
                                :}
        |               id_name:n DOT SUPER DOT ID_NAME:id LPAREN arg_lst_opt:a RPAREN
                                {:
                                        Qd_Id_Name i = new Qd_Id_Name(idleft,idright,id,null);
                                        Qd_Expr_Id ei = new Qd_Expr_Id(nleft,nright,n);
                                        RESULT = new Qd_Expr_Mtd_Call(nleft,nright,ei,i,a,Qd_Expr_Mtd_Call.NAME_SUPER);
                                :}
        ;
//Qd_Expr_Ary_Elmnt
```

```
ary_elmnt ::=
                    id_name:n LBRACK expr:e RBRACK
                    {:
                              Qd_Expr_Id array_id_name = new Qd_Expr_Id(nleft,nright,n);
                              RESULT = new Qd_Expr_Ary_Elmnt(nleft,nright,array_id_name,e);
                    :}
          |         first_expr_no_ary:p LBRACK expr:e RBRACK
                    {:
                              RESULT = new Qd_Expr_Ary_Elmnt(pleft,pright,p,e);
                    :}
          ;
//Qd_Expr
post_expr ::=
                    first_expr:p
                    {:
                              RESULT = p;
                    :}
          |         id_name:n
                    {:
                              Qd_Expr_Id ei = new Qd_Expr_Id(nleft,nright,n);
                              RESULT = ei;
                    :}
          ;

//Qd_Expr
sgn_expr ::=
                    PLUS:p sgn_expr:u
                    {:
                              RESULT = new Qd_Expr_Un(pleft,pright,Qd_Expr_Un.PLUS,u);
                    :}
          |         MINUS:m sgn_expr:u
                    {:
                              RESULT = new Qd_Expr_Un(mleft,mright,Qd_Expr_Un.MINUS,u);
                    :}
          |         un_exprs:u
                    {:
                              RESULT = u;
                    :}
          ;
//Qd_Expr
un_exprs ::=
                    post_expr:p
                    {:
                              RESULT = p;
                    :}
          |         COMP:c sgn_expr:u
                    {:
                              RESULT = new Qd_Expr_Un(cleft,cright,Qd_Expr_Un.COMP,u);
                    :}
          |         NOT:n sgn_expr:u
                    {:
                              RESULT = new Qd_Expr_Un(nleft,nright,Qd_Expr_Un.NOT,u);
                    :}
          |         type_ch_expr:c
                    {:
                              RESULT = c;
                    :}
          ;
//Qd_Expr_Type_Ch
type_ch_expr ::=
                    LPAREN:l expr:e RPAREN un_exprs:u
                    {:
                              RESULT = new Qd_Expr_Type_Ch(lleft,lright,e,u,0);
                    :}
          |         LPAREN:l id_name:n dms:d RPAREN un_exprs:u
                    {:
                              Qd_Expr_Id ei = new Qd_Expr_Id(nleft,nright,n);
                              RESULT = new Qd_Expr_Type_Ch(lleft,lright,ei,u,d.intValue());
                    :}
          ;
//Qd_ExprOver
pow_expr ::=
                    sgn_expr:ue
                    {:
                              RESULT = ue;
                    :}
          |         sgn_expr:a POWER pow_expr:b
                    {:
```

```
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.POWER,b);
                                :}
        ;
//Qd_Expr_Two_Op:Qd_Expr
mul_div_mod_expr ::=
                                pow_expr:o
                                {:
                                        RESULT = o;
                                :}
        |               mul_div_mod_expr:a MULT pow_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.MULT,b);
                                :}
        |               mul_div_mod_expr:a DIV pow_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.DIV,b);
                                :}
        |               mul_div_mod_expr:a MOD pow_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.MOD,b);
                                :}
        ;
//Qd_Expr_Two_Op:Qd_Expr
add_sub_expr ::=
                                mul_div_mod_expr:c
                                {:
                                        RESULT = c;
                                :}
        |               add_sub_expr:a PLUS mul_div_mod_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.PLUS,b);
                                :}
        |               add_sub_expr:a MINUS mul_div_mod_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.MINUS,b);
                                :}
        ;
//Qd_Expr_Two_Op:Qd_Expr
shft_expr ::=
                                add_sub_expr:c
                                {:
                                        RESULT = c;
                                :}
        |               shft_expr:a LSHIFT add_sub_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.LSHIFT,b);
                                :}
        |               shft_expr:a RSHIFT add_sub_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.RSHIFT,b);
                                :}
        |               shft_expr:a URSHIFT add_sub_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.URSHIFT,b);
                                :}
        ;
//Qd_Expr
rel_expr ::=
                                shft_expr:c
                                {:
                                        RESULT = c;
                                :}
        |               rel_expr:a LT shft_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.LT,b);
                                :}
        |               rel_expr:a GT shft_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.GT,b);
                                :}
        |               rel_expr:a LTEQ shft_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.LTEQ,b);
                                :}
        |               rel_expr:a GTEQ shft_expr:b
                                {:
                                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.GTEQ,b);
                                :}
```

```
        |       rel_expr:a IN var_type:b
                {:
                        if (!Qd_Cen_Cntrl.addUsedClass(b)){
                                printError("Bu sınıfın use da tanımlanması gereklidir!!",bleft,bright);
                                 Qd_Cen_Cntrl.incErrorCount();
                        }
                        Qd_Expr_T et = new Qd_Expr_T(bleft,bright,b);
                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.IN,et);
                :}
        | rel_expr:a ISA var_type:b
          {:
                if (!Qd_Cen_Cntrl.addUsedClass(b)){
                                printError("Bu sınıfın use da tanımlanması gereklidir!!",bleft,bright);
                                 Qd_Cen_Cntrl.incErrorCount();
                        }
             Qd_Expr_T et = new Qd_Expr_T(bleft,bright,b);
                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.ISA,et);
          :}
        ;
//Qd_Expr
eq_or_noteq_expr ::=
                rel_expr:c
                {:
                        RESULT = c;
                :}
        |       eq_or_noteq_expr:a ISEQ rel_expr:b
                {:
                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.ISEQ,b);
                :}
        |       eq_or_noteq_expr:a NOTEQ rel_expr:b
                {:
                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.NOTEQ,b);
                :}
        ;
//Qd_Expr
bitand_expr ::=
                eq_or_noteq_expr:c
                {:
                        RESULT = c;
                :}
        |       bitand_expr:a BITAND eq_or_noteq_expr:b
                {:
                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.BITAND,b);
                :}
        ;
//Qd_Expr
xor_expr ::=
                bitand_expr:c
                {:
                        RESULT = c;
                :}
        |       xor_expr:a BITXOR bitand_expr:b
                {:
                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.BITXOR,b);
                :}
        ;
//Qd_Expr
bitor_expr ::=
                xor_expr:c
                {:
                        RESULT = c;
                :}
        |       bitor_expr:a BITOR xor_expr:b
                {:
                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.BITOR,b);
                :}
        ;
//Qd_Expr
and_expr ::=
                bitor_expr:c
                {:
                        RESULT = c;
                :}
        |       and_expr:a AND bitor_expr:b
                {:
                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.AND,b);
                :}
        |       and_expr:a BIGAND bitor_expr:b
```

```
                {:
                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.BIGAND,b);
                :}
        ;
//Qd_Expr
or_expr ::=
                and_expr:c
                {:
                        RESULT = c;
                :}
        |       or_expr:a OR and_expr:b
                {:
                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.OR,b);
                :}
        |       or_expr:a BIGOR and_expr:b
                {:
                        RESULT = new Qd_Expr_Two_Op(aleft,aright,a,Qd_Expr_Two_Op.BIGOR,b);
                :}
        ;
//Qd_Expr
asgn_expr ::=
                or_expr:c
                {:
                        RESULT = c;
                :}
        |       asgn:c
                {:
                        RESULT = c;
                :}
        ;
//Qd_Expr
asgn ::=
                l_expr:lhs EQ asgn_expr:e
                {:
                        RESULT = new Qd_Expr_Two_Op(lhsleft,lhsright,lhs,Qd_Expr_Two_Op.EQ,e);
                :}
        ;
//Qd_Expr
l_expr ::=
                id_name:n
                {:
                        Qd_Expr_Id ei = new Qd_Expr_Id(nleft,nright,n);
                        RESULT = ei;
                :}
        |       fld_var:f
                {:
                        RESULT = f;
                :}
        |       ary_elmnt:a
                {:
                        RESULT = a;
                :}
        ;

//Qd_Expr
expr_opt ::=

          | expr:e
                {:
                        RESULT = e;
                :}
        ;
//Qd_Expr
expr ::=
                asgn_expr:e
                {:
                        RESULT = e;
                :}
        ;
//Qd_Expr
cnst_expr ::=
                expr:e
                {:
                        RESULT = e;
                :}
        ;
```

EK 2- qoperator JCup Dosyası

```
package qop.parser;
import java_cup.runtime.*;
import qop.*;

parser code  {:

 public void report_error(String msg, Object sym) {
    StringBuffer m = new StringBuffer("HATA(ERROR): ");
                java_cup.runtime.Symbol s;
    if (sym instanceof java_cup.runtime.Symbol) {
      s = (java_cup.runtime.Symbol)sym;
      m.append(findSymbolText(s) );
      m.append(" ,satir:"+s.left);
      m.append(" : "+msg);
    }
      System.out.println(m);
  }
   private static String findSymbolText(Symbol sym){
        String symText = null;
        if (sym.value==null){
           symText="";
        }else{
           symText=" (\""+sym.value.toString()+"\")";
        }
        java.lang.reflect.Field [] flds = sym.class.getFields();
        for (int i = 0; i < flds.length ; i++){
           if (!java.lang.reflect.Modifier.isPublic(flds[i].getModifiers())) continue;
           try {
              if (flds[i].getInt(null) == sym.sym) return flds[i].getName()+symText;
           }catch (Exception ex) {
              return "";
           }
        }
        return sym.toString()+symText;
    }
   public void report_fatal_error(String msg, Object sym) {
     report_error(msg, sym);
     throw new RuntimeException("!...Yazım Hatası...");
   }
:};

scan with {: return getScanner().next_token(); :}

terminal LPAREN,RPAREN,LBRACE,RBRACE,LBRACK,RBRACK,COMMA,SEMICOLON,EQUAL;

terminal TENSOR,TENSOR_POWER,PLUS,MINUS,MULT,DIV,BAR,POWER;
terminal GT,LT,NOT,QUESTION,CROSS;

terminal IM,PI,EXP,SIN,COS,SUM,SQRT;
terminal OP_I,OP_X,OP_Y,OP_Z,OP_S,OP_T,OP_R,OP_H;
terminal OP_QFT,OP_SWAP,OP_CNOT12,OP_CNOT21,OP_TOFFOLI;
terminal OP_ROTX,OP_ROTY,OP_ROTZ;
terminal OP_ENTANGLE,OP_PHASE,OP_FLIP;

terminal java.lang.Number INT_LIT;
terminal java.lang.Number FL_LIT;
terminal java.lang.String ID_NAME; // name


// NON TERMINALLER BURAYA YAZILACAK
////////////////////////////////////////////////////////////////
non terminal Qop_Op_Unit goal;
non terminal Qop_Expr lit;
non terminal Qop_Expr com_num;
non terminal Qop_Expr expr_cross;
non terminal Qop_Expr expr_mtrx;
non terminal Qop_Expr mtrx_rows;
non terminal Qop_Expr mtrx_row;
non terminal Qop_Expr cols;
non terminal Qop_Expr expr_lit;
non terminal Qop_Expr expr_cons;
non terminal Qop_Expr expr_ops;
non terminal Qop_Expr expr_funs;
```

```
non terminal Qop_Expr expr_id_name;
non terminal Qop_Expr expr_ten_pow;
non terminal Qop_Expr expr_ten;
non terminal Qop_Expr expr_pow;
non terminal Qop_Expr expr_un;
non terminal Qop_Expr expr_mult_div;
non terminal Qop_Expr expr_add_sub;
non terminal Qop_Expr expr;
non terminal Qop_Expr expr_end;

start with goal;

//Qop_Op_Unit
goal ::=
                        expr:e
                        {:
                        RESULT=new Qop_Op_Unit(eleft,eright,e);
                        :}
        ;
//Qop_Expr
expr ::=
                        expr_add_sub:e
                        {:
                                RESULT  = e;
                        :}
        ;

//Qop_Expr_Two_Op : Qop_Expr
expr_add_sub ::=
                        expr_mult_div:e
                        {:
                                RESULT = e;
                        :}
        |               expr_add_sub:l PLUS expr_mult_div:r
                        {:
                                 RESULT = new Qop_Expr_Two_Op(lleft,lright,l,Qop_Expr_Two_Op.PLUS,r);
                        :}
        |               expr_add_sub:l MINUS expr_mult_div:r
                        {:
                                 RESULT = new Qop_Expr_Two_Op(lleft,lright,l,Qop_Expr_Two_Op.MINUS,r);
                        :}
        ;

expr_mult_div ::=
                        expr_pow:e
                        {:
                           RESULT = e;
                        :}
        |               expr_mult_div:l MULT expr_pow:r
                        {:
                                RESULT = new Qop_Expr_Two_Op(lleft,lright,l,Qop_Expr_Two_Op.MULT,r);
                        :}
        |               expr_mult_div:l DIV expr_pow:r
                        {:
                                RESULT = new Qop_Expr_Two_Op(lleft,lright,l,Qop_Expr_Two_Op.DIV,r);
                        :}
        ;

//Qop_Expr_Two_Op : Qop_Expr
expr_pow ::=
                        expr_un:e
                        {:
                          RESULT = e;
                        :}
        |               expr_un:r POWER expr_pow:l
                        {:
                          RESULT = new Qop_Expr_Two_Op(lleft,lright,l,Qop_Expr_Two_Op.POWER,r);
                        :}
        ;

//Qop_Expr_Two_Op : Qop_Expr
expr_un ::=
                        MINUS:p expr_un:r
                        {:
                         RESULT = new Qop_Expr_Two_Op(pleft,pright,null,Qop_Expr_Two_Op.UNARY_MINUS,r);
                        :}
        |               NOT:p expr_un:r
                        {:
```

```
                        RESULT = new Qop_Expr_Two_Op(pleft,pright,null,Qop_Expr_Two_Op.UNARY_NOT,r);
                    :}
        |           expr_ten:e
                    {:
                      RESULT = e;
                    :}
        ;

//Qop_Expr_Two_Op : Qop_Expr
expr_ten ::=
                    expr_ten_pow:e
                    {:
                      RESULT = e;
                    :}
        |           expr_ten:l TENSOR expr_ten_pow:r
                    {:
                      RESULT = new Qop_Expr_Two_Op(lleft,lright,l,Qop_Expr_Two_Op.TENSOR,r);
                    :}
        ;

// Qop_Expr
expr_ten_pow ::=
                    expr_mtrx:e
                    {:
                       RESULT = e;
                    :}
        |           expr_mtrx:l TENSOR_POWER expr_ten_pow:r
                    {:
                        RESULT = new Qop_Expr_Two_Op(lleft,lright,l,Qop_Expr_Two_Op.TENSOR_POWER,r);
                    :}
        ;

//Qop_Expr_Mtrx: Qop_Expr
expr_mtrx ::=
                    LT:l mtrx_rows:me GT
                    {:
                            RESULT = new Qop_Expr_Mtrx(lleft,lright,me);
                    :}
        |           expr_cross:e
                    {:
                            RESULT = e;
                    :}
        ;

//Qop_Expr_Mtrx_Row
mtrx_rows ::=
                    mtrx_row:in
                    {:
                            RESULT = in;
                    :}
        |           mtrx_rows:a SEMICOLON mtrx_row:b
                    {:
                            Qop_Expr_Mtrx_Row root = (Qop_Expr_Mtrx_Row)a;
                            while(root.getNext()!=null){
                                    root = root.getNext();
                            }
                            root.setNext(b);
                            RESULT = a;
                    :}
        ;
//Qop_Expr_Mtrx_Row
mtrx_row ::=
                    cols:e
                    {:
                            RESULT = new Qop_Expr_Mtrx_Row(eleft,eright,e,null);
                    :}
        ;
//Qop_Expr
cols ::=
                    expr:e
                    {:
                      RESULT = e;
                    :}
        |           cols:a COMMA expr:b
                    {:
                       Qop_Expr root = a;
                            while(root.getNext()!=null){
                                    root = root.getNext();
```

```
                                        }
                                        root.setNext(b);
                                        RESULT = a;
                                :}
        ;
//Qop_Expr_Cross
expr_cross ::=
    BAR:b expr:e1 CROSS expr:e2 BAR
    {:
      RESULT = new Qop_Expr_Cross(bleft,bright,e1,e2);
    :}
  | expr_lit:e
        {:
                RESULT = e;
        :}
  ;
//Qop_Expr
expr_lit ::=
                lit:e
                {:
                        RESULT = e;
                :}
        |       com_num:e
                {:

                        RESULT = (Qop_Expr)e;
                :}
        ;

//Qop_Lit
lit ::=
    INT_LIT:l
                {:
                  RESULT= new Qop_Lit(lleft,lright,l,Qop_Lit.INT_LIT);
                :}
        |       FL_LIT:l
    {:
                  RESULT= new Qop_Lit(lleft,lright,l,Qop_Lit.FL_LIT);
                :}
        ;

//Qop_Com_Num
com_num ::=
                LBRACE:l lit:l1 COMMA lit:l2 RBRACE
                {:
                   RESULT = new Qop_Com_Num(lleft,lright,l1,l2);
                :}
        |       LBRACE:l lit:l1 COMMA QUESTION:q RBRACE
                {:
                   RESULT = new Qop_Com_Num(lleft,lright,l1,null);
                :}
        |       LBRACE:l QUESTION:q1 COMMA lit:l2 RBRACE
                {:
                   RESULT = new Qop_Com_Num(lleft,lright,null,l2);
                :}
        |  LBRACE:l QUESTION:q1 COMMA QUESTION:q2 RBRACE
                {:
                   RESULT = new Qop_Com_Num(lleft,lright,null,null);
                :}
  |     expr_id_name:e
        {:
                RESULT = e;
        :}
        ;


//Qop_Expr
expr_id_name ::=
        ID_NAME:n
                {:
                        Qop_Expr_Id_Name ei = new Qop_Expr_Id_Name(nleft,nright,n);
                        RESULT = ei;
                :}
        |       PI:c
                {:
                  RESULT = new Qop_Expr_Op(cleft,cright,Qop_Expr_Op.CONST_PI);
                :}
        |       IM:c
```

```
                {:
                  RESULT = new Qop_Expr_Op(cleft,cright,Qop_Expr_Op.CONST_I);
                :}
        | QUESTION:c
                {:
                  RESULT = new Qop_Expr_Op(cleft,cright,Qop_Expr_Op.CONST_QUESTION);
                :}
        |                       OP_I:op
                {:
                  RESULT = new Qop_Expr_Op(opleft,opright,Qop_Expr_Op.OP_I);
                :}
        |       OP_X:op
                {:
                  RESULT = new Qop_Expr_Op(opleft,opright,Qop_Expr_Op.OP_X);
                :}
        |       OP_Y:op
                {:
                  RESULT = new Qop_Expr_Op(opleft,opright,Qop_Expr_Op.OP_Y);
                :}
        |       OP_Z:op
                {:
                  RESULT = new Qop_Expr_Op(opleft,opright,Qop_Expr_Op.OP_Z);
                :}
        |       OP_S:op
                {:
                  RESULT = new Qop_Expr_Op(opleft,opright,Qop_Expr_Op.OP_S);
                :}
        |       OP_T:op
                {:
                  RESULT = new Qop_Expr_Op(opleft,opright,Qop_Expr_Op.OP_T);
                :}
        |       OP_H:op
                {:
                  RESULT = new Qop_Expr_Op(opleft,opright,Qop_Expr_Op.OP_H);
                :}
        |       OP_CNOT12:op
                {:
                  RESULT = new Qop_Expr_Op(opleft,opright,Qop_Expr_Op.OP_CNOT12);
                :}
        |       OP_CNOT21:op
                {:
                  RESULT = new Qop_Expr_Op(opleft,opright,Qop_Expr_Op.OP_CNOT21);
                :}
        |       OP_SWAP:op
                {:
                  RESULT = new Qop_Expr_Op(opleft,opright,Qop_Expr_Op.OP_SWAP);
                :}
        |       OP_TOFFOLI:op
                {:
                  RESULT = new Qop_Expr_Op(opleft,opright,Qop_Expr_Op.OP_TOFFOLI);
                :}
        |                       SIN:f LBRACK expr:e RBRACK
                {:
                  RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.SIN,e);
                :}
        |       COS:f LBRACK expr:e RBRACK
                {:
                  RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.COS,e);
                :}
        |       EXP:f LBRACK expr:e RBRACK
                {:
                  RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.EXP,e);
                :}
        | SQRT:f LBRACK expr:e RBRACK
                {:
                  RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.SQRT,e);
                :}
        |       OP_QFT:f LBRACK expr:e RBRACK
                {:
                  RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.OP_QFT,e);
                :}
        |       OP_PHASE:f LBRACK expr:e RBRACK
                {:
                  RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.OP_PHASE,e);
                :}
        |       OP_ENTANGLE:f LBRACK expr:e RBRACK
                {:
                  RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.OP_ENTANGLE,e);
```

```
                    :}
        |           OP_FLIP:f LBRACK expr:e RBRACK
                    {:
                      RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.OP_FLIP,e);
                    :}
        |           OP_R:f LBRACK expr:e RBRACK
                    {:
                     RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.OP_R,e);
                    :}
        |           SUM:f LBRACK ID_NAME:i EQUAL expr:e1 COMMA expr:e2 COMMA expr:e3 RBRACK
                    {:
                       Qop_Expr_Id_Name identifier = new Qop_Expr_Id_Name(ileft,iright,i);

                       RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.SUM,identifier,e1,e2,e3);
                    :}
        |           OP_ROTX:f LBRACK expr:e RBRACK
                    {:
                      RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.OP_ROTX,e);
                    :}
        |           OP_ROTY:f LBRACK expr:e RBRACK
                    {:
                      RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.OP_ROTY,e);
                    :}
        |           OP_ROTZ:f LBRACK expr:e RBRACK
                    {:
                      RESULT = new Qop_Expr_Fun(fleft,fright,Qop_Expr_Op.OP_ROTZ,e);
                    :}
        |           expr_end:e
                    {:
                      RESULT = e;
                    :}
        ;

//Qop_Expr
expr_end ::=
                    LPAREN:l expr:e RPAREN
                    {:
                              RESULT = new Qop_Expr_Pa(lleft,lright,e);
                    :}
        ;
```

EK 3- qoperator Yorumlayıcı Kodları

```
package qop;

import java.util.ArrayList;
import java.util.List;

import org.jscience.mathematics.number.Complex;
import org.jscience.mathematics.vector.ComplexMatrix;
import org.jscience.mathematics.vector.Matrix;

import qop.interpreter.MatrixParser;
import qop.interpreter.QOPComplexMatrixes;
import qop.interpreter.QOPInterpreter;

public class Qop_InterpretVisitor implements QOPVisitor {

        private List<String> errors;

        public Qop_InterpretVisitor() {
                errors = new ArrayList<String>();
        }
        public List<String> getErrors() {
                return errors;
        }

        public void report_Error(String error) {
                errors.add(error);
        }

        public static int LN(int sayi) {
                int res = 0;
                if (sayi <= 2)
                        return 2;
                if (sayi % 2 == 1) {
```

```
                sayi = sayi + 1;
            }
            while (sayi != 0 || sayi != 1) {
                sayi = sayi / 2;
                res++;
            }
            res = res - 1;
            return (res);
        }

        @Override
        public void accept(Qop_Expr_Cross in) {
            Qop_Expr l = in.getExpr_left();
            Qop_Expr r = in.getExpr_right();
            int row, col;
            int rw;
            if (l != null)
                l.visit(this);
            if (r != null)
                r.visit(this);
            Complex[][] c = null;
            if (!l.isDouble() || !r.isDouble()) {
                report_Error("|><| isleminde int kullanilmalidir");
            } else {
                row = (int) l.getDouble();
                col = (int) r.getDouble();
                rw = Math.max(row + 1, col + 1);
                rw = LN(rw);
                c = QOPComplexMatrixes.createZeroNxNMatrix(rw);
                c[row][col] = Complex.ONE;
            }
            in.setComplexMatrix(ComplexMatrix.valueOf(c));
            in.setAsComplexMatrix();
        }

        @Override
        public void accept(Qop_Expr_Fun in) {
            Qop_Expr_Id_Name id_name = in.getIdentifier();
            Qop_Expr start = in.getStart();
            Qop_Expr stop = in.getStop();
            Qop_Expr op = in.getOperator();
            if (id_name != null) {
                id_name.visit(this);
            }
            if (start != null) {
                start.visit(this);
            }
            if (stop != null) {
                stop.visit(this);
            }
            if (op != null) {
                op.visit(this);
            }
            switch (in.getOPType()) {
            case Qop_Expr_Fun.SIN: {
                if (op.isDouble()) {
                    in.setDoubleValue(Math.sin(in.getOperator().getDouble()));
                    in.setAsDouble();
                } else {
                    report_Error("SIN parametre double olmalıdır");
                }
            }
                break;
            case Qop_Expr_Fun.COS: {
                if (op.isDouble()) {
                    in.setDoubleValue(Math.cos(in.getOperator().getDouble()));
                    in.setAsDouble();
                } else {
                    report_Error("COS parametre double olmalıdır");
                }
            }
                break;
            case Qop_Expr_Fun.EXP: {
                if (op.isDouble()) {
                    in.setDoubleValue(Math.exp(in.getOperator().getDouble()));
                    in.setAsDouble();
                } else if (op.isComplex()) {
                    Complex c = op.getComplex();
```

```
                        double re, im;
                        double EPS = 1.5E-16;
                        // buraya dikkat et
                        re = Math.cos(c.getImaginary());
                        im = Math.sin(c.getImaginary());
                        if (im < EPS) {
                                im = 0.0;
                        }
                        in.setComplex(Complex.valueOf(re, im));
                        in.setAsComplex();
                } else {
                        report_Error("EXP parametre double olmalıdır");
                }
        }
                break;
        case Qop_Expr_Fun.SQRT: {
                if (op.isDouble()) {
                        in.setDoubleValue(Math.sqrt(in.getOperator().getDouble()));
                        in.setAsDouble();
                } else {
                        report_Error("EXP parametre double olmalıdır");
                }
        }
                break;
        case Qop_Expr_Fun.OP_QFT: {
                if (op.isDouble()) {
                        Complex[][] c = QOPComplexMatrixes.createQFT((int) op
                                        .getDouble());
                        in.setComplexMatrix(ComplexMatrix.valueOf(c));
                        in.setAsComplexMatrix();
                } else {
                        report_Error("QFT de parametre int olmalıdır");
                }
        }
                break;
        case Qop_Expr_Fun.OP_PHASE: {
                if (op.isDouble()) {
                        Complex[][] c = QOPComplexMatrixes.createPHASE((int) op
                                        .getDouble());
                        in.setComplexMatrix(ComplexMatrix.valueOf(c));
                        in.setAsComplexMatrix();
                } else {
                        report_Error("PHASE donusumunde parametre int olmalıdır");
                }
        }
                break;
        case Qop_Expr_Fun.OP_ENTANGLE: {
                if (op.isDouble()) {
                        Complex[][] c = QOPComplexMatrixes.createENTANGLE((int) op
                                        .getDouble());
                        in.setComplexMatrix(ComplexMatrix.valueOf(c));
                        in.setAsComplexMatrix();
                } else {
                        report_Error("ENTANGLE de parametre int olmalıdır");
                }
        }
                break;
        case Qop_Expr_Fun.OP_FLIP: {
                if (op.isDouble()) {
                        Complex[][] c = QOPComplexMatrixes.createFLIP((int) op
                                        .getDouble());
                        in.setComplexMatrix(ComplexMatrix.valueOf(c));
                        in.setAsComplexMatrix();
                } else {
                        report_Error("FLIP de parametre int olmalıdır");
                }
        }
                break;
        case Qop_Expr_Fun.OP_R: {
                if (op.isDouble()) {
                        Complex[][] c = QOPComplexMatrixes.createR(op.getDouble());
                        in.setComplexMatrix(ComplexMatrix.valueOf(c));
                        in.setAsComplexMatrix();
                } else {
                        report_Error("R de parametre double olmalıdır");
                }
        }
                break;
```

```
		case Qop_Expr_Fun.OP_ROTX: {
			if (op.isDouble()) {
				Complex[][] c = QOPComplexMatrixes.createRotX(op.getDouble());
				in.setComplexMatrix(ComplexMatrix.valueOf(c));
				in.setAsComplexMatrix();
			} else {
				report_Error("ROTX de parametre double olmalıdır");
			}
		}
			break;
		case Qop_Expr_Fun.OP_ROTY: {
			if (op.isDouble()) {
				Complex[][] c = QOPComplexMatrixes.createRotY(op.getDouble());
				in.setComplexMatrix(ComplexMatrix.valueOf(c));
				in.setAsComplexMatrix();
			} else {
				report_Error("ROTX de parametre double olmalıdır");
			}
		}
			break;
		case Qop_Expr_Fun.OP_ROTZ: {
			if (op.isDouble()) {
				Complex[][] c = QOPComplexMatrixes.createRotX(op.getDouble());
				in.setComplexMatrix(ComplexMatrix.valueOf(c));
				in.setAsComplexMatrix();
			} else {
				report_Error("ROTZ de parametre double olmalıdır");
			}
		}
			break;
		case Qop_Expr_Fun.SUM: {

		}
			break;
		default:
			break;
		}
	}

	@Override
	public void accept(Qop_Expr_Id_Name in) {

	}

	@Override
	public void accept(Qop_Expr_Mtrx_Row in) {
		List<Qop_Expr> cols = in.getAllColumnElements();
		for (Qop_Expr cel : cols) {
			if (cel != null)
				cel.visit(this);
		}
		for (Qop_Expr cel : cols) {
			if (!cel.isDouble() || !cel.isComplex()) {
				report_Error("Sütun elemanları sayı olmalıdır!");
			}
		}

	}

	@Override
	public void accept(Qop_Expr_Mtrx in) {
		List<Qop_Expr_Mtrx_Row> rows = in.getAllMatrixRows();
		for (Qop_Expr_Mtrx_Row row : rows) {
			if (row != null)
				row.visit(this);
		}
		int nocol = -1;
		int norow = rows.size();
		for (Qop_Expr_Mtrx_Row row : rows) {
			if (!row.isDouble()) {
				report_Error("Matris sayilardan olusmalidir!");
			}
		}
		for (Qop_Expr_Mtrx_Row row : rows) {
			if (row.getNoColumns() > nocol) {
				nocol = row.getNoColumns();
			}
		}
```

```
                int i, j;
                Complex[][] c = new Complex[norow][nocol];
                for (i = 0; i < norow; i++) {
                        for (j = 0; j < nocol; j++) {
                                c[i][j] = in.getElement(i, j).getComplex();
                        }
                }
                in.setComplexMatrix(ComplexMatrix.valueOf(c));
                in.setAsComplexMatrix();
        }

        @Override
        public void accept(Qop_Expr_Op in) {
                switch (in.getOPType()) {
                case Qop_Expr_Op.CONST_I: {
                        in.setComplex(Complex.I);
                        in.setAsComplex();
                }
                        break;
                case Qop_Expr_Op.CONST_PI: {
                        in.setDoubleValue(Math.PI);
                        in.setAsDouble();
                }
                        break;
                case Qop_Expr_Op.CONST_QUESTION: {
                        report_Error("Parametrelere atama yapılmalıdır!!!");
                        return;
                }

                case Qop_Expr_Op.OP_CNOT12: {
                        in.setComplexMatrix(ComplexMatrix
                                        .valueOf(QOPComplexMatrixes.OP_CNOT12));
                        in.setAsComplexMatrix();
                }
                        break;
                case Qop_Expr_Op.OP_CNOT21: {
                        in.setComplexMatrix(ComplexMatrix
                                        .valueOf(QOPComplexMatrixes.OP_CNOT21));
                        in.setAsComplexMatrix();
                }
                        break;
                case Qop_Expr_Op.OP_H: {
                        in.setComplexMatrix(ComplexMatrix.valueOf(QOPComplexMatrixes.OP_H));
                        in.setAsComplexMatrix();
                }
                        break;
                case Qop_Expr_Op.OP_V: {
                        in.setComplexMatrix(ComplexMatrix.valueOf(QOPComplexMatrixes.OP_V));
                        in.setAsComplexMatrix();
                }
                        break;
                case Qop_Expr_Op.OP_I: {
                        in.setComplexMatrix(ComplexMatrix.valueOf(QOPComplexMatrixes.OP_I));
                        in.setAsComplexMatrix();
                }
                        break;
                case Qop_Expr_Op.OP_S: {
                        in.setComplexMatrix(ComplexMatrix.valueOf(QOPComplexMatrixes.OP_S));
                        in.setAsComplexMatrix();
                }
                        break;
                case Qop_Expr_Op.OP_SWAP: {
                        in.setComplexMatrix(ComplexMatrix
                                        .valueOf(QOPComplexMatrixes.OP_SWAP));
                        in.setAsComplexMatrix();
                }
                        break;
                case Qop_Expr_Op.OP_T: {
                        in.setComplexMatrix(ComplexMatrix.valueOf(QOPComplexMatrixes.OP_T));
                        in.setAsComplexMatrix();
                }
                        break;
                case Qop_Expr_Op.OP_TOFFOLI: {
                        in.setComplexMatrix(ComplexMatrix
                                        .valueOf(QOPComplexMatrixes.OP_Toffoli));
                        in.setAsComplexMatrix();
                }
                        break;
```

```
		case Qop_Expr_Op.OP_X: {
			in.setComplexMatrix(ComplexMatrix.valueOf(QOPComplexMatrixes.OP_X));
			in.setAsComplexMatrix();
		}
			break;
		case Qop_Expr_Op.OP_Y: {
			in.setComplexMatrix(ComplexMatrix.valueOf(QOPComplexMatrixes.OP_Y));
			in.setAsComplexMatrix();
		}
			break;
		case Qop_Expr_Op.OP_Z: {
			in.setComplexMatrix(ComplexMatrix.valueOf(QOPComplexMatrixes.OP_Z));
			in.setAsComplexMatrix();
		}
			break;
		default:
			break;
		}
	}

	@Override
	public void accept(Qop_Expr_Pa in) {
		Qop_Expr e = in.getParanthesis_expr();
		if (e != null)
			e.visit(this);
		if (e.isComplexMatrix()) {
			in.setComplexMatrix(e.getComplexMatrix());
			in.setAsComplexMatrix();
		} else if (e.isComplex()) {
			in.setComplex(e.getComplex());
			in.setAsComplex();
		} else {
			in.setDoubleValue(e.getDouble());
			in.setAsDouble();
		}
	}

	@Override
	public void accept(Qop_Expr_Two_Op in) {
		Qop_Expr left = in.getLeft_operand();
		Qop_Expr right = in.getRight_operand();
		if (left != null) {
			left.visit(this);
		}
		if (right != null) {
			right.visit(this);
		} else {
			report_Error(" SAĞ OPERAND YOK!!");
		}
		switch (in.getOperator()) {
		case Qop_Expr_Two_Op.DIV: {
			if (left.isComplexMatrix() && right.isComplexMatrix()) {
				Matrix<Complex> c;
				ComplexMatrix r = right.getComplexMatrix();
				ComplexMatrix l = left.getComplexMatrix();
				c = l.divide(r);
				in.setComplexMatrix(ComplexMatrix.valueOf(c));
				in.setAsComplexMatrix();
			} else if (!left.isComplexMatrix() && right.isComplexMatrix()) {
				Matrix<Complex> c;
				ComplexMatrix r = right.getComplexMatrix();
				ComplexMatrix l;
				if (left instanceof Qop_Com_Num) {
					l = left.getComplexMatrix();
				} else {
					l = Qop_Helper.getAsMatrix(r, left.getDouble());
				}
				c = l.divide(r);
				in.setComplexMatrix(ComplexMatrix.valueOf(c));
				in.setAsComplexMatrix();
			} else if (left.isComplexMatrix() && !right.isComplexMatrix()) {
				Matrix<Complex> c;
				ComplexMatrix l = left.getComplexMatrix();
				ComplexMatrix r;
				if (right instanceof Qop_Com_Num) {
					r = right.getComplexMatrix();
				} else {
					r = Qop_Helper.getAsMatrix(l, right.getDouble());
```

```
                        }
                        c = l.divide(r);
                        in.setComplexMatrix(ComplexMatrix.valueOf(c));
                        in.setAsComplexMatrix();

                } else {
                        if ((right.isComplex()) && (left.isComplex())) {
                                Complex r = ((Qop_Com_Num) right).getComplex();
                                Complex l = ((Qop_Com_Num) left).getComplex();
                                in.setComplex(l.divide(r));
                                in.setAsComplex();
                        } else if (!(right.isComplex()) && (left.isComplex())) {
                                Complex r = Complex.valueOf(right.getDouble(), 0.0);
                                Complex l = ((Qop_Com_Num) left).getComplex();
                                in.setComplex(l.divide(r));
                                in.setAsComplex();
                        } else if ((right.isComplex()) && !(left.isComplex())) {
                                Complex l = Complex.valueOf(right.getDouble(), 0.0);
                                Complex r = ((Qop_Com_Num) right).getComplex();
                                in.setComplex(l.divide(r));
                                in.setAsComplex();
                        } else {
                                double l = left.getDouble();
                                double r = right.getDouble();
                                in.setDoubleValue(l / r);
                                in.setAsDouble();
                        }
                }
        }
                break;
        case Qop_Expr_Two_Op.MINUS: {
                if (left.isComplexMatrix() && right.isComplexMatrix()) {
                        Matrix<Complex> c;
                        ComplexMatrix r = right.getComplexMatrix();
                        ComplexMatrix l = left.getComplexMatrix();
                        c = l.minus(r);
                        in.setComplexMatrix(ComplexMatrix.valueOf(c));
                        in.setAsComplexMatrix();
                } else if (!left.isComplexMatrix() && right.isComplexMatrix()) {
                        Matrix<Complex> c;
                        ComplexMatrix r = right.getComplexMatrix();
                        ComplexMatrix l = Qop_Helper.getAsMatrix(r, left);
                        c = l.minus(r);
                        in.setComplexMatrix(ComplexMatrix.valueOf(c));
                        in.setAsComplexMatrix();
                } else if (left.isComplexMatrix() && !right.isComplexMatrix()) {
                        Matrix<Complex> c;
                        ComplexMatrix l = left.getComplexMatrix();
                        ComplexMatrix r = Qop_Helper.getAsMatrix(l, right);
                        c = l.minus(r);
                        in.setComplexMatrix(ComplexMatrix.valueOf(c));
                        in.setAsComplexMatrix();
                } else {
                        if ((right.isComplex()) && (left.isComplex())) {
                                Complex r = ((Qop_Com_Num) right).getComplex();
                                Complex l = ((Qop_Com_Num) left).getComplex();
                                in.setComplex(l.minus(r));
                                in.setAsComplex();
                        } else if (!(right.isComplex()) && (left.isComplex())) {
                                Complex r = Complex.valueOf(right.getDouble(), 0.0);
                                Complex l = ((Qop_Com_Num) left).getComplex();
                                in.setComplex(l.minus(r));
                                in.setAsComplex();
                        } else if ((right.isComplex()) && !(left.isComplex())) {
                                Complex l = Complex.valueOf(right.getDouble(), 0.0);
                                Complex r = ((Qop_Com_Num) right).getComplex();
                                in.setComplex(l.minus(r));
                                in.setAsComplex();
                        } else {
                                double l = left.getDouble();
                                double r = right.getDouble();
                                in.setDoubleValue(l - r);
                                in.setAsDouble();
                        }

                }
        }
                break;
```

```
case Qop_Expr_Two_Op.MULT: {
        if (left.isComplexMatrix() && right.isComplexMatrix()) {
                Matrix<Complex> c;
                ComplexMatrix r = right.getComplexMatrix();
                ComplexMatrix l = left.getComplexMatrix();
                c = l.times(r);
                in.setComplexMatrix(ComplexMatrix.valueOf(c));
                in.setAsComplexMatrix();
        } else if (!left.isComplexMatrix() && right.isComplexMatrix()) {
                Matrix<Complex> c;
                ComplexMatrix r = right.getComplexMatrix();
                Complex l = Qop_Helper.getComplex(left);
                c = r.times(l);
                in.setComplexMatrix(ComplexMatrix.valueOf(c));
                in.setAsComplexMatrix();
        } else if (left.isComplexMatrix() && !right.isComplexMatrix()) {
                Matrix<Complex> c;
                ComplexMatrix l = left.getComplexMatrix();
                Complex r = Qop_Helper.getComplex(right);
                c = l.times(r);
                in.setComplexMatrix(ComplexMatrix.valueOf(c));
                in.setAsComplexMatrix();
        } else {
                if ((right.isComplex()) && (left.isComplex())) {
                        Complex r = ((Qop_Com_Num) right).getComplex();
                        Complex l = ((Qop_Com_Num) left).getComplex();
                        in.setComplex(l.times(r));
                        in.setAsComplex();
                } else if (!(right.isComplex()) && (left.isComplex())) {
                        Complex r = Complex.valueOf(right.getDouble(), 0.0);
                        Complex l = ((Qop_Com_Num) left).getComplex();
                        in.setComplex(l.times(r));
                        in.setAsComplex();
                } else if ((right.isComplex()) && !(left.isComplex())) {
                        Complex l = Complex.valueOf(left.getDouble(), 0.0);
                        Complex r = right.getComplex();
                        in.setComplex(l.times(r));
                        in.setAsComplex();
                } else {
                        double l = left.getDouble();
                        double r = right.getDouble();
                        in.setDoubleValue(l * r);
                        in.setAsDouble();
                }
        }
}
        break;
case Qop_Expr_Two_Op.PLUS: {
        if (left.isComplexMatrix() && right.isComplexMatrix()) {
                Matrix<Complex> c;
                ComplexMatrix r = right.getComplexMatrix();
                ComplexMatrix l = left.getComplexMatrix();
                c = l.plus(r);
                in.setComplexMatrix(ComplexMatrix.valueOf(c));
                in.setAsComplexMatrix();
        } else if (!left.isComplexMatrix() && right.isComplexMatrix()) {
                Matrix<Complex> c;
                ComplexMatrix r = right.getComplexMatrix();
                ComplexMatrix l = Qop_Helper.getAsMatrix(r, left);
                c = l.plus(r);
                in.setComplexMatrix(ComplexMatrix.valueOf(c));
                in.setAsComplexMatrix();
        } else if (left.isComplexMatrix() && !right.isComplexMatrix()) {
                Matrix<Complex> c;
                ComplexMatrix l = left.getComplexMatrix();
                ComplexMatrix r = Qop_Helper.getAsMatrix(l, left);
                c = l.plus(r);
                in.setComplexMatrix(ComplexMatrix.valueOf(c));
                in.setAsComplexMatrix();
        } else {
                if ((right.isComplex()) && (left.isComplex())) {
                        Complex r = ((Qop_Com_Num) right).getComplex();
                        Complex l = ((Qop_Com_Num) left).getComplex();
                        in.setComplex(l.plus(r));
                        in.setAsComplex();
                } else if (!(right.isComplex()) && (left.isComplex())) {
                        Complex r = Complex.valueOf(right.getDouble(), 0.0);
                        Complex l = ((Qop_Com_Num) left).getComplex();
```

```
                                        in.setComplex(l.plus(r));
                                        in.setAsComplex();
                                } else if ((right.isComplex()) && !(left.isComplex())) {
                                        Complex l = Complex.valueOf(right.getDouble(), 0.0);
                                        Complex r = ((Qop_Com_Num) right).getComplex();
                                        in.setComplex(l.plus(r));
                                        in.setAsComplex();
                                } else {
                                        double l = left.getDouble();
                                        double r = right.getDouble();
                                        in.setDoubleValue(l + r);
                                        in.setAsDouble();
                                }

                        }
                }
                        break;
                case Qop_Expr_Two_Op.POWER: {
                        if (left.isComplexMatrix() && right.isComplexMatrix()) {
                                report_Error(" ^ operatörünün sağ tarafı int olmalı!");
                                return;
                        } else if (!left.isComplexMatrix() && right.isComplexMatrix()) {
                                report_Error(" ^ operatörünün sağ tarafı int olmalı!");
                                return;
                        } else if (left.isComplexMatrix() && !right.isComplexMatrix()) {
                                Matrix<Complex> c;
                                ComplexMatrix l = left.getComplexMatrix();
                                int r = (int) right.getDouble();
                                c = l.pow(r);
                                in.setComplexMatrix(ComplexMatrix.valueOf(c));
                                in.setAsComplexMatrix();
                        } else {
                                if ((right.isComplex()) && (left.isComplex())) {
                                        Complex r = ((Qop_Com_Num) right).getComplex();
                                        Complex l = ((Qop_Com_Num) left).getComplex();
                                        in.setComplex(l.pow(r));
                                        in.setAsComplex();
                                } else if (!(right.isComplex()) && (left.isComplex())) {
                                        Complex r = Complex.valueOf(right.getDouble(), 0.0);
                                        Complex l = ((Qop_Com_Num) left).getComplex();
                                        in.setComplex(l.pow(r));
                                        in.setAsComplex();
                                } else if ((right.isComplex()) && !(left.isComplex())) {
                                        Complex l = Complex.valueOf(right.getDouble(), 0.0);
                                        Complex r = ((Qop_Com_Num) right).getComplex();
                                        in.setComplex(l.pow(r));
                                        in.setAsComplex();
                                } else {
                                        double l = left.getDouble();
                                        double r = right.getDouble();
                                        in.setDoubleValue(Math.pow(l, r));
                                        in.setAsDouble();
                                }
                        }
                }
                        break;
                case Qop_Expr_Two_Op.TENSOR: {
                        if (left.isComplexMatrix() && right.isComplexMatrix()) {
                                Matrix<Complex> c;
                                ComplexMatrix r = right.getComplexMatrix();
                                ComplexMatrix l = left.getComplexMatrix();
                                c = l.tensor(r);
                                in.setComplexMatrix(ComplexMatrix.valueOf(c));
                                in.setAsComplexMatrix();
                        } else if (!left.isComplexMatrix() && right.isComplexMatrix()) {
                                Matrix<Complex> c;
                                ComplexMatrix r = right.getComplexMatrix();
                                ComplexMatrix l = left.getComplexMatrix();
                                c = l.tensor(r);
                                in.setComplexMatrix(ComplexMatrix.valueOf(c));
                                in.setAsComplexMatrix();
                        } else if (left.isComplexMatrix() && !right.isComplexMatrix()) {
                                Matrix<Complex> c;
                                ComplexMatrix l = left.getComplexMatrix();
                                ComplexMatrix r = right.getComplexMatrix();
                                c = l.tensor(r);
                                in.setComplexMatrix(ComplexMatrix.valueOf(c));
                                in.setAsComplexMatrix();
```

```
                } else {
                        System.out
                                        .println("tensor carpimi matrisler arasi yapilabilir!!");
                        return;
                }
        }
                break;
        case Qop_Expr_Two_Op.TENSOR_POWER: {
                if (left.isComplexMatrix() && right.isComplexMatrix()) {
                        System.out
                                        .println("iki matris arasında tensor power uygulanamaz!");
                        return;
                } else if (!left.isComplexMatrix() && right.isComplexMatrix()) {
                        System.out
                                        .println("iki matris arasında tensor power uygulanamaz!");
                        return;
                } else if (left.isComplexMatrix() && !right.isComplexMatrix()) {
                        ComplexMatrix l = left.getComplexMatrix();
                        ComplexMatrix total = l;
                        if (right.isDouble()) {
                                int kackez = (int) right.getDouble();
                                int i;
                                for (i = 0; i < kackez - 1; i++) {
                                        total = total.tensor(l);
                                }
                                in.setComplexMatrix(total);
                                in.setAsComplexMatrix();
                        } else {
                                report_Error("sağ taraf complex olmamalı!");
                                return;
                        }
                } else {
                        System.out
                                        .println("iki sayı arasında tensor power uygulanamaz!");
                        return;
                }
        }
                break;
        case Qop_Expr_Two_Op.UNARY_MINUS: {
                if (right.isComplexMatrix()) {
                        Matrix<Complex> c;
                        ComplexMatrix r = right.getComplexMatrix();
                        ComplexMatrix l = Qop_Helper.getAsMatrix(r, -1.0);
                        c = r.times(l);
                        in.setComplexMatrix(ComplexMatrix.valueOf(c));
                        in.setAsComplexMatrix();
                } else if (!right.isComplexMatrix()) {
                        if (right.isComplex()) {
                                Complex neg = Complex.valueOf(-1.0, 0.0);
                                Complex r = right.getComplex();
                                in.setComplex(r.times(neg));
                                in.setAsComplex();
                        } else {
                                double r = right.getDouble();
                                in.setDoubleValue(-1.0 * r);
                                in.setAsDouble();
                        }
                }
        }
                break;
        case Qop_Expr_Two_Op.UNARY_NOT: {
                if (right.isComplexMatrix()) {
                        Matrix<Complex> c;
                        ComplexMatrix r = right.getComplexMatrix();
                        c = r.adjoint();
                        in.setComplexMatrix(ComplexMatrix.valueOf(c));
                        in.setAsComplexMatrix();
                } else if (!right.isComplexMatrix()) {
                        if (right.isComplex()) {
                                Complex r = right.getComplex();
                                in.setComplex(r.conjugate());
                                in.setAsComplex();
                        } else {
                                double r = right.getDouble();
                                in.setDoubleValue(r);
                                in.setAsDouble();
                        }
                }
```

```
                        }
                                break;
                        default:
                                break;
                        }
                }

                @Override
                public void accept(Qop_Op_Unit in) {
                        Qop_Expr e = in.getS();
                        if (e != null)
                                e.visit(this);
                        if (e.isComplexMatrix()) {
                                in.setComplexMatrix(e.getComplexMatrix());
                                in.setAsComplexMatrix();
                        } else if (e.isComplex()) {
                                in.setComplex(e.getComplex());
                                in.setAsComplex();
                        } else {
                                in.setDoubleValue(e.getDouble());
                                in.setAsDouble();
                        }
                }

                @Override
                public void accept(Qop_Expr in) {

                }
}
```

EK 4- QDil Temel Sınıflarının Kodları

```
package qdil.lang;

use qdil.lang.Object;
use qdil.lang.Class;
use qdil.lang.boolean;
use qdil.lang.string;
use qdil.lang.int;

class Object{
  + const extdef getClass():Class;
  # extdef clone():Object;
  + extdef eql(Object o):boolean;
  + extdef hash():int;
  + def to_str():string;
}
```

```
package qdil.compiler.types;

import qdil.compiler.QdClass;

public class Qd_Object {
        private QdClass clazz;

        public Qd_Object(QdClass qdcl){
                this.clazz  = qdcl;
        }
        public Qd_Object(){

        }
        public QdClass getQdClass(){
                return clazz;
        }
        public void setQdClass(QdClass klz){
                clazz = klz;
        }
        public Qd_Class Object_getClass(){
                return null;
        }
        public Qd_Object Object_clone(){
                return null;
        }
        public Qd_boolean Object_eql(Qd_Object o){
                return null;
        }
        public Qd_int Object_hash(){
```

```
                return null;
        }
}
```

```
package qdil.lang;

use qdil.lang.Object;
use qdil.lang.Number;
use qdil.lang.byte;
use qdil.lang.short;
use qdil.lang.int;
use qdil.lang.long;
use qdil.lang.float;
use qdil.lang.double;
use qdil.lang.complex;
use qdil.lang.binary;

abstract class Number:Object{
        + extdef to_str():string;
        + abstract def to_byte():byte;
        + abstract def to_short():short;
        + abstract def to_int():int;
        + abstract def to_long():long;
        + abstract def to_float():float;
        + abstract def to_double():double;
        + abstract def to_complex():complex;
        + abstract def to_binary():binary;

        + abstract extdef @+(Number b):Number;
        + abstract extdef @-(Number b):Number;
        + abstract extdef @/(Number b):Number;
        + abstract extdef @*(Number b):Number;
        + abstract extdef @%(Number b):Number;
        + abstract extdef @^(Number b):Number;
        + abstract extdef @&(Number b):Number;
        + abstract extdef @|(Number b):Number;
        + abstract extdef @<<(Number b):Number;
        + abstract extdef @>>(Number b):Number;
        + abstract extdef @>>>(Number b):Number;
        + abstract extdef @~(Number b):Number;
        + abstract extdef @xor(Number b):Number;
        + abstract extdef @?=(Number b):Number;
        + abstract extdef @!=(Number b):Number;
}
```

```
package qdil.lang;

use qdil.lang.Object;
use qdil.lang.string;
use qdil.lang.boolean;
use qdil.lang.byte;
use qdil.lang.exceptions.ConvertException;
use qdil.lang.exceptions.NumberConvertException;

const class byte:Number{

        byte byte.min=-128 {+get;-set;}
        byte byte.max=127 {+get;-set;}

        + extdef to_byte():byte;
        + extdef to_short():short;
        + extdef to_int():int;
        + extdef to_long():long;
        + extdef to_float():float;
        + extdef to_double():double;
        + extdef to_complex():complex;
        + extdef to_binary():binary;

        + extdef str_to_byte(string str):byte raises ConvertException;

        + extdef eql(Object o):boolean;

}
```

```
package qdil.compiler.types;

import org.jscience.mathematics.number.Complex;

import qdil.exceptions.Qd_Exception;

public class Qd_byte extends Qd_Number{
```

```
private byte value;

public Qd_byte(byte value) {
	this.value = value;
}
public byte getValue(){
	return value;
}
private final static Qd_byte byte_byte_min = new Qd_byte((byte) -128);
private final static Qd_byte byte_byte_max = new Qd_byte((byte) 127);

public Qd_byte get_byte_byte_min() {
	return byte_byte_min;
}

public Qd_byte get_byte_byte_max() {
	return byte_byte_max;
}

public Qd_byte byte_byte_to_byte() {
	return this;
}

public Qd_short byte_byte_to_short(){
	short s = (short) value;
	return new Qd_short(s);
}

public Qd_int byte_byte_to_int(){
	int i = (int) value;
	return new Qd_int(i);
}

public Qd_long byte_byte_to_long() {
	long l = (long) value;
	return new Qd_long(l);
}

public Qd_float byte_byte_to_float() {
	float f = (float) value;
	return new Qd_float(f);
}

public Qd_double byte_byte_to_double(){
	double d = (double) value;
	return new Qd_double(d);
}

public Qd_complex byte_byte_to_complex(){
	double re = (double) value;
	Complex c = Complex.valueOf(re, 0.0);
	return new Qd_complex(c);
}

public Qd_binary byte_byte_to_binary(){
	int ii = (int) value;
	String str = Integer.toBinaryString(ii);
	String sub = str.substring(str.length()-8, str.length());
	Qd_boolean[] bits = new Qd_boolean[sub.length()];
	int i;
	for (i = 0; i < sub.length(); i++) {
		if (sub.charAt(i) == '0') {
			bits[i] = Qd_boolean.FALSE;
		} else {
			bits[i] = Qd_boolean.TRUE;
		}
	}
	return new Qd_binary(bits);
}

public Qd_byte byte_str_to_byte(Qd_string str) throws Qd_Exception{
	byte val = 0;
	String s = str.getValue();
	try {
		val = (byte)Integer.parseInt(s);
	} catch (NumberFormatException e) {
		throw new Qd_Exception("ConvertException");
	}
```

```
                return new Qd_byte(val);
        }

        public Qd_boolean byte_op_eq(Qd_Object o) {
                if (o instanceof Qd_byte) {
                        if (((Qd_byte) o).value == this.value) {
                                return Qd_boolean.TRUE;
                        }
                }
                return Qd_boolean.FALSE;
        }

}
```

```
package qdil.lang;

use qdil.lang.Object;
use qdil.lang.string;
use qdil.lang.boolean;
use qdil.lang.short;
use qdil.lang.exceptions.ConvertException;
use qdil.lang.exceptions.NumberConvertException;

const class short:Number{

        short short.min=-32768 {+get;-set;}
        short short.max=32767 {+get;-set;}

        + extdef to_byte():byte;
        + extdef to_short():short;
        + extdef to_int():int;
        + extdef to_long():long;
        + extdef to_float():float;
        + extdef to_double():double;
        + extdef to_complex():complex;
        + extdef to_binary():binary;

        + extdef str_to_short(string str):short raises ConvertException;

        + extdef eql(Object o):boolean;
}
```

```
package qdil.compiler.types;

import org.jscience.mathematics.number.Complex;

import qdil.exceptions.Qd_Exception;

public class Qd_short extends Qd_Number{
        private short value;
        public Qd_short(short value){
                this.value = value;
        }
        private final static Qd_short short_short_min = new Qd_short((short) -32768);
        private final static Qd_short short_short_max = new Qd_short((short) 32767);

        public Qd_short get_short_short_min() {
                return short_short_min;
        }

        public Qd_short get_short_short_max() {
                return short_short_max;
        }

        public Qd_byte short_short_to_byte() {
                byte b = (byte) value;
                return new Qd_byte(b);
        }

        public Qd_short short_short_to_short(){
                return this;
        }

        public Qd_int short_short_to_int(){
                int i = (int) value;
                return new Qd_int(i);
        }

        public Qd_long short_short_to_long() {
                long l = (long) value;
```

```
                return new Qd_long(l);
        }

        public Qd_float short_short_to_float() {
                float f = (float) value;
                return new Qd_float(f);
        }

        public Qd_double short_short_to_double(){
                double d = (double) value;
                return new Qd_double(d);
        }

        public Qd_complex short_short_to_complex(){
                double re = (double) value;
                Complex c = Complex.valueOf(re, 0.0);
                return new Qd_complex(c);
        }

        public Qd_binary short_short_to_binary(){
                int ii = (int) value;
                String str = Integer.toBinaryString(ii);
                String sub = str.substring(str.length()-16, str.length());
                Qd_boolean[] bits = new Qd_boolean[sub.length()];
                int i;
                for (i = 0; i < sub.length(); i++) {
                        if (sub.charAt(i) == '0') {
                                bits[i] = Qd_boolean.FALSE;
                        } else {
                                bits[i] = Qd_boolean.TRUE;
                        }
                }
                return new Qd_binary(bits);
        }

        public Qd_short short_str_to_short(Qd_string str) throws Qd_Exception{
                short val = 0;
                String s = str.getValue();
                try {
                        val = (short)Integer.parseInt(s);
                } catch (NumberFormatException e) {
                        throw new Qd_Exception("ConvertException");
                }
                return new Qd_short(val);
        }

        public Qd_boolean short_op_eq(Qd_Object o) {
                if (o instanceof Qd_short) {
                        if (((Qd_short) o).value == this.value) {
                                return Qd_boolean.TRUE;
                        }
                }
                return Qd_boolean.FALSE;
        }
}
```

```
package qdil.lang;

use java.lang.Object;
use java.lang.string;
use java.lang.boolean;
use java.lang.double;
use qdil.lang.exceptions.NumberConvertException;

const class complex:Number{

                complex complex.zero = new complex(0.0,0.0){+get;-set;}
                complex complex.one = new complex(1.0,0.0){+get;-set;}
                complex complex.I = new complex(0.0,1.0){+get;-set;}
                double real {+get;+set;}
                double imag {+get;+set;}

                + extdef complex(double re,double im);

                + extdef to_byte():byte;
                + extdef to_short():short;
                + extdef to_int():int;
                + extdef to_long():long;
                + extdef to_float():float;
```

```
                + extdef to_double():double;
                + extdef to_complex():complex;
                + extdef to_binary():binary;

                + extdef complex.getComplex(double re,double im);
                + extdef complex.isUndefined(complex value):boolean;
                + extdef changeSigns();
                + extdef inverse();
                + extdef conjugate();
                + extdef magnitude():double;
                + extdef argument():double;
                + extdef sqrt():complex;
                + extdef exp():complex;
                + extdef log():complex;
                + extdef pow(complex val):complex;
                + extdef eql(complex val):boolean;
                + extdef eql(complex val,tolerance tol):boolean;

                + extdef @+(complex val):complex;
                + extdef @-(complex val):complex;
                + extdef @*(double val):complex;
                + extdef @*(complex val):complex;
                + extdef @/(double val):complex;
                + extdef @/(complex val):complex;
                + extdef @?=(complex val):boolean;

                + extdef eql(Object o):boolean;
}
```

```
package qdil.compiler.types;

import org.jscience.mathematics.number.Complex;

import qdil.Qd_Bool_Lit;
import qdil.exceptions.Qd_Exception;

public class Qd_complex extends Qd_Number{
        private Complex value;

        public Qd_complex(Complex value){
                this.value = value;
        }
        public Complex getValue(){
                return value;
        }
        public void setValue(Complex value){
                this.value = value;
        }
        private final static Qd_complex complex_complex_min = new
Qd_complex(Complex.valueOf(Double.MIN_VALUE,Double.MIN_VALUE));
        private final static Qd_complex complex_complex_max = new
Qd_complex(Complex.valueOf(Double.MAX_VALUE,Double.MAX_VALUE));

        //constructor
        public void complex_complex(Qd_double re,Qd_double im){
                Complex c = Complex.valueOf(re.getValue(), im.getValue());
                this.value = c;
        }
        public Qd_complex get_complex_complex_min(){
          return complex_complex_min;
        }
        public Qd_complex get_complex_complex_max(){
                return complex_complex_max;
        }

        public Qd_double get_complex_real(){
                return new Qd_double(value.getReal());
        }
        public Qd_double get_complex_imag(){
                return new Qd_double(value.getImaginary());
        }
        public void set_complex_imag(Qd_double im){
                Complex c = Complex.valueOf(value.getReal(), im.getValue());
                this.value = c;
        }
        public void set_complex_real(Qd_double re){
                Complex c = Complex.valueOf(re.getValue(),value.getImaginary());
                this.value = c;
        }
```

```
public Qd_byte complex_complex_to_byte() throws Qd_Exception{
		byte b = (byte)value.byteValue();
		return new Qd_byte(b);
}
public Qd_short complex_complex_to_short() throws Qd_Exception{
		short s = (short)value.shortValue();
		return new Qd_short(s);
}
public Qd_int complex_complex_to_int() throws Qd_Exception{
		int i = (int) value.intValue();
		return new Qd_int(i);
}
public Qd_long complex_complex_to_long() throws Qd_Exception{
		long l = (long) value.longValue();
		return new Qd_long(l);
}
public Qd_float complex_complex_to_float() throws Qd_Exception{
		float f = (float) value.floatValue();
		return new Qd_float(f);
}
public Qd_double complex_complex_to_double() throws Qd_Exception{
		return new Qd_double(value.doubleValue());
}
public Qd_complex complex_complex_to_complex() throws Qd_Exception{

		return this;
}
public Qd_binary complex_complex_to_binary() throws Qd_Exception{
		long re =Double.doubleToRawLongBits(value.getReal());
		long im = Double.doubleToRawLongBits(value.getImaginary());
		String str_re = Long.toBinaryString(re);
		String str_im = Long.toBinaryString(im);
		String str = str_re.concat(str_im);
		Qd_boolean[] bits = new Qd_boolean[str.length()];
		int i;
		for (i=0;i<str.length();i++){
			if (str.charAt(i)=='0'){
				bits[i] = Qd_boolean.FALSE;
			}else{
				bits[i] = Qd_boolean.TRUE;
			}
		}
		return new Qd_binary(bits);
}
// a +i b ; a -i b
public Qd_complex complex_str_to_complex(Qd_string str) throws Qd_Exception{
		Complex c=null;
		String s = str.getValue();
		int i1=-1,i2=-1;
		i1= s.indexOf('+');
		i2 = s.indexOf('-');
		if (i2!=-1) {
			i1= i2;
		}
		i2 = s.indexOf('i');
		String s1=null,s2=null;
		Double re = 0.0,im = 0.0;
		s1 = s.substring(0, i1-1);
		s2 = s.substring(i2+1,s.length());
		try {
			re = Double.parseDouble(s1);
			im = Double.parseDouble(s2);

		} catch (NumberFormatException e) {
			throw new Qd_Exception("ConvertException");
		}
		return new Qd_complex(Complex.valueOf(re, im));
}
public Qd_complex complex_getComplex(Qd_double re,Qd_double im){
		Complex c = Complex.valueOf(re.getValue(),im.getValue());
		return new Qd_complex(c);
}
public Qd_boolean complex_isUndefined(Complex value){
		if (Double.isNaN(value.getReal()) || Double.isNaN(value.getImaginary())){
			return Qd_boolean.TRUE;
		}
		return Qd_boolean.FALSE;
```

```
	}
	public void complex_changeSigns(){
		value = value.inverse();
	}
	public void complex_inverse(){
		value = value.inverse();
	}
	public void complex_conjugate(){
		value = value.conjugate();
	}
	public Qd_double complex_magnitude(){
		return new Qd_double(value.magnitude());
	}
	public Qd_double complex_argument(){
		return new Qd_double(value.argument());
	}
	public void complex_sqrt(){
		value = value.sqrt();
	}
	public void complex_exp(){
		value = value.exp();
	}
	public void complex_log(){
		value = value.log();
	}
	public void complex_pow(Qd_complex val){
		value = value.pow(val.getValue());
	}
	public Qd_boolean complex_eql(Qd_complex val){
		if (value.equals(val.getValue())){
			return Qd_boolean.TRUE;
		}
		return Qd_boolean.FALSE;
	}
	public Qd_boolean complex_eql(Qd_complex val,Qd_double tol){
		if (value.equals(val.getValue(), tol.getValue())){
			return Qd_boolean.TRUE;
		}
		return Qd_boolean.FALSE;
	}

	public Qd_complex complex_op_add(Qd_complex val){
		Complex c = value.plus(val.getValue());
		return new Qd_complex(c);
	}
	public Qd_complex complex_op_minus(Qd_complex val){
		Complex c = value.minus(val.getValue());
		return new Qd_complex(c);
	}
	public Qd_complex complex_op_multiply(Qd_complex val){
		Complex c = value.times(val.getValue());
		return new Qd_complex(c);
	}
	public Qd_complex complex_op_divide(Qd_complex val){
		Complex c = value.divide(val.getValue());
		return new Qd_complex(c);
	}
	public Qd_boolean complex_op_eq(Qd_Object o){
		if (o instanceof Qd_complex){
			if (((Qd_complex)o).value.equals(this.value)){
				return Qd_boolean.TRUE;
			}
		}
		return Qd_boolean.FALSE;
	}
}
```

```
package qdil.lang;

use qdil.lang.Number;
use qdil.lang.boolean;
use qdil.lang.binary;
use qdil.lang.byte;
use qdil.lang.short;
use qdil.lang.int;
use qdil.lang.long;
use qdil.lang.float;
use qdil.lang.double;
use qdil.lang.complex;
```

```
use qdil.lang.exceptions.NumberConvertException;
use qdil.lang.exceptions.IndexException;

const class binary:Number{

	boolean[] value{-get;-set;}

	+ extdef binary(int n);


	+ extdef to_byte():byte;
	+ extdef to_short():short;
	+ extdef to_int():int;
	+ extdef to_long():long;
	+ extdef to_float():float;
	+ extdef to_double():double;
	+ extdef to_complex():complex;
	+ extdef to_binary():binary;

	+ extdef @+(binary):binary; //mod 2 ye gore
	+ extdef @[](int i):boolean raises IndexException;
	+ extdef @and(binary val):binary;
	+ extdef @or(binary val):binary;
	+ extdef @!():binary;
   + extdef @xor(binary):binary;

   + extdef eql(Object o):boolean;
}
```

```
package qdil.compiler.types;

import org.jscience.mathematics.number.Complex;

import qdil.exceptions.Qd_Exception;

public class Qd_binary extends Qd_Number{
	private int number_of_bits;
	private Qd_boolean[] bits;
	public Qd_binary(int nob) {
		this.number_of_bits = nob;
		bits = new Qd_boolean[nob];
	}
	public Qd_binary(Qd_boolean[] bits){
		this.bits = bits;
		this.number_of_bits = bits.length;
	}
	public int getNOBS(){
		return number_of_bits;
	}
	public Qd_byte binary_binary_to_byte() {
		int i=0;
		byte num = 0;
		int n=0;
		if (number_of_bits>=8){
			for (i=0;i<8;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}else{
			for (i=0;i<number_of_bits;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}
		num=(byte)n;
		return new Qd_byte(num);
	}

	public Qd_short binary_binary_to_short(){
		int i=0;
		short num = 0;
		int n=0;
		if (number_of_bits>=16){
			for (i=0;i<16;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}else{
			for (i=0;i<number_of_bits;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}
```

```
		num=(short)n;
		return new Qd_short(num);
	}

	public Qd_int binary_binary_to_int(){
		int i=0;
		int n=0;
		if (number_of_bits>=32){
			for (i=0;i<32;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}else{
			for (i=0;i<number_of_bits;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}
		return new Qd_int(n);
	}

	public Qd_long binary_binary_to_long() {
		int i=0;
		long n=0;
		if (number_of_bits>=64){
			for (i=0;i<64;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}else{
			for (i=0;i<number_of_bits;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}
		return new Qd_long(n);
	}

	public Qd_float binary_binary_to_float() {
		int i=0;
		int n=0;
		float f=0;
		if (number_of_bits>=32){
			for (i=0;i<32;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}else{
			for (i=0;i<number_of_bits;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}
		f = Float.intBitsToFloat(n);
		return new Qd_float(f);
	}

	public Qd_double binary_binary_to_double(){
		int i=0;
		long n=0;
		double d=0;
		if (number_of_bits>=32){
			for (i=0;i<32;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}else{
			for (i=0;i<number_of_bits;i++){
				n = n |  bits[i].getNum()<<(number_of_bits-i-1);
			}
		}
		d = Double.longBitsToDouble(n);
		return new Qd_double(d);
	}

	public Qd_complex binary_binary_to_complex(){
		int i=0;
		long n=0;
		double d=0;
		if (number_of_bits>=32){
			for (i=0;i<32;i++){
				n = n | bits[i].getNum()<<(number_of_bits-i-1);
			}
		}else{
			for (i=0;i<number_of_bits;i++){
```

```
                                        n = n |  bits[i].getNum()<<(number_of_bits-i-1);
                                }
                        }
                        d = Double.longBitsToDouble(n);
                   Complex c = Complex.valueOf(d, 0.0);
                        return new Qd_complex(c);
                }

                public Qd_binary binary_binary_to_binary(){
                        return this;
                }
                public static boolean isBinary(String str){
                        char[] kontrol = {'1','0'};
                        boolean res = true;
                        int i;
                        for (i=0;i<str.length();i++){
                                if (str.charAt(i)!=kontrol[0] || str.charAt(i)!=kontrol[1]){
                                        res = false;break;
                                }
                        }
                        return res;
                }
                public Qd_binary binary_str_to_binary(Qd_string str) throws Qd_Exception{
                        int i=0;
                        String val = str.getValue();
                        if (isBinary(val)==false){
                                throw new Qd_Exception("ConvertException");
                        }
                        Qd_boolean[] bits = new Qd_boolean[val.length()];
                        for (i=0;i<val.length();i++){
                                if (val.charAt(i)=='1'){
                                        bits[i] = Qd_boolean.TRUE;
                                }else{
                                        bits[i] = Qd_boolean.FALSE;
                                }
                        }
                        return new Qd_binary(bits);
                }

                public Qd_boolean binary_eql(Qd_Object o) {
                        if (o instanceof Qd_binary) {
                                Qd_binary bin = (Qd_binary)o;
                                int i=0;
                                if (bin.getNOBS()!=this.getNOBS()){ return Qd_boolean.FALSE;}
                                for (i=0;i<this.number_of_bits;i++){
                                        if (!bin.bits[i].equals(this.bits[i])){ return Qd_boolean.FALSE;}
                                }
                        }
                        return Qd_boolean.TRUE;
                }
                //@+   mod 2 ye gore toplama
                public Qd_binary binary_op_add(Qd_binary val){
                        int i=0;
                        int siz = Math.min(bits.length, val.getNOBS());
                        for (i=0;i<siz;i++){
                                bits[i] = bits[i].boolean_op_xor(val.bits[i]);
                        }
                        return this;
                }
                //@[]
                public Qd_boolean binary_op_index(Qd_int i) throws Qd_Exception{
                        int ii = i.getValue();
                        if (ii<0 ||ii>=this.getNOBS()) {
                                throw new Qd_Exception("IndexException");
                        }
                        return bits[ii];
                }
                //@and
                public Qd_binary binary_op_and(Qd_binary val){
                        int i=0;
                        int siz = Math.min(bits.length, val.getNOBS());
                        for (i=0;i<siz;i++){
                                bits[i] = bits[i].boolean_op_and(val.bits[i]);
                        }
                        return this;
                }
                //@or
                public Qd_binary binary_op_or(Qd_binary val){
```

```
                int i=0;
                int siz = Math.min(bits.length, val.getNOBS());
                for (i=0;i<siz;i++){
                        bits[i] = bits[i].boolean_op_or(val.bits[i]);
                }
                return this;
        }
        //@or
        public Qd_binary binary_op_not(Qd_binary val){
                int i=0;
                int siz = Math.min(bits.length, val.getNOBS());
                for (i=0;i<siz;i++){
                        bits[i] = bits[i].boolean_op_not();
                }
                return this;
        }
        //@or
        public Qd_binary binary_op_xor(Qd_binary val){
                int i=0;
                int siz = Math.min(bits.length, val.getNOBS());
                for (i=0;i<siz;i++){
                        bits[i] = bits[i].boolean_op_xor(val.bits[i]);
                }
                return this;
        }

}
```

EK 5- QDil Derleyici Temel Sınıfları

```
package qdil.compiler;

public class QdClass {
        public static final byte TYPE_CLASS = 1;
        public static final byte TYPE_INTERFACE = 2;
        public static final byte TYPE_CONSTCLASS = 4;
        public static final byte TYPE_ABSTRACTCLASS = 8;

        private byte version_maj, version_min;
        // bu class ın qvm deki durumunu belirliyor
        private byte class_qvm_state;
        private String package_name;
        private String class_name;
        private byte class_type;
        private String super_name; // süper sınıfının paket.adı
        private byte interface_count;
        private int field_count;
        private int method_count;
        private int used_class_count; // kac tane sınıf degisken tipi olarak kullanıldı

        private String[] used_class_names;
        private byte[] interface_indexs;// used class names icindeki indexleri
        QdField[] fields;
        QdMethod[] methods;
        int constant_list_size;

        QdConstant[] constant_list;

        public byte getVersion_maj() {
                return version_maj;
        }

        public void setVersion_maj(byte version_maj) {
                this.version_maj = version_maj;
        }

        public byte getVersion_min() {
                return version_min;
        }

        public void setVersion_min(byte version_min) {
                this.version_min = version_min;
        }

        public byte getClass_qvm_state() {
```

```
		return class_qvm_state;
	}

	public void setClass_qvm_state(byte class_qvm_state) {
		this.class_qvm_state = class_qvm_state;
	}

	public String getPackage_name() {
		return package_name;
	}

	public void setPackage_name(String package_name) {
		this.package_name = package_name;
	}

	public String getClass_name() {
		return class_name;
	}

	public void setClass_name(String class_name) {
		this.class_name = class_name;
	}

	public byte getClass_type() {
		return class_type;
	}

	public void setClass_type(byte class_type) {
		this.class_type = class_type;
	}

	public String getSuper_name() {
		return super_name;
	}

	public void setSuper_name(String super_name) {
		this.super_name = super_name;
	}

	public byte getInterface_count() {
		return interface_count;
	}

	public void setInterface_count(byte interface_count) {
		this.interface_count = interface_count;
	}

	public int getField_count() {
		return field_count;
	}

	public void setField_count(int field_count) {
		this.field_count = field_count;
	}

	public int getMethod_count() {
		return method_count;
	}

	public void setMethod_count(int method_count) {
		this.method_count = method_count;
	}

	public int getUsed_class_count() {
		return used_class_count;
	}

	public void setUsed_class_count(int used_class_count) {
		this.used_class_count = used_class_count;
	}

	public String[] getUsed_class_names() {
		return used_class_names;
	}

	public void setUsed_class_names(String[] used_class_names) {
		this.used_class_names = used_class_names;
	}
```

```
	public byte[] getInterface_indexs() {
		return interface_indexs;
	}

	public void setInterface_indexs(byte[] interface_indexs) {
		this.interface_indexs = interface_indexs;
	}

	public QdField[] getFields() {
		return fields;
	}

	public void setFields(QdField[] fields) {
		this.fields = fields;
	}

	public QdMethod[] getMethods() {
		return methods;
	}

	public void setMethods(QdMethod[] methods) {
		this.methods = methods;
	}

	public int getConstant_list_size() {
		return constant_list_size;
	}

	public void setConstant_list_size(int constant_list_size) {
		this.constant_list_size = constant_list_size;
	}

	public QdConstant[] getConstant_list() {
		return constant_list;
	}

	public void setConstant_list(QdConstant[] constant_list) {
		this.constant_list = constant_list;
	}

	public int getFieldIndex(String fieldName) {
		int i = -1;
		if (field_count > 0) {
			for (i = 0; i < field_count; i++) {
				if (fields[i].getField_name().equals(fieldName)) {
					return  i;
				}
			}
		}
		return -1;
	}

	public byte getMethodIndex(String methodName, byte param_number,
				String[] param_types) {
		int i = 0,j=0;
		byte[] param_type_indexes=null;
		boolean ok=true;
		if (method_count > 0) {
			for (i = 0; i < method_count; i++) {
				if (methods[i].getMethod_name().equals(methodName) &&
						methods[i].getParam_number()==param_number){
					param_type_indexes = methods[i].getParam_type_indexes();
					for (j=0;j<param_type_indexes.length;j++){
					    if (!param_types[j].equals(used_class_names[param_type_indexes[j]])){
						ok = false;break;
					    }
					}
					if (ok){return (byte)i;}
				}
			}
		}
		return (byte)-1;
	}

}
```

```
package qdil.compiler;
```

```
public class QdField {
	public static final byte OBJECT_FIELD = 1;
	public static final byte CLASS_FIELD = 2;

	public static final byte ACCESS_PLUS = 4;
	public static final byte ACCESS_MINUS = 8;
	public static final byte ACCESS_SHARP = 16;
	public static final byte ACCESS_PACKAGE = 32;

	private QdClass class_of_field; // bu field hangi classın nesnesi
	private byte class_index; // bu fieldin ait oldugu sınıfın index numarası
	private byte field_type_index; // field ın tipi clas içindeki used listten
	private byte access_type_get; // +,-,#,E
	private byte access_type_set; // +,-,#,E
	private byte kind_of_field; // OBJECT OR CLASS FIELD
	private int field_init_cons_index; // initialize kod constant list icinde nerede
	private String field_name;
	private QdMethod fieldGetMethod;
	private QdMethod fieldSetMethod;

	public QdClass getClass_of_field() {
		return class_of_field;
	}

	public void setClass_of_field(QdClass class_of_field) {
		this.class_of_field = class_of_field;
	}

	public byte getClass_index() {
		return class_index;
	}

	public void setClass_index(byte class_index) {
		this.class_index = class_index;
	}

	public String getField_name() {
		return field_name;
	}

	public void setField_name(String field_name) {
		this.field_name = field_name;
	}

	public byte getField_type_index() {
		return field_type_index;
	}

	public void setField_type_index(byte field_type_index) {
		this.field_type_index = field_type_index;
	}

	public byte getAccess_type_get() {
		return access_type_get;
	}

	public void setAccess_type_get(byte access_type_get) {
		this.access_type_get = access_type_get;
	}

	public byte getAccess_type_set() {
		return access_type_set;
	}

	public void setAccess_type_set(byte access_type_set) {
		this.access_type_set = access_type_set;
	}

	public int getField_init_cons_index() {
		return field_init_cons_index;
	}

	public void setField_init_cons_index(int field_init_cons_index) {
		this.field_init_cons_index = field_init_cons_index;
	}

	public QdMethod getFieldGetMethod() {
		return fieldGetMethod;
```

```
	}

	public void setFieldGetMethod(QdMethod fieldGetMethod) {
		this.fieldGetMethod = fieldGetMethod;
	}

	public QdMethod getFieldSetMethod() {
		return fieldSetMethod;
	}

	public void setFieldSetMethod(QdMethod fieldSetMethod) {
		this.fieldSetMethod = fieldSetMethod;
	}

	public byte getKind_of_field() {
		return kind_of_field;
	}

	public void setKind_of_field(byte kind_of_field) {
		this.kind_of_field = kind_of_field;
	}

}
```

```
package qdil.compiler;

public class QdMethod {
	public static final byte TYPE_DEF=1;
	public static final byte TYPE_EXTDEF=2;
	public static final byte TYPE_ORADEF=4;
	public static final byte TYPE_OPERATOR=8;
	public static final byte TYPE_OBJ_MTD=16;
	public static final byte TYPE_CLS_MTD=32;
	public static final byte TYPE_FLD_MTD=64;

	private QdClass class_of_method;
	private int class_index;
	private String method_name;
	private byte method_type; //OBJ_MTD,CLASS_MTD,FLD_MTD+(DEF,EXTDEF,ORADEF,OPERATOR)
	private byte mtd_features; //PACKAGE,MINUS,PLUS,SHARP,CONST,ABSTRACT Qd_Acc_Type
	private byte raise_count; //raise deyimi varsa
	private byte[] raises_indexes; //throws deyimi varsa
	private QdCatchTable catch_table;
	//used_class_names icinde verilen indislerde parametrelerin
	//tipleri sınıf ismi olarak var.
	private byte param_number;
	private  byte[] param_type_indexes; // types in used_class_names
	                                                                    // soldan saga sıralı
	private int return_type_index;
	private int code_size;
	private byte[] code; // method codes
	public QdClass getClass_of_method() {
		return class_of_method;
	}
	public void setClass_of_method(QdClass class_of_method) {
		this.class_of_method = class_of_method;
	}
	public int getClass_index() {
		return class_index;
	}
	public void setClass_index(int class_index) {
		this.class_index = class_index;
	}
	public String getMethod_name() {
		return method_name;
	}
	public void setMethod_name(String method_name) {
		this.method_name = method_name;
	}
	public byte getMethod_type() {
		return method_type;
	}
	public void setMethod_type(byte method_type) {
		this.method_type = method_type;
	}
	public byte getMtd_features() {
		return mtd_features;
	}
	}
```

```
	public void setMtd_features(byte features) {
		this.mtd_features = features;
	}
	public byte getRaise_count() {
		return raise_count;
	}
	public void setRaise_count(byte throw_count) {
		this.raise_count = throw_count;
	}
	public byte[] getRaises_indexes() {
		return raises_indexes;
	}
	public void setRaises_indexes(byte[] raises_indexes) {
		this.raises_indexes = raises_indexes;
	}
	public QdCatchTable getCatch_table() {
		return catch_table;
	}
	public void setCatch_table(QdCatchTable catch_table) {
		this.catch_table = catch_table;
	}
	public byte getParam_number() {
		return param_number;
	}
	public void setParam_number(byte param_number) {
		this.param_number = param_number;
	}
	public byte[] getParam_type_indexes() {
		return param_type_indexes;
	}
	public void setParam_type_indexes(byte[] param_type_indexes) {
		this.param_type_indexes = param_type_indexes;
	}
	public int getReturn_type_index() {
		return return_type_index;
	}
	public void setReturn_type_index(int return_type_index) {
		this.return_type_index = return_type_index;
	}
	public int getCode_size() {
		return code_size;
	}
	public void setCode_size(int code_size) {
		this.code_size = code_size;
	}
	public byte[] getCode() {
		return code;
	}
	public void setCode(byte[] code) {
		this.code = code;
	}
}
```

```
package qdil.compiler;

public class QdCatchTable {
	private byte catch_table_entry_count;
	private QdCatchTableEntry[] entries;

	public QdCatchTable() {
		catch_table_entry_count =0;
		entries = null;
	}

	public byte getCatch_table_entry_count() {
		return catch_table_entry_count;
	}

	public void setCatch_table_entry_count(byte catch_table_entry_count) {
		this.catch_table_entry_count = catch_table_entry_count;
	}

	public QdCatchTableEntry[] getEntries() {
		return entries;
	}

	public void setEntries(QdCatchTableEntry[] entries) {
		this.entries = entries;
```

```
		}

}
```

```
package qdil.compiler;

public class QdCatchTableEntry {
		private int start_ins_number; //bu instruction number ile
		private int end_ins_number;//bunun arasında
		private int exp_type_index; //bu fırlarsa
		private int jump_code_offset; //code da bastan(0) itibaren buraya atla

		public int getStart_ins_number() {
			return start_ins_number;
		}
		public void setStart_ins_number(int start_ins_number) {
			this.start_ins_number = start_ins_number;
		}
		public int getEnd_ins_number() {
			return end_ins_number;
		}
		public void setEnd_ins_number(int end_ins_number) {
			this.end_ins_number = end_ins_number;
		}
		public int getExp_type_index() {
			return exp_type_index;
		}
		public void setExp_type_index(int exp_type_index) {
			this.exp_type_index = exp_type_index;
		}
		public int getJump_code_offset() {
			return jump_code_offset;
		}
		public void setJump_code_offset(int jump_code_offset) {
			this.jump_code_offset = jump_code_offset;
		}
}
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Mustafa ŞAHİN

Doğum Yeri : Çan / ÇANAKKALE

Doğum Tarihi : 09.02.1979

EĞİTİM DURUMU

Üniversite	Fakülte	Bölümü	Derece	Mezuniyet Yılı
Çanakkale Onsekiz Mart Üniversitesi	Mühendislik-Mimarlık Fakültesi	Bilgisayar Mühendisliği	Yüksek Lisans	2005
Çanakkale Onsekiz Mart Üniversitesi	Mühendislik-Mimarlık Fakültesi	Bilgisayar Mühendisliği	Lisans	2001

BİLİMSEL FAALİYETLERİ

BİLDİRİLER
M. Şahin. Java, Python ve Ruby Dillerinin Karşılaştırılması, Arademik Bilişim 2007. Dumlupınar Üniversitesi.
Şahin M. Mammadov M., Tahmin ve Sınıflandırma Problemlerinde Farklı Geriye Yayılım Algoritmalarının Performansları, 4. İstatistik Kongresi Bildiri ve Poster Özetleri Kitabı, st. 246-247, Rekmay Ltd., Ankara, 2005.
M. Şahin, Java 1.5 de koşut zamanlı programlama yenilikleri ve performansları, Akademik Bilişim 2005, Gaziantep Üniversitesi, Gaziantep.
İ. Yılmaz, İ. Türkyılmaz, Ç. Camcı, C. Aktaş, M.Şahin and M. Battaloğlu, Solutions Of The Topological Structure In The Early Universe Via Conformal Motions, International Conference of Mathematical Sciences, 2009.
KİTAP ÇEVİRİSİ
K. Ökmen, İ. Türkyılmaz, M. Şahin ve İ. Yılmaz. 2011. Kuantum Bilgisayarlarda Hesaplamaya Giriş. Maltepe Üniversitesi Yayınları. İstanbul.
PROJE DENEYİMİ
Karo Yapılı Çok Çekirdekli İşlemcilerde Dizin Temelli Önbellek Tutarlılığı Verimliliğinin Arttırılmasına Yönelik Donanım/Yazılım Mekanizmaları, Bursiyer.

İŞ DENEYİMİ

Çanakkale Onsekiz Mart Üniversitesi	Öğretim Görevlisi	2007-
Çanakkale Onsekiz Mart Üniversitesi	Araştırma Görevlisi	2001-2007

İLETİŞİM

E-posta Adresi : msahin@comu.edu.tr