JAVA'nın Tarihçesinden

Java, genel amaçlı nesneye yönelik bir programlama dilidir. Sözdizimi C ve C++'a çok benzer. Başlangıçta Java, küçük sistemleri tüketici aygıtlarına gömme yoluyla yazılımın kurulmasında karşılaşılan sorunları adreslemek için geliştirilmişti. Heterojen iletişim ağları, birden fazla ana mimari yapılar ve güvenli servis için tasarlanmıştı. Bu tür gereksinimlere karşılık verebilmek için, Java herhangi bir sistemde çalışmalı, ağlar arasında iletişimi ve kullanıcının güvenli bir ortamda çalışmasını sağlamalıydı.

World Wide Web'in yaygınlaşması Java'nın bu özelliklerinin açığa çıkıp herkes tarafından bilinmesine yardımcı olmuştur. Son zamanlarda insanların ilgi odağı olan İnternet'e erişimi basit yollarla kolaylaştırmıştır. Mosaic gibi Web browser'lar milyarlarca insanın Net'te dolaşmasını ve sörf yapmasını sağlamaktadır. Yüksek hızlı bir modemle network ortamına Mac, PC ya da UNIX bir makine ile bağlandığınızda en azından size, duyduklarınızın ve gördüklerinizin aynı olmasını sağlamalıdır.

Web'in günden güne nüfusu artmaktadır. Ancak Web fanatikleri, Web'in HTML doküman formatı tarafından desteklenen içeriğinin çok sınırlı olduğunu gördüler. HTML uzantıları, browser'ın kullanıcının istediği tüm özellikleri sunmadığını daha açık bir şekilde ortaya çıkardı. Bu sırada Java programlama dili aranan bir başka uygulamayı buldu.

Java programlarını Web sayfalarına gömerek Java'nın ilginç özelliklerini kullanıma sunan Sun'ın HotJava browser'ı geliştirildi. Java'nın bu tür programlarına applet adı verilmektedir. Bu programlar görüntülendiği HTML sayfalarıyla birlikte HotJava browser'a yüklenir. Browser tarafından kabul edilmeden önce applet'lar, güvenilir olduğundan emin olmak için dikkatli bir şekilde kontrol edilmelidir. HTML sayfaları gibi derlenmiş Java programları da iletişim ağları ve bağımsız platformlardır. Applet programlar yüklendikleri makineleri ve geldikleri yerleri dikkate almadan çalışırlar.

Java'nın C programlama diline çok benzediğini söylemiştik. Ancak C'nin tersine Java'da belirteç kullanılmamaktadır. Bu durum güzel bir avantaj sağlamaktadır: Programın iyileştirilmesi sırasında hata oranı azalmaktadır. Böylelikle uzun zaman alan Boot ve Debuging'ler ortadan kalkmıştır. Bu farkı özellikle C kullanıcıları daha iyi görmektedir.

Bunun yanı sıra türlerin birbirlerine dönüştürülmesi işleminde Java, oldukça geniş olanaklar sunmaktadır. Bu da karmaşık işlemlerde hataların en baştan yok edilmesine neden olmaktadır.

Web toplumu Java'da bazı şeylerin yeni ve oldukça önemli olduğuna çok çabuk dikkati çekti. Web sayfalarını güçlendiren Java kullanıcıları makinelerine zarar vermeden güvenli bir ortamda çalışmanın keyfini çıkarmaktadırlar.

Java tüm applet programlarıyla birlikte İnternet kullanımında yeni bir yol sunmanın yanı sıra kullanımını da oldukça kolaylaştırmaktadır.

Günümüzde JAVA

Java'nın yaratıcısı Sun Microsystems'in mühendislerinden biri olan James Gosling, bu uğurda saçlarını ağartmış ileri görüşlü biri. Java tasarlanırken amacın piyasadaki bir çok programlama diline bir yenisini eklemek olduğunu zannetmiyorum, burada amaç bilgisayar dünyasına yeni bir ufuk açmak, yeni fikirler getirmek. Java'nın en büyük problemi kendisinden önce insanlara ulaşan bu yeni fikirlerin arasında kendini tanımlayamaması oldu. Çoğu insan Java'yı bu fikirlerle aynı kefeye koydu, bir başka kesimse yüzüne bile bakmadı. Bundan dolayı ilk olarak Java ile Java'nın getirdiği fikirleri birbirinden ayırarak işe başlamak istiyorum.

Java, gerçek anlamda bir programlama dilidir. Java ile günümüzün popüler dili C++ ile ne yapılıyorsa hepsinin yapılması mümkündür. İçinizden olur mu hiç öyle şey dediğinizi duyar gibiyim. Bal gibi de olur. Bilgisayar dilleri, syntax'ı hariç birbirlerinden çok farklı değillerdir. Önemli olan sizin programı yazmak istediğiniz platform için elinizde bulunan kütüphane(library)'lerdir. C++ bu konuda çok geniş olup Java ise kısıtlıdır. Peki o zaman Java ile her şey nasıl yazılır? Java, başka bir dilde yazılmış bir kütüphaneyi çok rahat kullanabilmektedir. Bu da yeni doğmuş bu dilin işini oldukça kolaylaştırmaktadır. "Olur mu öyle şey o zaman Java programımız bütün platformlarda çalışmaz!" işte bu cümle yine Java ile Java'nın felsefesini karıştıran cahilliğin eseridir. Kim demiş ki biz Java programlarımızı her platformda çalıştırmak istiyoruz diye. Bilgisayar programları satan bir yerden bir yazılım (software) alırken hangi platform için yazıldığına bakarak alırız, ve de gidip aynı software'i cep bilgisayarımızda da güle oynaya kullanabileceğimiz fikrine kapılmayız (tabi eğer program cep bilgisayarı için yazılmadıysa). Aynı mantıktan yola çıkarak Java ile Java derleyicisini elde edebileceğimiz herhangi bir platform için spesifik olan bir yazılım çıkarmak hiçte yanlış bir davranış değildir.

Microsoft, Java'ya bir programlama dili olarak verilmesi gereken önemi daha çok önceden anlamış, ve çıkardığı Visual J++ adlı software'inde PC platformuna spesifik program yazmak için gereken bir çok gereci içine koymuştur. Sun Microsystems ise kanımca yanlış bir strateji izleyerek bunu baltalamaya çalışmaktadır.

İşte Sun Microsystems ile Microsoft arasında uzun süredir süren davaların nedeni bundan ileri gelmektedir. Peki ama Sun'ın, Microsoft'la alıp veremediği nedir? Niçin faydalı birşeyi baltalamaya çalışmaktadır? Bunu anlamak için Java'nın getirdiği fikirleri anlamak gerekir.

Java'nın bir programlama dili olduğunu anladık. Peki getirdiği fikirler nelerdir? Java tasarlanırken amaç, bu dille yazılan bir programı kolay bir şekilde diğer platformlara taşıyabilmekti. Bir platform için yazılmış bir yazılımın başka bir platformda çalışır hale getirmek çok emek gerektirir. İşte Java bu probleme ilaç olmak için düşünüldü. Peki bu iş nasıl olur? Öncelikle, bütün dünyadaki bilgisayar tasarlayıcılarının bir araya gelip ortak, herkesi memnun edecek bir bilgisayar çıkartması ufukta görünmediği için, bu işi yazılım bazında halletmek gerekir. Bunun için Sun "Virtual Machine" dediği sanal bir bilgisayar tasarlamıştır. Bu sanal bilgisayar, sizin gerçek bilgisayarınızda çalışan bir program olup, Java programlarınızı çalıştıracaktır. Burada ortaya atılması gereken en büyük soru şu olmalıdır: "İyi de bu Virtual Machine denen programı butun platformlar için nasıl ve kim yazacak?" İşte Sun Microsystems ve birçok Java programcısının gözünden kaçan da bu olmuştur. Su anda bu sanal bilgisayar bir elin parmaklarını geçmeyecek kadar değişik platform için hazırda vardır, gelecekte bu sayının artması beklentimiz olmalıdır. Üstüne üstlük bu platformlardaki Virtual Machine'in en son versionlarını sadece iki platformda bulmak mümkündür. PC ve Sun'ın ürettiği Solaris. Apple maalesef yarışta geride kalmıştır. Bu nedenden dolayı Sun'ın Microsoft'u, Java'yı güçlü bir programlama dili olarak sunmasını baltalamaya çalışması yanlıştır. Nede olsa, Java ile yazılan programlar hiç bir zaman dünyadaki bütün platformlarda çalışmayacaktır. Bunu düşünmek aptallık olur.

Olayın biraz daha detayına inelim. Yukarıdaki paragrafta aslında dikkatli incelenirse bazı yanlışlar vardır, aslında basite indirgemek desek daha doğru olur. Virtual Machine'i baska bir platforma aktarmak zor olan bir şey değildir ve de ilk çıktığından bu yana fazla yeni sürümüde çıkmamıştır. Şimdi yine bazı okuyucular bana gülüyorlar biliyorum, hatta içlerinden şu cümle geçiyor: "Çıktığından beri değiştirmedikleri yeri kalmadı, bir öğrendiğimiz ertesi gün yok!" Programcılar bu kaygılarını dile getirirken aslında bir şeyi unutmaktadırlar. Değişen Virtual Machine

değildir. Değişen Virtual Machine ile birlikte Sun'in tasarladığı standart kütüphanelerdir. Bir dili kullanılır yapan öğenin kütüphanelerinin genişliği olduğunu daha önce belirtmiştim. İşte bu kütüphaneleri değişik platformlara taşımak gerçek bir azaptır, ve de bir gecede olmayacaktır, çünkü bir PC de ekrana bir pixel basmakla, bir Macintosh'ta ekrana bir pixel basmak farklı prosedürlerdir.

Peki Sun kendine Microsoft'a karşı destekçi nasıl bulmaktadır. Bu o kadar zor bir şey değildir. Microsoft piyasadaki monopol olarak zaten bir çok kullanıcının ve programcının nefretini kazanmayı başarmıştır. Dakka başı yeniden başlatmak zorunda kaldığımız Windows'da bunun cabasıdır. Tek yapılması gereken piyasaya bir slogan atmaktır: "Visual J++ ile yapılan Java programlarınız başka platformlarda çalışmaz!". Bu doğru bir cümle değildir, ama amacına ulaşmaktadır. Burada bizlerin anlaması gereken, Java'ya yeni başlayan birinin ne yaptığını bilmeden bu yazılımı kullanıp yazdığı bir Java programının başka platformlarda çalışmama olasılığının bulunduğudur. Şahsen ben Visual J++' in son sürümünü beğenerek kullanıyorum, ama kesinlikle bu işe yeni başlayan birine tavsiye etmiyorum. Eleştiriler çok da yersiz değil. Ancak, önemli noktayı kaçırmamak gerekir. Microsoft, kanımca Java konusunda kendisine özgü olmayan bir şey yapıyor: gerçekten faydalı olmaya çalışıyor. Sun ise Microsoft'a pazarı kaptırmanın verdiği hiddetle saldırıyor. Olaydan zararlı çıkan aslında yine biz programcılarız.

Yazımı biraz da Java'yı popüler yapan "Java Applet" lerinden bahsederek tamamlamak istiyorum, çünkü maalesef etrafta bilir bilmez konuşulan kavramların başında geliyor. Java Applet'lerinin Java programlarından farkı, web browser'ınız tarafından çalıştırılabilmesi, ve de Internette dolaşırken bir web sayfasında karşımıza çıkabileceği için güvenliğiniz düşünülerek sınırlandırılmış olmalıdır. Konun detayları, bu sınırlandırmanın nasıl yapıldığı, veya nasıl kaldırılabildiği yine bu yazının içeriği değildir. Bunların haricinde Java Applet'leri, Java programlarından hiç farklı değildir, onlar da Virtual Machine ile çalıştırılırlar. Önemli olan nokta Java Applet'lerini yazarken platform spesifik kütüphaneler kullanılmaması gerektiğidir. Çünkü sizin web sayfanızı ziyaret eden kişinin ne marka bilgisayar kullandığını bilemezsiniz. İşte Sun'ın standart kütüphanelerdinden şaşmamanız gereken yer burası. Yok hayır, beni sadece PC

kullanıcıları bağlar derseniz, o zaman istediğiniz yapmakta yine serbestsiniz. Kalkıp Microsoft'un DirectX kütüphanesini kullanıp 3 boyutlu mekanlarda insanları dolaştırmak mı istiyorsunuz, yada insanların kulağına Directional Audio ile hitap mi etmek istiyorsunuz? Bunların hepsi mümkün. Unutulmaması gereken Applet'lerin uzunca bir süre gerçek programların yerini alamayacağı. Düşünülmesi gereken bir başka nokta ise, Appletler'in bu amaçla kullanılmasının ne kadar doğru olduğu. Bunu size düşünme konusu olarak bırakıyor, bir başka yazıda tartışmayı diliyorum. Java'yı desteklemeyi bırakmayın.

Herkese iyi programlamalar.

JAVA DİLİNE GİRİŞ

Java' ya Başlarken

Derleme birimi olarak adlandırılan bir Java kaynak kütüğü bir ya da birkaç sınıf tanımı içeren bir metin kütüğüdür. Java derleyici, bu kütüklerin .java kütük adı uzantısı ile yüklenmesini bekler. Java kaynak kodu derlendiği zaman, her bir sınıf kendi çıktı kütüğüne yerleştirilir. Java'da genel fonksiyonlar ve değişkenler olmadığı için, bir Java kaynak kütüğünde olabilecek tek şey bir ya da birkaç sınıfın tanımıdır.

İlk Java programımızı yazalım.

```
class HeyDünya {

public static void main(String args[ ]) {
 System.out.println("Hey Dünya!");
                              }
                    }
```

Java, tüm kodların adı olan bir sınıfta yer almasına gereksinim duyar. Kütük adının harf düzeninin sınıf adına uygun olduğundan emin olduktan sonra, bu metni HeyDünya.java adlı bir kütüğe yüklemelisiniz. Kütük adlarının sınıf adına uygun olması. programınızı çalıştırmada çok yardımcı olacaktır. Java derleyici javac'ı çalıştırarak ve komut satırı üzerinde kaynak kütüğün adını belirterek bu metni derleyebilirsiniz; aşağıda gösterildiği gibi:

```
C:\> javac HeyDünya.java
```

Javac derleyici, yeni programın bayt kodundan bağımsız olarak derlenmiş bir işlemciyi kapsayan ve adı HeyDünya.class olan bir kütük hazırlayacaktır. Bayt kod, Java Sanal Makina (Java Virtual Machine) yorumlayıcı için makine dili komutlarını içeren programınızın bir ara sunumudur. Bu programı çalıştırmak için, bu yeni sınıfı

önce yükleyen, sonra çalıştıran bir Java yürütme zamanınızın olması gerekir. Bunu yapmak için ise, java komut satırında argüman yerine HeyDünya adlı sınıfa geçiş yapmanız gerekir.

```
C:\>java HeyDunya
Hey Dünya!
```

Bu noktada önemli olan, Java derleyicinin ve yürütme zamanının doğru olarak, kurulmasıdır.

Şimdi, yazdığımız programı satır satır inceleyelim.

"HeyDunya" basit bir program olmakla birlikte ilk kez kullandığınız yepyeni bir programlama dili ile tanışmış oldunuz. Bunun yanısıra yeni kavramlar ve ayrıntılar için attığınız ilk adımdır. İşte ilk satırımız:

```
class HeyDunya {
```

Bu satır, yeni bir sınıfın tanımlandığını bildirmek için class sözcüğünü kullanır. HeyDunya, sınıfı tanıtmak amacıyla kullandığımız geçerli bir tanıtıcıdır. Açık küme (kıvrımlı) parantezi ({) ile kapalı küme (kıvrımlı) parantezi (}) arasında sınıfın tam adı, kod ve veri yer alır. Java'da bu parantezlerin kullanımı C ve C++ yapısı ile eşdeğerdir.

```
public static void main(String args[ ]) {
```

Bu basit programın ikinci satırı genel fonksiyonların olmadığını, yalnızca sınıfların yer aldığını bildiren Java tasarımı ile zorlaştırılır.

Alan ve yöntemler biraraya gelerek sınıfları oluşturur. Sınıfın tanımı sınıfın yapısını ve hareketini belirler. Yöntemler fonksiyonlardan biraz daha farklı olup kendi sınıflarının veri alanlarına (değişkenlerine) erişebilirler.

public

public anahtar sözcüğü bir erişim belirleyici olup, programcıya her bir değişkenin ya da yöntemin kontrolünü verir. Burada public anahtar sözcüğü herhangi bir sınıfın main yöntemini görebildiğini göstermektedir.

private

Bu tip bir sınıfa yalnızca paket içerisinde ulaşılabilir. Bir sınıfın tanımlanması sırasında public ya da private olduğu belirtilmediği zaman, sınıfın otomatik olarak private olduğu kabul edilir.

abstract

Soyut yöntemlerin yer aldığı sınıfları belirtir. Soyut yöntemler program kodu olmayan anlamına gelmektedir. Soyut sınıflar diğer sınıflar için yapı oluşturur.

static

Bir sınıfa uygulanan yöntem ve değişkenlerin static olduğu bildirilir. Yöntem, sınıfın özel bir örneğinin çalıştırılmasına gerek kalmadan çağrılabilir. Main durumunda örneklerden önce yorumlayıcı tarafından çağrıldığı için, static olduğunu bildirmek gerekir. Static olduğu bildirilen yöntemler, statik ve yerel değişkenlerden başka herhangi bir şeye doğrudan başvuramayabilir. Nesneye yönelik programlama dillerinde statik değişken ve yöntemlere sınıf değişkenleri ve sınıf yöntemleri adı verilmektedir.

void

Genellikle birtakım özel türlerin değerleri için yöntemler kullanılır. Örneğin tamsayı değerler için int, gerçel sayı değerleri için float ya da daha karmaşık değerler için bir sınıfın özel adları kullanılmaktadır.

main

Java, tüm anahtar sözcük ve tanıtıcılara göre harfe duyarlı bir yapıya sahiptir. Bir başka deyişle Java'da Main ile main farklı yöntemleri ifade etmektedir.

Java derleyici, bir main yöntemi olmayan sınıfları derler. Ancak Java yorumlayıcı bu sınıfları çalıştırmaz. Bu programı yanlış yazdığınızı ya da Java tanıtıcılarının harfe duyarlı olduğunu bilmediğinizi varsayalım ve main yerine Main yazdığınızı kabul edelim. Derleyici HeyDunya adlı sınıfınızı derler. Kullandığınız Java yürütme zamanı, çalışmayı başlatacak yöntemi bilmediği için derlemeyi gerçekleştirir.

Örneğin main sözcüğünü hatalı yazdıysanız, programı çalıştırmak istediğinizde aşağıdaki mesajı alırsınız:

```
C:\> java HeyDunya
HeyDunya sınıfında: void main (String argv[ ]) tanımsız
```

Yöntemlerin bir kısmı parametrelere gereksinim duyar. Bu parametreler parantez içerisinde belirtilir. Verilen yöntem için parametreye gereksinim yoksa, parantezi boş olarak yerleştirmeniz gerekir. Bizim örneğimizde gördüğümüz gibi kullanılan bir parametre vardır. String args [ ], args adlı bir parametreyi göstermektedir. Tanıtıcıdan sonra kollanılmış olan köşeli parantezler ([ ]), bunun türünün bir dizi olduğunu göstermektedir. String karakter dizilimleri depolayan bir sınıftır.

Yorumlayıcının başlangıç yeri main yöntemidir. Program bir düzine sınıfa sahip olabilir, ancak bunlardan biri main yöntemine sahip olabilir. Applet'lar (Web browser'larda gömülü olan küçük Java programlarına denmektedir) hazırlamaya başladığınız zaman, Web browser'ın Java yürütme zamanı farklı bir kalıba sahip olduğundan main yöntemi kullanılmaz.

```
System.out.println("Hey Dünya!");
```

Bu satır out içerisinde println yöntemini çalıştırır. Out, bir OutputStream sınıfı elemanı olup, System sınıfında kullanılmaktadır.

OutputStream sınıfının println yöntemi dizilimi basar. Bu yöntem, bir başka çıktının yeni satır üzerinde basıldığından emin olmak için baskıdan önce, bir yeni satır karakterini dizilimin sonuna ekler.

Sözsel Bileşenler

Java programları komutlar, anahtar sözcükler, tanıtıcılar, hazır bilgiler, işletmenler, ayırıcılar ve boşluğun bileşeninden oluşmaktadır.

Boşluk (Whitespace)

Java bağımsız bir dildir. Düzgün çalışmasını sağlamak amacıyla herhangi bir şeyi içerden yazmak zorunda değilsiniz. HeyDunya sınıfı tek bir hizada yazılmıştır:

```
Public class HeyDunya{public static void main(String a[])
{System.out.println("HeyDunya!");} }
```

Aşağıda verilen örnek aynı .class sınıfını üretir:

```
public
class
HeyDunya
{
public
static
void
main
(String
```

```
args
[
]
 )
{
 System
.
out
.
println
(
"HeyDunya!"
)
;
}
}
```

Komutlar

Komutlar bir programın yürütme zamanına katkıda bulunmasalar bile, uygun bir şekilde kullanıldığında kaynak kodun en değerli kısmını oluşturmaktadır. Geniş programların anlaşılması da oldukça zordur. Ancak bir sınıfın ya da bir yöntemin ne ve niçin sorularına yanıt verebilen komutları kullanıcılar için oldukça yararlıdır. Bir kod satırını basit bir şekilde yineleyen komutlar, çok fazla değer eklemez, ancak algoritmayı gösteren ya da açıklayan komutlar programcılar için işaretçidir.

Kaynak kod komutları üç türde ele alınır: Tek satırlı, çok satırlı ve dokümantasyon komutları. Tek satırlı komutlar, // ile başlar ve satır sonunda biter. Bu komut türü. tek satırlı bir kodun kısa açıklamalarında kullanışlıdır.

```
a = 42;          // 42 yanıt ise, soru nedir?
```

Daha uzun komutlar için, a/* ile başlayıp a*/ ile biten uzun çok satırlı komutlar hazırlayabilirsiniz. a/* ile */ arasında yer alan herhangi bir şey bir komut olarak ele, alınır ve derleyici tarafından göz ardı edilir.

```
/*
    * Ankara Türkiye'nin başkentidir...
    * Açıklanmasına izin verin:
    */
```

Özel bir komut türü javadoc adlı bir araç ile kullanılmaktadır. Javadoc, bir sınıfın genel arabirimi için dokümantasyonu otomatik olarak üretmek amacıyla Java derleyicinin bileşenlerini kullanır. Javadoc'u desteklemek için kullanılan komut kuralı, genel sınıf, yöntem ve değişken deklerasyonundan önce bir dokümantasyon komutunu göstermek amacıyla /** komutunu kullanmaktır. Bu komut uzun çok satırlı komutlarda olduğu gibi */ ile biter. Javadoc, @ ile gösterilen birkaç özel değişkeni tanır.

Özel Amaçlı Anahtar Sözcükler

Özel amaçlı anahtar sözcükler özel tanıtıcılardır ve programın tanımlanmasını kontrol etmek için Java dili tarafından kullanılmaktadır. Yerleşik türleri, değiştiricileri ve akış kontrolü için mekanizmayı tanıtmak için kullanır. Bunlar toplam 59 tane özel amaç için ayrılmış sözcüktür. İşletmen ve ayırıcıların sözdiziminin birleştirilmesi ile oluşturulmuş olan bu anahtar sözcükler, Java dilinin tanımlanmasını biçimlendirir. Bu anahtar sözcükler yalnızca istenen amaca hizmet ederler ve bir değişken, bir sınıf ya da bir yöntem adı için tanıtıcı olarak kullanılamazlar.

Abstract	boolean	break	byte	byvalue
Case	cast	catch	char	class
Const	continue	default	do	doule
Else	extenhds	false	final	finally
Float	for	future	generic	goto
İf	implements	import	inner	instanceof

İnt	interface	long	native	new
Null	operator	outer	package	private
Protected	public	rest	return	short
Static	super	switch	synchronized	this
Throw	throws	transient	true	try
Var	void	volatile	while	

Java' nın özel amaçlı sözcükleri

Tanıtıcılar

Tanıtıcılar sınıf, yöntem ve değişken adları için kullanılır. Bir tanıtıcı büyük ve küçük harf, sayılar, altçizgi ve dolar işareti karakterinden oluşan bir tanımlayıcı dizidir. Bir sayı ile başlamamalıdır.

Java'nın harfe duyarlı olduğunu biliyorsunuz. VALUE ile Value birbirinden farklı tanıtıcılardır. Geçerli tanıtıcılar içerisinde timeOfDay, temp_val, a4 ve $_'ı söyleyebiliriz. 1 more, 3$, a:b, #foo ve @2 geçerli olmayan değişken adlarıdır.

Özel ve yerel değişkenler için tanıtıcılar altçizgi ile birlikte küçük harf olarak kullanılır, next_val ya da temp_val gibi. Sabitleri gösteren sonuç (final) değişkenler için tümüyle büyük harf kullanılır, TOK_BRACE, DAY_FRIDAY ya da GREEN gibi.

Hazır Bilgiler

Java'da bir sabit değer, kendisinin hazır bilgisi kullanılarak hazırlanabilir. Tamsayılar, virgüllü sayılar, boolean değerler, karakterler ve dizilimler Java kaynak kodunda herhangi bir yere yerleştirilebilir. Bu hazır bilgilerin her birinin sahip olduğu uygun bir türü vardır. Hazır bilgi bir türün belli bir değerini gösterir.

İşletmenler

Bir ya da birkaç argümandan oluşan bir işletmen sonuca ulaşmak için bu argümanları işletir. İşletmenler hazır bilgilerin yanında ya da tanıtıcıların arasında görünür. Aşağıdaki tabloda Java'da kullanılan işletmenler listelenmiştir.

+	+=	-	-=
*	*=	/	/=
|	|=	^	^=
&	&=	%	
>	>=	<	
!	!=	++	
>>	>>=	<<	
>>>	>>>=	&&	
==	==	~	?:
.	İnstanceof	[]	

Java işletmenleri

Ayırıcılar

Doğru bir Java programında görünebilen yalnızca bir çift farklı karakter dizisi vardır. Bunlar birkaç basit ayırıcı olup, kullandığınız kodun şeklini ve işlevselliğini tanımlar.

Aşağıda verilen tabloda ayırıcılar görevleri ile birlikte açıklanmaktadır:

Sembol	**Adı**	**İşlevi**
()	parantezler	Yöntemin tanımı ve çalıştırılması için gereken parametre listesini yerleştirmek için kullanılır. Ayrıca ifadelerde öncelik sırasını tanımlamak içinde kullanılır.

{ }	küme parantezi	Otomatik olarak ilk kullanıma hazırlanan dizilerin değerlerini yerleştirmek için kullanılır. Ayrıca sınıf, yöntem ve yerel kapsamlar için bir kod bloğunu tanımlamak için kullanılır.
[]	köşeli parantez	Dizi türlerini bildirmek ve dizi değerlerin uygulamasında kullanılır.
;	noktalı virgül	Deyimleri ayırır.
,	virgül	Değişken bildiriminde yer alan ardışık tanıtıcıları ayırır. Deyimleri bir for deyimine ardışık yerleştirmek için kullanılır.
.	nokta	Paket adlarını alt paketlerden ve sınıflardan ayırmak, bir değişken ya da yöntemi bir başvuru değişkeninden ayırmak için kullanılır.

Java ayırıcıları

Değişkenler

Bir Java programında temel bellek birimi değişkenlerdir. Bir tanıtıcı, tür ve kapsam birleşerek bir değişkeni tanımlayabilir. Değişkenlerinizi bildirdiğiniz yere bağlı olarak, geçici bir kapsam için yerel olabilir ya da sınıftaki tüm yöntemlere erişebilen örnek değişkenler olabilir. Yerel kapsamlar küme parantezi ile gösterilir.

Bir Değişkenin Kullanılması

Bir değişken aşağıdaki şekilde ifade edilebilir:

tür tanıtıcı [= değer] [, tanıtıcı [= değer] ...] ;

Tür bir bayt, short, int, long, char, float, double ve boolean değer ya da bir sınıf veya arabirimin adı olabilir.

int x, y, z;	x, y ve z olan üç tamsayıyı bildirir.
int a = 3, b, c = 5;	a ve c'yi başlatan üç tamsayıyı daha bildiriyor.
byte z=2z;	z'i çalıştırır.
double pi= 3.14159;	Pi sayısını (tam olmasa da yaklaşık olarak) bildirir.
char x='x';	x değişkeni 'x' değerine sahip.

Value (değer), bir türün değerini elde eden hazır bilgi ya da bir ifadedir. Aşağıdaki örnekte bir dik açının kenarlarını gösteren üç değişken hazırlanmakta ve ardından Pisagor Teoremi kullanılarak hipotenüs uzunluğu hesaplanmaktadır.

```
class Degişken {
public static void main(String args[ ] ) () {
double a = 3;
double b = 4;
double c;
c = Math.sqrt(a*a + b*b);
System.out.println("c = " + c);
}
}
```

Değişkenin Kapsamı

Java' da bileşik deyim blokları iki küme parantezi ({}) ile belirtilir. Java değişkenleri yalnızca bileşik deyimlerin sonuna kadar bildirilen yerden ge4erlidir. Bu bileşik deyimler içiçe yerleştirilebilir ve her birinin kendisine ait yerel değişken ifadesi olabilir.

Aşağıdaki örnekte ismi aynı olan iki ayrık değişken ifade edilmeye çalışılmaktadır. C ve C++' da bu değişkenler farklı kapsamlara sahip olduğu için birbirinden farklı olduğu kabul edilir. Java' da ise bu durum yasal değildir.

```
class Kapsam {
public static void main(String args[ ] ) ( ){
                                        int bar = 1 ;
{                       // yeni bir kapsam hazırlar
int bar = 2;                    // zaman hatasını derler...
                                        }
                                }
                        }
```

TÜRLER

Tür nedir? Tür içeriğe ve karakteristik özelliğe bağlı bir sınıflandırmadır. Programlamada tür, bir ifade ya da bir değişken olarak tanımlanır.

Tamsayı türünün asla pi sayısını tam olarak göstermediğinden emin olabilirsiniz.

Bu bölümümüzde Java'da kullanılan ana türler ele alınacak, değişkenlerin nasıl bildirildiği, değerlerin atanması ve türlerin ifade olarak birleştirilmesi gösterilecektir.

Basit Türler

Çoğu nesneye yönelik programla dillerinde her bir tür bir nesneye dönüştürülür. Örneğin 2 + 2 ifadesi, iki objesi üzerinde toplama yöntemini çalıştırmaktadır. Java dilini daha iyi anlayabilmek için, Java'da herşeyin bir obje olduğunu ve sistem performansının Java'nın gelişiminde esas amaç olduğunu kabul edin. Bu karar Java'nın basit türlerinin hazırlanmasına neden olmaktadır. Aslında bunların hepsi nesneye yönelik olmamakla birlikte diğer nesneye yönelik olmayan dillerdeki basit türlere çok benzer. Basit türler, tamsayılar, virgüllü değerler, karakter ve boolean değerler gibi atomik tek değerli ifadeleri göstermektedir. Bunların yanısıra Java'da bir de dizi türü vardır ki bunlar tek bir türün sabit boyutlu bütünüdür.

Java'da sekiz basit tür yer almaktadır: byte, short, int, long char, float, double ve boolean. Bunlar dört ana grupta toplanmaktadır:

- **Tamsayılar:** byte, short, int ve long tam değerli sayılardır.
- **Virgüllü sayılar:** float ve double virgüllü sayıları gösterir.
- **Karakterler:** char sembolleri harf ve basamağa benzer şekilde karakterlerle gösterir.
- **Boolean değerler:** boolean mantıksal değerleri gösteren özel bir türdür.

Bir programda çoğunlukla kullanılan tür tamsayılardır. Tamsayılar küçük sayma sayıları için temel olduğu için, kullanıcılar yasal aralığı belirlerken bu sayıları korkmadan kullanabilmektedir. Java'nın güvenilir ve güçlü olması nitelikle hazırlanmış

olan dilinden kaynaklanmaktadır. Java'da her bir ifadenin bir türü vardır, her değişkenin bir türü vardır ve her tür çok güçlü bir şekilde tanımlanmaktadır. Değişkenler için yapılan ifade atamalarının türünün uygunluğu denetlenir. Java derleyici, türlerin doğruluğunu sağlamak amacıyla derlediği tüm kodu denetler. Tür uygunsuzlukları derleyici uyarıları değildir, düzeltilmesi gereken hatalardır.

Sayısal Türler

Sayısal türler isminden de anlaşılacağı üzere sayılardan oluşur. Sayısal ifadenin sonucunu saklamak isterseniz, sayısal türde bir değişken hazırlamanız gerekir. İki tane sayısal tür vardır: Biri tamsayılar olarak bildiğimiz tam değerli sayılar, diğeri virgüllü sayılar dediğimiz kesirli kısımları saklanan sayılardır. Bu her iki tür için de duyarlılık derecesi değişebilmektedir.

Java, ANSI C'nin tüm tüm standart sayısal türlerini desteklemektedir. Ancak önemli bir fark vardır. Java'da kullanılan türlerin hepsi açık aralık ve matematiksel harekete göre tanımlanır. Çoğu C ya da C++ kodu, int türünün özel olmayan hareketini kapsayan sorunlarla karşılaşmaktadır. Bu sorunun temeli makine sözcük boyutu adı verilen düşünceden kaynaklanmaktadır. Verilen bir CPU'nun sözcük boyutu, temel yazmaçlarını göstermek için kullandığı bit sayısı ile belirlenir. 70'li yılların sonlarına doğru üretilen PC'ler 8 bitlik sözcük boyutuna sahiptir. 8086 PC'lerin gelmesiyle sözcük boyutu 16 bite yükseldi. 486'lar ve ardından çıkan Pentium makineler ile sözcük boyutu 32 bit oldu. Bugünlerde üretilen Pentium Pro ile UItraSPARC çiplerle sözcük boyutunu 64 bite taşıyoruz.

Tamsayılar

C ve C++ programcıları int türünü bit maskeleri ve grafik piksel değerleri gibi sayısal olmayan değerleri göstermek amacıyla kullanmaktadır. Bu nedenle işaretsiz olma düşüncesi yüksek dereceden bitin (bir int'in işaretini tanımlar) hareketini belirlemek için kullanılmıştı. Java bu düşünceyi ortadan kaldırmıştır. Tüm Java sayısal türleri işaretli değerlerdir.

Java'da işaretsiz bir türün olmaması tamsayı türlerini yarı yarıya azaltmaktadır. Bu türlerin herbiri 1, 2, 4 ve 8 baytlık belleği göstermektedir. Byte, short, int ve long olan bu türlerin doğal kullanma kategorisi vardır. Şimdi bu türlere kısa. bir göz atalım.

byte

Byte 8 bitlik işaretli türdür. Aralığı –128'den 127'e kadar olan bölgedir. Bir iletişim ağından ya da bir kütükten yabancı bir bayt akışı ile karşılaştığınız zaman, hu türü kullanmanız yararlı olacaktır.

Bayt değişkenler byte anahtar sözcüğü kullanılarak bildirilir. Örneğin aşağıda verilen b ve c olarak adlandırılmış iki bayt değişken bildiriliyor ve c ondalık basamak değeri 0x55'e yerleştiriliyor:

```
byte b;
byte c = 0x55;
```

short

Short 16 bitlik işaretli türdür. –32768'den 32767 aralığına sahiptir. .Java'nın türleri arasında en az kullanılandır.

Short değişkeni ile ilgili aşağıdaki örneği inceleyebilirsiniz:

```
short s;
short t = 0x55aa;
```

int

32 bitlik işaretli türdür. Sahip olduğu aralık –2,147,483,648'den 2,147,483,647'dir. Java'nın türleri arasında en çok kullanılandır. bu tür dizilerin yinelenmesinde ve sayılmasında çok kullanışlıdır.

int değişkenlerinin bildirilmesi ile ilgili örnek aşağıda verilmektedir:

```
int i;
int j = 0x55aa0000;
```

Bu türün çok yönlü ve verimli olmasından dolayı, dizilerin indekslenmesinde ya da matematiksel ifadelerin hesaplanmasında kullanabileceğiniz en ideal türdür.

long

long 64 bitli işaretli bir türdür. Yeterince geniş bir aralıda sahiptir. Bir int türünün istenen değeri tutamayacak kadar geniş olmadığı bir durum karşısında long, birtakım fırsatlar sunmaktadır. Büyük sayılara sahip tamsayı ifadelerin hesaplanmasında çarpma işlemi sonucunda trilyonlara varan ara değerler üretebilmektedir. Ayrıca süre hesaplandığı zaman, bir yıl milisaniye olarak ölçüldüğünde 30 milyarı aşmakta ve 32 bitlik bir int değeri geçmektedir. Böyle durumlarda kullanmanız gereken tür long'dur.

Aşağıdaki örneği inceleyin:

```
long m;
long n = 0x55aa000055aa0000;
```

Bir tamsayı (int) türünün genişliği kullandığı bellek miktarı ile ölçülmemelidir: bunun yerine bu türün ifadelerinin ve değişkenlerinin hareketi ölçüm olarak ele alınmalıdır.

Bu dört türün özelliğini genel olarak aşağıdaki tabloda gösterebiliriz:

Adı	Genişliği	Aralığı
Long	64	-9,223,372,036,854,775,808'den 9,223,372,036,854,775,807'e
İnt	32	-2,147,483,648'den 2,147,483.647'e
Short	16	-32,768'den 32.767'e
byte	8	-128'den 127'e

Virgüllü Sayılar

Gerçel sayılar olarak bilinen virgüllü sayılar (floating point numbers), kesirli duyarlılığa gereksinim duyan fonksiyonların hesaplanmasında kullanılır. Karakök ya da sinüs veya kosinüs gibi trigonometrik fonksiyonları içeren karmaşık hesaplamalar, duyarlılığı virgüllü türe gereksinim duyan bir değeri sunmaktadır. Java bu türün standart setini (IEEE-754) ve işletmenlerini çalıştırır. Bu tür biri float diğeri doıble olmak üzere iki gruba ayrılmaktadır. Bu iki grup aşağıdaki gibi tanımlanabilir:

Adı	**Genişliği**	**Aralığı**
Double	64	1.7e-308'den 1.7e+308'e
Float	32	3.4e-038'den 3.4e+038'e

float

float anahtar sözcüğü ile belirlenen tek duyarlılık bir değeri depolamak için 32 bit kullanır. Tek duyarlılık bazı işlemcilerde daha hızlıdır. Bazı kesirli değerlerin kaydedilmesine ihtiyacı olan basit hesaplamalar, özellikle tam sonuç istediğiniz zaman, float türü kullanmanızda yarar vardır.

float değişkeni gösteren örnek aşağıda verilmektedir:

```
float f;
float f2 = 3.14f;
```

double

Çift duyarlılık double anahtar sözcüğü ile gösterilmekte olup, bir değeri depolamak için 64 bite ihtiyaç duyar. Bazı modern işlemcilerde çift duyarlılık tek duvarlılıktan daha hızlıdır. Tüm trigonometrik fonksiyonların hesaplanmasında double değerler kullanılır.

Bu değişken için hazırlanan örnek aşağıda verilmektedir:

```
double d;
double pi = 3.14159365358979323846;
```

Karakterler

Java'da en küçük tamsayı türü byte'dır ve C ile C++'da bir char türde bulunan 8 bitlik değerleri depolar. Java bir dizilimdeki karakterleri göstermek için Unicode kullandığı için, char türü işaretsiz 16 bitten oluşur ve uluslararası Unhicoed karakter setinin binlerce karakterinden on tanesini depolamak için kullanılır. Bir char'ın aralığı 0 ile 65536 arasıdır. Negatif karakter yoktur. ASCII olarak bilinen standart karakter setinin aralığı 0 ile 127 arasıdır.

Not: Unicode hakkında daha fazla bilgi edinmek için http://www.unicode.org ve htlp://www.stonehand.com/unicode.html adreslerine başvurabilirsiniz.

char bildirimlerine örnekler aşağıda verilmektedir:

```
char c;
char c2 = 0xf132;
char c3 ='a';
```

```
char c4 ='\n';
```

Karakterleri tamsayı olarak kullanamasanız da tamsayı olarak işletebilirsiniz. İki karakteri birlikte toplayabilirsiniz ya da bir karakter değişkenin değerini artırabilirsiniz.

Aşağıda verdiğimiz örnekte taban karakterden başladık ve istediğimiz sayıyı gösteren bir tamsayıyı ekledik:

```
int iki = 2;
char bir ='1';
char üç = (char) (iki + bir);
```

Boolean Değerler

Java'da mantıksal değerler için adı boolean olan bir tür vardır. Olası olan iki değerden yalnızca birine sahip olabilir: Doğru (true) ve yanlış (false). Bunlar özel bir amaç için ayrılmış sözcüklerdir. Boolean, if, while ve do gibi koşullu akış kontrol işletmenleri tarafından gereksinim duyulan bir türdür.

Bir boolean aşağıdaki gibi bildirilir:

```
boolean done = false;
```

Şimdi bu türlerin her birinin yer aldığı basit bir örnek üzerinde çalışalım.

```
class Örnek {
public static void main(String args[ ] ) {
byte b = 0x55;
short s = 0x55ff;
int I = 1 000000;
long I = OxffffffffL;
char c ='a';
```

```
float f = .25f;
double d = .OOOO1234;
boolean b00l = true;
System.out.println("byte b = " + b);
System.out.println("short s = " + s);
System.out.println("int i ="+i);
System.out.println("long I ="+I);
System.out.println("char c ="+c);
System.out.println("float f ="+f);
System.out.println("double d ="+d);
System.out.println("boolean bool ="+ bool);
}
}
```

Bu programı çalıştırdığınız zaman, aşağıdaki çıktıyı alırsınız:

```
C:\>java Örnek
byte b = 85
short s = 22015
int I = 1000000
long I = 4294967295
char c = a
float f = 0.25
double d = 1234e-005
boolean bool = true
```

Diziler

Dizi benzer tür değişkenlerin oluşturduğu bir gruptur. Diziler önemli bir programla özelliğidir.

Diziler, aynı türün bir grup değişkenini bir araya getiren özel bir türdür. 12 tamsayısının bir dizisini bildirmek isterseniz, özel bir tür hazırlamanız gerekir, bu tür bir int dizisidir. Bu tür double dizisinden çok farklıdır. Kaşeli parantezler bir dizi türünü bildirmek için kullanılır. Aşağıdaki örnek int dizisinin yanısıra month_days değişkenini bildirmektedir:

İnt month_days[];

Diğer tüm değişken bildirimlerinde olduğu gibi month_days değeri varsayılan değer olan sıfıra yerleştirilir. Diziler için null (sıfır) adı verilen özel bir değer vardır. Null, değeri olmayan bir diziyi göstermektedir. Peki month_days'e değer nasıl verilir? Özel bir işletmen olan new işletmeni diziye boşluk yerleştirmek. için kullanılır. Bu işletmeni kullanmak için, öncelikle bir tür ve negatif olmayan bir tamsayı sağlamalısınız. Aşağıdaki örnekte month_days tarafından kullanılan bir diziye 12 tamsayıyı yerleştirir.

Month_days = new int[12];

Bu örneğe göre month_days 12 tamsayıdan oluşan bir diziyi kullanır. Köşeli parantezler dizi elemanlarına birer birer başvurulması için kullanılır. Dizi hazırlandıktan sonra, sıfırdan başlayarak dizi elemanlarına ulaşılır. Aşağıdaki örnekte her bir ayın günlerinden oluşan bir dizi hazırlanmaktadır:

```
class Örnek2 {
        public static void main(String args[]) {
int month_days[ ];
        month_days[0j = 31;
month_days[1] = 28;
month_days[2] = 31;
month_days[3] = 30;
month_days[4] = 31;
month_days[5] = 30;
```

```
month_days[6] = 31;
month_days[7) = 31;
month_days[8] = 30;
month_days[9] = 31;
month_days[10] = 30;
month_days[11] = 31;
System.out.println("April has"+ month_days[3] +" days.");
}
}
```

Bunun yanısıra basit türleri çalıştırmak için kullandığınız yolu kullanarak dizileri de otomatik olarak çalıştırabilirsiniz.

Aşağıdaki kod bir tamsayı dizisini hazırlamaktadır. Bir dizi kullanım başlatıcısı olan küme parantezleri ile çevrili virgülle ayrılmış ifadelerin listesidir. Virgül dizi elemanlarını birbirinden ayırmak için kullanılır.

```
class Dizi {
public static void main(String args[]) {
int month_days[ ] = { 31, 28, 31, 30, 31, 30, 31, 31,
30, 31, 30, 31 };
System.out.println("Nisan "+ month_days[3] + " days.");
                        }
                }
```

Çok Yönlü diziler

Java'da çok yönlü diziler vardır. Çok yönlü dizileri, dizilerin dizileri olarak düşünebilirsiniz. Çok yönlü diziler bloklara ayrılır. Örneğin X'den Y'e ve Z'ye olmak üzere üç boyutlu bir matris için gerekli olan bellek X, Y ve Z'nin çarpımı olup her bir hücreye depolanan tür boyutuna eşittir. Java'da bir değişkeni üç boyutlu olarak bildirebilirsiniz.

Aşağıdaki kod, her biri sıfır olan 16 double'ı hazırlamaktadır. Bu matris double dizilerinin dizisi şeklinde çalıştırılmaktadır.

double matrix[ ] [ ] = new double [4] [4];

Aşağıda verilen kod ise, farklı boyutların içiçe yerleştirilmiş dizilerini göstermek amacıyla aynı bellek miktarını işletmektedir.

```
double matrix[ ][ ]= new double [4] [ ];
matrix[0] = new double [4];
matrix[1] = new double [4];
matrix[2] = new double [4];
matrix[3] = { 0, 1, 2, 3 );
```

Şimdiki örneğimizde dörde dörtlük bir matris diagonal değerleri 1 olarak hazırlanmıştır.

```
class Matris {
public static void main(String args[]) {
double m[ ] [ ];
m = new double [4] [4];

m[0] [0] = 1;
m[1] [1] = 1;
m[2] [2] = 1;
m[3] [3] = 1;

System.out.println(m([0] [0] +" "+ m[0] [1] +" "+ m[0] [2] +" " + m[0] [3] );
System.out.printın(m([1] [0] +" "+ m[1] [1] +" "+ m[1] [2] +" " + m[1] [3] );
System.out.println(m([2] [0] +" "+ m[2] [1] +" "+ m[2] [2] +" " + m[2] [3] );
System.out.println(m([3] [0] +" "+ m[3] (1] +" "+ m[3] [2] +"  "+ m[3] [3] );
```

```
}
}
```

Bu programı çalıştırdığınız zaman, aşağıdaki çıktıyı elde edersiniz.

```
C:1> java Matris
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

Sıfır olmasını istediğiniz değerleri çalıştırmanız gerekmemektedir. Aşağıdaki program satır ve sütunun çarpımını içeren elemanlardan oluşan yeni bir matris üretmektedir.

```
class YeniMatris (
public static void main(String args[ ]) {

double m[ ] [ ] = {
        { 0*0, 1*0, 2*0, 3*0 },
{ 0*1. 1*1, 2*1, 3*1 },
{ 0*2, 1*2, 2*2, 3*2 },
{ 0*3, 1*3, 2*3, 3*3 },
};

System.out.printin(m([0] [0] +" "+ m[0] [1 ] +" "+ m[0] [2] +" " + m[0] [3] );
System.out.printın(m([1] [0] +" "+ m[1] [1 ] +" "+ m[1] [2] +" " + m[1] [3] );
System.out.printın(m([2] [0] +" "+ m[2] [1 ] +" "+ m[2] [2] +" " + m[2] [3] );
System.out.println(m([3] [0] +" "+ m[3] [1 ] +" "+ m[3] [2] +" " + m[3] [3] );
}
}
```

Bu programı çalıştırdığınız zaman, aşağıdaki çıktıyı elde edersiniz.

```
C:\> java YeniMatris

0 0 0 0
0 1 2 3
0 2 4 6
0 3 6 9
```

İŞLETMENLER

İşletmen (bir diğer adıyla operatör) nedir? Java işletmenleri özel karakterlerdir. Bu karakterler, işlenen üzerinde yapmak istediğiniz bir işlemi derleyiciye yöneltir. İşlem komutları işletmenler tarafından belirlenir. Peki işlenenler nelerdir? İşlenenler değişkenler, ifadeler ya da hazır bilgi değerleridir. Bazı işletmenler yalnızca bir işlenen üzerinde çalışır ki bunlara birli işletmenler adı verilir. Bazı işletmenler işlenenden önce gelir ki bunlara önek işletmenler denir. Bazı işletmenler ise işlenenden sonra gelir ki bunlara da sonek işletmenler denmektedir. Çoğu işletmen iki işlenen arasında yer alır. Bu tür işletmenlere araya yerleştirilmiş ikili işletmenler denir. Bu türler genellikle cebirde kullanılmaktadır.

Java'da 44 tane yerleşik işletmen vardır. Bu işletmenlerin her biri, matematiksel ya da mantıksal bir işlemi derleyiciye ulaştıran komuttur. Aritmetik, bitwise, ilişkisel ve mantıksal olmak üzere dört temel işletmen vardır. Bu bölümümüzde bu türlere ayrıntılı değineceğiz.

Aritmetik İşletmenler

Bu işletmenler matematiksel işlemlerde kullanılır. İşlenenler sayısal bir tür olmalıdır. Bu nedenle bu işletmenleri boolean türde kullanmanız mümkün değildir.

Ancak char türünü int türün alt kümesi olmasından dolayı kullanabilirsiniz.

İşletmen	**Sonuç**	**İşletmen**	**Sonuç**
+	toplama	+=	Toplayıcı atama
-	Çıkarma (ayrıca Birli çıkarma)	-=	Çıkarıcı atama
*	çarpma	*=	Çarpma ataması
/	bölme	/=	Bölme ataması
%	modülüs	&=	Modülüs ataması
++	artırma	--	Azaltma

Dört İşlem İşletmenleri

Toplama, çıkarma, çarpma ve bölme işletmenini bildiğiniz sayısal türlerde kullanabilirsiniz. Birli çıkarma işletmeni tüm işlenenlerin olumsuzunu alır. Aşağıda işletmenleri daha iyi görebilmeniz açısından basit bir örnek sunulmaktadır. Burada işletmenlerin diğer değişkenlerin yanı sıra hazır bilgi tamsayıları üzerinde çalıştığına dikkat edin.

```
class TemeIMath
        public static void main(String args[ ]) {
                int x=2+2;
int y = x * 3;
        int z=y/2;
        int k =-z;
        System.out.println("x ="+ x);
        System.out.println("y = " + y);
        System.out.println("z = " + z);
        System.out.println("k = " + k);
}
        }
```

Bu programı çalıştırdığınız zaman, aşağıdaki çıktıyı elde edersiniz:

```
C:1> java TemeIMath
x=4
y=12
z=6
k=-6
```

Modülüs İşletmenler

Modülüs ya da kısaca mod işletmeni (%) bir bölme işleminde kalanı sunar. C++'ın tam tersi olarak Java'da mod fonksiyonu virgüllü sayılar ve tamsayılar üzerinde çalışır. Mod işletmeninin nasıl çalıştığını görmek için aşağıdaki örneğimizi inceleyin.

```
class Modül {
public static void main(String args[ ]) {
int x = 42;
double y = 42.3;
System.out.println("x mod 10 = " + x % 10);
System.out.println("y mod 10 = " + y % 10);
}
}
```

Bu program aşağıdaki sonucu üretir.

```
C:1> java Modül
xmod10=2
ymodl0=2.3
```

Aritmetik İşletmen Atamaları

Her aritmetik işletmenin işlemle kurulu ataması olan birleşik bir şekli vardır. Örneğin aşağıda verilen şekli;

a=a+4:

ayrıca şu şekilde de kullanabilirsiniz:

a+=a;

Yine aynı şekilde;

a=a%2;

ifadesi şu şekilde de yazılabilir:

a%=2:

Bu şekil tüm ikili işlemler için geçerlidir.

Artırma ve Azaltma

Bu işletmenler C dilinde üretilmişti. Bir işlenenden 1 çıkarmak ya da 1 eklemek için kullanılmaktadır. Bu işlemin uzun şekli şöyledir:

x=x+1;

Aynı ifadeyi kısaca şu iki şekilde yazabilirsiniz:

x+= 1;

++X;

Aynı şekilde çıkarma işlemini de gerçekleştirebilirsiniz.

Tamsayı Bitwise İşletmenleri

Tamsayı sayısal türleri long, int. short, char ve byte'ın değerlerini oluşturan bitlerini değiştirebilen ve denetleyebilen işletmenleri vardır. Bu işletmenleri şu şekilde özetleyebiliriz:

İşletmen	**Sonucu**	**İşletmen**	**Sonucu**
~	Bitwise birli NOT		
&	Bitwise AND	&=	Bitwise AND ataması
|	Bitwise OR	|=	Bitwise OR ataması
^	Bitwise exclusive OR	^=	Bitwise exclusive OR ataması
>>	Sağa kaydırma	>>=	Sağa kaydırma ataması
>>>	Sağa sıfır doldurarak kaydırma	>>>=	Sağa sıfır doldurarak kaydırma ataması
<<	Sola kaydırma	<<=	Sola kaydırma ataması

Aritmetik işletmenlerin tersi olarak bitwise işletmenler bir değerdeki her bir bit üzerinde bağımsız olarak çalışır.

Bu tamsayı türlerinin hepsi bit genişlikleri değişik olan ikili sayılarla gösterilir. Örneğin ikili sisteme göre 42 sayısının bit değeri 00101010'dur. Her bir konum 2°'dan ya da 1'den (en sağdaki bit) başlayarak ikinin üssünü ifade eder. Sol taraftaki bit

konumu 2^1 ya da 2 olup, yine sol tarafa doğru ilerleyerek 2^2 ya da 4, sonra 8, 16, 32 vb. olarak devam eder. 42 sayısı 1, 3 ve 5 konumlarında 1 bit setine sahip olmasından dolayı 42, 2^1+2^3+2^5'in toplamı yani 2+8+32'dir.

Tüm tamsayı türleri, char türü hariç, işaretli tamsayılardır. Bu, pozitif değerlerin yanı sıra negatif değerlerin de olabileceği anlamına gelmektedir. Java ikinin tümleyeni olarak adlandırılan bir kodlama sistemini kullanır. Bir başka deyişle negatif sayılar ters çevrilerek ve ardından sonuca 1 ekleyerek gösterilir.

Örneğin -42 sayısı, 42 sayısındaki bitler ters çevrilerek gösterilir. 00101010 kodlaması ters çevrildiğinde 11010101 olur. Bu sonuca 1 eklendiğinde sonuç 11010110 olur.

NOT

Birli NOT işletmeni işlenenin tüm bitlerini olumsuz hale (tersine) dönüştürür. Örneğin aşağıdaki bit kalıbına sahip olan 42 sayısı:

00101010

NOT işletmeninden sonra şu şekle dönüşür:

11010101

AND

AND işletmeni, işlenenlerin her ikisi de 1 olduğunda 1 bit üreten bitleri birleştirir. Diğer durumlarda sıfır üretilir. Aşağıda verilen örneği inceleyin:

```
  00101010 42
& 00001111 15
= 00001010 10
```

OR

OR işletmeni, işlenenlerinden en az biri 1 olan bitleri birleştirir ve sonuç biti 1 olur. Aşağıda gösterilen örneği inceleyin:

```
  00101010 42
| 00001111 15
= 00101111 47
```

XOR

Bu işletmen, bir işlenen 1 olduğunda bitleri birleştirir, ancak her iki işlenen, 1 olduğunda sıfır sonucunu verir. Aşağıdaki örnekte gösterilmektedir:

```
  00101010 42
^ 00001111 15
= 00100101 37
```

Bit İşleme Örneği

Aşağıda verilen tablo, bitwise işletmenlerinin işlenen bitlerin birleştirilmesi üzerinde etkisini göstermektedir.

A	B	OR	AND	XOR	NOT A
0	0	0	0	0	1
1	0	1	0	1	0
0	1	1	0	1	1
1	1	1	1	0	0

Aşağıda sunulan örnekte yukarıda gösterilen işletmenlerin bir programda kullanımını görebilirsiniz.

```
class BitÖrnek {
                public static void main (String args [ ]) {
                        String binary[ ] = {
"0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111", "1000", "1001",
"1010", "1011", "1100", "1101", "1110", "1111",
};
int a = 3; // 0 + 2 + 1 ya da ikili sistemde 0011
int b = 6; // 4 + 2 + 0 ya da ikili sistemde 0110
int c=a|b;
int d = a & b;
int e=a^b;
int f = (~a & b) | (a &~b)
int g =~a & 0x0f;
System out.println("  a ="+binary[a];
System out.println("  b ="+binary[b];
System out.println("a|b ="+binary[c];
System out.println("a&b ="+binary[d];
System out.println("a^b ="+binary[e];
System out.println("~a&b|a&~b ="+ binary[f];
System out.println(" a & b = " + binary[d];
Systemout.println("  ~a ="+binary[g];
}
}
```

Bu program çalıştırıldığında elde edilen sonuç şöyledir:

```
C:\> java BitÖrnek
a = 0011
b=0110
a|b=0111
a&b = 0010
```

a^b = 0101
~a&6|a&~b = 0101
~a = 1100

Sola kaydırma

Sola kaydırma işletmeni sol işlenendeki bitlerin hepsini sağ işlenen tarafından belirlenen konumlara sol tarafa taşır. Sola kaydırdığınız her bit konumu için, çoğu bit işleneninin sol sonunda kaybolur ve bir kısmı sıfır biti işlenenin sağ tarafını doldurur. İşlenenler int ya da daha küçük olduğu zaman, ifadeler otomatik olarak int türe geçirilir. Eğer long işlenenler varsa ifadeler long türe geçirilir. Bu şu anlama gelmektedir: int işlenenleri üzerinde yapılan sola kaydırma ifadelerinde bitler 31 bit konumuna kaydırıldığında başlangıcı kaybolur. İşlenen bir long ise, 63. bit konumuna geçinceye kadar bitler korunur.

Sağa Kaydırma

Sağa kaydırma işletmeni sol işlenendeki bitleri sağ tarafa taşır. Aşağıda verilen örnekte kod parçası 32 değerini 2 bit konumu kadar sağ tarafa kaydırmaktadır.

```
int a = 32;
a=a>>2;
```

Bir değer sözcüğün sağ ya da sol tarafına kaydırılmış bitlere sahip olduğunda, bu bitler kaybolur. Bir sonraki kod parçası 35 değerini iki bit sağa kaydırır.

```
int a = 35;
a=a>>2;
```

Aynı işlem ikili sisteme göre aşağıdaki gibi yapılır:

00100011 35

```
>> 2
00001000 8
```

İşaretsiz Sağa Kaydırma

Sayısal bir değeri göstermeyen bir bit kalıbını işletiyorsanız, üst bitleri sıfırlara kaydırmak isteyebilirsiniz. Bu işlem özellikle grafik işlemlerinde maske hazırlamada çok yararlıdır. Buna işaretsiz kaydırma adı verilir. Java'nın sıfır doldurarak sağa kaydırma işletmeni bu durumda doğru şeyi yapar. Bu kod parçasında a –1'e yerleştirilir, 32 bit ikili sistemde 1'e yerleştirilir. Bu değer daha sonra 24 bit sağa kaydırılır ve üstteki 24 bit sıfırlarla doldurulur. Burada a 255 değerine yerleştirilir.

```
int a =-1;
a = a>>> 24;
```

Aşağıda verilen ise aynı işlemin ikili sisteme göre eşdeğeridir.

```
11111111111111111111111111111111      -1
>>>24
00000000000000000000000011111111      255
```

Eğer işaretli sağa kaydırmayı kullanarak -1 kaydırırsanız, sonuç her zaman -1 kalır. İşaretsiz sağa kaydırma işletmeni beklediğiniz kadar kullanışlı değildir. Daha küçük değerlerin otomatik olarak ifadelere geçirilmesinden dolayı yalnızca 32 ve 64 bit değerlerini etkiler. Aşağıda verilen örneği inceleyebilirsiniz.

```
class BaytÖrnek {
static public void main(String args[ ]) {
char hex[ ] = {
'0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
'a', 'b', 'c', 'd', 'e', 'f' };
byte b = (byte) 0xf1;
```

```
byte c = (byte) (b >> 4);
byte d = (byte) (b >>> 4);
byte e = (byte) ((b & Oxff >> 4);
System.out.println(" b = Ox"
+ hex[ (b >> 4) & OxOf] + hex[b & OxOf]);
System.out.println(" b >> 4 = Ox'
+ hex[ (c >> 4) & OxOf] + hex[c & OxOf]);
System.out.println(" b>>>4 = Ox"
+ hex[ (d >> 4) & OxOf] + hex[d & OxOf]);
System.out.println(" (b & Oxff)>> 4 = Ox'
+ hex[ (e >> 4) & OxOf] + hex[e & OxOf]);
}
}
```

Bu programın çıktısı şu şekildedir:

```
C:\>java BaytÖrnek
b = Oxf1
b>>4= Oxff
b>>> 4 = Oxff
(b & Oxff)  4 = OxOf
```

Bitwise İşletmen Atamaları

İkili bitwise işletmenleri cebirsel işletmenlerinkine benzer şekle sahiptir. Sonucun atanmasını sol taraftaki işlenen otomatik olarak gerçekleştirir. Aşağıdaki örnekte gösterilen ve değeri sağ tarafa dört bit kaydırmakta olan iki deyim birbirine eşittir.

```
a=a>>4;
a>>=4;
```

Aşağıdaki örnek program birkaç tane tamsayı değişken hazırlıyor ve ardından bu değişkenleri işletmek için bitwise işletmen atamalarının kestirme şekillerini kullanıyor.

```
class BitÖrnek2 {
public static void main(String args[]) {
int a = 1;
int b = 2;
int c = 3;
a |= 4:
b >>= 1;
c<<=1;
a ^= c;
System.out.println("a ="+ a);
System.out.println("b ="+ b);
System.oul.println("c = " + c);
}
}
```

Bu program çalıştırıldığında şu sonuç üretilir:

```
C:\> java BitÖrnek2 {
a=3
b=1
c=6
```

İlişkisel İşletmenler

Java'da iki değeri karşılaştırmak amacıyla, eşitliği ve sıralamayı tanımlayan ilişkisel işletmenler vardır.

Bu işletmenleri aşağıdaki şekilde tablolayabiliriz:

İşletmen	Sonucu
==	Eşittir
!=	Eşittir değil
>	-den büyük
<	-den küçük
>=	-den büyük ya da eşit.
<=	-den küçük ya da eşit

Java'da herhangi bir tür (bunlar tamsayılar, virgüllü sayılar, karakterler, boolean değerler ve başvurular olabilir) eşitlik (==) ve eşitsizlik (!=) testi kullanılarak karşılaştırılabilir.

Yalnızca. sayısal türler sıralama türleri kullanılarak karşılaştırılabilir. Tamsayılar, virgüllü sayılar ve karakter işlenenleri bir diğerinden küçük ya da büyük olduğunu görmek amacıyla karşılaştırılabilir.

Boolean Mantıksal İşletmenleri

Boolean mantıksal işletmenleri aşağıdaki tabloda özetlenmektedir. Tüm ikili mantıksal işletmenler iki boolean değeri birleştirir ve sonuç yine bir boolean değer olur.

İşletmen	Sonucu	İşletmen	Sonucu
&	Mantıksal AND	&=	AND atması
|	Mantıksal OR	|=	OR ataması
^	Mantıksal XOR (exclusive OR)	^=	XOR ataması
||	Kısa devreli OR	==	Eşitlik
&&	Kısa devreli AND	!=	Eşitsizlik
!	Mantıksal birli NOT	?:	İf-then-else üçlüsü

Mantıksal boolean işletmenleri AND, OR ve XOR boolean değerler üzerinde çalışır. Mantıksal NOT işletmeni boolean deyimini ters çevirir, bir başka deyişle, !false (yanlış) == true (doğru) ve !true (doğru) ==false (yanlış) olur.

A	B	OR	AND	XOR	NOT A
Yanlış	Yanlış	Yanlış	Yanlış	Yanlış	Doğru
Doğru	Yanlış	Doğru	Yanlış	Doğru	Yanlış
Yanlış	Doğru	Doğru	Yanlış	Doğru	Doğru
Doğru	Doğru	Doğru	Doğru	Yanlış	Yanlış

Aşağıda bir örnek program verilmektedir. Daha önce yazdığımız BitLogic örneğimize benzemektedir. Bu kez ikili bitlerin yerine boolean mantıksal değerleri kullanılmıştır.

```
class BoolMantık {
public static void main (String args [ ]) {
boolean a = true;
boolean b = false;
boolean c = a|b;
boolean d = a & b;
boolean e = a ^ b;
boolean f = (!a & b) | (a & !b);
boolean g = !a;
System out.println(" a ="+a);
System out.println(" b ="+b);
System out.println(" a|b ="+c);
System out.println(" a & b ="+d);
System out.println(" a ^ b ="+ e);
System out.println(" !a&b|a&!b ="+f);
System out.println(" !a ="+g);
}
}
```

Bu programı çalıştırdığınız zaman, ayın mantıksal kuralların bitlerde olduğu gibi boolean değerler üzerine de uygulanabildiğini görebilirsiniz. Bir Java boolean değerinin dizilim gösterimi true (doğru) ya da false (yanlış) mantıksal sonucundan biridir.

```
C:\>java BoolMantık
a = true
b = false
a | b = true
a & b = false
a ^ b = true
!a&b|a&!b= true
!a = false
```

Kısa Devre Mantıksal İşletmenler

Boolean işletmenlerinde iki ek işletmen daha vardır. Bunlar boolean AND ve OR işletmenlerinin ikincil uyarlamaları olup, kısa devreli mantıksal işletmenler adı verilir. Tablodan da anlayacağımız üzere işlenenlerden herhangi birinin doğru olması durumunda OR işletmeni true (doğru) mantıksal sonucunu üretmektedir. Yine aynı şekilde işlenenlerden herhangi birinin yanlış olması durumunda AND işletmeni yanlış (false) mantıksal sonucunu üretmektedir.

| ve & yerine || ile && şekillerini kullanırsanız, ifade çıktı sol işlenenden sağlandığı zaman, Java sağ taraftaki işleneni değerlendirmeye almayacaktır. Şağ taraftaki işlenenin mantıksal sonucunun doğru ya da yanlış olması sol taraftaki işlenenlerden birine bağlı olduğunda bu oldukça yararlı bir özelliktir.

Aşağıda verilen kod parçası bir bölme işleminde kısa devre mantıksal değerlendirme işleminin avantajından nasıl yararlanabileceğinizi göstermektedir.

```
if (denom != 0 && num / denom > 10)
```

AND işletmeninin kısa devre şekli kullanıldığı için, sıfırla bölme işlemi yapmaya çalıştığınız zaman herhangi bir çalışma zamanı olağan dışı durumu ile karşılaşma olanağınız olmayacaktır. Ancak aşağıdaki örneği incelediğinizde her iki tarafta değerlendirmeye alındığında bir çalışma zamanı olağan dışı durumunun ortaya çıktığını görebilirsiniz.

```
if (denom !=0 & num / denom > 10)
java.lang.ArithmeticException: / by zero
```

if-then-else İşletmeni

C ve C++'da yer alan aynı üç yollu işletmenler Java'da da bulunmaktadır. Kullanımı yaygın olan bu işletmenin genel şekli şöyledir:

```
ifade ? deyim1 : deyim2:
```

Burada ifade (expression), herhangi bir ifadeyi bir boolean değer olarak değerlendirebilir. Sonuç doğru olduğunda deyim1 çalışır, tersi durumda deyim2 çalışır.

Hangi ifadenin değerlendirileceğini seçmek amacıyla bazı değerlerin test edilmesinde oldukça düzgün çalışır. Şu örnekte olduğu gibi:

```
ratio = denom == 0 ? 0 : num / denom;
```

Java bu kod parçasını değerlendirmeye aldığı zaman soru işaretinin sol tarafındaki ifadeye bakar. Eğer denom sıfıra eşitse, soru işareti ile iki nokta arasındaki ifade değerlendirmeye alınır ve tüm ifadenin değeri olarak kullanılır. Eğer denom sıfıra eşit değilse, iki noktadan sonra gelen ifade değerlendirmeye alınır ve tüm ifadenin değeri olarak kullanılır.

Aşağıda verilen örnekte bu işletmen kullanılarak ifadenin değerlendirilmesinden önce paydanın sıfır olup olmadığı denetlenmektedir.

```
class Örnek3 {
public static void main(String args[ ]) {
int a = 42;
int b = 2;
int c = 99;
int d = 0;
int e = (b==0)? 0:(a/b);
int f = (d==0)? 0:(c/d);
System.out.println("a = " + a);
System.out.println("b = " + a);
System.out.println("c = " + a);
System.out.println("d = " + a);
System.out.println("a / b = " + e);
System.out.println("c / d = " + f);
}
}
```

Bu programı çalıştırdığınız zaman şu sonucu alırsınız.

```
C:\>java Ornek3
a=42
b=2
c=99
d=0
a/b=21
c/d=0
```

f'ye atama yapan satır bu işletmen kullanılmadan yazılsaydı, aşağıda gösterildiği gibi;

int f =c/d; bir çalışma zamanı olağan dışı durumu gerçekleşirdi:

```
C:\> java Örnek3
java.lang.ArithmeticException: / by zero
        at Ternary.main(Ternary.java:8)
```

İşletmenlerin Önceliği

Bir ifadede kesin bir sıra ya da işlem önceliği vardır. Örneğin cebirde çarpma ve bölme öncelik sırasına sahip olan işlemler olup toplama ve çıkarma ardından gelir. Bu düzen bilgisayar programlama dillerinde de geçerlidir. Ancak programlama sırasında kullanılan çok sayıda işlem vardır. Bu nedenle bu öncelik sırasını görebilmeniz açısından aşağıdaki tabloyu sunuyoruz.

En yüksek derece			
( )	[ ]	.	
++	--	~	!
*	/	%	
+	-		
>>	>>>	<<	
>	>=	<	<=
= =	!=		
&			
^			
|			
&&			
||			
?:			
=	op=		
En düşük derece			

AKIŞ KONTROLÜ

Akış kontrolü, bir programlama dilinin çalışma akışına neden olan yoldur. Dallama, yineleme, dağıtım ve alt programlar hepsi birer akış kontrolüdür. Java'da akış kontrolü C ve C++'da olduğu gibi hemen hemen eşdeğerdir. Bu benzerlikten dolayı çoğu C ya da C'++ programları oldukça kolay bir yolla Java programlarına dönüştürülebilir. Programınızı çalıştırma kontrolünü yapabilmeniz için gereken tüm yollar bu bölümümüzde verilmektedir.

Dallara Ayırma

Şimdiye kadar gördüğünüz örnekler tümüyle doğrusaldı. Deyimler bitinceye kadar birer birer çalıştırılır. Çoğu programlar belli deyimlerin çalıştırılmasına gereksinim duyar. Java, çalıştırdığınız kod parçanızın kontrolünü başarmanız için birkaç mekanizma sunmaktadır. Bunların herbiri incelenmektedir.

if-else

İf-else kurulumu, farklı kod parçalarının çalıştırılmasına neden olan boolean durumlarının akışını sağlar. Bir if-else deyiminin genel şekli şöyledir:

if (boolean-ifade) deyim1; [else deyim2;]

else yantümcesi isteğe bağlıdır. Deyimlerin her biri küme parantezlerine alınmış bir bileşke deyimdir. Bir boolean-ifade sonucu bir boolean değer olan ifadedir. Aşağıdaki kod parçasında gösterildiği gibi, boolean olarak bildirilmiş basit bir değişken olabilir.

```
boolean dataAvailable;
// ...
if (dataAvailable)
ProcessData( );
```

```
else
        waitForMoreData( );
```

Boolean-ifade, aşağıda gösterildiği gibi bir karşılaştırma sonucunu üretmek amacıyla ilişkisel operatörleri kullanan bir ifade olabilir.

```
int bytesAvailable;
//...
if (bytesAvailable > 0)
ProcessData( );
else
        waitForMoreData( );
```

Unutmayın, if ya da else'den sonra yalnızca bir deyim görünebilir. Daha fazla deyim yerleştirmek istediğiniz zaman, küme parantezlerini kullanmanız gerekir.

```
int bytesAvailable;
//...
if (bytesAvailable > 0) {
ProcessData();
bytesAvailable -= n;
} else
waitForMoreData();
```

Çoğu programcı, her bir yantümcede yalnızca bir deyim olmasına rağmen, küme parantezlerini ({}) kullanmanın daha güvenli olduğu görüşündedir.

break

Java'da bir goto deyimi yer almamaktadır. Diğer dillerde goto keyfi olarak dallara ayırmak için kullanılan bir yoldur, üstelik anlaşılması oldukça da güçtür ve belli derleyici düzenlemelerine de izin vermemektedir. Buna karşın goto'nun yararlı olduğu

kullanım alanları da vardır. Özellikle akış kontrolünün uygun bir şekilde gerçekleştirilmesini sağlamaktadır. Java'da bulunan break deyimi ise bu tür durumlar için tasarlanmıştır. Break, bir kod bloğunu kesme görevini üstlenir.

Bir bloğu adlandırmak için, yasal bir tanıtıcı seçin ve etiketlendirmek istediğiniz bloktan önce yerleştirin. Ardından iki nokta (:) karakterini ekleyin. Daha sonra bu etiketi bir argüman olarak kullanabilirsiniz. Aşağıda verilen örneği inceleyin. Bu program içiçe yerleştirilmiş üç bloğu göstermektedir. Her bir blok kendi tanıtıcısı ile etiketlendirilmiştir.

```
class Break {
public static void main(String args[ ]) {
boolean t = true;
a:        {
b:                  {
c:                          {
System.out.println("Kesmeden önce");
if (t)
break b;
System.out.println("Çalıştırmaz");
}
System.out.println("Çalıştırmaz");
}
System.out.println("b'den sonra");
}}}
```

Bu programı çalıştırdığınız zaman, break'in çalışma şeklini görebilirsiniz.

```
C:\> java Break
Kesmeden önce
b'den sonra
```

switch

switch deyimi, tek bir değişken ya da ifadeye bağlı bir kodun farklı parçalarını dağıtmanın daha güvenli bir yolunu sunmaktadır. Bu deyimin genel formu şöyledir:

```
switch(expression){
case valuel:
break;
case value2:
break;
case valueN
break;
default:
}
```

Burada yer alan expression (ifade), herhangi basit bir türü sunabilir ve case deyimlerinde belirtilen her bir değer uyumlu bir tür olmalıdır. Her değer birbirinden farklı hazır bilgi olmalıdır. Yedek durum değerleri sağlarsanız, derleyici hata verir. Switch deyimi şu şekilde çalışır: İfadenin değeri case deyimlerinde yer alan her bir hazır bilgi değerleri ile karşılaştırılır. Bir uygunluk bulunursa. o case deyimini izleyen kod dizisi çalışır. Case deyimlerinden herhangi biri ifadenin değeri ile uygun olmazsa default deyimi çalışır. Aslında default deyiminin kullanımı isteğe bağlıdır.

Break anahtar sözcüğü çoğunlukla etiketsiz switch deyimlerinde kullanılır. Bir switch deyiminde etiketsiz bir break'in kullanımı, switch deyiminin sonuna geçişi sağlayan çalışma işlemini gerçekleştirir.

Aşağıda verilen örneği inceleyin. Bir girdi diziliminde yer alan farklı karakter kodlarına bağlı olarak göndermek amacıyla kullanılan switch deyimini göstermektedir. Bu program, basit bir metin diziliminin karakterlerini, satırlarını ve sözcüklerini saymaktadır. UNIX wc (sözcük sayma) komutuyla eşdeğerdir.

```
class SözcükSayma {
        static String text = "Ülkeye yararlı bir \n" +
    "insan olmak için\n" +
    "vergi ödemenin şimdi\n"
    "tam zamanıdır\n";
static int len = text,length( );
static public void main(String args[ ]) {
boolean inWord = false;
int numChars = 0;
int numWords = 0;
int numLines = 0;
for (int I = 0; I < len; I++) {
char c = text.charAt(i);
numChars++;
switch (c) {
        case '\n':
numLines++;
// FALLSTHROUGH
case'\t': // FALLSTHROUGH
case'':
        if (inWord) {
numWords++;
inWord = false;
}
break;
default:
inWord = true;
}}
System.out.println("\t" + numLines +
                    "\t" + numWords + "\t"  + numChars);
}}
```

Bu sözcük sayma programı dizilimlerle ilgili birkaç kavram kullanır. Örnekte text.lenght(), text diziliminin uzunluğunu bir tamsayı olarak sonuçlandırmaktadır. Buna keza text.charAt(i), text dizilimindeki i. karakteri çıkarır. Bu karakterlerin her biri numChars++ ile sayılır. Yeni satır karakterlerini ('\n') bulmak için kullanılan switch deyimi, numLines aracılığı ile girdi diziliminin satır sayısını belirler. Kod, numWords ile sözcükleri saymaktadır. Örneğimizde yer alan default durumu tümn karakterler için kullanılmış olup, tüm karakterlerin bir sözcüğün parçası olduğunu göstermek amacıyla inWord değişkenini yerleştirmektedir.

return

İleriki bölümlerimizde de göreceğiniz gibi, Java'da yöntem adı verilen alt programlar kullanılmaktadır. Yöntem, sınıfları nesnelleştirmek için işlemsel bir arabirimi çalıştırır. Örneklerimizde kullandığımız main aslında bir statik yöntemdir. Bir yöntemin herhangi bir yerinde kullanılan return deyimi, yöntem çağırıcısının çalışmasına neden olur. Aşağıdaki örnek return deyimi çağırıcının return'e dönmesini sağlamaktadır.

```
class ReturnÖrnek
public static void main(String args[ ]) {
boolean t = true;
System.out.println("return'den önce");
if (t)
return;
System.out.println("Çalışmaz");
}
}
```

Döngü

İşlemi bitiren koşullar karşılaşıncaya kadar, sürekli aynı kod bloğununun çalıştırılmasına döngü adı verilir. Bir döngü dört kısımdan oluşur: Giriş (initialization),

gövde (body), yineleme (iteration) ve bitiş kısmı (termination). Bir döngünün başlangıç koşullarını kuran kod döngünün giriş kısmını oluşturur. Gövde, tekrarlamak istediğiniz deyimdir. Gövdeden sonra döngüden önce çalıştırmak istediğiniz kod, döngünün yineleme bölümüdür. Yineleme genellikle sayaçları artırmak ve azaltmak için kullanılır. Döngünün bitiş bölümü bir boolean ifade olup, döngüyü durduran deyimi görmek amacıyla döngüyü denetler. Java'da üç tane döngü kurulumu vardır. Bunlar while, do-while ve for'dur.

while

While döngüsü Java'nın en temel döngü deyimidir. Basitçe yerleştirilen while tek bir deyim üzerinde sürekli çalışır ta ki doğru (true) olan bir boolean ifade ile karşılaşıncaya dek. Bu deyimi aynı zamanda küme parantezlerini kullanarak bir deyim bloğu şeklinde de yerleştirebilirsiniz. While'ın genel şekli şöyledir:

```
[giriş;]
while ( bitiş ) {
gövde;
[yineleme]
}
```

Giriş ve yineleme bölümleri isteğe bağlıdır. Bitirme bölümü doğru sonucunu buluncaya kadar gövde deyimi çalışmaya devam eder. Aşağıdaki örnek while döngüsün'den saymaya başlayarak on satırlık işareti basmaktadır.

```
class WhileÖrnek {
public static void main(String args[ ]) {
int n=10;
while (n > 0) {
System.out.println("işaret " + n);
n--;
}}}
```

Bu programı çalıştırdığınız zaman, şu çıktıyı elde edersiniz:

```
C:\> java WhileÖrnek
işaret 10
işaret 9
işaret 8
işaret 7
işaret 6
işaret 5
işaret 4
işaret 3
işaret 2
işaret 1
```

do-while

Bazen boolean ifade yanlış sonucunu verse bile, bir while döngüsünün gövdesinin en az bir kez çalıştırılması istenir. While döngüsündeki gibi bitirme ifadesini başlangıç kısmında değil de döngünün sonunda test etmek daha verimli olabilir. İşte do-while döngüsü tam olarak bu koşulu kurmaktadır. Genel şekli şöyledir:

```
[giriş;]
do {gövde; [yineleme;]} while (bitiş);
```

Aşağıda verilen örnekte bitirme bölümünün ilk kez değerlendirmeye alınmasından önce gövde kısmı çalıştırılmaktadır. Böylelikle yineleme ve bitirme bölümleri birleştirilebilir.

```
class DoWhileÖrnek {
public static void main(String args[ ]) {
int n=10;
```

```
do {
System.out.println("işaret " + n);
} while (--n > 0);
}
}
```

Örnekte yer alan (--n > 0) ifadesi do-while döngüsünün bitirme bölümünün test edilmesi için kullanılmaktadır.

for

For deyimi bir döngüyü açıklamak için kullanılan sıkıştırma bir yoldur. Döngüyü oluşturan dört parça for deyimi ile birbirine oldukça yakındır. Temel şekli şöyledir:

```
for (giriş ; bitirme ; yineleme) gövde;
```

Gördüğünüz gibi for ve while deyimleri aynı temel yapıyı oluşturmalarına rağmen birbirinden farklı ifadelerdir. Bir for döngüsü, while döngüsüne bağlı olarak ifade edilebilir. Başlangıç koşulları bitirme bölümünün birinci kezde doğru sonucu sunmasını sağlamıyorsa, gövde ve yineleme deyimleri asla çalışmaz. Yasal bir for döngüsü küçükten büyüğe tamsayıları sayar.

```
class ForÖrnek {
public static void main(String args[ ]) {
for (int i = 1; I <= 10; i++)
System.out.println("i = " + i);
                }
        }
```

Program çalıştırıldığında elde edilecek çıktı aşağıda verilmektedir:

```
C:\> java ForÖrnek
i=1
i=2
i=3
i=4
i=5
i=6
i=8
i=9
i=10
```

While ve do-while örneklerinden alınan işaret programı aşağıdaki şekilde for döngüsü için uyarlanabilir:

```
class ForİşaretPublic static void main(String args[ ]) {
for (int n = 10; n > 0; n--)
System.out.println("işaret " + n);
}}
```

Değişkenleri for deyiminin giriş kısmına bildirebilirsiniz. Aşağıda verilen örnek yukarıdaki örnekle aynı işleve sahiptir.

```
class For {
public static void main(String args[ ]) {
{
int n;
for (n = 10; n > 0; n--)
System.out.println("işaret " + n);
}}}
```

Aşağıda verilen mevsim örneğini inceleyin. Örnekte for döngüsünün genel şekli kullanılmıştır.

```
class Aylar
static String aylar[ ] = {
"Ocak", "Subat", "Mart", "Nisan", "Mayıs", "Haziran", "Temmuz", "Agustos", "Eylul",
"Ekim", "Kasım", "Aralık"
};
static int günler[ ] = { 31 , 28, 31, 30, 31 , 30, 31 , 31, 30,
31,30,31};
static String ilkbahar = "ilkbahar";
static String yaz = "yaz";
static String sonbahar = "sonbahar";
static String kıs = "kıs";
static String mevsimler[ ] = {
kıs, kıs, ilkbahar, ilkbahar, ilkbahar, yaz, yaz,
yaz, sonbahar, sonbahar, sonbahar, kıs
};
public static void main(String args[ ]) {
for (int ay = 0 < 12; ay++) {
System.out.println(gunler[ay] + "günlü " +
      aylar[ay] + " bir " +
      mevsimler(ay] + " aydır. ");
}}}
```

Bu program aşağıdaki çıktıyı üretir.

```
C:\> java Aylar
Ocak 31 gunlu bir kıs ayıdır.
Subat 28 gunlu bir kıs ayıdır.
Mart 31 gunlu bir ilkbahar ayıdır.
```

Nisan 30 gunlu bir ilkbahar ayıdır.
Mayıs 31 gunlu bir ilkbahar ayıdır.
Haziran 30 gunlu bir yaz ayıdır.
Temmuz 31 gunlu bir yaz ayıdır.
Agustos 31 gunlu bir yaz ayıdır.
Eylul 30 gunlu bir sonbahar ayıdır.
Ekim 31 gunlu bir sonbahar ayıdır.
Kasım 30 gunlu bir sonbahar ayıdır.
Aralık 31 gunlu bir kıs ayıdır.

Virgül Deyimleri

Giriş ve yineleme bölümlerine birden fazla deyim yer(eştirmek istediğiniz ortamlar olabilir. Ancak bir for deyiminde küme parantezlerini kullanarak bileşik bir blok deyimini başlatamayacağınız için, Java tarafından sunulan alternatif bir yol bulunmaktadır. Bir for deyiminin parantezleri içerisinde virgül kullanarak birden fazla deyimi ayırabilirsiniz.

Aşağıda verilen örnekte giriş ve yineleme bölümlerinde birden fazla deyim kullanılmıştır.

```
class Virgul {
public static void main(string args[ ]) {
int a, b;
for (a = 1 , b = 4; a < b; a++, b--){
System.out.println("a = " + a);
System.out.println("b = " + b);
}
}
}
```

Bu örnek öncelikle giriş deyimlerini çalıştırır, sonra bitirme bölümünü ve ardından gövde kısmını işletir. Virgülle ayrılmış iki yineleme deyimi çalıştırılır ve bu şekilde döngü devam eder. Bu örnekte döngü iki kez çalıştırıldığında aşağıdaki çıktı elde edilir.

```
C:\> java Virgul
a=1
b=4
a=2
b=3
```

continue

Bir döngüden vaktinden önce çıkmak isteyebileceğiniz gibi döngünün devam etmesini de isteyebilirsiniz. Ancak durdurma işlemi yineleme için kodun kalan kısmıdır. Gerçi bu bir döngünün gövde kısmına geçişi sağlayan goto deyimidir. Java'nın continue deyimi tam olarak bu işlevi gerçekleştirmektedir. While ve do-while döngülerinde kontrol continue deyiminden transfer edilerek gövdenin kalan kısmından bitirme ifadesine geçirilir. Bir for döngüsünde for deyiminin üçüncü aşaması continue deyiminden sonra çalıştırılır. Aşağıda verilen örnek programda görebileceğiniz gibi continue deyimi iki sayının her bir satır üzerine basılmasını sağlamaktadır.

```
class ContinueDemo {
public static void main(String args[ ]) {
for (int i = 0; i < 10; I++) {
System.out.print(i + " ");
if (I % 2 ==0)
continue;
System.out.println(" ");
}
}
}
```

Bu kod yeni bir satır yerleştirmeksizin System-out-print komutunu kullanmıştır. İndeksin çift olup olmadığını ya da bir başka deyişle 2 ile bölünür olup olmadığını denetlemek için mod işletmenini (%) kullanmıştır. Çift olduğunda döngü, yeni satır basmadan devam etmektedir.

Bu programın çıktısı şöyledir:

```
C:\> java ContinueDemo
0 1
2 3
4 5
6 7
8 9
```

JAVA PAKETLERİ

Temel ilke olarak Java, nesne tabanlı programlama, iletişim ağı ortamı ve çok yönlü sistemin gereksinimlerine karşılık verebilmek amacıyla tasarlandığı için, çok sayıda sınıf ve arabirimi bünyesinde taşımaktadır. Bu sınıf ve arabirimleri bu bölümümüzden başlayarak ayrıntılı bir şekilde bulabilirsiniz.

Bir Java programı kaynak kütükleri farklı birkaç Java sınıfından oluşabilir. Sınıf bir objenin yapısını, bir diğer deyişle şekil ve hareketini ve yöntem olarak adlandırılan işlevselliğini tanımlar. Bir Java programı yazılıp çalıştırılmaya başlandıktan sonra sistem, sınıf tanımlarını kullanarak objeleri yani sınıf örneklerini hazırlar. Genel olarak bir sınıfın tanımı şöyledir:

```
class sınıfınadı extends üstsınıfınadı {
tür örnek-değişken:;
tür yöntem(parametre){
yöntem-gövde
}
}
```

Burada adı geçen extends anahtar sözcüğü, sınıfadı'nın üstsınıfınadı'nın alt sınıfı olduğunu göstermektedir.

Bir sınıf kurucu ve yöntemlerle tanımlanabilir. Yöntemi bir sınıfın işlevsel arabirimi ve sınıf tanımlarına iliştirilmiş alt yordam olarak tanımlayabiliriz. Sınıf tanımları içerisinde kullanılan yöntemlerin genel şekli şöyledir:

```
tür yöntem(parametre ya da liste){
yöntem-gövde;
}
```

Esasen kurucular da birer yöntemdir. Bir kurucu sınıfı ile aynı isme sahiptir. Obje hazırlandıktan sonra kurucu otomatik olarak çağrılır.

java.lang Paketi

Java.lang paketi, Java dilinin ve Java Sanal Makinenin çekirdeğini oluşturan sınıf ve arabirimleri sunmaktadır. Örneğin Object, String ve Thread hemen her programda kullanılmaktadır. Diğer java.lang sınıfları Java Sanal Makinede çıkabilecek olağan dışı durumları ve durumları tanımlamaktadır.

ClassLoader, Process, Runtime, SecurityManage ve System gibi sınıflar sistem kaynaklarına erişimi sağlamaktadır.

Java.lang paketi tüm JAVA programlarına otomatik olarak gönderilebilir. JAVA'nın en temel düzeyidir ve bu dilin çekirdeğini oluşturan sınıf ve arabirimleri sunar.

Java.lang paketinde yer alan sınıf ve arabirimleri genel olarak şu şekilde gruplandırabiliriz:

Temel sınıflar arasında yer alan Object, Class, String ve StringBuffer sınıfları hemen her program tarafından kullanılır.
Boollean, Charecter, Double Float, Integer, Long ve Nummer sınıfları kap (container) sınıflarıdır. Daha önceki türleri korumak için kullanılır.

ClassLoader, Math, Process, Runtime, SecurtyManager ve System sınıfları sistem fonksiyonlarına ve kaynaklarına erişmeyi sağlar.

java.io Paketi

Kütüklere veya I/O kaynaklarına veri yazmaya ya da okumak için kullanılan girdi ve çıktı (I/O) akışlarının setini hazırlamaktadır.

JAVA akışları bayta yöneliktir. Burada tanımlanan sınıflar, daha karmaşık akış işlevselliğini gerçekleştirmek için zincirleme olarak kullanılabilir.

I/O (input/output) sözcüklerinden de anlaşılacağı gibi bu paket girdi ve çıktı işlevlerini kapsamaktadır. Tek düze akış modeli sağlayan Java.io paketi, verinin kütüklere yazılamasını ve okunmasını sağlar ve diğer girdi ile çıktı kaynaklar için kullanılan girdi ve çıktı akış kümesini sunar.

Bir kütük sistemine ya da iletişim ağı veya bir girdi aygıtına başvurduğunuz zaman, gereksinim duyacağınız tek şey InputStream ve OutputStream objelerini kullanmak olacaktır.

Bu pakette sınıflar birkaç kategoriye ayrılmaktadır:

InputStream ve OutputStream sınıfları sistemdeki genel girdi/çıktı akışını gerçekleştirir. Girdi ve çıktının süzme işlemini gerçekleştiren FilteredInputStream ve FilteredOutputStream sınıfları girdi ve çıktı işlevselliğinin etkilenmesini de sağlamaktadır.

java.util Paketi

Java.util paketi, destek sınıfları ve ilgili arabirimleri sunmaktadır. Dictionary, Hashtable, Stack, Vector gibi geniş kapsamı olan veri yapılarını, StringTokenizer gibi dizilim işletme ve Date gibi tarih ve takvim hizmet programlarını temin eden sınıfları kapsamaktadır.

Ayrıca bu pakette Observer arabirimi ile Observable sınıfı da yer almaktadır. Bu sınıf ve arabirim aracılığı ile objeler, değişikliğe uğradıklarını ve bir başka objeye rahatlıkla bildirebilmektedir. JAVA'nın destek paketi olup karmaşık veri yapıları ve bu yapıların yöntemlerini kapsamaktadır. Genel olarak veri yapıları, denetim tabloları, yığınlara ve dizilere benzemektedir.

java.net Paketi

Java.net paketinde iletişim ağı çalışmalarını sağlayan sınıf ve arabirimler yer almaktadır. JAVA İnternet'in TCP / IP protokolünü hem önceden kurulmuş olan akış I / O arabirimini çıkararak hem de I / O objeler kurmak için gereken özellikleri ekleyerek desteklemektedir. Bunun yanı sıra bir URL ve URL bağlantısını gösteren sınıflar ile soket bağlantısını ve İnternet adresini gösteren sınıflarda bulunmaktadır.

Girdi / çıktı akışları ve Internet adresleri bir başka sistemden alınabilir ya da gönderilebilir. Java.net paketi, bir uygulamanın bilgiyi iletişim ağına aktarmasını kolaylaştıran sınıf ve arabirimleri kapsamaktadır. TCP ve UDP'nin her ikisini desteklemektedir. Web'deki bir bilgiye Tekdüze Kaynak Yerleştiriciler kullanılarak kolayca erişebilir.

java.awt Paketi

Java.awt paketi standart grafik kullanıcı arabirimi elemanlarını sunmaktadır. Bu elemanlar düğme, liste, menü ve metin alanlarıdır. Ayrıca pencere ve menü çubukları gibi kaplar ile kütüklerin açılması ve saklanması için kullanılan diyalog pencereleri gibi yüksek düzey bileşenleri de içermektedir. AWT (Abstract Window Toolkit – Soyut Pencere Araçkiti) serisi içerisinde java.awt.image ve java.awt.peer olmak üzere iki paket daha yer almaktadır.

Soyut Pencere Araçkiti (AWT) JAVA'nın paketleri arasında en ağır olanı diyebiliriz. Bu sınıflar temel Machintosh 84, Windows 95, X/Motif 88 ve Xeror PARC 80 grafik kullanıcı arabirim bileşenlerini işletir.

Java.awt, standart grafik kullanıcı arabirimi (GUI) elemanlarının kolay kullanımını sağlayan bir pakettir. Bu paket hem temel bileşenleri hem de üst düzey arabirimleri içermektedir. Bunun yanı sıra uygulamalar kendi bileşenlerini de kurabilir. Bu pakette tüm menü, kaydırma çubukları, düğmeler ve diğer bileşenler yer almaktadır.

Ayrıca daha ayrıntılı resim işlemine ya da renk işlemine gereksinim duyan uygulamalar, java.awt.image paketi içinde yer alan sınıfları da kullanabilir. Bileşenlerin daha farklı görünmesini ve hareket etmesine gereksinimi olan uygulamalar ise, java.awt.peer paketinde bulunan tanımlı arabirimlerle birlikte Toolkit sınıfı kullanılabilir.

java.awt.image Paketi

Java.awt.image paketinde karmaşık işlemlerini gerçekleştirmek için gerekli olan sınıf ve arabirimler yer almaktadır. Bu sınıflar ve arabirimler, resim ve renkler üzerinde düzenlemeler veya değişiklik yapmaya gereksinim duyan uygulamalar tarafından kullanılabilir.

Daha karmaşık resim işlemleri için kullanılan sınıf ve arabirimleri kapsamaktadır.

java.awt.peer Paketi

Java.awt.peer paketi, AWT bileşenlerini pencere sistemine özel çalışmalara bağlamak için kullanılan arabirimleri kapsamaktadır. AWT'nin pencere sistemine özel çalışmalarını hazırlamazsanız, java.awt.peer paketindeki arabirimleri kullanmanız gerekir.

Bu pakette yer alan her bir arabirim java.awt paketinde uygun bir bileşene sahiptir.

java.applet Paketi

İnternet hizmetlerine ulaşabilen, net üzerinden veriyi nakledebilen, otomatik olarak kurulabilen ve bir web dokümanı gibi çalıştırılabilen küçük uygulamalara applet adı verilmektedir.

Bu küçük uygulamaları hazırlamak için java.applet paketindeki sınıf ve arabirimlerini kullanacağız. Bundan sonra sizin applet'lerin çalışması ile temel bilgilere sahip olduğu kabul edilecektir.

Basit bir Applet yazılımı

Appletler java ile yazılıp HTML sayfalarında yayınlanabilen sınıflardır.HTML sayfalarına dinamik bir görünüm sağlarlar. Appletler java.applet.Applet sınıfından türerler.

Basit bir applet aşağıdaki gibi türetilebilir.

```
import java.applet.*;
                public class PMyApplet extends Applet
    {

}
```

Applet internet browser'ınıza yüklendiği zaman browser tarafından Applet'e ait bazı fonksiyonlar çağırılır. Bunlar;

public void init() :

Browser tarafından applet sisteme yüklendiği zaman çağırılır. Applet'in kendini hazırlaması için gerekli işlemler bu fonksiyon içerisinde yapılır. Örneğin applet tarafından kullanılacak olan nesnelerin yaratılması, bunlara ilk değerlerinin atılması. Bu metod sadece bir defa çağırılır.

public void start ( ) :

Browser tarafından applet'in icra edilmesi (execute) gerektiği anda çağırılır. İcra edilme işlemi applet sisteme yüklendikten hemen sonra olabileceği gibi, kullanıcının

browser'ında bir sonraki sayfaya geçip applet'in olduğu sayfaya geri döndüğünde de olabilir.

Public void stop ( )

Browser tarafından appletin icra edilmesinin son bulması gerektiğinde çağırılır. Applet browser sayfasında görünmediği zamanlarda çağırılır.

public void destroy( ) :

Applet browser tarafından hafızadan kaldırılmadan önce çağırılır. Bu adımda applet tarafından yaratılan kaynakların hafızadan atılması işlemi gerçekleştirilir. init( ) metodunda olduüu gibi destroy metodu'da sadece bir defa çağırılır.

Yukarıda metodların dışında işimize yarayacak bir diğer metod ise Applet'in java.awt.Container'dan miras aldığı public void paint(Graphics) metodur. Bu metod içerisinde appletin nasıl boyanacağını belirtebiliriz.

Yukarıdaki metodlarıda PMyAppelt sınıfımız içerisine erleştirirsek basit bir Applet sınıfının iskeleti aşağıdaki gibi oluyor.

```
import java.applet.*;
import java.awt.*;
public class PMyApplet extends Applet
{
        public void init( )
  {
    // Applete ait kaynakları burada yaratın.
  }
        public void start ( )
  {
        // Applete başladğında yapılacak işlemler.
```

```
    }
        public void paint( Graphics g )
    {
        // Appleti boyayalım.
    }
         public void stop ( )
    {
        // Applete durduğunda yapılacak işlemler.
    }
        public void destroy ( )
    {
         // Applet yok edileceğinde yapılacak işlemler.
    }
}
```

JDBC (Veri Tabanı)

Bilgi çağında, veri tabanı verinin toplanması ve işlenmesi için kullanılan bir araçtır ve birçok şirketin alt yapısını oluşturur. Veri tabanı sisteminin bilgi depolama ve erişimine iyi uyarlanmış olmasına karşın, depolanan veriyi görmek ve kullanmak için bir çeşit ön-uç uygulamasına ihtiyaç vardır.

Problem, birçok şirkette bulunan bilgisayarların heterojen doğası yüzünden karmaşıklaşır. Sanat ve pazarlama departmanları Macintosh sistemlerine, mühendisler yüksek-uç Unix iş istasyonlarına sahiptir ve satış temsilcileri PC'leri kullanır. Veriyi veri tabanında göstermek için, geliştiriciler yayılmak istedikleri sistemlerin tüm değişik permütasyonlarını ele almalıdır.

Bu bölümde Java, tek ve tutarlı bir uygulama programlama arabirimi, Java Veri Tabanı Bağlanabilirliği (Java Database Connectivity) API'sini sağlayarak veri tabanı ön-uçlarının "hayali tasarısını" çözümleyen bir yol olarak ele alınacaktır.

Bir Veri Tabanı Ön-Ucu Olarak Java

Bir veri tabanı sunucusu için bir ön-uç uygulamasının oluşturulmasında Java, geliştiriciye birkaç avantaj sunar. Java bir "bir kere yaz, her yerde çalıştır" programlama dilidir. Bunun anlamı, Java Sanal Makinesini çalıştıran her işletim sistemi ve bilgisayar mimarisi üzerinde Java programlarının değişiklik yapılmaksızın yayılabilmesidir. Büyük şirketler için sadece ortak bir geliştirme platformuna sahip olmak bile büyük bir tasarruf anlamına gelmektedir: Artık programcıların büyük bir şirketin sahip olabileceği birçok platform için kod yazması gerekmez. Java ayrıca üçüncü-parti geliştiriciler için de caziptir; tek bir Java programı büyük bir müşteri grubunun ihtiyaçlarına cevap verebilir.

Ek olarak, şirketin sahip olduğu her sistemin (istemci) yazılım ve donanımının kurulması ve bakımı ile ilgili bir maliyet söz konusudur Windows PC'leri, Macintosh ve Unix sistemleri gibi masaüstü-merkezli istemcilerin (ağır istemciler) makine başına

şirkete maliyeti 10,000...15,000$ arasındadır. Java teknolojisi şimdi herhangi bir şirketin daha küçük bir sistem alanı kullanmasını mümkün kılmaktadır. Bu sistemler bir Java çip setine dayanır ve yerleşik bir Java işletim sisteminden herhangi bir programı ve tüm Java programlarını çalıştırabilir.

Çok düşük miktarda sabit kaynaklar ile çalışan ve yine de tüm Java ortamını çalıştırabilen Java-tabanlı istemcilerin (hafif istemciler) makine başına 2,500$'dan daha az bir maliyete sahip olması beklenmektedir. Değişik çalışmalara göre, 10,000 tane ağır istemciyi hafif istemcilere taşıyan bir şirket için tasarruf miktarı yıllık 100 milyon $ kadar olabilmektedir.

Bunun ardından, birlikte çalışan sistemler için Java-tabanlı uygulama ve applet'lar oluşturmak için teşvikin yüksek olması gelir. Şirketler uygulamalarını mimari ve işletim sistemine özgü modellerden ağ-merkezli modellere kaydırmak konusuyla oldukça ilgilenmektedir Java, kaynak maliyetlerinin korunması konusunda bir uzun-dönem stratejisi ortaya koyar.

Geliştirici için, Java büyük bir pazar fırsatı sunmaktadır. Yapılan işlerin bir bölümü için veri tabanı kullanmayan çok az sayıda orta ve geniş ölçekli organizasyon varken, bunların birçoğu da, insan kaynaklarından ön planda müşteri satışlarına kadar, işlerinin tümünde veri tabanlarını kullanır.

Bu bölümde, şu andaki Java Database Connectivity (JDBS) API'sinin Java uygulama ve applet'larını veri tabanı hizmetlerine bağlamak için nasıl kullanıldığı ile birlikte JDBS ele alınacaktır.

Veri Tabanı İstemci/Sunucu Metodolojisi

İlişkisel veri depolamanın gelişimi, veri parçaları arasındaki ilişkileri tanımlamak için 12 kural öneren Dr E. F Codd'un çalışmaları ile 1970'de başladı. Verinin ilişkisel olarak modellenmesi için Codd'un önerdiği kurallar, veri yönetim sistemlerinin gelişimi için temel oluşturdu. Bugün, İlişkisel Veri Tabanı Yönetim

Sistemleri (Relational Database Management Systems, RDBMS) Codd'un vizyonunun sonucudur.

Bir RDBMS içindeki veriler, tablolarda ayrı veri satırları olarak depolanır Veriyi sorgulamak (erişmek), depolamak ve değiştirmek için yapısal bir dil kullanılır. Yapısal Sorgulama Dili (Structured Query Language, SQL) bir ANSI standardıdır ve tüm ana RDBMS sağlayıcıları SQL komutlarını kullanmak için mekanizmalar sağlar.

RDBMS uygulamalarının ilk geliştirme örnekleri kullanıcı arabirimi kodu, uygulama kodu ve veri tabanı kütüphanelerinden oluşan birleşik bir modelden yararlandı. Bu iki kısımdan meydana gelen tek model sadece yerel bir makine üzerinde (tipik olarak bir mainframe) çalıştırıldı. Bu uygulamalar basit, ancak verimsizdi ve LAN'lar üzerinde çalışmadı. Model ölçeklenmemişti ve uygulama ve kullanıcı arabirimi kodu veri tabanı kütüphanelerine sıkı olarak bağlıydı.

Ayrıca, bu homojen yaklaşım uygulamanın birkaç örneğinin birbiri ile haberleşmesine izin vermedi. Bu yüzden uygulamanın örnekleri arasında genellikle çekişme vardı.

NOT: RDBMS ve DBMS'in (Database Management System) birbiri yerine kullanılması karakteristiktir, çünkü hemen hemen tüm ticari veri tabanları ilişkiseldir ve kullanıcının veri tabloları arasındaki ilişkileri sorgulamasını sağlamak için bir SQL biçimini destekler.

İki Katlı Veri Tabanı Tasarımı

İki-katlı veri tabanı modelleri, sunucu teknolojisinin gelişi ile beraber ortaya çıktı. Haberleşme-protokolünün gelişimi ve yerel ve geniş alan ağlarının yaygın olarak kullanılması, veri tabanı geliştiricisinin arka-uç sunucusuna bir bağlantı (soket) aracılığıyla veriye erişen bir uygulama ön-ucu oluşturmasını sağladı.

İstemci programları (bir kullanıcı arabirimi uygulayarak) veri tabanı sunucusuna SQL istekleri gönderir. Sunucu, uygun sonuçları geri gönderir ve istemci verinin şekillendirilmesinden sorumludur. İstemciler buna rağmen, sunucu ve istemci arasındaki haberleşmeyi yöneten, sağlayıcı tarafından verilen fonksiyon kütüphanesini kullanır. Bu kütüphanelerin çoğu C dilinde yazılır.

Ticari veri tabanı sağlayıcıları veri tabanı sunucusuna "beceri" ekleme imkanının farkına vardı. Sağlayıcılar, veri tabanı geliştiricisinin basit veri işletimi için makro programları geliştirmesine izin veren özel teknikler oluşturdu. Depolanmış prosedürler olarak adlandırılan bu makrolar, bakım ve sürüm kontrolü ile ilgili problemlere neden olabilir. Depolanmış bir prosedür veri tabanı üzerinde bulunan yürütülebilir bir program olduğundan, tablo değiştirilirken veri tabanı tablosunun adlandırılmış sütunlarına erişmeye çalışabilir. Örneğin, id adlı bir sütunun adı cust_id olarak değiştirilirse, orijinal depolanmış prosedürün anlamı kaybolur. Belirli bir tablo veya tablolardaki harekete bağlı olarak otomatik olarak yürütülen depolanmış prosedürler olan tetikleyiciler, bir sorgulamadan gönderilen veri beklenen veri olmadığında (bu durum, yine, tetikleyicinin değiştirilmiş bir tablo sütununu okumasının sonucu olabilir), bu problemleri çözebilir. İstemci/sunucu mimarisinin başarısına rağmen, iki-katlı veri tabanı modelleri birkaç sınırlama ile karşı karşıyadır:

Bu modeller üretici tarafından verilen kütüphane ile sınırlıdır, bir veri tabanı üreticisinden diğerine geçilmesi, istemci uygulama kodunun önemli bir miktarının yeniden yazılmasını gerektirir.

Sürüm kontrolü bir sorundur. Sağlayıcı istemci tarafındaki kütüphaneleri güncelleştirildiğinde, veri tabanından faydalanan uygulamalar yeniden derlenmeli ve yeniden dağıtılmalıdır.

Üretici kütüphaneleri alt-düzey veri işletimi ile uğraşır. Tipik olarak, taban kütüphane verinin sadece tek satır ve sütunlarında alıp getirilmesi ve güncelleştirilmesi ile ilgilenir. Bu davranış, sunucu tarafında depolanmış bir prosedür oluşturularak artırılabilir, ancak bu kez de sistemin karmaşıklığı artar.

Verinin işlenmesi ve kullanılması ile ilgili tüm beceriler, istemci tarafında uzun çalıştırma zamanları oluşturarak, istemci uygulamasında gerçekleştirilir. Bu ise her bir istemci makinenin maliyetini artırır.

Üç Katlı Veri Tabanı Tasarımı

Günümüzde çok-katlı tasarıma büyük bir ilgi söz konusudur. Sadece üç katın olması gerekmez, ancak kavramsal olarak bir sonraki aşama budur. Bir çok-katlı veri tabanı tasarımında istemci, RDBMS'den soyutlama katmanı sağlayan, bir ara sunucu ile haberleşir.

Ara katman birkaç istemci isteğini işlemek ve bir veya daha fazla veri tabanı sunucusu bağlantısını yönetmek için tasarlanır. Ara kat tasarımı, ikikatlı veri tabanı tasarımına göre birkaç avantaj sağlar. Ortadaki kat:

Aynı anda birkaç istemci bağlantısını yönetmek amacıyla çok kanallıdır.

Üreticiden bağımsız çeşitli protokollerdeki (HTTP'den TCP/IP'ye kadar) istemci bağlantılarını kabul edebilir, sonra istekleri uygun sağlayıcıya özgü veri tabanı sunucularına göre düzenler ve cevapları uygun istemcilere gönderir.

Verinin işletimini yöneten bir "iş kuralları" seti ile programlanabilir; iş kuralları, verinin belirli bölümlerine erişimi kısıtlamaktan, verinin yerleştirilmeden veya güncelleştirmeden önce doğru olarak şekillendirildiğinden emin olunmasına kadar her şeyi içerebilir.

Yoğun-işlem görevlerini merkezileştirerek istemcinin ağırlaşmasını ve veri gösterimini daha üst bir düzeye soyutlamasını önler.

İstemci uygulamasını veri tabanı sisteminden ayırır ve bir şirketi, iş kurallarını tekrar düzenlemek zorunda kalmadan, veri tabanları arasında geçiş yapma konusunda serbest bırakır.

Eş-zamansız olarak istemciye bu andaki bir veri tablosunun veya satırının bir durum bilgisini sağlar.

Bu son noktaya örnek olarak, bir istemci uygulamasının belirli bir tablonun sorgulamasını henüz bitirmiş olduğunu düşünelim. Eğer diğer bir istemci tarafından yürütülen bir sonraki faaliyet veriyi değiştirirse, ilk istemci akıllı bir orta-kat programından bu konuda bir bildirim alabilecektir.

JDBC API

JDBC API'si geliştiricilerin, sürekli olarak kodu yeniden yazmak zorunda kalmadan, veri tabanı önuçları oluşturmasını sağlamak için tasarlanmıştır. ANSI komitesi tarafından belirlenen standartlara rağmen, her bir veri tabanı sistemi üreticisinin sisteminin bağlantısını yapmak ve bazı durumlarda, sistemi ile haberleşmek için özel bir yöntemi vardır.

Güçlü, platformdan bağımsız uygulamalar oluşturma yeteneği ve Webtabanlı applet'lar, geliştiricileri Java'yı kullanarak ön-uç bağlanırlık çözümleri geliştirmek konusunda harekete geçirdi. Başlangıçta üçüncü-parti yazılım geliştiricileri uygun çözümler sağlayarak, istemci tarafındaki kütüphaneleri birleştirmek için doğal yöntemler kullanarak veya yeni bir protokol ve bir üçüncü kat oluşturarak ihtiyacı karşıladı.

Java ürünlerinin geliştirilmesinden sorumlu Sun Microsystems iş birimi JavaSoft, kullanılan özel veri tabanını düşünmeksizin geliştiricilerin istemci tarafındaki uygulamalarını yazmasına izin veren DBMS'den bağımsız bir mekanizma oluşturmak için, veri tabanı ve veri tabanı-araç sağlayıcıları ile birlikte çalıştı. Sonuç, ilk sürümünde çekirdek JDK 1.1'in parçası olan JDBC API'dir.

JDBC uygulama geliştiricilerine, uniform ve veri tabanından bağımsız olan bir single API sağlar. API, kod yazmak için bir standart ve tüm değişik uygulama tasarımlarını hesaba katan bir standart sağlamaktadır. İşlemin özünde, bir sürücü

tarafından gerçekleştirilen bir Java arabirimleri seti yatmaktadır. Sürücü standart JDBC çağrılarının, sürücünün desteklediği veri tabanı tarafından ihtiyaç duyulan özel çağrılara dönüştürülmesini ele alır. Bunun devamında, uygulama bir kez yazılır ve değişik sürücülere taşınır. Uygulama aynı olarak kalır; sürücüler değişir. Sürücüler, middleware olarak da bilinen, çok-katlı veri tabanı tasarımının orta katını geliştirmek için kullanılabilir.

Geliştiricilere tekbiçimli ve DBMS'den bağımsız bir yapı sağlamaya ek olarak, JDBC geliştiricilerin veri tabanı üreticilerinin sunduğu özel işlevselliği sürdürmesine de imkan sağlar. JDBC sürücüleri ANSI SQL-2 Entry Level'ı desteklemelidir, ancak JDBC geliştiricilerin sorgulama katarlarını doğrudan bağlanan sürücüye göndermelerine izin verir. Bu katarlar ANSI SQL veya (hatta) SQL olabilir ya da olmayabilir. Bu katarların kullanımı alttaki sürücüye bağlıdır. Bu özelliğin kullanımı tabii ki, uygulama geliştiricisinin veri tabanı arka uçlarını değiştirme özgürlüğünü sınırlar.

JDBC, Microsoft'un Open Database Connectivity (ODBC) spesifikasyonundan türetilmemiştir. JDBC tümüyle Java'da yazılmıştır ve ODBC bir C arabirimidir. Bununla birlikte, hem JDBC ve hem de ODBC, X/Open SQL Command Level Interface'ine (CLI) dayanır. Aynı kavramsal temele sahip olmak, API'deki işin daha hızlı ilerlemesini sağlar ve API'nin kabulünü daha kolay hale getirir. JavaSoft, JDBC'den ODBC'ye dönüştüren bir JDBC-ODBC köprüsü sağlar. Doğal yöntemlerle gerçekleştirilen bu uygulama, oldukça küçük ve verimlidir.

Genel olarak JDBC API'de iki arabirim düzeyi vardır: Geliştiricinin SQL aracılığıyla veri tabanına çağrılar yaptığı ve sonuçları aldığı Uygulama Katmanı (Application Layer) ve özel bir sürücü uygulaması ile tüm haberleşmeyi yöneten Sürücü Katmanı (Driver Layer).

Her JDBC uygulaması (veya applet'ı) en az bir JDBC sürücüsüne sahip olmalıdır ve her bir sürücü kullanılan DBMS tipine özgüdür. Ancak bir sürücünün doğrudan bir veri tabanı ile ilişkili olması gerekmez.

API Bileşenleri

Daha önce bahsedildiği gibi, JDBC API içinde iki ayrı katman vardır: Veri tabanı uygulama geliştiricilerinin kullanacağı Uygulama Katmanı ve sürücü sağlayıcılarının gerçekleştirdiği Sürücü Katmanı. Uygulama katmanında kullanılan nesnelerin bir kısmı sürücü tarafından oluşturulduğundan Sürücü katmanının anlaşılması önemlidir.

Şans eseri, uygulama geliştiricisi JDBC uyumluluğunu garanti etmek için sadece standart API arabirimlerini kullanmaya ihtiyaç duyar. Sürücü geliştiricisi, veri tabanına arayüz oluşturan ve JDBC uygulama düzeyi çağrılarını destekleyen kodu geliştirmekten sorumludur.

Her sürücü katmanının gerçekleştirmesi gereken dört temel arabirim ve Uygulama ve Sürücü katmanları arasında bağlantı kuran bir sınıf vardır. Dört tane arabirim Driver, Connection, Statement ve ResultSet arabirimleridir. Driver arabirimi uygulaması veri tabanı bağlantısının yapıldığı yerde gerçekleştirilir. Birçok uygulamada, Driver arabirimine (geliştirici için bir tane daha soyutlama katmanı sağlayarak) DriverManager sınıfı vasıtasıyla erişilir.

Connection, Statement ve ResultSet arabirimleri sürücü üreticisi tarafından gerçekleştirilir, ancak bu arabirimler uygulama geliştiricisinin gerçek nesne sınıfları olarak davranacağı yöntemleri ortaya koyar ve geliştiricinin deyimler oluşturmasına ve sonuçlara erişmesine izin verir. Bu nedenle bu kısımda Uygulama ve Sürücü katmanları arasındaki ayrılık yapaydır; fakat nesnelerin nereden geldiğini veya uygulamanın hangi özel sürücüyü kullanacağını düşünmek zorunda kalmadan, geliştiricinin veri tabanı uygulamaları oluşturmasına imkan verir.

Sürücü Katmanı

Veri tabanı ve JDBC Sürücüsü arasında bire-bir haberleşme vardır. Bu yaklaşım çok-katlı tasarımlarda yaygındır. Driver sınıfı sürücü sağlayıcısı tarafından

gerçekleştirilen bir arabirimdir. Diğer önemli bir sınıf Sürücü ve Uygulama katmanları üzerinde yer alan DriverManager sınıfıdır. DriverManager sınıfı, sürücüleri yüklemek ve geri yüklemekten ve sürücüler vasıtasıyla bağlantılar yapmaktan sorumludur. Bu sınıf ayrıca logging ve veri tabanı oturum zaman-aşımı süreleri ile ilgili özellikler sağlar.

NOT: Sürücü doğrudan bir veri tabanına bağlanmak zorunda değildir ve bir çok-katlı veri tabanı tasarımı için yeni bir protokolü destekleyebilir.

Her JDBC uygulaması en az bir tane JDBC sürücü uygulamasına sahip olmalıdır. Driver arabirimi, DriverManager ve JDBC Uygulama katmanlarının, kullanılan özel veri tabanından bağımsız olarak mevcut olmasını sağlar. Bir JDBC sürücüsü Driver arabirimi sınıfının bir uygulamasıdır.

Sürücüler veri tabanının yerini belirlemek ve ona erişmek için bir katar kullanır. Bu katarın sözdizimi bir URL katarına çok benzer. Bir JDBC URL katarının amacı, uygulama geliştiricisini sürücü geliştiricisinden ayırmaktır. JavaSoft sürücü URL'leri için aşağıdaki amaçları tanımlar:
Sürücü-erişim URL adı, kullanılmakta olan veri tabanının tipini tanımlamalıdır.

Kullanıcı (uygulama geliştiricisi) bir veri tabanı bağlantısının oluşturulmasının yönetiminden sorumlu olmamalıdır; bu nedenle, herhangi bir veri tabanı bağlantı bilgisi (makine, port, veri tabanı ismi, kullanıcı erişimi ve şifreler) URL içinde kodlanmalıdır.
Kullanıcının özellikle doğru makine adını ve veri tabanının port numarasını kodlamak zorunda kalmasını önlemek için bir ağ adlandırma sistemi kullanılabilir.
World Wide Web tarafından kullanılan URL sözdizimi bu amaçları karşılayan standart bir sözdizimini destekler JDBC URL'leri aşağıdaki sözdizimine ve yapıya sahiptir:

jdbc: <subprotocol>:<subname>

Burada <subprotocol > sürücünün tipini tanımlar ve <subname> ağ kodlu ismi sağlar Örneğin:

jdbc:oracle:prodıacts

Burada veri tabanı sürücüsü bir Oracle sürücüsüdür ve alt-ad products adında bir yerel veri tabanıdır. Bu sürücü, Oracle veri tabanına bağlantı yaparken alt adı nasıl kullanılacağının bilinmesi için tasarlanır.

Bir ağ adlandırma servisi özel bir veri tabanı sürücü adı kullanmak yerine, alt-protokol olarak da belirtilebilir. Bu durumda alt-protokol, adlandırma servisini belirtir:

jdbc:localnaming:human-resources

Burada alt-protokol bir veri tabanı sunucusuna human-resources altadını çözebilen yerel bir servis tanımlar Bu yaklaşım, uygulama geliştiricisi kullanıcıyı gerçek yerleşim, ad, veri tabanı kullanıcı adı ve veri tabanı Şifresinden uzak tutmak istediği zaman oldukça faydalı olabilir. Bu URL localnaming adında bir sürücünün tayin edildiğini belirtir; bu, basit bir büyük-dosya araması içeren, human-resources ifadesini hrdatabasel.eng:888/personnel'a çeviren ve user kullanıcı adını ve matilda şifresini kullanmayı bilen bir Java programı olabilir. Bağlantının detayları kullanıcıdan saklı tutulur.

Tipik olarak uygulama geliştiricisi özellikle veri tabanının nerede bulunduğunu bilecek ve yerini belirlemek için ağı ters yönde kullanmayabilecektir. Bu durumda URL makinenin yerleşimini ve özel portu ve veri tabanı bilgisini içerecek şekilde genişletilebilir:

jdbc.:msql//dbserver.eng:1112/bugreports

Burada eng domain'indeki dbserver adında bir sunucunun yerini belirlemek için bir msql veri tabanı sürücü tipi kullanılır ve varsayılan kullanıcı adı ve şifre kullanılarak, bugreports adında bir veri tabanını içeren port 1112'deki bir veri tabanı sunucusuna bağlanmaya çalışır.

NOT: Alt-protokol adları neticede örtüşecektir ve kullanılmış olan adlardan oluşan bir kayda ihtiyaç olacaktır. Bir JDBC altprotokol adının kaydedilmesi hakkında bilgi edinmek için JDBC spesifikasyonuna başvurunuz.

Driver arabirimi, sürücü üreticisi tarafından, aşağıdaki her bir arabirim yöntemi için yöntemler oluşturarak gerçekleştirilir:

İmza:interface java.sql.Driver

public abstract Connection connect (String url,
Properties info) throws SQLException

Bu yöntemin sürücü uygulaması, bu sürücü ile eşleşme için gönderilen URL katarının alt-protokol adını kontrol etmelidir. Eğer bir eşleşme varsa, sürücü bundan sonra URL katarının geri kalanında gönderilen bilgiyi kullanarak veri tabanına bir bağlantı kurmaya çalışmalıdır. Başarılı bir veri tabanı bağlantısı, bir Connection arabiriminin (nesne) sürücü uygulamasının bir örneğini gönderecektir. SQLException sadece, eğer sürücü URL alt-protokolünü tanır, ancak veri tabanı bağlantısı yapamazsa gönderilmelidir. Eğer URL, sürücünün beklediği bir URL'ye uymazsa bir null değeri gönderilir. Kullanıcı adı ve şifre Properties adındaki bir kap sınıfı içindedir.

public abstract boolean acceptsURL(String url)
throws SQLException

Sürücüye URL'nin geçerli olup olmadığını açıkça "sormak" da mümkündür, fakat bu yöntemin uygulamasının (tipik olarak) bağlantının yapılıp yapılmadığını değil, sadece URL'de belirtilen alt protokolü kontrol ettiğine dikkat edilmelidir.

Driver sınıfının connectt() yöntemi en önemli yöntemdir ve bir Connection nesnesinin elde edilmesi için DriverManager sınıfı tarafından çağrılır. Connection nesnesi, sorgulamaları gerçekleştiren Statement nesnelerinin oluşturulması için kullanılır.

Driver connect( ) yöntemi tipik olarak şu işlemleri yerine getirir:

Verilen URL katarının geçerli olup olmadığını kontrol eder.
Belirtilen makineye ve port numarasına bir bağlantı açar.
Adlandırılan veri tabanı tablosuna (eğer varsa) erişmeye çalışır.
Bir Connection nesnesinin bir örneğini gönderir.

NOT: Connection bir Java arabirimidir, dolayısıyla gönderilen nesne aslında Connection arabirimi sürücüsünün uygulamasının bir örneğine bir başvurudur.

DriverManager Sınıfı

DriverManager sınıfı aslında JDBC sürücülerini yönetmek için kullanılan bir yardımcı sınıftır. Sınıf bir sürücü aracılığıyla bir bağlantının elde edilmesi, sürücülerin kaydedilmesi ve geri kaydedilmesi, logging'in ayarlanması ve veri tabanı erişimi için oturum zaman-aşımı sürelerinin belirlenmesi için yöntemler sağlar. Aşağıda listelenen DriverManager sınıfındaki tüm yöntemler statiktir ve yöntemlere şu sınıf adı ile işaret edilebilir:

İmza:public class java.sql.DriverManager
public static synchronized Connection
getConnection(String url, Properties info)
throws SQLException

Bu yöntem (ve diğer getConnection() yöntemleri) Connection arabiriminden gerçekleştirilen bir nesneye bir başvuru göndermeye çalışır. Yöntem, URL katarını ve Properties info nesnesini her birine sırayla göndererek, saklı Driver sınıflarının oluşturduğu bir vektörü tarar. Bir Connection nesnesi gönderen ilk Driver sınıfı kullanılır. info, genellikle kullanıcı/şifre şeklinde olan etiket/değer çiftinin bir Properties kap nesnesine bir başvurudur. Bu yöntem, vektördeki her sürücü için yetkili kılınmış bir bağlantı yapmak amacıyla birkaç girişimde bulunulmasına izin verir.

public static synchronized Connection
    getConnectiontString url) throws SQLException

Bu yöntem boş bir Properties nesnesi ile (info) yukarıdaki getConnection(url, irıfo) yöntemini çağırır.

public static synchronized Connection
    getConnection(String url, String user, String password)
    throws SQLException

Bu yöntem bir Properties nesnesi oluşturur, kullanıcı adı ve şifre katarlarını onun içinde saklar ve sonra yukarıdaki getConnection(url,info) yöntemini çağırır.

public static void
    setlogstream(java.io.PrintStream out)

Bu yöntem, yönteme gönderilen PrintStream nesnesi ne statik özel bir java.io.PrintStream başvurusu belirler.

İPUCU: Sürücü uygulaması setType yöntemleri aracılığıyla saklanan ve sürücü tarafından getType yöntemleri aracılığıyla erişilen iki statik nesne başvurusundan faydalanabilir: Oturum zaman-aşımı süresini belirten bir tamsayı ve sürücü bilgisini kaydetmek için kullanılan bir PrintStream nesnesi.
Sürücüler, ya DriverManager sınıfının başlatılması sırasında ya da sürücünün bir örneği oluşturulduğunda, DriverManager sınıfı ile kaydedilir.

DriverManager sınıfı yüklendiğinde, sınıf içindeki bir statik kod bölümü çalıştırılır ve jdbc.drivers adındaki bir Java özelliği içinde listelenen sürücülerin sınıf adları yüklenir.

Bu özellik aşağıdaki gibi, ":" sembolü ile birbirinden ayrılmış sürücü sınıf adlarından oluşan bir listeyi tanımlamak için kullanılabilir:

jdbc.drivers=imaginary.sql.Driver:oracle.sql.Driver:
weblogic.sql.Driver

Her bir sürücü adı, kullanılan CLLASSPATH aracılığıyla DriverManager sınıfının yüklemeye çalışacağı, paket bildirimini de içeren bir sınıf adıdır. DriverManager, adı belirtilen sınıfın yerini belirlemek, yüklemek ve bağlanmak için aşağıdaki çağrıyı kullanır:

Class.forName(driver);

Eğer jdbc.drivers özelliği boş ise (belirtilmemişse), bu durumda uygulama programcısı sürücü sınıfının bir örneğini oluşturmalıdır.

Her iki durumda, Driver sınıfı aşağıdaki çağrıyı kullanarak DriverManager ile birlikte kendisini kaydetmelidir:

DriverManager.registerDriver(this):

Aşağıda hayali sürücünün (mini-SQL veri tabanı için) bir kod parçası görülmektedir Driver, hayali sürücünün bir örneği oluşturulduğunda, her defasında kendisini kaydeder:

```
public class iMsqlDriver implements java.sql.Driver
{
static {
try
{
new iMsqlDriver();
```

```
}
catch (SQLException e)
{
e.printStackTrace();
}
/**
* Yeni bir sürücü oluşturur ve onu JDBC taslak protokolü ile
* belirtildiği gibi java.sql.DriverManager.registerDriver()
* ile kaydeder.
*/
public iMsqlDriver() throws SQLException
{
        java.sql.DriverManager.registerDriver(this);
}
...
```

DriverManager sınıfının öncelikli kullanımı getConnectiont() yöntemi aracılığıyla bir Connection nesnesinin elde edilmesidir:

```
Connection conn=null ;
        Conn=DriverManager.getConnection
                ("jdbc:sybase://dbserver:8080/billing.", dbuser, dbpasswd);
```

Bu yöntem kayıtlı sürücülerin listesini gözden geçirir ve sürücünün connect( ) yöntemi aracılığıyla her bir sürücüye URL katarını ve parametreleri gönderir Eğer sürücü alt-protokol ve alt-ad bilgisini desteklerse, bir Conneetion nesne başvurusu gönderilir.

Bir Connection nesnesi doğrudan Driver sınıfından alınabildiğinden, DriverManager sınıfı JDBC uygulamaları oluşturmak için gerekmez:

```
Connection conn=null;
    Conn=new Driver().connect
        ("jdbc:sybase://dbserver:8080/billing", props);
```

Bunun anlamı, bir bağlantının elde edilmesinin engelsiz olmadığı ve uygulama geliştiricisinin güvenlik kontrollerini sağlamak için Driver uygulamasına bağımlı bırakıldığıdır.

Uygulama Katmanı

Uygulama katmanı Sürücü katmanında gerçekleştirilen, ancak uygulama geliştiricisi tarafından kullanılan üç arabirimi içerir. Java'da arabirim, özel bir nesneyi göstermek için genel bir ad kullanılmasına imkan sağlar. Bu genel ad, özel nesne sınıfları tarafından gerçekleştirilmesi gereken yöntemleri tanımlar. Uygulama geliştiricisi için bunun anlamı, özel Driver sınıfı uygulamasının konuyla ilgisiz olduğudur. Sadece standart JDBC API'lerin kodlanması yeterli olacaktı. Bu, elbette sürücünün JDBC uyumlu olduğunu kabul etmektedir. Aynı zamanda hatırlanırsa, bunun anlamı veri tabanının en az ANSI SQL-2 Entry Level'da olduğudur.

Üç ana arabirim Connection, Statement ve ResultSet arabirimleridir. Bir Connection nesnesi, DriverManager.getConnection( ) yöntem çağrısı aracılığıyla sürücü uygulamasından elde edilir. Bir Connection nesnesi bir kez gönderildiğinde, uygulama geliştiricisi veri tabanına karşı göndermek için bir Statement nesnesi oluşturabilir. Bir Statement'ın sonucu, belirli deyimin (eğer varsa) sonuçlarını içeren bir ResultSet nesnesidir.

Connection Arabiriminin Esasları

Conrıection arabirimi Driver tarafından sağlanan veri tabanı bağlantısı ile birlikte bir oturum sunar. Tipik veri tabanı bağlantıları, hareketler aracılığıyla depolanan gerçek veriye yapılan değişiklikleri kontrol etme yeteneği içerir. Bir transaction, sıra ile tamamlanan bir takım işlemlerdir. Bir commit eylemi, veri tabanındaki verilerin

saklanması (veya değiştirilmesi) işlemlerini yapar. Bir rollback eylemi, işlenmeden önce bir önceki hareketi geri alır. Oluşturulma sırasında, JDBC Connections auto-commit kipindedir; rollback mümkün değildir. Bu nedenle sürücüden bir Connection nesnesi alındıktan sonra, geliştirici setAutoCommit(boolean b) yöntemi ile auto-commit işlevini false olarak ayarlamalıdır.

Auto-commit devreden çıkarıldığında Connection, Connection.commit() ve Connection.rollback() yöntem çağrılarını destekleyecektir. Transaction yalıtımının desteklenme düzeyi, veri tabanı içindeki transaction'lar için alttaki desteğe bağlıdır.

Connection arabirimi tanımının bir kısmı aşağıdadır:

İmza: public interface Connection

Statement createStatement() throws SQLException

Connection nesne uygulaması bir Statement nesne uygulamasının bir örneğini gönderecektir. Statement nesnesi daha sonra sorgulamalar yapmak için kullanılır.

PreparedStatement prepareStatementString sql) throws SQLException

Connection nesne uygulaması, gönderilen sql katarı ile konfigüre edilen PreparedStatement nesnesinin bir örneğini gönderecektir. Eğer veri tabanı (sürücüsü) önceden derlenen deyimleri işliyorsa, sürücü bundan sonra deyimi veri tabanına gönderebilir. Aksi takdirde, sürücü PreparedStatement bir yürütme yöntemi tarafından yürütülünceye dek bekleyebilir. Eğer sürücü ve veri tabanı önceden derlenen deyimleri işlemiyorsa, bir uyarı mesajı verilebilir.

CallableStatement prepareCall(String sql) throws SQLException

Connection nesnesi bir CallableStatement nesnesinin bir örneğini gönderecektir. CallableStatement nesneleri depolanmış prosedürleri işlemek için optimize edilir. Sürücü uygulaması prepareCall ( ) sona erdiğinde hemen sql katarını gönderebilir veya bir yürütme yöntemi çağrısı meydana gelinceye dek bekleyebilir.

void setAutoCommit(boolean autoCommit) throws SQLException

Sürücü uygulamasında bir bayrağı, commit/rollback işlevlerini kullanılabilir kılan false değeri olarak veya tüm transaction'ların hemen işlenmesi işlevini geçerli kılan true değeri olarak belirler.

void commit() throws SQLException

Bu andaki transaction'ın (ya bağlantının açılması ya da son commit( ) veya rollback ( ) çağrılarından beri) başlangıcından beri yapılan tüm değişiklikleri yerine getirir.

void rollback() throws SQLException

Bu andaki transaction'ın başlangıcından beri yapılan tüm değişiklikleri iptal eder.

Canrıection arabiriminin temel kullanımı bir deyimin oluşturulmasıdır:

```
Connection msqlConn=null;
Statement stmt: = nul ;
MsqlConn=DriverManager.getConnection(url);
stmt = msqlConn.createStatement();
```

Bu deyim, bir ResultSet nesne başvurusunda tek bir sonuç seti gönderen SQL deyimlerinin gönderilmesi için kullanılabilir. Küçük değişikliklerle birkaç defa çağrılması gereken deyimler, bir PreparedStatement kullanılarak daha etkin bir şekilde yürütülebilir. Connection arabirimi ayrıca, depolanmış prosedürleri yürütmek amacını taşıyan bir CallableStatement oluşturmak için de kullanılabilir.

Geliştirici çoğu zaman veri tabanı planını önceden bilir ve bu plana dayanan uygulamayı oluşturu. Ancak, JDBC bir veri tabanı planının dinamik olarak belirlenmesi için kullanılabilecek bir arabirim sağlar. Connection arabiriminin getMetaData() yöntemi bir DatabaseMetaData nesnesi gönderecektir. Arabirimi gerçekleştiren sınıfın

örneği, tablolar ve prosedürler hakkındaki erişim bilgisini, sütun adlarını, veri tiplerini vs de içeren, veri tabanının tümü hakkında bilgi sağlar.

DatabaseMetaData uygulama detayları, veri tabanı sağlayıcısının bu tip bilgiyi gönderme yeteneğine bağlıdır.

Statement Arabiriminin Esasları

Deyimler (Statement), veri tabanına SQL sorgulamaları göndermek ve bir sonuç setine erişmek için kullanılan araçtır. Deyimler SQL güncelleştirmeleri (update), eklemeler (insert), silme işlemleri (delete) ve sorgulamalar (Select ile) olabilir. Statement arabirimi, sorgulamaların veri tabanına daha kolay yazılması işini yapmak için tasarlanmış birkaç yöntem içerir.

İmza:public interface Statement

ResultSet executeQuery(String sql) throws SQLException

Tek bir SQL sorgulamasını yürütür ve sonuçları ResultSet tipinde bir nesnede gönderir.

int executeUpdate(String sql) throws SOLException

Bir sonuç takımı değil, etkilenen satırların sayısını gönderen tek bir SQL sorgulamasını yürütür.

boolean execute(String sql) throws SQLException

Birkaç sonuç seti ve/veya güncelleştirme sayısını gönderebilen genel bir SQL deyimidir. Bu yöntem Out ve in0ut parametrelerini gönderen depolanmış prosedürleri yürütmek için kullanılır. getResultSet(), getUpdateCount() ve getMoreResults() yöntemleri gönderilen veriye erişmek için kullanılır.

İPUCU: In parametreleri bir işleme gönderilen parametrelerdir. Out parametreleri bir başvuru tarafından gönderilen parametrelerdir; bunların başvuru tipini göndermesi

beklenir. Inout parametreleri, bir işlemin sonucu olarak değişebilen bir ilk değeri içeren Out parametreleridir. JDBC her üç parametre tipini de destekler.

ResultSet getResultSet() throws SQLException

Bu andaki veriyi, bir deyim yürütmesinin sonucu olarak ve bir ResultSet nesnesi olarak gönderir. Eğer okunacak hiçbir sonuç yoksa veya sonuç bir güncelleştirme sayısı ise, bu yöntemi null değerini gönderdiğine dikkat edilmelidir. Ayrıca bir kez okunduğunda sonuçların temizlendiğine de dikkat edilmelidir.

int getUpdateCount() throws SoLException

Güncelleştirme, silme veya ekleme sorgulamasının veya depolanmış bir prosedürün durumunu gönderir. Gönderilen değer etkilenen satır sayısıdır. Eğer bir güncelleştirme sayısı yoksa veya gönderilen veri bir sonuç takımı ise, bir -1 değeri gönderilir. Birkez okunduğunda, güncelleştirme sayısı temizlenir.

boolean getMoreResults() throws SDLException

Bir birkaç sonuç/güncelleştirme sayısı seti içinde bir sonraki sonuca hareket eder. Eğer sonraki sonuç bir ResultSet nesnesi ise, bu yöntem true değerini gönderir. Ayrıca yöntem, dal önce okunmuş her ResultSet sonucunu kapayacaktır.

Deyimler kullanılan Statement yöntemine bağlı olarak bir ResultSet nesnesini gönderebilir veya göndermeyebilir. Örneğin, executeUpdate( ) yöntemi bir satır sayısı durumu dışında bir son beklemeyen SQL deyimlerini yürütmek için kullanılır:

```
int rowCount;
rowCount=stmt.executeUpdate (DELETE FROM Customer WHERE Customer ID='McG10233'");
```

Tek bir sonuç seti gönderen SQL deyimleri executeQuery( ) yöntemini kullanabilir. Bu yöntem tek bir ResultSet nesnesi gönderir. Nesne, sorgulamanın sonucu olarak gönderilen satır sayısını gösterir:

```
ResultSet results;
Results=stmt.executeQuery("SELECT * FROM Stock");
```

Depolanmış bir prosedürü yürüten (veya tetikleyen) SQL deyimleri birden fazla sonuç sei gönderebilir. execute( ) yöntemi, tek bir sonuç setini veya birkaç sonuç setini gönderebilen genel amaçlı bir yöntemdir. Yöntem daha fazla sonuç seti olup olmadığını belirlemek için kullanılan bir mantık bayrağını gönderir. Bir sonuç seti veri ya da bir satır sayısı gönderen bir işlem sayısını içerdiğinden, getResult5et(), getMoreResults() ve getUpdateCount( ) yöntemleri kullanılır.

Örneğin:

```
// SQL katarının birkaç sonuç setini gönderdiği kabul ediliyor
// Bir sonuç seti (ResultSet) gönderilirse, true
int count;
if (stmt.execute (SQLString))
{
results=stmt.getResultSet();
// false, bir gGncelleme sayısı (UpdateCount) gönderildi.
}
else
count=stmt.getUpdateCount();
// İ1k sonuçiar burada işleniyor...
// Şimdi, başka hiçbir güncelleme sayısı veya sonuç kalmayıncaya
// dek döngü içinde işlem yapılacak.
do
{
// Sonraki sonuç bir ResulSet mi?
if (stmt.getMoreResults())
{
results=stmt.getResultSet();
}
```

```
else
{
count=stmt.getUpdateCount()=
}
// Burada sonraki sonuçlar işlenir...
} while ((rasutts!=null) && (count!=-1));
```

PreparedStatement arabirimi Statement arabiriminden miras alınır. Küçük değişikliklerle tekrarlanması gereken bir SQL deyimi olduğunda preparedStatement, parametreleri kullanan önceden derlenmiş bir SQL deyimini göndermek için etkin bir mekanizma sağlar.

public interface PreparedStetement extends Statement

PreparedStatement parametreleri bir SQL deyimine veri göndermek için kullanılır, bu nedenle bunlar In parametreleri olarak değerlendirilir ve setType yöntemleri kullanılarak doldurulur.

NOT: setType yöntemleri bir PreparedStatement içindeki parametrelerin (soru işaretleri ile gösterilen) değerlerini doldurur. Bu parametreler 1'den n'e kadar indekslenir.

```
// priceList, %10 oranında indirim uygulanması gereken
// fiyatların oluşturduğu bir dizi ve reducedItems ürün
// bilgilerinin oluşturduğu bir dizi olarak alınıyor.
int reduction=10;
PreparedStatement ps=msqlConn.prepareStatement ("UPDATE Catalog SET Price=? WHERE ıtemID=?");
// Güncellemeler bir döngü içinde yapılıyor.
for (int i=0; i<reducedltems.lenght(); i++)
{
// setType yöntemlerinin S0L deyiminde ? ile belirtilen
// parametrelerin değerlerini ayarladığına dikkat ediniz
```

```
// Bunlar 1'den n'e kadar indekslenmiştir.
ps.setFloat (l, (priceList[i]*((float)(100-reduction)/100)));
ps.setString (2, reducedItems[i]);
if (ps.executeUpdate( )== 0)
{
throw new SQLException ("No Item ID: " + reducedItems[i]);
}
}
```

Parametreler ya yeni bir setType yöntemi çağrılıncaya dek ya da PreparedStatement nesnesi için clearParameters() yöntemi çağrılıncaya dek, bu andaki değerlerini tutar. Statement arabiriminden miras alınan execute( ) yöntemlerine ek olarak, PreparedStatement aşağıdaki setType yöntemlerini bildirir. Her bir yöntem iki argüman alır:

Bir parametre indeksi ve aşağıdaki gibi bir ilkel veri veya sınıf tipi.

Yöntem imzası	**Java Tipi**	**Veri Tabanından gönderilen SQL Tipi**
void setByte(int Index, byte b)	byte	TINYINT
void setShort(int ındex, short x)	short	SMALLINT
void setlnt(int index, int i)	int	INTEGER
void setLong(int index,long l)	long	BIGINT
void setFloat(int index, float f)	float	FLOAT
void setDuble(int index, double d)	double	DOUBLE
void setBigDecimal(int index,BigDecimal x)	Java.math.BigDecimal	NUMERIC
void setString(int index, String s)	Java.Wng.String	VARCHAR ya da LONGVAR CHAR
void setBytes(int index,byte x[])	byte dizisi	VARBINARY ya da LONGVAR BINARY
void setDate(int index, Date d)	Java.sql.Date	DATE

void setTime(int index, Time t)	Java.sql.Time	TİME
void setTimestamp (int index, Timestamp ts)	Java.sql.Timestamp	TIMESTAMP
void setNull(int index, int sqlType)	-	Java.sql tipleri, SQL tiplerini rakam ile belirtir ve NULL tamsayı 0(sıfır)' dır
void setBoolean (int index, boolean b)	boolean	BIT

SetType yöntemleri

CallabeStatement arabirimi depolanmış SQL prosedürlerini yürütmek için kullanılır. CallableStatement arabirimi PreparedStatement arabiriminden miras alınır, bu nedenle tüm yürütme ve setType yöntemleri mevcuttur. Depolanmış prosedürlerin veri tabanı sağlayıcıları arasında değişen bir sözdizimi vardır, bu nedenle JDBC tüm RDBMS'lerin depolanmış prosedürleri çağırması için standart bir yol sağlar:

public interface CallableStatement extends PreparedStatement

JDBC parametrelerim In ve Out parametreleri olarak gönderilmesini sağlayan bir escape sequence kullanır. Sözdizimi ayrıca bir sonucun gönderilmesine izin verir ve eğer bu sözdizimi kullanılırsa parametre bir Out parametresi olarak kaydedilmelidir. Aşağıda bir Qut parametresi gönderen bir CallableStatement görülmektedir:

```
CallableStatement cs=conn.prepareCall("(call getQuote (?,?))");
cs.setString (1. stockName);
// java.sqi tipleri Out parametreleri olarak gönderilen
// SQL veri tiplerini tanımlar.
Cs.registerOutParameter (2, Types.FLOAT);
stmt.executeUpdate();
float quote=stmt.getFloat (2);
```

CallableStatement veri tabanından gönderilen SQL tiplerini Java tiplerine dönüştüren bir getType yöntem takımı tanımlar. Bu yöntemler aşağıda gösterildiği gibi, PreparedStatement tarafından bildirilen setType yöntemlerine uyar.

Yöntem İmzası	**Java Tipi**	**Veri Tabanından gönderilen SQL Tipi**
boolean getBoolean(int Index)	boolean	BIT
byte getByte(int index)	byte	TINYINT
short getShort(int index)	short	SMALLINT
int getint(int index)	int	INTEGER
long getLong(int index)	long	BIGINT
float getFloat(int index)	float	FLOAT
double getDouble(int index)	double	DOUBLE
BigDecimal getBigDecimal (int index, Int scale)	java.math.BigDecimal	NUMERIC
String getString(int Index)	String	CHAR, VAR CHAR ya da LONGVAR CHAR
byte[] getBytes(int index)	byte dizisi	BINARY ya da VAR BINARY
Date GetDate(int index)	Java.sql.Date	DATE
Time getTime(int index)	Java.sql.Time	TIME
Timestamp getTimestamp(int index)	Java.sql.Timestamp	TIMESTAMP

GetType yöntemleri

NOT: getType yöntemleri sonuç bir sorgulama olacak şekilde, her bir sütundaki veriye erişir. Her bir sütuna ya satırdaki konumuna göre, i'den n'e kadar numaralanmış sütunlar, veya adı ile, custID gibi, erişilebilir.

NOT: Veri tabanından SQL veri tipleri olarak gönderilen verilerin Java tiplerine dönüştürülmesinin JDBC sürücüsünün sorumluluğunda olduğuna dikkat edilmelidir.

ResultSet Arabiriminin Esasları

ResultSet arabirimi, bir deyimin yürütülmesinin sonucu olarak üretilen veri tablolarına erişmek için yöntemler tanımlar. ResultSet sütun değerlerine herhangi bir sırayla erişilebilir; bunlar indekslenmiştir ve sütunun adı veya numarası (1'den n'e kadar numaralanmış) ile seçilebilir.

ResultSet gönderilen ilk veri satırı ile başlayarak, bu andaki satırın konumunu muhafaza eder. next( ) yöntemi bir sonraki veri satırına geçer.

ResultSet arabiriminin bazı yöntemleri aşağıdadır:

İmza: public interface ResultSet

boolean next() throws SQLException

ResultSet'i bir sonraki satırda konumlandırır; ResultSet satır konumu en başta sonuç takımının ilk satırıdır.

ResultSetMetaData getMetaData() throws SQLException

Bu andaki sonuç setinin bir tanımını içeren bir nesne gönderir: Bu tanım sütun sayısını, her bir sütunun tipini ve sonuçların özelliklerini içerir.

void close() throws SQLException

Diğer bir Statement yürütüldüğünde doğal olarak ResultSet kapatılır, ancak kaynakların daha önce serbest bırakılması istenebilir.

Yukarıda CallableStatement arabiriminde olduğu gibi, sonuçlanan veri, getType yöntemleri aracılığıyla okunabilir. Örneğin:

```
// Deyim nesnesine bir sorgulama gönderiliyor
ResultSet rs=stmt.executeQuery("SELECT * FROM Stock WHERE quantity=0");
// Sonuçlar Java tipleri olarak alınıyor.
```

```
// Sütunların 1'den başlayarak bir tamsayı ile veya
// "ItemID" gibi sütun adı ile indekslendiğine dikkat ediniz.
System.out.println ("Stock replenisment list");
while (rs.next())
{
System.out.println ("ItemID: " + rs.getString ("ItemID"));
System.out.println ("Next ship date: " + rs.getDate (2));
System.out.println ("");
}
```

ResultSetMetaData

JDBC bir ResultSet nesnesinden veri okuyabilmenin yanı sıra, geliştiricinin hangi tip verinin gönderildiğini belirlemesine imkan veren bir arabirim sağlar. ResultSetMetaData arabirimi içerik olarak DatabaseMetaData arabirimine benzer, fakat bu andaki ResultSet nesnesine özgüdür. DatabaseMetaData arabiriminde olduğu gibi, birçok uygulama veri tabanı planı ve sütun adlan ve değerleri anlaşılarak yazıldığı için, mühtemelen birçok geliştirici bu arabirimi kullanmayacaktır. Ancak ResultSetMetaData, depolanmış bir prosedürden gönderilen bir ResultSetMetaData değerlerini dinamik olarak belirlemeye yarar.

Büyük Veri Yığınlarının Yollanıp Alınması

SQL LONGVARBINARY ve LONGVARCHAR veri tipleri keyfi bir boyutta olabilir. getBytes() ve getString() yöntemleri sürücü tarafından düzenlenen sınırlara kadar bu tipleri okuyabilir. Sınırlar Statement.getMaxFieldSize() yöntemi aracılığıyla okunabilir. Daha geniş veri öbekleri için, JDBC veriyi yığınlar halinde göndermek için geliştiricilerin java.io.Input akımlarını kullanmasına izin verir.

İPUCU: Akımlar sorgulama yürütmesinin hemen ardından okunmalıdır; bunlar bir s6nraki sonuç takımının alınmasında otomatik olarak kapatılır.

Büyük veri öbeklerinin gönderilmesi parametreler olarak java.io.0utputStream kullanılarak mümkündür. Bir deyim yürütüldüğünde JDBC sürücüsü veriyi akımlar içinde iletmek ve okumak için tekrarlanan çağrılar yapar.

JDBC Kullanımının Sınırlamaları (Applet'lara karşı Uygulamalar)

Java dünyasında iki tip program vardır: Uygulamalar ve applet'lar. Her program tipinin avantajları vardır ve her birinin kullanımı genellikle, geliştiricinin kullanıcının programa erişmesini istediği yol ile belirlenir.

Uygulamalar

Uygulamalar tek-başına yürütülebilir programlar olarak geliştirilen Java programlarıdır. Kullanıcının programın yürütülebilir koduna (sınıf dosyası) ve yerel olarak Java yorumlayıcısına erişimi olması beklenir. Bir intranettabanlı veri tabanı ön-ucu için, bu yöntem daha hızlı başlangıç avantajlarını (sınıf dosyalan yereldir) ve yerel disk kullanımını önerir.

Ek olarak, Java uygulamaları güvenilirdir ve istemci programlarının uzaktaki sunucular üzerinde birkaç veri tabanı sistemine erişmesi mümkün kılınarak, soket bağlantıları ile daha esnek olmalarına izin verilmiştir.

GUI geliştirilmesi için araçlar ortaya çıktıkça ve tam zamanında (Just-inTime, JIT)derleyici/yorumlayıcıları aracılığıyla hız artışı sağlandıkça, Java uygulamaları daha da yaygın hale gelmektedir. Uygulamalar ayrıca, tarayıcı güvenlik sorunlarını ve Java zımbırtılarının (widget) tarayıcının uygulaması içindeki farklılıklarını ortadan kaldırabilir veya azaltabilir.

Applet'lar

Applet'lar çalıştırılmak için Java-destekli bir tarayıcıya ihtiyaç duyan minik Java programcıklarıdır. Tarayıcı, çizim ve gösterim kaynaklarını doğrudan tarayıcı

sayfasında içererek, applet'ın içinde çalıştırılabilmesi için bir ortam sağlar. Bir kullanıcı bir applet içeren bir tarayıcı sayfasından hareket ettiğinde veya bu sayfayı ziyaret ettiğinde, applet otomatik olarak yürütülür. İşlem JDBC sürücülerini ve uygulama katmanı yazılımını içeren gereken Java applet kodunun indirilmesini, kod üzerinde otomatik olarak güvenlik kısıtlamalarının kontrol edilmesini ve her şey yolunda ise applet'ın çalıştırılmasını içerir. Applet'lar, uygulamalar ile kıyaslandığında birkaç önemli avantaj sağlar:

Versiyon kontrolü: HTML sayfasındaki başvurularda sınıf dosyası yerleştirilerek neredeyse hareket halinde iken bir applet değiştirilebilir.

Daha kolay yürütme modeli: En karmaşık tarayıcıları bile öğrenmek ve bir ön-uç istemcisini yürütmek çok az bir emek gerektirir; kullanıcı sadece uygulamanın bulunduğu sayfayı dolaşır.

Çevrimiçi yardım: Çalıştırılan programın bir tarayıcı HTML sayfasında oluşturulması, çalıştırılan programdan ayrı olarak geliştirilebilen yardım bağlantılarının gömülmesini oldukça kolaylaştırır.

Applet'lar tipik olarak, gönderilen verinin kritik olmadığı ve erişimin iki-katlı bir model ile sınırlanabildiği (üç-katlı modeller kullanmak da mümkündür, ancak bu modeller daha karmaşık katman oluşturma düzenleri içerir) büyük bir organizasyonda eğitim amacıyla kullanılabilir. Diğer bir kullanım, yine veri miktarının çok fazla olmadığı ve veri iletisinin güvenliğinin çok önemli olmadığı, verinin İnternet topluluğuna basit sunumudur.

Bununla birlikte, applet'lar tarayıcı ortamı tarafından ağır bir şekilde kısıtlanır. Applet'ların aşağıdakileri yapmasına izin verilmez:

Yerel dosyalara erişim. Bu, applet'ın yaşamı boyunca yerel ön belleğe alma işlevini, tablo işletimini ve hafıza depolamasını sınırlar.

Keyfi makinelere bağlanmak. Sadece applet ile applet'ın kaynaklandığı makine arasındaki soket bağlantılarına izin verilir. Doğal yöntemler içeren sürücülerin yüklenmesi veya çalıştırılması (C dili çağrıları).

Ek olarak, applet kodunun bir İnternet (geniş alan) ağ bağlantısı üzerinden yüklenmesi ile sağlanan hatırı sayılır bir performans başarısı vardır.

Bu kısıtlamaların bir kısmı güvenilir applet'ların ve onları kabul eden tarayıcıların ortaya çıkışı ile azaltılabilir veya artırılabilir. Güvenilir applet'lar şifreli bir anahtar ile kod imzalı veya güvenilir bir yerde saklanmış olabilir. Eğer tarayıcı ortamı applet'ın kaynağının güvenilir olduğuna inanırsa, hala Java güvenlik yöneticisi ile ilgili olmayan İnternet üzerinde veri tabanlarının yerleşimine ilişkin sınırlar olabilmesine rağmen, bu durumda güvenlik amaçları açısından applet'lara uygulamalar gibi davranılabilir. Güvenilir applet'lar Java güvenlik modelinin gelecekteki değerlendirme konusudur.

Bugün daha somut ve kullanılabilir olan diğer bir alternatif, üç-katlı veri tabanı modelinin kullanılmasıdır. Bu yaklaşımda applet, hem HTML sayfasını hem de HTTP sunucusunu ve birkaç istemci için soket bağlantılarını sağlayan, ve sonuçta uzaktaki veri tabanı sistemleri ile bağlantı kuran çok kanallı bir uygulamayı (Java, C veya C++) sağlayan bir middleware katından yüklenir.

Üçüncü kata yapılan çağrılar özel (uygun) bir protokol geliştirilerek, Uzak Yöntem Çağrımı (Remote Method Invocation, RMI) kullanılarak veya bir Nesne İsteği Aracı (Object Request Broker ORB) kullanılarak yönetilebilir. Bu bölümdeki Alternatif Bağlantı Yöntemleri konusuna bakınız.

Güvenlik Meseleleri

JDBC API standart Java güvenlik modelini izler. Kısaca, uygulamalar güvenilir ve applet1ar güvenilmeyen kod olarak değerlendirilir. Genel olarak, güvenli bir JDBC sürücüsünün yazılması işi sürücü sağlayıcısına bırakılır.

Java Sanal Makinesi güvenilmeyen applet'lar için daha önce bahsedilen kısıtlamaları içeren kendi iyi-belgelenmiş güvenlik kontrollerini kullanır Ancak, eğer bir JDBC sürücüsü üreticisi sürücüsüne özellikler ekleyerek modeli genişletmek isterse (örneğin, bir veri tabanı ile "konuşmak" için birkaç applet'ın aynı TCP soketi bağlantısını kullanmasına izin vermek gibi) bu durumda, her bir applet'ın bağlantıyı kullanmasına izin verilmesini kontrol etmek üreticinin sorumluluğundadır.

Java güvenlik modelinin bütünlüğünü muhafaza etmeye ek olarak, hem JDBC sürücüsü üreticisi hem de JDBC uygulama geliştiricisi, JDBC API'sinin bir ağ güvenlik modeli tanımlamadığını ve veri tabanı çağrılarını yürütmek için bir araç sağladığını akıında tutmalıdır. Kablodan veri tabanına gönderilen veri ve ortaya çıkan tablo bilgisi (örneğin, müşteri kredi kartı bilgisi isteğine karşılık) yayılmaktadır ve ağı merakla ve gizlice gözetleyebilen her terminal tarafından okunabilir.

JDBC Veri Tabanı Örneği

Aşağıdaki örnek şimdiye dek bu bölümde anlatılan kavramları kullanmaktadır; örnek yapaydır ve Statement, PreparedStatement ve CallableStatement arabirimlerinin kullanımını göstermek amacındadır.

Ele alınan basit veri tabanı, aşağıdaki tablo'da gösterilen plana sahip Customers adında bir tabloya sahiptir.

CustomerID	VARCHAR
LastName	VARCHAR
FirstName	VARCHAR
Phonenumber	VARCHAR
StreetAddress	VARCHAR
Zipcode	VARCHAR

Customer veri tabanı

Bu tablo geniş bir katalog sıralama sistemi ile ilgili bilgiyi depolayan daha büyük bir veri tabanının parçasıdır. Aşağıda iki temel yöntem, insertNewCustomer( ) ve getCustomer( ) ile birlikte basit bir Customer nesnesinin tanımı görülmektedir:

İmza:public class Customer

public Customer(Connection conn)

Sınıfın kurucusudur. Customer kurucusu, Statement başvurularını oluşturmak için kullandığı bir Connection nesnesini alır. Ek olarak kurucu bir PreparedStatement ve üç tane CallableStatement oluşturur.

public String insertNewCustomer(String lname, String fname, String pnum, String addr, String zip) throws insertFailedException, SQLException

Yeni bir kimlik bilgisi ile yeni bir müşteri kaydı açar. Kimlik bilgisi, müşteri kimlik bilgilerinin bu andaki listesini okuyan ve yeni bir başvuru oluşturan bir depolanmış prosedür aracılığıyla oluşturulur. Yöntem oluşturulan yeni kimlik bilgisini gönderir veya eğer ekleme başarısız olursa bir kural dışı durum uyarısı gönderir.

public CustomerInfo getCustomer(String custID) throws SQLException, selectException

Müşteri tablosundaki veriyi içeren bir nesne gönderir. Eğer gönderilen müşteri kimlik bilgisi mevcut değilse veya uygun olarak şekillendirilmemişse ya da SQL deyimi başarısız olursa, bir kural dışı durum uyarısı gönderir.

public static synchronized boolean validateZip(String zip) throws SQLException

Posta kodunun geçerliliğini denetlemek için yardımcı bir koddur. Posta kodu veri tabanındaki ZipCode tablosunda mevcut ise true değeri gönderilir.

public static synchronized boolean validateID(String id) throws SQLException

Bir müşteri kimlik bilgisinin geçerliliğini denetlemek için yardımcı bir yöntemdir. Eğer bu bilgi mevcutsa, yöntem true değerini gönderir.

Kaynak kodu CD-ROM'da javadevhdbk\ch11\Customer.java dosyasında bulunmakta ve aşağıda görülmektedir:

Customer.java

```
// Customer kayıt sınıfı
// bu sınıf veri tabanında müşteri verilerini saklamak ve erişmeye yarar
import java.sql.*;
public class Customer {
private Connection conn;
private PreparedStatement insertNewCustomer;
private CallableStatement getNewID;
public static CallableStatement checkZip;
public static CallableStatement checkID;
// Customer kurusu: Connection nesnesinin yerel bir kopyasını saklar
// sonra kullanmak üzere deyimler yaratır
public Customer (Connection c) {
connec;
try {
insertNewCustomer=conn.prepareStatement
("INSERT INTO Customers VALUES (?, ?, ?, ?, ?, ?)");
getNewID=conn.prepareCall ("{call getNewID (?)}");
checkID=conn.prepareCall ("{call checkID (?,?)}");
checkZip=conn.prepareCall ("{call checkZip (?,?)}");
}
catch (SQLException e) {
System.out.println
("Unable to create prepared and callable statements");
}}
```

```
// yeni müşteri kaydı yaratmak için yöntem
public String insertNewCustomer (String lname, String fname, String pnum,
String addr, String zip) throws insertFailedException, SQLException {
String newID;
// kayıtlı prosedürü kutlanarak yeni bir müşteri kodu üret
if ((newID = getNewID()) = = null) {
throw new insertFailedException ("could not get new ID");
}
// yeni customer ID'yi ekle
insertNewCustomer.setString (1, newID);
insertNewCustomer.SetString (2, lname);
insertNewCustomer.setString (3. fname);
insertNewCustomer.setString (4, pnum);
insertNewCustomer.setString (5. addr);
insertNewCustomer.setString (6, zip);
// deyimi icra et
if (insertNewCustomer.executeUpdate()! = 1) {
throw new insertFailedException ("could nöt execute insert");
}
return (newID);
}
// bu ID'ye sahip tek customer kaydını al
public CustomerInfo getCustomer (String custID)
throws selectException, SQLException  {
// önce ID'yi kontrol et
if (!validateID (custID)) {
throw new selectException ("no customer with ID: " + custID);
}
// select cümlesi yarat
Statement stmt = conn.createStatement();
// sonuçları al
ResultSet rs = stmt.executeQuery
```

```
("SELECT FROM Customer WHERE CustID =" + custID);
// bir CustomerInfo kap nesnesi yarat
CustomerInfo info = new CustomerInfo ();
info.CustomerID = rs.getString (1);
info.LastName = rs.getString (2);
info.FirstName = rs.getString (3);
info.PhoneNumber = rs.getString (4);
info.StreetAddress = rs.getString (5);
info.Zip = rs.getString (6);
return (info);
}
// müşteri posta kodunun testi
public static synchronized boolean validateZip (String zip)
throws SQLException {
checkZip.setString (1, zip);
checkZip.registerOutParameter (2, Types.BIT);
checkZip.executeUpdate();
return (checkZip.getBoolean(2));
}
// customer ID kontrolü
public static synchronized boolean validateID (String id)
throws SQLException {
checkID.setString (1, id);
check.ID.registerOutParameter (2. Types.BIT);
checkID.executeUpdate();
return (checkID.getBoolean(2));
}
private String getNewID () throws SOLException {
GetNewID.registerOutParameter (1. Types.VARCHAR);
getNewID.executeUpdate();
return (getNewID.getString (1));
}}
```

```
// istisnalar
// insertFailedException, SQL insert sorunu için genel bir istisnadır
class insertFailedExceptlon extends SQLException {
public insertFailedException (String reason){
super (reason);
}
public insertFailedException() {
super ();
}}
// selectException SQL select sorunu için genel bir istisnadır
class selectException extends SQLException {
public selectException(String reason){
super (reason);
}
public selectException() {
super ();
}}
```

İPUCU: CustomerInfo sınıfı basit bir kap nesnesidir. Kap sınıfları, veri tabanındaki müşteri tablosunu işleyen her yöntemden veya her yönteme, tüm bir müşteri kaydının gönderilmesini kolaylaştırır. Veri kap sınıfında saklanabilir ve her öğeyi tek bir başvuru olarak göndermek zorunda kalmak yerine, tek bir nesne başvurusu olarak gönderilebilir.

Aşağıdaki kod CD-ROM'da javadevhdbk\ch11\CustomerInfo.java dosyasında bulunabilir.

CustomerInfo.java

```
// Customer tablosu için bir kap nesnesi
public class CustomerInfo {
String CustomerID;
String LastName;
```

```
String FirstName;
String PhoneNumber;
String StreetAddress;
String Zip;
}
```

Son olarak, basit Customer sınıfını test etmek için aşağıda, bir Sysbase sürücüsünün yüklenmesini gösteren, sonra bir bağlantı yapan ve bir Customer nesnesinin yeni bir örneğine gönderilen Connection nesnesini gönderen basit bir Java uygulaması yer almaktadır.

CD-ROM'da javadevhdbk\ch11\Example.java dosyasında bulunan kod aşağıdadır:

Example.java

```
// DriverManager, Driver, Connection, Statement and ResultSet
// kullanımını gösteren basit bir örnek

import java.sql.*;
import sybase.sql.*;
public class Example {
Connection sybaseConn;
// main
public static void main (String arg[]) {
// url, username ve password iste
if (arg.length < 3) {
System.out.println ("Example use:");
System.out.println ("java Example <url> <username> <password>");
System.exit (1);
}
// sınıf örneğini yarat
```

```
Example ex = new Example ();
// bağlantıyı başlat
ex.initdb (arg[0], arg[1), arg[2]);
// bağlantıyı test et
ex.testdb ();
}
// veri tabanı bağlantısını başlatmak
public void initdb (String url, String user, String passwd) {
// veri tabanını aç ve bağlantı kur
try {
// bağlantı yarat
sybaseConn = DriverManager.getConnection(url, user, passwd);
}
catch (SQLException e) {
System.out.println ("Database connection failed:");
System.out.println (e.getMessage());
System.exit (1);
}}
// Customer class yöntemlerinin testi
public void testdb () {
String custID = null;
// Customer sınıf örneğini yarat
Customer cust = new Customer (sybaseConn);
try {
// yeni müşteri ekle
custID = cust.insertNewCustomer
("Jones", "8ill", "555-1234", "5 Main Street", "01234");
}
catch (SQLException e) {
System.out.println ("Insert failed:");
System.out.println (e.getMessage());
System.exit (1);
```

```
}
try {
// geri oku
CustomerInfo info = cust.getCustomer (custID);
}
catch (SQLException e) {
System.out.println ("Read failed:");
System.out.println (e.getMessage());
System.exit (1);
}}}
```

Bu örnek, posta kodunun ve müşteri kimliğinin geçerliliğini denetleyen depolanmış prosedür çağrılarını yapmak için CallableStatement ve her bir ekleme ile değişecek parametreler ile bir Insert SQL deyimini kullanmak için PreparedStatement arabirimlerinin kullanılmasını göstermektedir.

Örnek ayrıca, Customer sınıfında kullanılan depolanmış prosedürleri destekleyecek her JDBC sürücüsü ile çalıştırılacak kodu açıklamaktadır. Sürücü sınıf adları jdbc.drivers özelliğinden yüklendiğinden, tekrar derleme gerekli değildir.

JDBC Sürücüleri

JDBC API'sini cazip kılan özelliklerinden biri, sürücüler oluşturmak için tüm ana veri tabanı üreticilerinin paralel olarak çalıştıklarını bilerek uygulamalar geliştirilebilmesidir. Belirli sayıda sürücü, hem veri tabanı sağlayıcılarından hem de üçüncü-parti geliştiricilerden sağlanabilir. Birçok durumda en iyi özellikler, maliyet ve destek için çevreden alış-veriş yapmak akıllıcadır.

Sürücüler, yapılarına ve desteklemeleri amaçlanan veri tabanının tipine göre değişik özelliklerde gelir JavaSoft veri tabanı sürücülerini dört yolla sınıflandırır:

1. ODBC ikili kodu ve bazı durumlarda da bir istemci kütüphanesi ile gerçekleştirilir. Köprü sürücüsü üç kısımdan oluşur:

JDBC'yi ODBC sürücü yöneticisine bağlayan C kütüphaneleri takımı;
ODBC sürücü yöneticisi ve ODBC sürücüsü.

2. Doğal kütüphaneden-Java'ya uygulaması. Bu sürücü JDBCyi doğal istemci kütüphanesine çevirmek için doğal C dili kütüphane çağrılarını kullanır. Bu sürücüler, sağlayıcıya özgü işlevsellik sağlayan C dili kütüphanelerini kullanır ve bu kütüphaneleri (doğal yöntemler aracılığıyla) JDBC'e bağlar. Bu sürücüler ilk olarak Oracle, Sybase, Informix, DB2 ve diğer istemci tabanlı RDBM'ler için piyasaya sürülmüştür.
3. Ağ-Protokolü sürücüsü. JDBC çağrıları bu sürücü tarafından DBMS'den bağımsız bir protokole dönüştürülür ve bir soket üzerinden bir orta-kat sunucusuna gönderilir. Orta kat kodu istemci adına değişik veri tabanları ile bağlantı kurar. Bu yaklaşım en yaygın ve (büyük bir farkla) en esnek yöntem haline gelmektedir. Bu yaklaşım ayrıca, özel olarak firewall'lar aracılığıyla veri gönderilmesini içeren ağ güvenliği ile ilgili konuları ele alır.
4. Doğal-protokol Java sürücüsü. JDBC çağrıları doğrudan DBMS sunucusu tarafından kullanılan ağ protokolüne dönüştürülür. Bu sürücü senaryosunda, veri tabanı sağlayıcısı bir ağ soketini destekler ve JDBC bir soket bağlantısı üzerinden doğrudan veri tabanı sunucusu ile haberleşir. İstemci tarafındaki kod Java'da yazılabilir. Bu çözüm, uygulanması en basit olan çözüm olma avantajına sahiptir ve intranet kullanımı için oldukça pratiktir. Bununla birlikte, ağ protokolü sağlayıcı tarafından tanımlandığından ve özel olduğundan, sürücü genellikle sadece veri tabanı sağlayıcısından gelir.

JDBC-ODBC Köprüsü

JDBC-ODBC köprüsü JDBC çağrılarını ODBC işlemlerine dönüştürmeyi sağlayan bir JDBC sürücüsüdür. ODBC'yi destekleyen belirli sayıda DBMS vardır; Microsoft ölçeğinde bir firma veri tabanı erişimi için bir standart oluşturduğunda, mutlaka bunu izleyecek sağlayıcılar olacaktır ve gerçekte mevcut olan 50'den fazla farklı ODBC sürücüsü vardır.

Daha önce bahsedildiği gibi, hem JDBC hem de ODBC X/Open CLI'a dayanır, bu nedenle JDBC ile ODBC arasındaki dönüşüm nispeten açıktır. ODBC istemci tarafındaki bir kütüphaneler takımıdır ve istemcinin işletim sistemine ve, bazı durumlarda, makine mimarisine özgü bir sürücüdür.

Geliştiricinin perspektifinden bakıldığında, bir JDBC-ODBC köprüsünün kullanılması kolay bir seçimdir; uygulamalar yine de doğrudan JDBC arabirimi sınıfları ile haberleşecektir, dolayısıyla bu tamamen diğer bir JDBC sürücüsü kullanılması ile aynıdır. Ancak, bir JDBC-ODBC köprüsü uygulaması, geliştiricinin uygulamayı çalıştırmak için neye ihtiyaç duyulduğunu bilmesini gerektirir. ODBC çağrıları ikili C çağrıları kullanılarak yapıldığından, istemci ODBC sürücüsünün, ODBC sürücü yöneticisinin ve istemci tarafındaki kütüphanelerin yerel bir kopyasına sahip olmalıdır.

JavaSoft bu nedenlerle JDBC-ODBC köprüsünün Web-tabanlı veri tabanı erişimi için kullanılmamasını tavsiye eder. İntranet erişimi için, geliştirici Java programını ya bir Java uygulaması olarak ya da yerel istemci dosya sisteminden güvenilir bir kaynak olarak çalıştırılacak olan bir Java appiet'ı olarak istemci makinelere dağıtmalıdır.

Kullanılan JDBC Sürücüleri

JDBC sürücüleri birçok üretici tarafından ve o kadar hızlı bir şekilde sağlanmaktadır ki kesin bir liste vermek pratik olmayacak ve verildiği tarihte bile bu liste eskimiş olacaktır. Şu andaki sürücü üreticileri, ürün adları ve hangi veri tabanlarını destekledikleri konusunda bilgi edinmek için iyi bir kaynak:

http://splash.javasoft.com/jdbc/Jdbc.drivers.html

Alternatif Bağlanırlık Yöntemleri

JDBC veri tabanı uygulaması geliştirirken, zamanı ve gelecekteki yatırımları koruyan iyi bir yöntem ortaya koyar. API, JDBC standardına göre yazılan bir istemci programının her JDBC uyumlu sürücü ve veri tabanı bileşimi ile çalışacağını garanti

etmektedir. Sonraki kısımda, JavaSoft tarafından önerilen ve bir geliştirme yatırımını koruyan esnek bir yol sağlayan iki alternatif teknoloji ele alınmaktadır: Uzak Yöntem Çağırma (Remote Method Invocation, RMI) ve Ortak Nesne İsteği Aracı Mimarisi (Common Object Request Broker Architecture, CORBA).

Uzak Yöntem Çağırma (RMI)

Hem RMI hem de CORBA, istemci uygulamalarını veri tabanına sağlamak için kullanılabilir, bununla birlikte ele alınması gereken birkaç önemli nokta vardır. RMI, Uzak Prosedür Çağrılarına (Remote Procedure Calls, RPC) benzer, ancak RPC dağıtık nesne sistemleri için tasarlanmamışken, bu RMI'ın güçlü olduğu noktadır. RMI, istemci uygulamalarının uzaktaki bir sunucu üzerinde bulunan nesnelerin yöntemlerini sanki bu nesneler yerel nesnelermiş gibi yürütmesine izin verir.

Veri tabanı bağlanabilirliği için bunun anlamı, nesneler aslında veri tabanı sunucu makinesi üzerinde gerçekleştirilmiş olsa bile, geliştiricinin bu veri tabanı nesnelerine doğrudan erişen bir uygulama yazabilmesidir. RMI, nesnelerin serileştirilmiş akımlar olarak gönderilmesine imkan veren mekanizmalar sağladığından, bu akımların firewall'lar aracılığıyla gönderilmesi için oluşturulan protokolleri de destekler. RMI Java'dan-Java'ya bir çözüm olduğundan, çok-katlı tasarımlarda en iyi çözümü elde etmek için JDBC ve RMI birleştirilebilir. Mesela, eğer JDBC sürücüsü RMI kullanılarak yazılırsa, bu durumda standart bir veri tabanı arabirimi tanımı yazılabilir ve RMI aracılığıyla nesne sürekliliği ve uzak çağrılar kullanılabilir, bu suretle JDBC modeli genişletilebilir.

NOT: RMI hakkında daha fazla bilgi için şu JavaSoft Web sayfasına başvurabilirsiniz: http://chatsubo.javasoft.com/current/rmi/.

Ortak Nesne İsteği Aracı Mimarisi (CORBA)

Ortak Nesne İsteği Aracı Mimarisi (Common Object Request Broker Architecture, CORBA), Nesne Yönetim Grubu (Object Management Group, OMG)

tarafından yıllarca yürütülen çalışmaların ürünüdür. OMG, dağıtık nesne topluluklarına erişmek amacıyla farklı bilgisayar mimarileri üzerinde farklı bilgisayar dillerine izin veren bir haberleşme alt yapısı için bir spesifikasyon oluşturmuş olan 500'den fazla firmanın oluşturduğu bir konsorsiyumdur.

Veri tabanı uygulama geliştiricisi için, CORBA heterojen bir geliştirme ortamında en yüksek düzeyde esneklik sağlar. Sunucu C veya C++'da geliştirilebilir ve istemci bir Java applet'ı olabilir. Şu anda JavaSoft, bir CORBA 2.0 IDL dosyasını alan ve bir istemci uygulaması için gerekli çekirdek dosyalarını oluşturan bir Java Arabirim Tanımlama Dili (Interface Definition Language, IDL) derleyicisi oluşturma çalışmalarını sürdürmektedir.

CORBA sağlayıcıdan ve dilden bağımsız bir tanımlama dili tanımlayan bir standarttır. IDL, istemci ve sunucu uygulaması arasında bir anlaşma oluşturmak için kullanılır. IDL'nin kendisi bir uygulama dili değildir; sadece bu hizmetlerin bir uygulamasında gerçekleştirilebilen, nesne servislerini ve işlemlerini tanımlar. CORBA'nın çekirdeğinde Nesne İsteği Aracı (Object Request Broker, ORB) bulunur. ORB, bir CORBA uygulamasının istemcisi ve sunucusu arasında bilgi iletimi (işlem istekleri ve sonuçları) için temel bir bileşendir. ORB isteklerin düzenlenmesini yönetir, sunucuya bir bağlantı tesis eder, veriyi gönderir ve sunucu tarafında istekleri yürütür. Aynı işlem, sunucu çalışmanın sonuçlarını gönderdiğinde de meydana gelir.

CORBA 2.0 spesifikasyonu, ORB aracılığıyla istemci ve sunucu arasındaki bağlantının protokolünü tanımlayan bir İnternet Karşılıklı Çalışılabilirlik Protokolü (Internet Interoperability Protocol, IIOP) tanımlar. Bu protokol, geliştiricilerin iki farklı sağlayıcıdan bir istemci ve sunucu IDL derleyicisi seçmesine izin verir.

JavaSoft'un yanı sıra, IIOP ve Java IDL'yi içeren, CORBA 2.0 uyumluluğu sağlayan birkaç sağlayıcı vardır.

NOT: CORBA'yı daha yakından tanımak ve OMG konsorsiyumu hakkında daha fazla bilgi edinmek için şu Web sayfasına başvurabilirsiniz: http://www.omg.org/.

Nesne Veri Tabanlarına Bağlanırlık

RMI ve CORBA'nın yanı sıra diğer bir alternatif, özellikle Java'nın nesne modelini destekleyen bir nesne veri tabanının kullanılmasıdır. Java için birkaç nesne veri tabanı ürünü bulunmaktadır, ancak şu anda hiçbir standart tanımlanmamıştır. Şans eseri, Java nesne veri tabanları için bir API standardı üzerinde çalışan bir nesne veri tabanı standardı komitesi, Nesne Veri Tabanı Yönetim Grubu (Object Database Management Group, ODMG) bulunmaktadır.

NOT: ODMG'nin standart bir Java nesne veri tabanı üzerindeki çalışması hakkında bilgi edinmek için şu Web adresine başvurunuz: http://www.odmg.org/java.html

Web-Tabanlı Veri Tabanlarıyla Bağlanırlık

Özel olarak JDBC ve genellikle Java ile ilgili olmayan, diğer bir alternatif yöntem veri tabanlarına Web sayfalarından erişilmesidir. CGI script'lerine bilgi göndermek için HTML sayfaları kullanılabilir. CGT script'leri daha sonra, veri tabanına bağlanır ve sonuçları HTML sayfasına gönderir Webtabanlı veri tabanı pazarındaki sağlayıcılar, veri tabanı bağlantısı ve sorgulamaları yöneten, C veya C++'da yazılmış çok-kanallı uygulamalar ile CGI performansını artırmak için çeşitli yöntemlere sahiptir.

JavaSoft'da, CGI performansını artıran bir teknoloji sağlar. Jeeves API'si, bir sunucu üzerinde bir görevi gerçekleştirmek için küçük servlet'leri (Java'da yazılmış) yürüten applet'ları sağlayan bir sınıf takımı sunar. Servlet'ten, bir veri tabanı bağlantısını açması, sonuçları alması ve veriyi applet'a göndermesi istenebllir.

NOT: Servlet teknolojisi hakkında daha fazla bilgi için şu Web sayfasından faydalanabilirisiniz: http://java.sun.com/products/jeeves/.

NOT: Hem ücretsiz hem de ticari Web-tabanlı veri tabanı bağlantısı ürünlerini içeren şu Web sayfasını da ziyaret edebilirsiniz:
http://www.stars.com/Vlib/Providers/database.html.

İNTERNET'TE JAVA

Yazı Türleri ve Renkleri

Font Kontrolü

Yaratacağınız Java applet'lerinde değişik yazı türlerini kullanmanız gerekebilir. Bu yazı türleri değişik boyut ve özelliklerde olabilir. Bu bölümdede değişik yazı türleri işlenecektir.

Java dilinde, Sun Microsystems tarafından hazırlanmış çeşitli alt programlar bulunur. Herhangi bir yazıyı applet içerisinde değişik fontlar ve puntolarla yazabiliriz. Bunun için sisteme:

Hangi yazı türünü (font), yazı türünün hangi özelliğini (kalın, yatık v.s), yazı türünün boyutunu bildirmek zorundayız. Bu işlemi gerçekleştirebilmek için font türünden bir değişken tanımlamak gerekir. Bu tanımlamanın bir örneği aşağıda görülmektedir.

```
Font yaziTipi=new Font("TimesRoman", Font.BOLD,18);
```

Yukarıdaki program satırında her üç özellik de bulunmaktadır:
"TimesRoman" genel yazı tipini, Font.BOLD parametresi yazı tipinin özelliğini, 18 parametresi yazı tipinin puntosunu belirtir.

Sun Microsystems firmasına göre, Java sistemindeki bir alt program olan getFontList ile sistemde bulunan tüm fontlar bulunabiliyor.

Kullanılan yazı türü genelde 4 değişik özelliğe sahip olabiliyor:

Normal : Font.PLAIN
Normal : Font.BOLD
Normal : Font.ITALIC
Normal : Font.BOLD + Font.ITALIC

Şimdi yazacağımız bir Html dosyası ve hazırlayacağımız bir applet ile TimesRoman türünden bir düz yazıyı değişik özellikleri ile belirli bir puntoda ekrana getireceğiz. Programımızı doğru olarak çalıştırdıktan sonra değerleri değiştirip çeşitli denemeler yapabilirsiniz. Örnekten sonra kullanılan değişik alt programlar açıklanacaktır.

Örnek için önce bir Html dosyası hazırlayın ve bu dosyaya fontlar.html adını verin. Aşağıda fontlar.html web sayfası kaynak kodu yazılmıştır.

```
<html>
<title>fontlar.html</title>
<body>
<applet code=fontlar.class width=250 height=150>
</applet>
</body>
</html>
```

Şimdi de aşağıda kaynak kodunu gördüğünüz fontlar.java applet'ini yazınız.

```
//Bu applet ile değişik font örnekleri ekrana yansıtılacaktır.

import java.awt.*;
import java.applet.*;
public class fontlar extends Applet {
public void paint(Graphics temp) {
Font normalFont=new Font("TimesRoman", Font.PLAIN,18);
Font kalinFont=new Font("TimesRoman", Font.BOLD,18);
Font yatikFont=new Font("TimesRoman", Font.ITALIC,18);
Font yatikkalinFont=new Font("TimesRoman", Font.BOLD+Font.ITALIC,18);
temp.setFont(normalFont);
temp.drawString("NORMAL FONT", 10, 25);
```

```
temp.setFont(kalinFont);
temp.drawString("KALIN FONT",10,50);
temp.setFont(yatikFont);
temp.drawString("YATIK FONT",10,75);
temp.setFont(yatikkalinFont);
temp.drawString("YATIK ve KALIN FONT",10,100);
}}
```

Şimdi applet'imizi derleyelim. Bunun için MS-DOS Komut istemindeyken aşağıdaki komutu yazınız.

Javac fontlar.java

Derleme işleminden sonra appletimizi appletviewer programı ile görelim bunun için:

appletviewer fontlar.html komutunu yazınız.

Şimdi örneğimizi inceleyelim:

```
Font normalFont=new Font("TimesRoman", Font.PLAIN,18);
Font kalinFont=new Font("TimesRoman", Font.BOLD,18);
Font yatikFont=new Font("TimesRoman", Font.ITALIC,18);
Font yatikkalinFont=new Font("TimesRoman", Font.BOLD+Font.ITALIC,18);
```

satırları ile dört değişik yazı türü tanımlanmıştır. Herhangi bir yazının ekranda belirmesi için drawString() metodunu kullandık. Fakat önce hangi yazı türünü seçtiğimizi sisteme:

graf.setFont(normalFont); adlı komut satırı ile bildirmemiz gerekir.

Aşağıda fontlarla ilgili bir metod tablosu hazırlanmıştır:

Metod Adı	Hangi Nesnede Olduğu	İşlevi
GetFont()	Graphics	Önceden setFont() ile belirlenmiş olan yazı türü parametrelerini elde etmeye yarar
GetName()	Font	Font adını bir dizi olarak geri gönderir.
GetSize()	Font	Font puntosunu integer olarak geri gönderir.
GetStyle	Font	Font özelliğini integer olarak geri gönderir. 0:normal, 1:kalın, 2:yatık, 3:kalın ve yatık
İsPlain()	Font	Font stili normal için true ve false değerini geri gönderir.
isBold()	Font	Font stili kalın için true ve falsedeğerini geri gönderir.
İsItalic()	Font	Font stili yatık için true ve false değerini geri gönderir.

Renk Kontrolu

Hazırladığımız Java Applet'lerinin yalnızca gri ve siyah renklerden oluşması, ekranda okunabilir bir bir görüntü sağlamasına rağmen diğer renklerin kullanılmasıda gerekebilir. Java sisteminde ekran renklerinin kontrolu üç ana renk olan kırmızı (red), yeşil (green) ve mavi (blue) nin karışımından oluşur. Her üç rengin dozajı 0-255 ünite oranında ayarlanabilir. Buna göre de 256*256*256=16,777,216 değişik renk olasılığı ortaya çıkar. Sistemdede belirli renkler tanımlanmıştır. Bu ön tanımlamaları color class'ından alabiliriz:

Tanımlanmış Hazır Renkler:

Renk Adı	**Renk**	**RGB değeri**
Color.white	Beyaz	255,255,255
Color.black	Siyah	0,0,0
Color.lightGray	açık gri	192,192,192
Color.gray	Gri	128,128,128

Color.darkGray	koyu gri	64,64,64
Color.red	Kırmızı	255,0,0
Color.green	Yeşil	0,255,0
Color.blue	Mavi	0,0,255
Color.yellow	Sarı	255,255,0
Color.magenta	Mor	255,0,255
Color.cyan	Mavi	255,175,175
Color.orange	Turuncu	255,200,0

Eğer applet'niz için yukarıdaki renkler yetersiz kalıyorsa sizler istediğiniz kombinasyonla yeni renkler tanımlayabilirsiniz.

Bu işlem için:

Color renk=new Color(100,150,250) biçiminde bir komut vermeniz gerekir. Yukarıdaki örnekte açık mavi bir renk elde edersiniz. Şimdi bir örnek yapalım, ekranda bir kaç renkli kare çizelim:

Renkli Kareler Programı

Bu program içinde kullanılacak rengi belirleyen setColor(), istenilen bir alanı seçilen renkle dolduran fillRect() ve bir dikdörtgen çizmeye yarayan drawRect() yöntemleri kullanılmıştır.

Aşağıdaki Html dosyasını kutular.html adı altında oluşturun:

```
<html>
<title>kutular.html /title>
<body>
<applet code=kutular.class width=150 height=100>
</applet>
</body>
</html>
```

Aşağıda kutular.java applet'inin kaynak kodunu görmektesiniz

```
//Bu program değişik renkte kutuları applet içine yerleştirir.

import java.awt.*;
import java.applet.*;
public class kutular extends Applet {
public void paint (Graphics sakla) {
int i,j;
i=5;
j=10;
sakla.setColor(Color.red);
sakla.fillRect (i,j,25,25);
sakla.setColor (Color.black);
sakla.drawRect (i-1,j-1,25,25);
i=i+30;
sakla.setColor (Color.cyan);
sakla.fillRect (i,j,25,25);
sakla.setColor (Color.black);
sakla.drawRect (i-1,j-1,25,25);
i=i+30;
sakla.setColor (Color.magenta);
sakla.fillRect (i,j,25,25);
sakla.setColor ( Color.black);
sakla.drawRect (i-1,j-1,25,25);
i=i+30;
sakla.setColor (Color.green);
sakla.fillRect (i,j,25,25);
sakla.setColor (Color.black);
sakla.drawRect (i-1,j-1,25,25);
}}
```

Programın içinde kullandığımız sakla.setColor(Color.red); komut satırı ile bu komutu takip eden komutların, hangi renkten olmasını istediğimizi belirtiyoruz.

sakla.fillRect(i,j,25,25); komut satırı ile sol kenardan i'nin değeri kadar, üst kenardan j'nin değeri kadar uzaklıkta olan, 25*25 pixel boyutlarındaki bir alanı, daha önceden belirlediğimiz renkle dolduruyoruz. Bu işlem bittikten sonra, renkle doldurduğumuz alanın etrafına siyah bir çerçeve çizmek istediğimizden önce siyah rengi sakla.setColor(Color.black); komutu ile belirliyor. Daha sonrada, sakla.drawRect(i-1,j-1,25,25); komutu ile bir dikdörtgen çiziyoruz. Dikdörtgenin boyutları 25*25 pixel olduğundan bir kare elde ediyoruz. Değişkenlerden i ve j'nin değerlerini bir pixel azaltmakla (i - 1ve j - 1) dikdörtgeni, renk ile doldurduğumuz alanların birer pixel gerisinde olmasını sağlıyoruz.

Gelişi Güzel Renkli Kutular

Yazacağımız yeni bir applet ile gelişigüzel renkli kutuları iç içe çift döngü kullanarak ekrana getirelim.

Bunun için yine önce Html kodlarını yazalım:

```
<html>
<title>birsurukutu.html></title>
<body>
<applet code=birsurukutu.class width=500 height=150>
</applet>
</body>
</html>
```

Java Kaynak Kodu :

```
// Bu program gelisiguzel kutuları applet içine yerleştirir...
import java.awt.*;
```

```
import java.applet.*;
public class birsurukutu extends Applet {
public void paint(Graphics temp){
int kirmiziorani,yesilorani,maviorani;
int i,j;
i=5;
j=10;
while(j<130) {
while(i<450) {
kirmiziorani=(int)Math.floor(Math.random()*256);
yesilorani=(int)Math.floor(Math.random()*256);
maviorani=(int)Math.floor(Math.random()*256);
temp.setColor(new Color(kirmiziorani,yesilorani,maviorani));
temp.fillRect(i,j,25,25);
temp.setColor(Color.black);
temp.drawRect(i-1,j-1,25,25);
i=i+30;
}
//i<450 dongusunun sonu
i=5;
j=j+30;
}
//j<130 dongusunun sonu
}
//paint(Graphics temp) sonu
}
//birsurukutu appletinin sonu
```

Html dosyasına birsurukutu.html, Java dosyasına ise birsurukutu.java adını verin. Applet her çalıştırıldığında değişik ranklerle karşılaşılır.

Program içinde integer türünden tanımladığımız kırmiziorani, yesilorani, maviorani değişkenlerine;

```
kirmiziorani=(int)Math.floor(Math.random()*256);
yesilorani=(int)Math.floor(Math.random()*256);
maviorani=(int)Math.floor(Math.random()*256);
```

komutları ile 0 ile 255 arasında rastgele (random) değerler veriliyor. Üç temel renk olan kırmızı, yeşil ve mavinin değişik oranları ile de değişik renkler elde ediyoruz ve elde ettiğimiz bu renklerle çizdiğimiz kutuların içini dolduruyoruz. Böylece değişik renklere sahip bir çok kutu elde etmiş oluyoruz.

Ön ve Arka Plan Renkleri

Bu zamana kadar yazdığımız Applet'lerin arka planı gri, Applet üzerindeki renkler ise siyahtı. Bu bölümde bu standart renkleri nasıl değiştireceğimizi göreceğiz. Bunu daha iyi anlayabilmek için hemen bir örnek yapalım. Html dosyası ve Java kaynak kodları aşağıda verilmiştir.

```
<html>
<title>renkliyazi.html<</title>
<body>
<applet code=renkliyazi.class width=450 height=150>
</applet>
</body>
</html>
```

Java kaynak kodu:

```
// Bu program ile degisik yazılar applet icine yerlestirilir
import java.awt.*;
import java.applet.*;
public class renkliyazi extends Applet {
public void paint(Graphics yazi) {
```

```
setBackground(Color.red);
setForeground(Color.blue);
Font kalin=new Font("TimesRoman",Font.BOLD,36);
Font yatikkalin=new Font("TimesRoman",Font.BOLD+Font.ITALIC,24);
yazi.setFont(kalin);
yazi.drawString("Ne mutlu Türküm diyene!",10,60);
yazi.setFont(yatikkalin);
yazi.drawString("Mustafa Kemal ATATÜRK",60,100);
}}
```

Program çalıştırıldığında applet'in arka planının kırmızı yazıların ise mavi olduğu görülür.

Bu programda kullanılan yeni komutlardan setBackground(Color.red); satırı ile applet arka plan rengini kırmızı; setForeground(Color.blue); satırı ile ise applet alanı içindeki yazıların mavi olmasını sağladık.

Grafik ve Resimler

Grafik

Java Koordinat Sistemi

Java applet'leri için tanımlanmış olan grafik koordinat sistemi diğer grafik koordinat sistemlerinden biraz farklıdır. Aşağıdaki şekilde java koordinat sistemi pixel birimi ile verilmiştir.

Applet penceresinin en üst sol noktası x=0 ve y=0, başka bir deyişle 0,0 koordinatlarına sahiptir. Pencerenin sağ tarafına ilerledikçe x koordinatının, alt tarafına ilerledikçe de y koordinatının değeri artar. Koordinatları ayarlarken önce x sonra y koordinat değerleri verilmelidir.

Resimler

Resim Dosyaları

İnternet dünyasında browserlar ile geziye çıkanlara en cazip gelen nokta şüphesiz ki eriştiği yazısal bilgilerin yanında gördüğü resimlerdir. Resimler genelde web sayfalarının estetiğini arttırdıkları gibi çeşitli bilgiler ve grafikleri içerebilirler. Bilgisayarda bir resim dosyası oluşturmanın birden fazla yolu vardır. Örneğin windows'un Paint programı ile herkes resim oluşturabilir. Fakat her program farklı tiplerde veya uzantılarda resim oluşturmaktadır. Paint programı ise .BMP uzanılı dosyalar hazırlanmaktadır. Bu tür dosyaların en büyük dezavantajları çok fazla yer kaplamaları ve internet üzerinden erişimleri sırasında erişim süresini uzatmalarıdır. İşte bu nedenle web sayfalarında .JPG ve .GIF uzantılı resim dosyaları kullanılmaktadır. Değişik grafik programları ile .BMP uzantılı dosyalar .JPG veya .GIF dosyalarına çevrilebilir.

Normal bir fotoğrafı bilgisayara aktarabilmek için tarayıcı (scanner) kullanılabilir. Tarayıcı programları tarama işlemi bittikten sonra, kullanıcının istekleri doğrultusunda istenilen tür ve özelliklerde resim dosyası yaratabilir. Aynı zamanda digital fotoğraf makinaları ile çekilen fotoğraflar anında bilgisayara aktarılabilir.

Resimli Web Sayfası örneği hazırlayacağımız yeni bir Html dosyası ve Java programı ile bir web sayfası üzerine yazı ve resim yerleştireceğiz. Aşağıda resim.html dosyasının kodu verilmiştir:

```
<html>
<title>resim.html </title>
<body>
<applet code="resim.class" width=300 height=400>
</applet>
</body>
</html>
```

resim.java kaynak kodu:

```
import java.awt.*;
import java.applet.*;
public class resim extends Applet{
Image resim1;
public void init(){
resim1=getImage(getCodeBase(), "./cindy.gif"); }
public void paint(Graphics resim){
setBackground(Color.blue);
resim.drawImage(resim1,10,10,this);
}}
```

Program anlaşılır olması açısından oldukça kısa hazırlanmıştır. Önce resim (Image) türünden bir değişken olan resim1'i aşağıdaki şekilde Image resim1; tanımlanıyor ve daha sonra,

```
public void init() {
resim1=getImage(getCodeBase(), "/cindy.gif"); }
```

yöntemi ile programın resim dosyasını okumasını sağlıyoruz. Kullanılan resim programın bulunduğu dizinde bulunmak zorundadır. Son olarakta,

```
public void paint(Graphics resim);{
setBackground(Color.blue);
resim.drawImage(resim1,10,10,this); }
```

program satırları ile arka plan renginin mavi olmasını ve resmin ekranda belirmesini gerçekleştiriyoruz.

Yukarıdaki appleti biraz değiştirip, bir döngü ve bir resim daha eklersek nasıl bir görüntü elde ederdik acaba, şimdi bunu deniyelim. Aşağıdaki Java Kaynak koduna kayanresim.java adını verdim.

```
import java.awt.*;
import java.applet.*;
public class kayanresim extends Applet{
Image resim1;
Image resim2;
public void init(){
resim1=getImage(getCodeBase(), "./cindy.gif");
resim2=getImage(getCodeBase(), "./claudia.gif"); }
public void paint(Graphics resim) {
int i;
setBackground(Color.blue);
for (i=10; i<700; i=i+10){
if(i<=530) resim.drawImage(resim1,i,10,this);
resim.drawImage(resim2,700-i,10,this);
}}}
```

Program çok kolay anlaşılır niteliktedir. Programda,

Image resim1;
Image resim2; image tipinde iki adet değişken tanımlanmıştır. Daha sonra,

```
public void init(){
resim1=getImage(getcodeBase(), "./cindy.gif");
resim2=getImage(getcodeBase(), "./claudia.gif"); }
```

yöntemi ile programın resim dosyalarını okuması sağlanmıştır.Daha sonrada

```
public void paint(Graphics resim)
int i;{
```

```
setBackground(Color.blue);
for (i=10; i<700; i=i+10){
if(i<=530) resim.drawImage(resim1,i,10,this);
resim.drawImage(resim2,700-i,10,this);
}}
```

program satırları ile resimlerin ekrana getirilmesi sağlanmış ve for döngüsü yardımıyla resimlerin applet penceresi sınırları içinde yer değiştirilerek çizilmesi sağlanmıştır. For döngüsündeki şartın (i<700) olmasının sebebi applet penceresinin genişliğinin 700 alınmış olması ve resimlerin applet penceresi dışına çıkmasını engellemek içindir. Resimler çizilirken y değişkeni sabit tutulmuş, x değişkeni ise i'ye bağlı olarak değiştirilmiştir.

Resim Boyutları Kontrolu

Applet içine yerleştirmek istediğimiz resimlerin boyutlarını Java programları içinden ayarlayabiliriz. Önce herhangi bir gif veya jpg türünden bir resim dosyası bulun, bulduğunuz bu resmi çalışmakta olduğunuz alana koyun (programı kaydettiğiniz dizinin içi). Benim bu bölümde kullandığım resim ana dizinde bulunan (C:/) demi.jpg dosyası. Bu resmin boyutları 297*480 boyutlarında ve 640*480 dpi çözünürlüğe sahip ve disk üzerinde 25240 byte yer kaplıyor. Eğer bu dosya gif formatında kaydedilseydi 96351 byte yer kaplayacaktı. Görüldüğü gibi JPG türü az yer kaplayan bir resim türüdür. Bu nedenle iletişim hızı diğer türlere göre daha fazladır bu nedenle internette en çok kullanılan türdür.

Şimdi aşağıdaki resimboyutlari.html dosyasını hazırlayalım.

```
<html>
<title>resimboyutlari.html</title>
<body>
<applet code=resimboyutlari.class width=500 height=450>
```

```
</applet>
<body>
</html>
```

Java programının kaynak kodu ise aşağıda verilmiştir:

```
import java.awt.*;
import java.applet.*;
public class resimboyutlari extends Applet {
//  TANIMLAMALAR-----------------------------------------------------------------------------
Image demi;
Font kalinfont12=new Font("TimesRoman",Font.BOLD,12);
Font kalinfont18=new Font("TimesRoman",Font.BOLD,18);
Font kalinfont24=new Font("TimesRoman",Font.BOLD,24);
// RESIM DOSYASINI OKU-----------------------------------------------------------------------
public void init(){
demi=getImage(getCodeBase(), "./demi.jpg"); }
// RESIM DOSYASINI EKRANA GETIR ------------------------------------------------------
public void paint(Graphics resim){
// Bu Metodta Kullanılan Degiskenler
int xKoordinati;
int yKoordinati;
int resminEni;
int resminYuksekligi;
int resminYkoordinati;
//Yazi tipini belirle
resim.setFont(kalinfont18);
//Resmin Enini ve Yüksekligini Belirle
resminEni=demi.getWidth(this);
resminYuksekligi=demi.getHeight(this);
//Baslangıc Koordinatlarini Ayarla
```

```
xKoordinati=10;
resminYkoordinati=10;
yKoordinati=resminYkoordinati;
//Resmi Ekrana Yerlestir
resim.drawImage(demi,xKoordinati,yKoordinati,this);
//Mesajın y Koordinatını Belirle
yKoordinati=resminYkoordinati+resminYuksekligi+20;
//Mesaj yaz
resim.drawString("Orjinal Boyutlar", xKoordinati,yKoordinati);
//Ikinci Resmin Sol Ust Koordinatlarini Belirle
xKoordinati=xKoordinati+resminEni+10;
yKoordinati=resminYkoordinati;
//Resmi %50
faltresim.drawImage(demi, xKoordinati, yKoordinati, resminEni/2, resminYuksekligi/2,
this);
//Mesajin y Koordinatini belirle
yKoordinati=resminYkoordinati+resminYuksekligi/2+20;
//Mesaj Yaz
resim.drawString("%50 ufak",xKoordinati,yKoordinati);
//Ucuncu Resmin Sol ust Koordinatlarini belirle
xKoordinati=xKoordinati+(resminEni/2)+10;
yKoordinati=resminYkoordinati;
//Resmi %75 Ufalt
resim.drawImage(demi,xKoordinati,yKoordinati,resminEni/4,resminYuksekligi/4,this);
//Mesajin y Koordinatini Belirle
yKoordinati=resminYkoordinati+resminYuksekligi/4+20;
//Mesaj Yaz
resim.drawString("%75 ufak",xKoordinati,yKoordinati);
}}
```

İlk resmin orjinal boyutunda ve sol üst koordinatları

```
xKoordinati=10;
```

resminYkoordinati=10;
Ykoordinati=resminYkoordinati;
komutları ile belirleniyor ve orjinal resim
resim.drawImage(demi,xKoordinati,yKoordinati,this);
komutu ile ekrandaki yerine yerleştiriliyor. Bu işlemden önce ise
resminEni=demi.getWidth(this);
resminYuksekligi=demi.getHeight(this);
komutları ile getWidth() ve getHeight() metotlarından yararlanılıyor, Orjinal resmin en ve yüksekliğini pixel olarak belirliyoruz.

Resmin altına resim.draw.String("Orjinal boyutlar",xKoordinatı,yKoordinatı);
komutu ile bir bilgi satırı yolluyoruz.

Diğer iki resim için yine drawImage() metodunu kullanıyoruz, fakat bu kez parametre sayıları biraz değişik:
resim.drawImage(demi,xKoordinati,yKoordinati,resminEni/2,resminYuksekligi/2,this);
satırında görüldüğü gibi yKoordinatı değişkeni ile this sözcüğü arasında iki parametre daha bulunuyor ve bu parametrelerle x ve y yönlerinde resmi hangi oranda küçültmek istediğimizi belirtiyoruz. İkinci resimde bu oran /2 olduğundan, orjinal resmin eni ve boyu ikiye bölünüyor ve %50 oranında ufalmış oluyor. Üçüncü resimde oran /4 ve resim dolayısıyla %75 oranında ufalmış oluyor.

Eğer bir resmin iki misli büyümesini istersek yukarıdaki program satırını aşağıdaki şekilde değiştirmeniz gerekiyor.

resim.drawImage(demi, xKoordinati, yKoordinati, resminEni*2, resminYuksekligi*2, this);

Resmin genel görüntüsünün bozulmaması için resmi küçültme veya büyültme işlemleri gerek x, gerekse y yönünde aynı olmalıdır. Aksi taktirde değişik görüntüler ortaya çıkabilir. Bu nedenle küçültme ve büyültme işlemlerine dikkat etmenizde yarar var.

Bu bölümde yapılan örneklerde yalnızca Internet'ten alınan çeşitli fotoğraflar kullanılmıştır. Tabii ki resim dosyaları çeşitli grafik ve çizim dosyalarınıda içerebilmektedir. Sizde bu bölüme kadar öğrendiklerinizle bazı denemeler yaparak çok değişik görünümde applet'ler üretebilirsiniz.

Ses

Sesin Önemi

Yirmi sene kadar önceki bilgisayarlarda henüz ekranlar gelişmediğinden, bilgisayarın kullanıcıya verdiği tüm yanıtlar kağıt üzerinde olurdu. Kullanıcılar sorunlarını delikli kartlar üzerinde hazırlarlar, cardreader denilen aygıtlarla bu kartların içerikleri bilgisayara iletilir, bilgisayarın yanıtları ise teleks türü yazıcılarla kağıt üzerine çıkardı. Klavye ve ekranların geliştirilmesi ile bu işlemler oldukça kolaylaştı. Artık kullanıcı bilgisayar ile bir tür diyalog kurabiliyor, sorunlarını klavye ile bilgisayara aktarıyor, yanıtları ise ekran üzerinde beliren yazılardan alıyordu. artık bu yöntemde zamanla yetersiz kaldı. Ekranda artık yalnızca yazılar değil, resimler, grafiklerde belirmeye başladı. Mouse ve dokunmatik ekranlar gibi aygıtlarla klavye kullanımı oldukça azaldı. Artık bir kullanıcı, özellikle Windows gibi grafiksel işletim sistemlerinde, klavyeden daha çok mouse'u kullanmaya başladı. Bilgisayardan alınan yanıtlarda görsel olmanın yanında, ses olarakta oluşmaya başlayınca, kullanım oldukça kolaylaştı. Yakın bir gelecekte eminim ki klavye ve mouse'un yerinide ses alacaktır ve bilgisayarla yapılan tüm diyaloglar ses ile gerçekleştirilecektir.

Java programlama dilide, sesli ortamı desteklemektedir. Bu bölümde Java programları içine nasıl, ses dosyalarını yerleştireceğimizi, bu dosyaları nasıl kullanabileceğimizi, özellikle applet üzerinde belirli alanları klikleyince belirli sesleri, sesli mesajları nasıl elde edeceğimizi öğreneceğiz.

Ses Dosyaları

Windows işletim sisteminin standart ses dosyaları .WAV cinsinden olan dosyalardır, fakat java bu tür dosyaları desteklemez. Java'nın desteklediği ses dosyaları .AU türünden olup 8 bit mono ses içerirler. Ses kaliteleri pek iyi değildir, buna karşılık boyutları ufak olduğundan Internet bilgi iletişiminde fazla bir zaman kaybına neden olmazlar. Burada yapacağımız örneklerde kullanacağımız .AU ses dosyalarını Internet'ten elde ettim.

```
<html>
<title>ses.html</title>
<body>
<applet code=ses.class width=130 height=130>
</applet>
</body>
</html>
```

Şimdide aşağıdaki Java kaynak kodunu yazın.

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class ses extends Applet {
public void paint(Graphics graf) {
graf.drawRect(10,10,100,100);
play(getCodeBase(), "eye.au");
}}
```

Programımızı derleyip appletviewer ile çalıştırırsak, bilgisayarınıza bir ses kartı ve hoparlör bağlı ise garip bir gülme sesi duyacaksınız.

Bu programda kullanılan ses dosyası eye.au adını taşıyor. Fakat siz dilediğiniz başka bir ses dosyasınıda kullanabilirsiniz. Bunun için Internet'in olanaklarından faydalanabilir ve bir çok değişik ses dosyasını bilgisayarınıza alabilirsiniz. İşte size birçok .AU dosyasını bulabileceğiniz bir adres:

http://info.fuw.edu.pl/multimedia/sounds/

Bu programda applet çalıştığı anada sesleri duymaya başlıyoruz. Belirli bir süre sonrada bu sesler kesiliyor. Eğer browser'ınızdaki scroolbarları (sağ tarafında bulunan yatay çubuk) applet gözden kaybolup yeniden ekrana gelecek şekilde hareket ettirirseniz sesler yeniden duyulmaya başlayacaktır. Applet penceresine her yeniden çiz komutu geldiğinde Java ses dosyasınıda yeniden çalacaktır.

Ses dosyasını:
play(getCodeBase(),"eye.au"); komutu ile devreye sokuyoruz. Programda dikdörtgen çizme nedenimiz, bu dikdörtgenin her çiziminde ses dosyasının Java tarafından çalınmasını sağlamaktı. Eğer bilgisayarınızda ses kartınız yoksa sadece dikdörtgen çizimini göreceksiniz.

Ses dosyalarının applet'in çalıştığı alanda bulunması zorunlu değildir. Programa ufak bir eklenti ile ses dosyalarınızı istediğiniz dizinde tutabilirsiniz. Bunun için programı:
play(getCodeBase(), "/audio/eye.au"); şeklinde değiştirsek eye dosyasının audio adlı dizinde olması gerekir. Burada dikkat etmeniz gereken en önemli özellik kullanılan taksim işaretinin normal taksim işareti (/) olduğudur. Eğer ters taksim kullanılırsa ses dosyası program tarafından bulunmayacak ve herhangi bir ses duyulamayacaktır.

Mouse Kontrollu Ses Kullanımı ve Mouse Koordinatları

Yazacağımız yeni appletin daha büyük bir alana ihtiyacı olduğundan önce yeni bir html dosyası oluşturalım.

```
<html>
<title>mouseses.html</title>
<body>
<applet code=mouseses.class width=400 height=300>
</applet>
</body>
</html>
```

Yazacağımız programın Java kaynak kodu:

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class mouseses extends Applet {
int xKoordinati;
int yKoordinati;
//-----------------------------------------------------
public void paint(Graphics graf) {
String xPozisyonu;
String yPozisyonu;
graf.drawRect(10,10,150,150);
graf.drawRect(175,10,150,150);
xPozisyonu="Farenin x pozisyonu:";
yPozisyonu="Farenin y pozisyonu:";
xPozisyonu=xPozisyonu+String.valueOf(xKoordinati);
yPozisyonu=yPozisyonu+String.valueOf(yKoordinati);
graf.drawString(xPozisyonu,25,200);
graf.drawString(yPozisyonu,25,250);
}
//paint class'inin sonu
```

```
public boolean mouseDown(java.awt.Event e,int x,int y) {
xKoordinati=x;
yKoordinati=y;
if ((x>9) & (x<161) & (y>9) & (y<161))
play(getCodeBase(),"inek.au");
else if ((x>174) & (x<326) & (y>9) & (y<161))
play(getCodeBase(),"kopek.au");
else
play(getCodeBase(),"kuzu.au");
repaint();
return true;
 }
//mouseDown metodunun sonu }
```

Soldaki dikdörtgen içerisinde mouse'u kliklerseniz, sağdaki dikdörtgen içindeki kliklemeden farklı bir ses duyarsınız. Her iki dikdörtgenin dışında kalan bir alanda butona bastığınızda değişik üçüncü bir ses duyarsınız.

Mouse'un her kliklenişinden sonra pencerenin alt bölümünde, klikleniş esnasında mouse cursor'unun hangi koordinatlarda olduğu ekranda belirecektir.

Şimdi programımızı açıklamaya geçelim:
Mouse koordinatlarını barındıran iki değişkeni:

```
int xKoordinati;
int yKoordinati;
```

program satırları ile tanımlıyoruz. Ekrana yansıyacak olan iki mesajın tanımı ise

```
String xPozisyonu;
String yPozisyonu;
```

satırları ile gerçekleştiriyoruz. Applet penceresi içine yerleştirmek istediğimiz 2 dikdörtgeni

```
graf.drawRect(10,10,150,150);
```

graf.drawRect(175,10,150,150);
satırları ile çiziyoruz. Ekranda belirmesi gereken mesajın içeriğini
xPozisyonu="Farenin x pozisyonu:";
yPozisyonu="Farenin y Pozisyonu";
xPozisyonu=xPozisyonu+String.valueOf(xKoordinati);
yPozisyonu=yPozisyonu+String.valueOf(yKoordinati);
satırları ile hazırlıyor ve
graf.drawString(xPozisyonu,25,200);
graf.drawString(yPozisyonu,25,250);
satırları ile bu mesajı ekrana getiriyoruz.

Bundan sonraki mouseDown metodu, applet üzerinde herhangi bir yer kliklendiğinde devreye giriyor. Klikleniş esnasında cursor kordinatlarını
xKoordinati=x;
yKoordinati=y;
satırları ile saklıyoruz. Sıra klikleniş koordinatlarına göre değişik bir ses dosyasını devreye sokmaya geliyor. Bu mantığı,
if ((x>9) & (x<161) & (y>9) & (y<161))
play(getCodeBase(),"inek.au");
else if ((x>174) & (x<326) & (y>9) & (y<161))
play(getCodeBase(),"kopek.au");
else play(getCodeBase(),"kuzu.au");
program satırları ile gerçekleştiriyoruz. Soldaki dikdörtgenin sol üst köşesi pencere içinde x=10 ve y=10 koordinatlarında bulunuyor. Eni ve yüksekliği 150 pixel olduğundan bu dikdörtgenin sınırlarını (x>9 & x<161 & y>9 & y<161) ifadeleri belirtir. Buna göre x koordinatlarının değeri 10 ile 160 arasında ise ve y koordinatlarının değeride 10 ile 160 arasındaysa, mouse butonu soldaki dikdörtgen içinde kliklenmiş olacaktır. Basit bir if tekniği ile bu belirleme yapılmıştır.
if (x>9 & x<161 & y>9 & y<161)
Bu if tekniğinde eğer karşılaştırma doğru ise fare soldaki dikdörtgende kliklenmiş olacak ve play(getCodeBase(),"inek.au"); program satırı ile inek.au dosyası çalınacak ve metod terkedilecektir. Sağdaki dikdörtgenin sol üst köşesi pencere içinde x=175 ve

y=10 koordinatlarında bulunuyor. Eni ve yüksekliğide 150 pixel olduğundan sınırları kolayca hesaplanıp basit bir if tekniği yazılmıştır. Buna göre x koordinatının değeri 175 ile 325 arasında ise ve y koordinatlarının değeride 10 ile 160 arasında ise, mouse butonunun sağdaki dikdörtgen içinde kliklendiği ortaya çıkıyor. Bunu if mantığı ile şu şekilde kullandık.

else if (x>174 & x<326 & y>9 & y<161) bu karşılaştırmanın sonucu doğru ise play(getCodeBase(),"kopek.au); program satırı ile kopek.au dosyasını çalıp , metodu terkediyoruz.

Eğer her iki dikdörtgenin içinde de mouse kliklenmemişse
else play(getCodeBase(),"kuzu.au");
program satırı ile kuzu.au dosyası çalınıp metod terkediliyor.

rapaint(); komutu ile ise mouse her kliklenişinde ses dosyalarının çalınmasını sağlıyoruz.

Susmayan Bilgisayar

Bir applet'in çaldığı bir ses dosyasının daha değişik bir kontrol yöntemini, aşağıda yaptığımız yeni programda görmektesiniz. Yine önce HTML dosyamızı oluşturalım.

```
<html>
<title>susmayan bilgisayar</title>
<body>
<applet code=music.class width=400 height=300>
</applet>
</body>
</html>
```

Şimdi de Java programımızın kaynak kodunu yazalım.

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class music extends Applet {
int xKoordinati;
int yKoordinati;
//---------------------------------------------------------
public void paint(Graphics graf) {
String xPozisyonu;
String yPozisyonu;
graf.drawRect(10,10,150,150);
graf.drawRect(175,10,150,150);
xPozisyonu="Farenin x pozisyonu:";
yPozisyonu="Farenin y pozisyonu:";
xPozisyonu=xPozisyonu+String.valueOf(xKoordinati);
yPozisyonu=yPozisyonu+String.valueOf(yKoordinati);
graf.drawString(xPozisyonu,25,200);
graf.drawString(yPozisyonu,25,250);
}
//paint class'inin sonu
public boolean mouseDown(java.awt.Event olay,int x,int y) {
xKoordinati=x;
yKoordinati=y;
AudioClip music1=getAudioClip(getCodeBase(),"music1.au");
AudioClip music2=getAudioClip(getCodeBase(),"music2.au");
AudioClip music3=getAudioClip(getCodeBase(),"ciglik.au");
if (x>9 & x<161 & y>9 & y<161) {
music2.stop();
```

```
music1.loop();
}
else if (x>174 & x<326 & y>9 & y<161) {
music1.stop();
music2.loop();
}
else {
music1.stop();
music2.stop();
music3.play();
}
repaint();
return true;
}
//mouseDown metodunun sonu
}
```

Bu programda,

```
AudioClip music1=getAudioClip(getCodeBase(),"music1.au");
AudioClip music2=getAudioClip(getCodeBase(),"music2.au");
AudioClip music3=getAudioClip(getCodeBase(),"ciglik.au");
```

satırları ile üç değişik ses dosyasını, üç parça olarak tanımlıyoruz. Bu üç değişik müzik dosyasına,
muzik1, muzik2 ve muzik3 adlarını veriyoruz.

Eğer bir parçayı bir kez çalmak istiyorsak, muzik1.play(); komutu yeterlidir. Sürekli olarak çalmak istiyorsak muzik1.loop(); komutunu kullanıyoruz. Sürekli olarak çalmakta olan bir ses dosyasını durdurmak için ise muzik1.stop(); komutundan yararlanıyoruz.

Örneğimizde sol dikdörtgen kliklenince sürekli bir müzik çalınır, aynı şekilde sağ dikdörtgen kliklenince de sürekli değişik bir müzik çalınır. Dikdörtgenlerin dışındaki bir alan kliklenincede bir çığlık duyularak bilgisayar susar.

Aşağıdaki örnekte, çoklu sunum ortamında kaydedilen bir sesin Java ile nasıl bir Web sayfasına aktarıldığını görmektesiniz. Bu işlem için çoklu sunu ortamının ses kaydedici modülü ve mikrofon ile ses .WAV formatında kaydedilir. WAV formatı ile kaydedilen ses .AU Formatına (Goldwave, v.b.yazılımlar kullanılarak) dönüştürülür. Aşağıda kodlaması verilen Java programı ile kullanıma sunulur.

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class benimsesim extends Applet {
public void paint(Graphics graf) {
graf.drawRect(10,10,100,100);
play(getCodeBase(),"ses.au");
}
}
```

Animasyon

Animasyonun Önemi

Ekranlar önce yalnızca yazılar içindi. Sonra ekranlar üzerinde basit grafikler olan ekranlar ortaya cıktı. 1985'te CGA (Color Graphics Adepter) içeren ekranlar piyasaya sürülünce, normal iş ortamları dışında kalanlar da bilgisayar sistemleri ile bilgilenmeye başladılar. 1986'da IBM VGA ve MCGA ekran tekniklerini tanıtmaya başlayınca ekrandaki görüntüler fotoğraf kalitesine yaklaştı. Günümüzdeki ekran kartları ile televizyon tekniğinden çok daha üstün görüntüler elde etmeye basladık. Sıra

ekranda hareket edebilen görüntülere geldi. Çesitli televizyon kanallarında izlediğimiz reklamlarda hatta Holywood'un görkemli filmleri bile bilgisayar desteği ile hazırlanmaya başlandı.

Burada film ve reklam değil, Internet'in web sayfalarında hareket edebilen görüntülerin nasıl hazırlandığının prensiplerini açıklamaya çalısıyoruz. Hareket eden, sabit olmayan görüntülere ise 'animasyon' adı verilmektedir.

JAVA dili ile animasyon programlarının nasıl hazırlanabileceğini bu bölümde öğreneceğiz. Yine bu bölümde JAVA programları içinde mouse desteğiyle nasıl çizim yapabileceğimizi örnekleri ile incelemek olanagı bulacağız.

Animasyon 1
Çizim Programı

İlk örneğimiz basit bir çizim programı olacak. Bu programda applet icerisinde bulunan bir dikdörtgende mouse butonu ile çizim yapacağız.

Bu bölümdeki tüm HTML dosyalaına cizim.htm tüm java programlarına da cizim.java adını veriyoruz. İşte aşağıda cizim.html dosyası:

```
<html>
<title>cizim.html</title>
<body>
<applet code=cizim.class width=350 height=350>
</applet>
</body>
</html>
```

Yazacağımız çizim.java programının kaynak kodu ise:

```
import java.awt.*;
```

```
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class cizim extends Applet {
//TANIMLAMALAR.....
int fareXeski;
int fareYeski;
int fareXyeni;
int fareYyeni;
Graphics cizim2;
boolean cizim2Tamam=false;
//DIKDÖRTGEN CİZ.....
public void paint (Graphics cizim) {
cizim.drawRect(10,10,330,330);
}
// Butona basılı hareket.....
public boolean mouseDrag (java.awt.Event olay,int x,int y) {
// Dikdörtgen dışında hiç bir şey yapma!
if (x<10|x>340|y<10|y>340)
return true;
// dikdörtgen icinde eski fare pozisyonu ile yeni
//fare pozisyonu arsinda bir cizgi ciz!
fareXyeni=x;
fareYyeni=y;
if (cizim2Tamam==false) {
cizim2=getGraphics();
cizim2Tamam=true;
}
cizim2.drawLine(fareXeski,fareYeski,fareXyeni,fareYyeni);
fareXeski=fareXyeni;
fareYeski=fareYyeni;
```

```
return true;
}
//BUTONA BASILI.....................................
public boolean mouseDown(java.awt.Event olay,int x,int y) {
fareXeski=x;
fareYeski=y;
return true;
}
}
```

Programımızı: javac cizim.java, appletViewer cizim.html komutları ile derleyip çalıştırın.

Şimdi programımız içinde kullandığımız teknikleri inceleyelim:

Programımızın başında tanımladığımız değişkenler

```
int fareXeski;
int fareYeski;
int fareXyeni;
int fareYyeni;
Graphics cizim2;
boolean cizim2Tamam=false;
```

Program içinde tüm bölümler için geçerli. Bu tür değişkenlere ingilizce kaynaklı programlama kitaplarında global variables (her yerde geçerli değişkenler) adı verilir. MouseDown() metodunu önceki bölümlerden (Ses Bölümü) tanıyoruz. Farenin herhangi butonu kliklediğimizde devreye giren bu alt program, bize butona basış anındaki farenin X ve Y koordinatlarını bildiriyor. MouseDrag() metodu ise fare hareket ettirildigi anda devreye giriyor. Bu mantığın içinde farenin hareketinden oluşan yeni X ve Y koordinatlarını belirliyoruz. Bu yeni koordinatlar ile eskileri arasını düz bir hatla birleştirerek çizgi çiziyoruz. Dikkat ederseniz mouseDrag() ve mouseDown metodu aynı parametrelere sahip fakat bu parametreler içinde Graphics türünden bir parametre

(örnegin cizim) yer almıyor. Graphics türünden bir degişken ile çizi en kolay sekilde yapılabileceğinden, getGraphics() metodundan yararlanmak zorunda kalıyoruz. Bu metod bir kez kullanıldığı anda bize program başında tanımladığımız Graphics türünden bir değişken olan cizim2 değişkeninden faydalanmamız olanağını sağlıyor.

Dikkat edilmesi gereken önemli bir nokta ise getGraphics() metodunu yalnızca bir kez devreye sokmamız zorunluluğu. Aksi takdirde bu metodu her çağırışımızda bilgisayarımızın ana belleğinden (RAM) belirli bir kısmını kaybetmiş olacağımız ve dolayısıyla belirli bir kullanım süresi sonunda bellekte yer kalmadığı mesajı ile karşılaşacağımızdır. Yine programımızın başında tanımladığımız boolean türünden bir degişken olan cizimTamam'ın baslangic degeri false ve getgraphics() metodu yalnızca bu değer false oldugu anda devreye giriyor. Bir kez devreye girdikten sonra graf2Tamam değişkeninin değerini true olarak sabitliyor ve gereken önlemi almış oluyoruz.

Renkli Çizim Programı

Son örneğimizi geliştirdim ve değişik renkleri klikledikten sonra büyük dikdörtgen içinde seçtiğim renkte çizim yapabileceğim bir program yazdım. Bu program içinde, önceki bölümlerde açıkladığımız tekniklerden yararlandım. Eğer bu örneği anlamakta zorluk çekmiyorsanız, şimdiye kadar incelediğimiz örnekleri iyice 'kavramış' durumdasınız demektir.

İşte son örneğin ve HTML ve JAVA kaynak kodları:

```
<html>
<title>cizim.html</title>
<body>
<applet code=cizim.class width=450 height=350>
</applet>
</body>
</html>
```

Java kaynak kodu:

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class cizim extends Applet {
//Tanimlamalar.....
int fareXeski;
int fareYeski;
int fareXyeni;
int fareYyeni;
Graphics cizim;
Graphics cizim2;
boolean cizim2Tamam=false;
//dikdörtgen ciz.....
public void paint (Graphics cizim) {
int ii=1;
int yKoor;
cizim.setColor(Color.black);
cizim.drawRect(10,10,330,330);
yKoor=10;
while(ii<11) {
cizim.setColor(Color.black);
cizim.drawRect(360,yKoor,20,20);
switch (ii){
case 1: cizim.setColor(Color.red);
break;
case 2: cizim.setColor(Color.green);
break;
```

```
case 3: cizim.setColor(Color.blue);
break;
case 4: cizim.setColor(Color.yellow);
break;
case 5: cizim.setColor(Color.magenta);
break;
case 6: cizim.setColor(Color.cyan);
break;
case 7: cizim.setColor(Color.pink);
break;
case 8:cizim.setColor(Color.orange);
break;
case 9: cizim.setColor(Color.black);
break;
case 10: cizim.setColor(Color.white);
break;
}
cizim.fillRect(361,yKoor+1,19,19);
yKoor=yKoor+30;
ii++;
}}
//BUTONA BASILI HAREKET.....
public boolean mouseDrag(java.awt.Event olay,int x,int y) {
//renk secimi kareleri kliklenince, secilen rengi ayarlar
if (x>361& x<380 &y> 10& y<30)
cizim2.setColor(Color.red);
if (x>361& x<380 &y> 40& y<60)
cizim2.setColor(Color.green);
if (x>361& x<380 & y> 70 & y<90)
cizim2.setColor(Color.blue);
if (x>361& x<380 & y> 100 & y<120)
cizim2.setColor(Color.yellow);
```

```
if (x>361& x<380 & y> 130 & y<150)
izim2.setColor(Color.magenta);
if (x>361& x<380 & y> 160 & y<180)
cizim2.setColor(Color.cyan);
if (x>361& x<380 & y> 190 & y<210)
cizim2.setColor(Color.pink);
if (x>361& x<380 & y> 220 & y<240)
cizim2.setColor(Color.orange);
if (x>361& x<380 & y> 250 & y<270)
cizim2.setColor(Color.black);
if(x>361& x<380 & y> 280 & y<300)
cizim2.setColor(Color.white);
//Fare butonuna cizim cercevesi disinda basiliysa birsey yapma
if (fareXeski<10| fareXeski>340|fareYeski<10|fareYeski>340)
return true;
//Fare butonuna cizim cercevesi disinda basiliysa birsey yapma
if (x <10|x >340|y <10|y>340)
return true;
fareXyeni= x;
fareYyeni=y;
if(cizim2Tamam==false) {
cizim2=getGraphics();
cizim2Tamam=true;
}
cizim2.drawLine(fareXeski,fareYeski,fareXyeni,fareYyeni);
fareXeski=fareXyeni;
fareYeski=fareYyeni;
return true; }
//BUTONA BASILDI.....
public boolean mouseDown(java.awt.Event olay,int x,int y) {
fareXeski=x;
fareYeski=y;
```

```
return true;
}}
```

Yay Çizimi

Son örneğimizde yalnızca düz çizgiler çizebilmiştik. Aşağıda kaynak kodunu gördüğünüz JAVA programı ile düz hat yerine yay çizimini gerçekleştirebiliriz. Kaynak kodunda yalnızca bir önceki örnekte bulunan public boolean mouseDrag() değişiklik yapılmıştır. Programın diğer bölümlerinde herhangi bir değişiklik yapılmamıştır.

Java kaynak kodu:

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class cizim extends Applet {
// Tanimlamalar.....
int fareXeski;
int fareYeski;
int fareXyeni;
int fareYyeni;
Graphics cizim;
Graphics cizim2;
boolean cizim2Tamam=false;
//dikdörtgen ciz.....
public void paint (Graphics cizim) {
int ii=1;
int yKoor;
cizim.setColor(Color.black);
cizim.drawRect(10,10,330,330);
```

```
yKoor=10;
while(ii<11) {
cizim.setColor(Color.black);
cizim.drawRect(360,yKoor,20,20);
switch (ii) {
case 1: cizim.setColor(Color.red);
break;
case 2: cizim.setColor(Color.green);
break;
case 3: cizim.setColor(Color.blue);
break;
case 4: cizim.setColor(Color.yellow);
break;
case 5: cizim.setColor(Color.magenta);
break;
case 6: cizim.setColor(Color.cyan);
break;
case 7: cizim.setColor(Color.pink);
break;
case 8:cizim.setColor(Color.orange);
break;
case 9: cizim.setColor(Color.black);
break;
case 10: cizim.setColor(Color.white);
break;
}
cizim.fillRect(361,yKoor+1,19,19);
yKoor=yKoor+30;
ii++;
}}
//BUTONA BASILI HAREKET.....
public boolean mouseDrag(java.awt.Event olay,int x,int y) {
```

```
//renk secimi kareleri kliklenince, secilen rengi ayarlar
if (x>361& x<380 &y> 10& y<30)
cizim2.setColor(Color.red);
if (x>361& x<380 &y> 40& y<60)
cizim2.setColor(Color.green);
if (x>361& x<380 & y> 70 & y<90)
cizim2.setColor(Color.blue);
if (x>361& x<380 & y> 100 & y<120)
cizim2.setColor(Color.yellow);
if (x>361& x<380 & y> 130 & y<150)
cizim2.setColor(Color.magenta);
if (x>361& x<380 & y> 160 & y<180)
cizim2.setColor(Color.cyan);
if (x>361& x<380 & y> 190 & y<210)
cizim2.setColor(Color.pink);
if (x>361& x<380 & y> 220 & y<240)
cizim2.setColor(Color.orange);
if (x>361& x<380 & y> 250 & y<270)
cizim2.setColor(Color.black);
if(x>361& x<380 & y> 280 & y<300)
cizim2.setColor(Color.white);
//Fare butonuna cizim cercevesi disinda basiliysa birsey yapma
if (fareXeski<10| fareXeski>340|fareYeski<10|fareYeski>340)
return true;
//Fare butonuna cizim cercevesi disinda basiliysa birsey yapma
if (x <10|x >340|y <10|y>340)
return true;
fareXyeni= x;
fareYyeni=y;
if(cizim2Tamam==false) {
cizim2=getGraphics();
cizim2Tamam=true;
```

```
}
cizim2.drawArc(fareXeski,fareYeski,50,50, 0,90);
fareXeski=fareXyeni;
fareYeski=fareYyeni;
return true;
}
//BUTONA BASILDI.....................
public boolean mouseDown(java.awt.Event olay,int x,int y) {
fareXeski=x;
fareYeski=y;
return true;
}}
```

Kullandığımız mouseDrag() metodundaki tek değişiklik, graf2.drawArc(fareXeski,fareYeski,50,50, 0,90); satırları arasında bulunuyor.

Kullandığımız mouseDrag() metodunda altı değişik paremetre bulunuyor.Bunlar sırası ile,
applet üzerinde baslangic xKoordinatı applet üzerinde baslangıç yKoordinatı
applet üzerindeki yayın eni
applet üzerindeki yayın yüksekliği
applet üzerindeki yayın başlangıç açısı
applet üzerindeki yayın bitiş açısı

Daire Çizimi

Son örneğimizdeki cizim2.drawArc(fareXeski,fareYeski,50,50, 0,90); komut satırını aşağıdaki biçimde değiştirisek daire çizimini elde ederiz.

```
cizim2.drawArc(fareXeski,fareYeski,50,50, 0,360);
```

Bitiş açısını 360 olarak belirledik. Bir yayın başlangıcı 0° de başlar ve bu yayın bitiş noktası 360° olarak belirlenirse sonuçta bir tam daire elde edilir.Biz de aslında bir yay çizerek , bir daire görüntüsü elde ettik. Bundan sonraki örneğimizde çizdiğimiz dairenin içini nasıl dolduracagımızı öğreneceğiz.

Daire çizimi Java kaynak kodu:

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class cizim extends Applet {
// Tanimlamalar.....
int fareXeski;
int fareYeski;
int fareXyeni;
int fareYyeni;
Graphics cizim;
Graphics cizim2;
boolean cizim2Tamam=false;
//dikdörtgen ciz.....
public void paint (Graphics cizim) {
int ii=1;
int yKoor;
cizim.setColor(Color.black);
cizim.drawRect(10,10,330,330);
yKoor=10;
while(ii<11) {
cizim.setColor(Color.black);
cizim.drawRect(360,yKoor,20,20);
switch (ii) {
```

```
case 1: cizim.setColor(Color.red);
break;
case 2: cizim.setColor(Color.green);
break;
case 3: cizim.setColor(Color.blue);
break;
case 4: cizim.setColor(Color.yellow);
break;
case 5: cizim.setColor(Color.magenta);
break;
case 6: cizim.setColor(Color.cyan);
break;
case 7: cizim.setColor(Color.pink);
break;
case 8:cizim.setColor(Color.orange);
break;
case 9: cizim.setColor(Color.black);
break;
case 10: cizim.setColor(Color.white);
break;
}
cizim.fillRect(361,yKoor+1,19,19);
yKoor=yKoor+30;
ii++;
}}
//BUTONA BASILI HAREKET.....
public boolean mouseDrag(java.awt.Event olay,int x,int y) {
//renk secimi kareleri kliklenince, secilen rengi ayarlar
if (x>361& x<380 &y> 10& y<30)
cizim2.setColor(Color.red);if (x>361& x<380 &y> 40& y<60)
cizim2.setColor(Color.green);
if (x>361& x<380 & y> 70 & y<90)
```

```
cizim2.setColor(Color.blue);
if (x>361& x<380 & y> 100 & y<120)
cizim2.setColor(Color.yellow);
if (x>361& x<380 & y> 130 & y<150)
cizim2.setColor(Color.magenta);
if (x>361& x<380 & y> 160 & y<180)
cizim2.setColor(Color.cyan);
if (x>361& x<380 & y> 190 & y<210)
cizim2.setColor(Color.pink);
if (x>361& x<380 & y> 220 & y<240)
cizim2.setColor(Color.orange);
if (x>361& x<380 & y> 250 & y<270)
cizim2.setColor(Color.black);
if(x>361& x<380 & y> 280 & y<300)
cizim2.setColor(Color.white);
//Fare butonuna cizim cercevesi disinda basiliysa birsey yapma
if (fareXeski<10| fareXeski>340|fareYeski<10|fareYeski>340)
return true;
//Fare butonuna cizim cercevesi disinda basiliysa birsey yapma
if (x <10|x >340|y <10|y>340)
return true;
fareXyeni= x;
fareYyeni=y;
if(cizim2Tamam==false) {
cizim2=getGraphics();
cizim2Tamam=true; }
cizim2.drawArc(fareXeski,fareYeski,50,50,0,360);
fareXeski=fareXyeni;
fareYeski=fareYyeni;
return true;
}
//BUTONA BASILDI.....
```

```
public boolean mouseDown(java.awt.Event olay,int x,int y) {
fareXeski=x;
fareYeski=y;
return true;
}}
```

Mouse kullanımı ile ilgili applet'ler yukarıda anlatılmaktadır. Eğer bu applet'lerin yazımını anladıysanız, değişik animasyonları kolayca yapabilirsiniz. İşte buna bir örnek:

Gerçek Animasyon

Bu zamana kadar yalnızca çeşitli çizim ve resimlerin ekranda belirmelerini ve bu nesnelerin mouse yardımıyla ekrandaki applet penceresi içerisinde hareket ettirilmelerini öğrendik. Resim veya nesnelerin pencere içinde bir 'sinema filmi ' gibi hareket etlmesi üzerinde henüz bir çalısma yapmadık. 'Gercek animasyon' diye adlandırdığımız bu tür bir çalışmayı bu bölümde yapacağız. Değişik, birbirlerinde az farkları olan resimleri ekran üzerine yerleştirmekle bir hareket izlemininin nasıl sağlanacağını araştıracağız. Türü gif olan bu resimlerden tümünü Java programlarını yazdığınız çalışma alanına kopyalayınız. ResimlerKopyalanan bu resimler diskinizde aşağıdaki şekilde yer alacaktır.

```
Directory of c:\
. <dir> 17/06/97 11:21
.. <dir> 17/06/97 11:21
t2 gif 1.840 13/06/97 2:54 t2.gif
t3 gif 1.840 13/06/97 2:54 t3.gif
t4 gif 1.840 13/06/97 2:54 t4.gif
t5 gif 1.840 13/06/97 2:54 t5.gif
t6 gif 1.840 13/06/97 2:54 t6.gif
t7 gif 1.840 13/06/97 2:54 t7.gif
t8 gif 1.840 13/06/97 2:54 t8.gif
```

t9 gif 1.840 13/06/97 2:54 t9.gif
t10 gif 1.840 13/06/97 2:54 t10.gif
t11 gif 1.840 13/06/97 2:54 t11.gif
t1 gif 1.840 21/07/97 2:54 t1.gif

Bu hareket izlenimi yaratabilmek için ardarda değişik resimleri ekran üzerine getirmeliyiz. Bunun için biraz önce kopyaladığınız .GIF dosyalarını kullanacağız. Bu resimleri t1.gif'ten başlayarak, t11.gif'e kadar ardarda ekran üzerinde aynı koordinatlarda gösterebiliriz.

Asağıdaki .html ve .JAVA kaynak kodlarını kullanarak ilk denememizi yapalım. Html kodu:

```
<html>
<body>
<title>animasyon.html</title>
<applet code=animasyon.class width=120 height=120>
</applet>
<body>
</html>
```

Java Kaynak Kodu:

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class animasyon extends Applet {
//Tanimlamalar.....
Graphics cizim;
boolean cizimTamam=false;
```

```
Image resim1, resim2,resim3,resim4,resim5;
Image resim6, resim7,resim8,resim9,resim10,resim11;
//Resim dosyalarini oku.....................................
public void init() {
resim1=getImage(getCodeBase(), "t1.gif");
resim2=getImage(getCodeBase(), "t2.gif");
resim3=getImage(getCodeBase(), "t3.gif");
resim4=getImage(getCodeBase(), "t4.gif");
resim5=getImage(getCodeBase(), "t5.gif");
resim6=getImage(getCodeBase(), "t6.gif");
resim7=getImage(getCodeBase(), "t7.gif");
resim8=getImage(getCodeBase(), "t8.gif");
resim9=getImage(getCodeBase(), "t9.gif");
resim10=getImage(getCodeBase(),"t10.gif");
resim11=getImage(getCodeBase(),"t11.gif");
}
//Resim dosyalarini ardarda göster
public void paint(Graphics cizim) {
cizim.drawImage(resim1,30,30,this);
cizim.drawImage(resim2,30,30,this);
cizim.drawImage(resim3,30,30,this);
cizim.drawImage(resim4,30,30,this);
cizim.drawImage(resim5,30,30,this);
cizim.drawImage(resim6,30,30,this);
cizim.drawImage(resim7,30,30,this);
cizim.drawImage(resim8,30,30,this);
cizim.drawImage(resim9,30,30,this);
cizim.drawImage(resim10,30,30,this);
cizim.drawImage(resim11,30,30,this);
}}
```

Bilgisayarlarlar günümüzde o kadar hızlı çalışıyorlar ki, son örneğimizde bir hareket elde demiyoruz. Ancak programımızı yapay olarak yavaşlatabiliriz.Ancak buda etkili bir çözüm olmayacaktır. Pekala ne yapmamız gerekiyor. Bunu yapmak için Java Programlama dilinde kullanılan Thread özelliğinden yararlanıp animasyonun sürekli hareketi sağlanabilmektedir. Animasyon-2 bölümünde Thread Yöntemi anlatılmaktadır

Animasyon 2
THREADS

Thread'ler bilgisayarın mikro işlemcisini esir almadan çalışan programlardır. Böyle bir program çalışırken mikroişlemci başka işlemlerde yapabilir. Bu program tekniğine multitasking özelliği denir. Java birden fazla Thread'ın aynı anda çalışmasına destekler, fakat aynı anda birden çok thread çalışmaktaysa sistem yavaşlayabilir. Thread programları sonsuz döngü içerebilirler; ne de olsa kullanıcı başka işlemler yapabileceğinden, sonsuz döngüler sistemi kilitlemezler.

Yazacağımız yeni bir program ile bu programlama tekniğini öğreneceğiz. Bu örneğimiz içinde integer türünden bir sayacı, bir sonsuz döngü içerisinde ekranda göstereceğiz, buna rağmen sistemimiz kilitlenmeyecek.

Resimli Thread Örneği

Bir önceki örnekte resimlerle yapmış olduğumuz program tatmin edici bir sonuca ulaşmamıştı. Şimdi thread niteliği ile beklediğimiz neticeye ulaşacağız.

İşte thread1 programının html dosyası:

```
<html>
<body>
<title>thread1.html</title>
<applet code=thread1.class width=250 height=150>
</applet>
```

```
<body>
</html>
```

Java Kaynak Kodu:

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class thread1 extends Applet implements Runnable {
Thread thread1=null;
String resimNo;
Image[] resimler=new Image[20];
int ii;
Image aktuelResim;
//---------------------------------------------------------------
//Resim dosyalarını okuyup resimler adlı dizi içine koy
public void init() {
ii=0;
while (ii<=11) {
ii=ii+1;
resimNo= "t"+String.valueOf(ii)+".gif";
aktuelResim=getImage(getCodeBase(),resimNo);
resimler[ii]=aktuelResim; }
ii=1;
}
//---------------------------------------------------------------
//Resmi ekrana getiren program mantığı
public void paint(Graphics graf) {
Font kalinFont=new Font("TimesRoman", Font.BOLD,16);
graf.setFont(kalinFont);
```

```
resimNo= "t"+String.valueOf(ii)+".gif";
aktuelResim=resimler[ii];
graf.drawString("Resim dosya ismi :" + resimNo,10,50);
graf.drawImage(resimler[ii],30,70,null);
ii++;
if (ii==12) {
ii=1;
}}
//---------------------------------------------------------------
public void start() {
thread1=new Thread(this);
thread1.start();
}
//---------------------------------------------------------------
public void stop() {
thread1.stop();
}
//---------------------------------------------------------------
public void run() {
while(true) {
try {
Thread.currentThread().sleep(200);
}
catch(InterruptedException ie) {
}
repaint();
}}}
```

Şimdi programın açoklamasına geçelim:

Herhangi bir java programını bir thread biçiminde hazırlamak için aşağıdaki tanımlamanın yapılması zorunludur.

public class thread1 extends Applet implements Runnable

Yukarıdaki tanımlamada gördüğümüz yenilik implements Runnable sözcükleridir.

Program içindeThread thread1=null; satırı ile thread1 adlı Thread cinsinden bir değişken tanımladık. Sistem bu değişkenin içinde bir thread tanıtım numarası (Tread ID Number) barındırır.

Start Metodu

Java, start( ) metodunu bir applet çalıştığı anda otamatik olarak başlatır. Bu nedenle thread programını, kendisini otamatik olarak çalıştıracak olan start( ) metodu içerisine yerleştiriyoruz.

```
public void start( ) {
thread1=new Thread(this);
thread1.start( );
}
```

Yukarıdaki kod içinde thread1=new Thread(this); komutu ile sistemin bir thread tanıtım numarası (Thread ID Number) belirlemesini gerçekleştiriyoruz. Thread programının devreye girmesini ise thread1.start( ); komutu ile sağlıyoruz.

Stop Metodu

Thread yöntemi, çalışan bir programın başlattığı başka bir program anlamına geldiğinden, çalışma program sona erdiği zaman, thread programının çalışmasını durdurmak gerekir. Bu işlemi ise stop( ) metodu ile aşağıdaki komut satırları ile gerçekleştiriyoruz:

```
public void stop( ) {
```

```
thread1.stop( );
}
```

Bu işlemi yapmayacak olursak, applet sona ermesine rağmen, thread çalışmasına devam edecektir.

Run Metodu

Bir thread'in esas çalışma yöntemi run( ) metodu içinde yapılır. Örneğimizde bu metod için aşağıdaki kod kullanılmıştır.

```
public void run( ) {
while(true) {
try {
Thread.currentThread( ).sleep(200);
}
catch(InterruptedException ie) {
}
repaint( );
}}
```

Yukarıdaki program bölümünü inceleyecek olursak, run( ) metodu için gereken tüm prensipleri görürüz. Bir while döngüsü içinde 3 ayrı komut satırı görüyoruz:

1-) İçinde sleep( ) (uyu, uykuya yat)metodunu bulunduran komut satırı thread programının kaç milisaniye, hiçbir işlem yapmadan beklemesini belirtiyot. Bu bekleme anında işetim sistemi diğer programların komutlarını icra edebiliyor.
2-) Catch komut satırı
3-) Catch komut satırını takip eden program satırı

İlk bölümdeki Thread.curentThread( ).sleep(200); satırı ile başlattığımız thread programının 200 milisaniye süre için devre dışı kalmasını sağlıyoruz. Bu sürenin resim

içeren animasyonlarda 200-3000 milisaniye arasında olması gerçeğe yakın hareket izlenimi yaratır.

Döngüyü dışardan gelen bir komutla sona erdirmek için: catch (InterruptedException ie){} komut satırını kullanıyoruz.

Thread programının yeniden sleep( ) metodu ile uykuya yatmasından önce, bu programın yapmasını istediğimiz komut satırlarını sıralıyoruz. Örneğimizde yalnızca bir komut satırı bulunuyor: repaint( )

Bu komut satırı ise paint( ) metodu ile ekrana getirilen bilgileri güncelleştiriyor.

Şimdi de örneğimizdeki komut satırlarına biraz göz atalım, programımızın aşağıdaki komut satırları:

```
public void init( ) {
ii=0;
while (ii<=11) {
ii=ii+1;
resimNo= "t"+String.valueOf(ii)+".gif";
aktuelResim=getImage(getCodeBase(),resimNo);
resimler[ii]=aktuelResim;
}
ii=1;
}
```

Hatırlarsak, gerçekleştirmek istediğimiz animasyonda, tam olarak 11 değişik resmimiz vardı. Bu resim dosyalarının türü .GIF türünden oluşmakta, her resim dosyasının baş harfi ise "t" harfi ile başlamaktaydı. Bu 11 resmi program içine yüklemek için, herbirinin adını sabit olarak belirtmedik; bir döngü ile resim dosyalarının adını hazırladık. Integer türünden bir değişken olan "ii" yi String.valueOf(ii) fonksiyonu ile string türüne çevirip,

resimNo="t"+String.valueOf(ii)+".gif";
komutu ile başına bir "t" harfi, sonuna da ".gif" yazısını ekleyerek Java programı içinde adını belirttiğimiz dosyayı hard disk'ten okumak istediğimizi belirttik. Okuma işlemini;
aktuelResim=getImage(getCodeBase(),resimNo);
komutu ile gerçekleştirip, okunan resmi, Image türünden bir değişken olan aktuelResim içine yerleştirdik; daha sonra bu resmi, resimler adlı resim tabelası içine, sırasıyla koyduk. Animasyon için resim resim dosyalarını her kullanılış anında hard disk'ten okuma işlemi çok zaman alacağından ve istenen hareket elde edilemeyeceğinden, programın geri kalan bölümünde bu resim tabelasından yararlanıyoruz.

HTML'den Paremetre Aktarımı

Bir önceki örnekte yazdığımız thread uyku süresiThread.curentThread().sleep(200); program satırında 200 milisaniye olarak belirlendi. Bu sayıyı değiştirerek, hareketin hızlanmasını veya yavaşlamasını sağlayabiliriz. Ne varki, programı her değiştirişimizde, yeniden derlememiz gerekiyor ve bu işlemde vakit alıyor. Eğer Html dosyasından, Java programına parametre olarak aktaracak olursak, Java programını sürekli olarak JavaC ile derlememize gerek kalmayacak .

Aşağıdaki thread2.html dosyasını hazırlayın:

```
<html>
<body>
<title>thread2.html</title>
<applet code=thread2.class width=250 height=150>
<param name=uykuSuresi Value="100">
</applet>
<body>
</html>
```

Yukarıda gördüğünüz gibi <param name=uykuSuresi Value="100"> satırı ile uykuSuresi adlı değişkenin değerini 100 olan bir parametre ile tanımladık. Şimdi bu parametre ve değerini, Java programına aktarmak istiyoruz. Java programının başına iki yeni değişken ekleyecek ve bu değişkenlere;

int miliSaniye;

String HTMLdenGelen;

adlarını vereceğiz. Program içindeki public void init( ) bölümünün başına ise:

HTMLdenGelen=getParameter("uykuSuresi");

miliSaniye=Integer.valueOf(HTMLdenGelen).intValue( );

satırlarını ekleyeceğiz. Son yapacağımız iş ise public void run( ) metodunun thread uyku satırını aşağıdaki biçimde değiştirmek olacak:

Thread.curentThread( ).sleep(milisaniye);

Bu yaptığımız değişiklikleri devreye sokabilmek için bir kez Java programını derliyelim. Bundan böyle yalnızca HTML dosyasının içindeki uykuSuresi parametresinin değerini değiştirmekle Java programımızın değişik süratlerde çalışmasını elde edebiliriz.

İşte son thread2.java programının kaynak kodu:

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class thread2 extends Applet implements Runnable {
int miliSaniye;
String HTMLdenGelen;
Thread thread1=null;
String resimNo;
Image[] resimler=new Image[20];
```

```
int ii;
Image aktuelResim;
//---------------------------------------------------------------
public void init() {
//HTML'den paremetre aktar
HTMLdenGelen=getParameter("uykuSuresi");
miliSaniye=Integer.valueOf(HTMLdenGelen).intValue();
//Resim dosyalarını okuyup resimler adlı dizi içine koy
ii=0;
while (ii<=11) {
ii=ii+1;
resimNo= "t"+String.valueOf(ii)+".gif";
aktuelResim=getImage(getCodeBase(),resimNo);
resimler[ii]=aktuelResim;
}
ii=1;
}
//---------------------------------------------------------------
//Resmi ekrana getiren program mantığı
public void paint(Graphics graf) {
Font kalinFont=new Font("TimesRoman", Font.BOLD,16);
graf.setFont(kalinFont);
resimNo= "t"+String.valueOf(ii)+".gif";
aktuelResim=resimler[ii];
graf.drawString("Resim dosya ismi :" + resimNo,10,50);
graf.drawImage(resimler[ii],30,70,null);
ii++;
if (ii==12) {
ii=1;
}}
//---------------------------------------------------------------
public void start( ) {
```

```
thread1=new Thread(this);
thread1.start();
}
//-------------------------------------------------------------
public void stop( ) {
thread1.stop( );
}
//-------------------------------------------------------------
public void run() {
while(true) {
try {
Thread.currentThread().sleep(miliSaniye);
}
catch(InterruptedException ie) {
}
repaint();
}}}
```

Kullanıcı ile Diyalog

Butonlar

Bir Çok Java programında kullanıcı ile bir diyalog kurulması, kullanıcının belirli opsiyonlardan birini seçmesi gerekebilir. Program akışıda kullanıcının aksiyonuna göre değişebilir

Hazırlayacağımız bir örnek ile kullanıcının, Java programı üzerindeki bir butonu sonucunda, Java programının değişik işlemler yapmasını gerçekleştireceğiz.

Önce diyalog.html adlı dosyayı hazırlayalım:

```
<html>
<body>
<title>animasyon.html</title>
<applet code=diyalog.class width=450 height=300>
</applet>
<body>
</html>
```

Şimdi de aşağıda kaynak kodunu gördüğünüz diyalog.java programını hazırlayalım:

```
import java.awt.*;
import java.applet.*;
public class diyalog extends Applet {
String mesaj;
int secim;
//-----------------------------------------------------------------------------------------------------
public void init() {
Panel buton=new Panel();
buton.add(new Button("ATACAN"));
buton.add(new Button("Besiktas"));
buton.add(new Button("Galatasaray"));
buton.add(new Button("Trabzonspor"));
buton.add(new Button("Fenerbahce"));
add("KULLANILMIYOR",buton);
mesaj="";
secim=0;
}
//-----------------------------------------------------------------------------------------------------
public void paint(Graphics graf) {
int i,j;
//ATACAN Renklerini Hazırla
i=5;
```

```
j=50;
graf.setColor(Color.green);
graf.fillRect(i,j,25,25);
i=i+25;
graf.setColor(Color.yellow);
graf.fillRect(i,j,25,25);
graf.setColor(Color.black);
graf.drawRect(i-26,j-1,50,25);
//Beşiktaş renklerini hazırla
i=5;
j=j+35;
graf.setColor(Color.black);
graf.fillRect(i,j,25,25);
i=i+25;
graf.setColor(Color.white);
graf.fillRect(i,j,25,25);
graf.setColor(Color.black);
graf.drawRect(i-26,j-1,50,25);
//Galatasaray Renklerini Hazırla
i=5;
j=j+35;
graf.setColor(Color.red);
graf.fillRect(i,j,25,25);
i=i+25;
graf.setColor(Color.yellow);
graf.fillRect(i,j,25,25);
graf.setColor(Color.black);
graf.drawRect(i-26,j-1,50,25);
//Trabzonspor Renklerini Hazırla
i=5;
j=j+35;
graf.setColor(Color.magenta);
```

```
graf.fillRect(i,j,25,25);
i=i+25;
graf.setColor(Color.cyan);
graf.fillRect(i,j,25,25);
graf.setColor(Color.black);
graf.drawRect(i-26,j-1,50,25);
//Fenerbahçe Renklerini Hazırla
i=5;
j=j+35;
graf.setColor(Color.blue);
graf.fillRect(i,j,25,25);
i=i+25;
graf.setColor(Color.yellow);
graf.fillRect(i,j,25,25);
graf.setColor(Color.black);
graf.drawRect(i-26,j-1,50,25);
//Ekrana bilgi yerleştir.....
Font kf=new Font("Arial",Font.BOLD,16);
Font kf2=new Font("Arial",Font.BOLD,24);
graf.setFont(kf);
graf.drawString("Yukaridaki butonlardan birini klikleyiniz",70,70);
graf.setFont(kf2);
graf.drawString(mesaj,150,150);
//kullanıcının seçimine göre büyük bir bayrak çiz...
switch(secim) {
case 1: {
graf.setColor(Color.green);
graf.fillRect(150,170,100,100);
graf.setColor(Color.yellow);
graf.fillRect(250,170,100,100);
graf.setColor(Color.black);
graf.drawRect(149,169,200,100);
```

```
break;
}
case 2: {
graf.setColor(Color.black);
graf.fillRect(150,170,100,100);
graf.setColor(Color.white);
graf.fillRect(250,170,100,100);
graf.setColor(Color.black);
graf.drawRect(149,169,200,100);
break;
}
case 3: {
graf.setColor(Color.red);
graf.fillRect(150,170,100,100);
graf.setColor(Color.yellow);
graf.fillRect(250,170,100,100);
graf.setColor(Color.black);
graf.drawRect(149,169,200,100);
break;
}
case 4: {
graf.setColor(Color.magenta);
graf.fillRect(150,170,100,100);
graf.setColor(Color.cyan);
graf.fillRect(250,170,100,100);
graf.setColor(Color.black);
graf.drawRect(149,169,200,100);
break;
}
case 5: {
graf.setColor(Color.blue);
graf.fillRect(150,170,100,100);
```

```
graf.setColor(Color.yellow);
graf.fillRect(250,170,100,100);
graf.setColor(Color.black);
graf.drawRect(149,169,200,100);
break;
}}}
//-----------------------------------------------------------------------------------------------
public boolean action(Event olay, Object sonuc) {
if ("ATACAN".equals(sonuc)) {
mesaj="En Büyük Eyüp ATACAN";
secim=1;
}
if ("Besiktas".equals(sonuc)) {
mesaj="En Büyük Besiktas!";
secim=2;
}
if ("Galatasaray".equals(sonuc)) {
mesaj="En Büyük Galatasaray!";
secim=3;
}
if ("Trabzonspor".equals(sonuc)) {
mesaj="En Büyük Trabzonspor!";
secim=4;
}
if ("Fenerbahce".equals(sonuc)) {
mesaj="En Büyük Fenerbahce!";
secim=5;
}
repaint();
return true;
}
}
```

Örneğimizde sistem otamatik olarak 5 butonu applet penceresi içine yerleştiyor. Eğer applet penceresinin konumu değiştirilecek olursa butonlarda otamatik olarak konumlandırılırlar.

Bu örnekte kullandığımız yeniliklerin bir bölümü public void init( ), diğer bölümü ise public boolean action(Event olay, Object sonuc) metotları içinde bulunuyor.

Bu iki metottan public void init( ) ile applet üzerine yerleştirmek istenilen butonlar tanımlanıyor. Önce bu metodun incelenmesine geçelim:

```
public void init( ) {
Panel buton=new Panel( );
buton.add(new Button("ATACAN"));
buton.add(new Button("Besiktas"));
buton.add(new Button("Galatasaray"));
buton.add(new Button("Trabzonspor"));
buton.add(new Button("Fenerbahce"));
add("KULLANILMIYOR",buton);
mesaj="";
secim=0;
}
```

Panel, applet içinde kullanılan butonların yerleştirileceği alan anlamına geliyor ve bu alanda kullandığımız bir değişken olan buton'u Panel buton=new Panel( ); program satırı ile tanımlıyoruz.

Bu işlemden sonra her buton.add(new Button(".....")); satırı ile yeni bir buton ve noktalı alana yazdığımız sözcük ile de bu butonun başlığını belirliyoruz. Yukarıdaki program satırında add sözcüğü ekle, new Button sözcükleri ise yeni buton anlamlarını taşıyor. Sistem, tanımladığımız butonları soldan sağa doğru applet penceresi içine yerleştiriyor. Son buton tanımı olan buton.add(new Button("Fenerbahce")); program satırından sonra add("KULLANILMIYOR", buton); komut satırı ile buton sayısını

Java'ya bildiriyoruz. Bu public void init() metodu ile applet'in başlangıç görüntüsü belirleniyor.

Kullanıcı applet üzerindeki butonlardan herhangi bir tanesini kliklediği anda public boolean action(Event olay, Object sonuc) metodu devreye giriyor. Eğer Besiktas başlıklı buton klikleniş ise bu metod içindeki if ("Besiktas".equals(sonuc)) {mesaj="En Büyük Besiktas!"; secim=2;} satırları icra ediliyor. Bu satırlar ise ekrana yansıyacak mesajı "En Büyük Besiktas" olarak belirliyor ve secim değişkeninin değeri 2 olarak sabitleniyor. Aynı metod içindeki repaint(); komutu gerek "En Büyük Besiktas" ekrana belirlenen puntolarda yansıtıyor, gerekse secim değişkeninin değerine göre Besiktas'ın renklerini içeren bir bayrağı bu mesajın altına bir dikdörtgen olarak çiziyor.

Kullanıcı Kontrollü Animasyon

Bir önceki bölümde yazdığımız animasyon örneğinde HTML dosyasından paremetre aktararak, hareket hızını değiştirmeyi incelemiştik. Bu örneğimizde ise HTML dosyasından gelen parametre yerine, JAVA programı içinden hareket hızını nasıl kontrol edebileceğimizi öğreneceğiz.

HTML dosyasına diyalog2.html adını kullandım:

```
<html>
<body>
<title>diyalog2.html</title>
<applet code=diyalog2.class width=250 height=250>
</applet>
<body>
</html>
```

diyalog2.java'nın kaynak kodu:

```
import java.awt.*;
```

```
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.*;
public class diyalog2 extends Applet implements Runnable {
int miliSaniye;
Thread thread1=null;
String resimNo;
Image[] resimler=new Image[20];
int ii;
Image aktuelResim;
String mesaj;
//---------------------------------------------------------------
public void init( ) {
//Butonların tanımı
Panel buton=new Panel();
buton.add(new Button("50"));
buton.add(new Button("100"));
buton.add(new Button("200"));
buton.add(new Button("300"));
buton.add(new Button("400"));
add("KULLANILMIYOR",buton);
//-----------------------------------------------------------------------------------------------
//Resim dosyalarını okuyup resimler adlı dizi içine koy
ii=0;
while (ii<=11) {
ii=ii+1;
resimNo= "t"+String.valueOf(ii)+".gif";
aktuelResim=getImage(getCodeBase(),resimNo);
resimler[ii]=aktuelResim;
}
ii=1;
```

```
miliSaniye=200;
}
//-----------------------------------------------------------------------------------------------------
public boolean action(Event olay, Object sonuc) {
if ("50".equals(sonuc)) {
mesaj="Uyku Süresi 50 Milisaniye";
miliSaniye=50;
}
if ("100".equals(sonuc)) {
mesaj="Uyku Süresi 100 Milisaniye";
miliSaniye=100;
}
if ("200".equals(sonuc)) {
mesaj="Uyku Süresi 200 Milisaniye";
miliSaniye=200;
}
if ("300".equals(sonuc)) {
mesaj="Uyku Süresi 300 Milisaniye";
miliSaniye=300;
}
if ("400".equals(sonuc)) {
mesaj="Uyku Süresi 400 Milisaniye";
miliSaniye=400;
}
repaint();
return true;
}
//-----------------------------------------------------------------------------------------------------
public void paint(Graphics graf) {
Font kalinFont=new Font("TimesRoman",Font.BOLD,16);
graf.setFont(kalinFont);
resimNo="t"+String.valueOf(ii)+".gif";
```

```
aktuelResim=resimler[ii];
graf.drawString("Uyku süresi:"+String.valueOf(miliSaniye),10,70);
graf.drawString("Resim dosya ismi:"+resimNo,10,90);
graf.drawImage(resimler[ii],30,110,null);
ii++;
if (ii==12) {
ii=1;
}}
//-----------------------------------------------------------------------------------------------------
public void start( ) {
thread1=new Thread(this);
thread1.start( );
//-------------------------------------------------------------
public void stop( ) {
thread1.stop( );
}
//-------------------------------------------------------------
public void run() {
while(true) {
try {
Thread.currentThread().sleep(miliSaniye);
}
catch(InterruptedException ie) {
}
repaint();
}}}
```

Applet içine 5 değişik değer taşıyan buton yerleştirdik. Başlangıç değeri olan thread'ın 'uyku süresini' 200 milisaniye olarak belirledik. Hangi butonu kliklersek, yazdığımız programa göre, o buton üzerindeki değer thrread'in uyku süresi olarak devreye girerek animasyonun hareket süratini değiştiriyor. Uyku süresi değeri ne kadar ufak olursa, animasyonun hareketi o kadar hızlanıyor.

Bu programla, Kullanıcının çeşitli butonları kullanarak Java Programının akışını nasıl değiştirebileceğine örnek verdik. Bu son programda yeni bir teknik kullanılmamıştır. Eğer bu programı rahatlıkla anlayabiliyorsanız anlatılan bütün konuları kavramışsınız demektir.

Chat

Chat ne demek? Bunu internetle birazcık alakalı olan herkes biliyordur herhalde! Chat, sohbet etmektir. Bu sohbet iki veya daha fazla kişi arasında olabilir. Internet ile dünyanın dört bir köşesindeki insanlarla sohbet edebilirsiniz. Bunun için elbette yazılıma ihtiyacınız var. Çeşitli yazılımlar mevcut : MIRC, PIRCH gibi. Özel chat yazılımları kullanmadan bu iş yapılamaz mı peki? Belki sizde rastlamışsınızdır, web sayfaları üzerinden chat bile yapmışsınızdır belkide? İşte bizde bu sayfalarda java ile böyle bir chat programı hazırlayacağız. Hazırsanız başlayalım.

Güvenlik

Applet'lerin, yani HTML sayfalarına bağlanmış java programlarının güvenlik nedeniyle bazı kısıtlamaları vardır. Örneğin: Applet'ler sadece yüklendikleri server ile bağlantı kurabilirler. Bir applet "www.javasayfasi.net"'den yüklenmiş ise, sadece "www.javasayfasi.net" ile bağlantı kurabilir. Aksi taktirde (başka bir server ile bağlantı kurmaya çalışırsa) browser'ın status (durum) çubuğunda "Applet can't start: Security violation: security.socket.connect" hatası belirir. Aynı hata bağlantı kurulmak istenen porttan cevap alınılamadığı durumlarda da ortaya çıkar. Örneğin chat-server çalışmıyorsa, chat server'ın dinlediği porttan cevap alınamaz.

URL'ler

Başka bir bilgisayar ile bağlantı kurmak için, o bilgisayarın internet adresinden faydalanıyoruz. Örneğin "www.javasayfasi.net" gibi. Bağlantıyı java ile kurmak için sınıf oluşturuyoruz. Aşağıdaki örnek java kodu "www.javasayfasi.net" adlı bilgisayardaki index.html dosyasına ulaşmak için kullanılıyor.

```
Try {
baglantiurl= new URL("http://www.javasayfasi.net/index.html");
}
 catch (MalformedURLExceoption e) {
getAppletContext().showStatus("Ulaşılamayan URL:" + baglantiurl);
}
```

Yukarıda görüldüğü gibi try { } catch { } bloğu ile bağlantı kurmayı deniyoruz ve hataları yakalayabiliyoruz. Hata oluşması durumunda status çubugunda "Ulaşılamayan URL: www.javasayfasi.net " şeklinde bir mesaj çıkmasını sağlıyoruz. Hata oluşmaması durumunda adresini verdiğimiz server'a, hatta bu durumda o server'daki bir dosyaya ulaşmış oluruz. Bundan sonraki adım ise verilerin alınması, bunu C++'dan tanıdığımız stream'lerle yapacağız.

Stream

Stream, basit olarak iki nokta arasındaki bağlantıdır. Bir taraf verileri gönderirken diğer taraf verileri alır. Stream'de güzel olan iki taraf içinde karşı tarafta ne olduğu önemli değildir. Örneğin verileri gönderen tarafta bir dosya, klavyeden girilmiş manuel bir bilgi veya başka bilgisayar ile kurulmuş bir veri bağlantısı olurken, alıcı taraf bir program bir dosya vb. olabilir.

Yukarıda kurulan bağlantıdan veri alabilmemiz için stream kullanmamız gerekiyor. URL sınıfının openStream( ) adlı metodu tam bu iş için. Aşağıdaki kod, yukarıdaki URL bağlantısından satır satır bilgi okuyup bunu gösteriyor.

```
Try {
InputStream is=baglantiurl.openStream();
DataInputStream dis=new DataInputStream(is);
}
catch (IOException e) {}
while(in.blabla()) {
```

```
try {
satir=in.readLine();
System.out.println(satir);
}
catch (IOException e) { }
}
```

Socketler

Streamler ile olan çözüm gerçi iletişim kurmak için bir yöntemdir ki veriler bir noktadan diğerine aktarılıyor. Sohbetlerde İletişimin çift tarfalı olması gerektiği bir gerçek, aksi taktirde bir taraf hep sukut halinde değil midir? Gerçi streamler ile server'a veri gönderilebilir. Server'da çalışan bir cgi programıda bu verileri alıp değerlendirir. Ama Internetin temelini oluşturan TCP/IP protokolü çift taraflı iletişimler için daha iyi bir yöntem sunuyor :

Socketler TCP/IP protokolünün bir özelliği verilerin paketler halinde alınıp verilmesini sağlamasıdır. Verilerin doğru adrese ulaşması için de her pakete bir IP adresi verilir. Internet teki tüm server'ların bir IP adresi vardır. Bir network ortamındaki bir çok kullanıcı farklı serverlar ile bağlantı kurarken, bir kullanıcıda bir çok server ile bağlantı kurabilir. Kurulan bağlantıların çakışmaması için her veri paketine IP adresinin yanında bir numara daha verilir. Bu numara port numarasıdır. IP adresi ve port numarası ile veri paketi doğrtu yerine ulaşır. Bu IP adresi ve port numarası kombinasyonuna Socket denir.

Bazı portlar standart hale getirilip, rezerve edilmiştir. Örneğin Telnet için port 23 , WWW için port 80 standarttır. 256 'nın altındaki tüm portlar standart ağ hizmetleri için rezerve edilmiştir. 256 ile 1024 arasındaki portlar UNIX hizmetlerine tahsis edilmiştir. Örneğin: rlogin
1024 ile 16384 arasındaki portlar serbesttir ve kullanılabilir. Bizim yazacağımız chat sisteminin standart bir portu yok. Kendimize 1024 iel 16384 arasında bir sayı seçebiliriz. Ben 8888 numaralı portu seçtim.

Sunucu / İstemci (Client / Server)

Socket'ler yardımı ile chat sistemimizi programlamaya başlayacağız. Belli bir appleti yükleyen kullanıcıların birbirleri ile sohbet etmesini sağlayacak bu sistemimiz.

Daha öncede bahsettiğim gibi güvenlik nedeniyle appletler birbirleri ile direkt iletişim kuramazlar. Bu nedenden dolayı chat sistemimizi Sunucu/İstemci mimarisine göre geliştireceğiz. Server üzerinde bir uygulama çalışacak devamlı bağlantı bekleyen ve bağlantı isteklerine cevap veren Applet'in bağlandığı HTML sayfasıda bu server üzerinde olmalıdır. Tabii ki bu server üzerinde Web server da kurulu olması gerekir. HTML sayfasi diğer bilgisayarlar (kullanıcılar) tarafından yüklediğinde, bu sayfanın içinde olan applet çalışacak ve server ile bağlantı kuracaktır. Girilen tüm veriler server'a gönderilecek, server ise gelen bilgileri, kendisine bağlı olan tüm kullanıcılara geri gönderecektir.

Chat Sistemi

Chat sistemimiz toplam 3 sınıftan oluşacak. Birincisi Chat Server, server üzerinde çalışacak ve bağlantı taleplerine cevap verecektir. İkincisi Baglantı Sınıfı, Bağlantı taleplerini karşılayacak ve kullanıcılar arasındaki veri iletişimini düzenleyecek. Üçüncüsü Chat Applet, kullanıcının bilgisayarında çalışacak.

Chat Server Sınıfı

Öncelikle kullanacağımız standart sınıf paketlerini yazalım:

```
import java.net.*;
import java.io.*;
import java.util.*;
```

java.net içerisinde Socket veya URL gibi internet iletişiminde gerekli olan sınıflar bulunuyor. java.io içerinde Giriş/Çıkış sınıfları, java.util içerisinde ise bir sürü gerekli sınıf bulunuyor.

```
public class chatserver implements Runnable {
...
}
```

Chat Server sınıfımız kendi başına çalışan bir thread şeklinde olacaktır. Bunun için implements Runnable kullanıyoruz.

```
public static final int PORT = 8888;
protected ServerSocket dinle;
protected Vector baglantilar;
Thread baglan;
```

Sınıf tanımından sonra değişken tanımlarını yaptık. PORT sabit bir değişken ve daha önceden karar verdiğimiz port numarası atanmış. "dinle" bağlantı taleplerini dinlemek için. "baglantilar" aktif bağlantıların tutulacağı liste için. "baglan" chat server sınıfının thread'ini tutuyor.

```
public static void main(String[] args){
new chatserver();
}
```

main metodumuzda chat server sınıfımızı oluşturuyoruz.

```
public chatserver( ) {
try {
dinle = new ServerSocket(PORT);
}
catch (IOException e) {
```

```
System.err.println("Socket oluşturulurken hata :"+e);
System.exit(1);
}
baglantilar = new Vector();
baglan = new Thread(this);
baglan.start();
}
```

Chat server sınıfımızın başlangıçında (constructor) verilen portta bir socket açılıyor ve olası hatalar yakalanıyor. Sonra bağlantıların tutulacağı liste ve thread oluşturulup çalıştırılıyor.

```
public void run() {
try {
while(true) {
socket istemci=dinle.accept();
baglanti b = new baglanti(this, istemci);
baglantilar.addElement(b);}
} catch (IOException e) {
System.err.println("Bağlantı talebi beklerken hata :"+e);
System.exit(1);
}}
```

run() metodu yani asıl çalışan kısım sonsuz döngü içinde bağlantı taleplerini bekliyor. Bağlantı talebi geldiğinde bu talep için baglanti adında yeni bir connection nesnesi oluşturuyor. Niye her bir talep için yeni connection nesnesi oluşturuyor diye bir soru geldi ise aklınıza, Server bundan sonraki bağlantı taleplerine cevap verebilsin diye.

```
public void mesajgonder(String msg) {
int i;
baglanti sen;
for (i=0; i<baglantilar.size(); i++) {
```

```
sen = (baglanti) baglantilar.elementAt(i);
sen.out.println(msg);
sen.out.flush();
}}
```

mesajgonder metodu tüm aktif bağlantılara msg parametresi ile gelen mesajı gönderir. Aktif bağlantılar chat server sınıfında tutulduğu için bu metod chat server sınıfında kullanılmadığı halde bu sınıfın içinde bulunması gerekiyor.

Bağlantı Sınıfı

Bağlantı sınıfı chat server sınıfı tarafından oluşturulur ve aktif bağlantılar arasındaki iletişimi kontrol eder.

```
import java.net.*;
import java.io.*;
class baglanti extends Thread {
...
}
```

Her bağlantı talebi için bağlantı nesnesi oluşturulur. Bu nesne bağlantı talebini alır ve bağlantı sonuna kadar çalıştırır. Bundan dolayı her bağlantı kendi başına bir thread olmalıdır.

```
protected Socket istemci;
protected BufferedReader in;
protected PrintWriter out;
protected chatserver server;
```

istemci Chat Server'dan Socket'i alır. in ve out adlı Giriş/Çıkış streamleri bu nesne ile bağlantılı olacaklardır. server Chat Server objesini gösterir.

```
public baglanti(chatserver server, Socket istemci) {
this.server=server;
this.istemci=istemci;
try {
in =new BufferedReader(new InputStreamReader(istemci.getInputStream()));
out = new PrintWriter(istemci.getOutputStream());
}
catch (IOException e) {
try {
istemci.close();
}
catch (IOException e2) {
System.err.println("Stream oluşturulurken hata : " + e);
return;
}
this.start();
}
```

Bağlantı sınıfının başlangıcında (constructor) parametre olarak alınan server ve istemci değişkene saklanıyor. Sonra bir giriş ve bir çıkış stream i oluşturulmak isteniyor. Bu sırada oluşacak hatalar catch( ) tarafından yakalanıyor.

```
public void run() {
String satir;
try {
while(true) {
satir=in.readLine();
if(satir!=null)
server.mesajgonder(satir);
}}
catch (IOException e){
System.out.println("Hata:" + e);
```

}}

Bağlantı sınıfının asıl çalışan kısmı gelen mesajları aktif bağlantılara gönderen sonsuz bir döngüden oluşuyor. Mesaj göndermek için chat server sınıfını mesajgonder metodu kullanılıyor.

Chat Applet

Toplam üç sınıfımızdan biri olan chat appleti, tek grafik arayüzüne sahip sınıftır. Bu yüzden java.applet paketinin yanında java.awt sınıf paketininide kullanması gerekiyor.

```
import java.net.*;
import java.io.*;
import java.awt.*;
import java.applet.*;
import java.awt.event*;
public class chatapplet extends Applet implements Runnable,ActionListener {
...
}
```

Chat Applet sınıfı Applet sınıfını miras alır ve Thread kullanır. Mesaj yazıldıktan sonra Enter'a basılıp basılmadığını kontrol etmek için ActionListener sınıfını kullanıyoruz

```
public static final int PORT = 8888;
Socket socket;
BufferedReader in;
PrintWriter out;
TextField inputfield;
TextArea outputarea;
Thread thread;
```

PORT, socket, in und out chatserver ve bağlantı da açıklanan ile aynı anlamı taşır. inputfield giriş nesnesini outputarea ise çıkış nesnesini tutuyor. thread kendine ait Thread'i saklıyor.

```
public void init() {
inputfield = new TextField();
outputarea = new TextArea();
outputarea.setFont( new Font("Dialog", Font.PLAIN, 12));
outputarea.setEditable(false);
this.setLayout(new BorderLayout());
this.add("South", inputfield);
this.add("Center", outputarea);
this.setBackground(Color.lightGray);
this.setForeground(Color.black);
inputfield.setBackground(Color.white);
outputarea.setBackground(Color.white);
inputfield.addActionListener(this);
}
```

Init metodunda grafik arayüz hazırlanıyor. Giriş sahası (Textfield) ve çıkış sahası (Textarea) oluşturulup, giriş sahası aşağıya gelecek şekilde yerleştiriliyor. Bunun akabinde renkler belirleniyor. En son satırda ise mesaj girilecek yazı alanında meydana gelen Enter'a basma olayını anlamak için ActionListener ekliyoruz.

```
public void start() {
try {
socket = new Socket(this.getCodeBase().getHost(), PORT);
in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
out = new PrintWriter(socket.getOutputStream());
} catch (IOException e) {
this.showStatus(e.toString());
```

```
say("Server ile bağlantı kurulamadı!");
System.exit(1);
}
say("Server ile bağlantı kuruldu...");
if (thread == null) {
thread = new Thread(this);
thread.setPriority(Thread.MIN_PRIORITY);
thread.start();
}}
```

Start metodunda applet server ile bağlantı kurulmaya çalışılıyor. Gerekli olan server adresi HTML dökümanı tarafından belirleniyor. Bu da daima HTML dökümanın bulunduğu serverın adresidir. Başka serverlar ile bağlantıya izin verilmediği için, başka server olmasınında hiç bir anlamı yok. Hata oluşursa veya bağlantı kurulursa, sonuç bir mesaj ile kullanıcıya bildiriliyor. Bundan sonra kendi thread'ini oluşturup başlatıyor.

```
public void stop() {
try {
socket.close();
}
catch (IOException e) {
this.showStatus(e.toString());
}
if ((thread !=null) && thread.isAlive()) {
thread.stop();
thread = null;
}}
```

Stop Metodu browser tarafıdan, kullanıcı sayfayı terk ettiği zaman otomatik olarak çağrılır. Socket kapatılıyor ve thread durduruluyor.

```
public void run() {
String satir;
Try {
while(true) {
satir = in.readLine();
if(satir!=null)
outputarea.append(satir+'\n' );
}}
catch (IOException e) {
say("Server ile bağlantı kesildi.");
}}
```

run metodunda sonsuz bir dögü içinde giriş sahasından girilen mesajlar, çıkış sahasında gösteriliyor.

```
public void actionPerformed(ActionEvent e){
if (e.getSource()==inputfield) {
String inp=inputfield.getText();
out.println(inp);
out.flush();
inputfield.setText("");
}}
```

actionPerformed, kullanıcı giriş yaptığında aktif olur. Kullanıcının giriş sahasına girip girmediğini kontrol ediyoruz. Bu taktirde girileni alıp servera gönderiyoruz. Giriş sahasını tekrar boşaltıyoruz.

```
public void say(String msg) {
outputarea.append("*** "+msg+" ***\n");
}
```

Say Metodu girilen mesajı başına ve sonuna yıldız ekleyerek gösteriyor.

Derleme & Çalıştırma

Şimdi chatserver ve chatappleti derleyelim.

```
javac chatserver.java
javac chatapplet.java
```

Bu işlem sonucunda üç class oluşacak, chatserver otomatik olarak baglanti class oluşturur.Bizim chat sistemimiz JDK 1.1 metodlarını kullanıyor. Bu yüzden JDK 1.1 desteklemeyen Netscape 3 hatta 2 gibi browserlarda appletimizin çalışmayacaktır.

Şimdi chat server'a başlatalım. Yoksa kullanıcılar bağlanamaz değil mi?

```
java chatserver
```

HTML Sayfası

Applet'imizi bağlayacağımız ufak bir HTML sayfası hazırlayalım. Siz bu sayfayı süsleye bilirsiniz.

```
<html>
<head>
<title>Sohbet</title>
</head>
<body bgcolor=000000 text=ffffff>
<center>
<applet code="chatapplet.class" width=95% height=60%>
</applet>
</center>
</body></html>
```

Bu HTML sayfasını appletin bulunduğu dizine kopyalayın ve en az iki browser penceresi açın. Her iki pencereye de yukarıdaki HTML sayfasını yükleyin.

İyi sohbetler.

Eğer sadece kendi kendinize sohbet etmek istemiyorsanız, kendinize bir web server kurmalı ve insanların bunu dışardan görebilmelerini sağlamalısınız. Dial-up bağlanıyor olsanız bile online olduğunuz zamanlar arkadaşlarınızı chate davet edebilirsiniz.

Her zaman ulaşılmasını istiyorsanız, web hosting yapan bir kuruluşla anlaşmanız gerekir. Ama çoğu kuruluş sizin chat server'ınızı çalıştırmak istemeyecektir.

Kim bilir belki siz chat server'ınızı çalıştırmalarını belki sağlayabilirsiniz.

Sadece Başlangıç

Bu sisteme bir sürü fonksiyon eklenbilir. IRC programlarını görenler, bir çok fonksiyon olduğunu bilirler. Örneğin her kullanıcıya bir isim atamak, kullanıcılar arasında özel mesajlaşma gibi...

Ama bu sadece işin başlangıcı, sizin bir chat sistemi nasıl yazılır, bunun temelini vermeye çalıştım.

Gerisi size kalmış.

ÖRNEK PROGRAM

Şimdiye anlatılanlardan ve verilen örneklerden yola çıkarak bir örnek program hazırlayalım. Örneğimiz birden fazla satırı olan bir metnin hizalanması ile ilgili.

Öncelikle yapmamız gereken ilk şey, uzun metni kendi sözcüklerimizle parçalara ayırmak olacaktır. Ardından etkin fonta göre sözcüğün uzunluğunu belirleyin. Sözcük çok uzun ise sonraki satır için dikey boşluğu geliştirin. Şimdi bu işlemleri gerçekleştirelim ve programımızı yazmaya başlayalım.

```
import.java.applet.*;
import.java.awt.*;
import.java.util.*;
import.java.lang.*;

public class MetinDüzeni extends Applet {
final int LEFT=0;
final int RIGHT=0;
final int CENTER=0;
final int LEFTRIGHT=0;
int align;
Dimension d;
Font f;
FontMetrics fm;
int fontSize;
int fy, tc;
int space;
String text;
public void init() {
setBackground(Color.yellow);
text=getParameter(“text”);
try {
```

```
fontSize=Integer.parseInt(getParameter(“fontSize”));
}
catch(NumberFormatException e) {
fontSize=14;
}
align=LEFT;
}
```

Önce burada hizalama stilini belirlemek amacıyla statik tamsayılar hazırladık. Font değişkenlerini, font yüksekliğini (fy) ve taban çizgisini (tc) belirledikten sonra, init yöntemini yerleştirdik. Burada init yöntemi metni alıyor ve try-catch bloğunda font boyutunu 14 punto olarak kullanıyor. Örneğimizi geliştirmeye devam edelim.

```
public void paint(Graphics g) {
update(g);
}
public void update(Graphics g) {
d = size();
g.setColor(getBackground());
g.fillRect(0,0,d.width, d.height);
if (f==null) f = new Font(getParameter("fontname"),
Font.PLAIN, fontSize);
g.setFont(f);
if (fm = null) {
fm = g.getFontMetrics();
tc = fm.getAscent();
fy = tc + fm.getDescent();
space = fm.stringWidth(" ");
}
```

Burada bir font metric objeden taban çizgisini, fontu ve font yüksekliğini yerleştirmiş olduk.

```
g.setColor(Color.black);
StringTokenizer st = new StringTokenizer(text);
intx=0;
int nextx;
inty=0;
String sözcük, sp,
int wordCount = 0;
String satır=” ”;
while (st.hasMoreTokens()) {
sözcük = st.nextToken();
if(sözcük.equals("<p>")) {
drawString(g, satır, wordCount,
fm.stringWidth(satır), y+tc);
line= “ ”;
wordCount=0;
x=0;
y=y+(fy*2);
}
else {
int w=fm.stringWidth(sözcük);

if(((nextx=(x+space+w))>d.width) {
drawString(g, satır, wordCount, fm.stringWidth(line), y+tc);
line= “ ”;
wordCount=0;
x=0;
y=y+(fy*2);
}
if(x!=0) {sp= “ ”;} else {sp= “ ”;}
line=line+sp+sözcük+w;
wprdCount++;
}}
```

Bir StringTokenizer objesi hazırlayarak sıradaki simgeyi (boşlukla ayrılan dizilim) yakaladık. Sıradaki simge <p> iken dikey boşluk yerleştirdik. Bir diğer taraftan etkin fontta iken simgenin uzunluğunun sütun genişliğini aşıp aşmadığını denetledik. Bir tam satırlık bir metnimiz olduğunda ya da simge sayısı çok olduğunda satırımızı drawString yöntemini kullanarak yerleştirdik. Burada wordCount sözcük sayımı, line satır, space boşluğu göstermektedir.

```
drawString(g, line, wordCount, fm.stringWidth(line), y+tc);
}
public void drawString(Graphics g, String line, int Wc, int lineW, int y) {
switch(align) {
case LEFT: g.drawString(line, 0, y);
break;
case RİGHT: g.drawString(line, d.width-lineW, y);
break;
case CENTER: g.drawString(line, (d.width-lineW)/2, y);
break;
case LEFTRIGHT:
if(lineW < (int) (d.width*.27)) {
g.drawString(line, 0, y);
}
else {
int toFill = (int) ((d.width - lineW)/wc);
int nudge = d.width - lineW - (toFill*wc);
int s = fm.stringWidth(" ");
StringTokenizer st = new StringTokenizer(line);
int x = 0;
while(st.hasMoreTokens()) {
String sözcük = st.nextToken();
g.drawString(sözcük, x, y);
if(nudge>0) {
x = x + fm.stringWidth(sözcük) + space + toFill + 1 ;
```

```
nudge--;
}
else {
x = x + fm.stringWidth(sözcük) + space + toFill;
}}}
break;
}}
public boolean mouseUp(java.awt.Event evt, int x, int y) {
align = (align + 1) % 4;
repaint();
return true;
}}
```

Dizilimi sağa, sola ya da merkeze hizalamak oldukça basit. Hem sola hem de sağa hizalamak biraz daha marifet isteyen bir iş. Bunun için dizilimin genişliği ile sütunun genişliği arasındaki farkı hesaplamalı (bu fark boşluk olarak tanımlanmaktadır) ve sonra bu boşluğu sözcüklerin arasına dağıtmak gerekir.

KAYNAKLAR

KESKİNKILIÇ, M., (1997). **Java ile Programlama**, Seçkin Yayınevi, Ankara.

TANYERİ, F.M., (1997). **Java – İnternet**, C.E.S. ENTERNASYONEL KİTABEVİ, İzmir.

AKIN, C., (1998). **Java Uygulama Geliştirme Kılavuzu**, Alfa Kitabevi, İstanbul.

JAVA, (1997). **Grup JAVA**, Beta Yayınları, İstanbul.

http://.www.javasayfasi.net
http://kitap.felab.itu.edu.tr
http://www.programlama.com
http://java.sun.com/

PC WORLD bilgisayar dergisi, Aralık 1999, İlk Adımlar, S.260.
PC WORLD bilgisayar dergisi, Ocak 2000, Java, S.204.