T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

JAVA PROGRAMLAMA DİLİNİN WEB ÜZERİNDEN SUNUMU

YÜKSEK LİSANS TEZİ

Ayhan KİRAZ

Enstitü Anabilim Dalı : Elektronik ve Bilgisayar Eğitimi

Tez Danışmanı : Yrd. Dç. Dr. Ali Fuat BOZ

Eylül 2006

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

JAVA PROGRAMLAMA DİLİNİN WEB ÜZERİNDEN SUNUMU

YÜKSEK LİSANS TEZİ

Ayhan KİRAZ

Enstitü Anabilim Dalı : Elektronik ve Bilgisayar Eğitimi

Bu tez 15/09/2006 tarihinde aşağıdaki jüri tarafından oybirliği ile kabul edilmiştir.

Yrd. Doç. Dr. Ali Fuat BOZ	Prof. Dr. Abdullah FERİKOĞLU	Yrd. Doç. Dr. Raşit KÖKER
Jüri Başkanı	Üye	Üye

TEŞEKKÜR

Tezin hazırlanış aşamasında bana her türlü desteği veren danışman hocam Sayın Yrd. Doç. Dr. Ali Fuat BOZ'a, uygulamanın yapımı aşamasında bana yardımlarını esirgemeyen iş arkadaşlarıma ve çalışmalarım esnasında sürekli yanımda olan sevgili eşime sonsuz teşekkürlerimi sunarım.

Eylül 2006

Ayhan KİRAZ

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

TCP/IP	: Geçiş Kontrol Protokolü Internet Protokolü(Transmission Control Protokol/Internet Protokol)
PDA	: Kişisel Dijital Yardımcı(Personal Digital Assistant)
CPU	: Merkezi İşlem Birimi(Central Processing unit)
HTML	: Text İşaretleme Dili(Hypertext Markup Language)
USB	: Universal Seri Yol(Universal Serial Bus)
ISO	: Uluslararası Standart Organizasyonu(International Standards Organization)
GUI	: Grafiksel Kullanıcı Arayüzü(Graphical User Interface)
I/O	: Giriş Çıkış(Input Output)
UDP	: Kullanıcı Datagram Protokolü(User Datagram Protocol)
AWT	: Genel Windows Araçları(Abstract Windows Toolkit)
UI	: Kullanıcı Ara yüzü(User interface)
FTP	: Dosya Transfer Protokolü(File Transfer Protocol)

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Kelimeler: Java, İnternet Destekli Eğitim, Web sitesi

Yaşadığımız bilişim çağında teknolojinin ilerlemesi zaman zaman takip edilemeyecek kadar hızlı olmaktadır. Özellikle bilgisayar ve iletişim alanındaki gelişmeler, diğer alanları da etkilemiş, dolayısı ile teknoloji yaşadığımız zamanı adeta kontrol eder hale gelmiştir. Bu alanlarda yaşanılan gelişmeler, yeni bir çağın başlangıcı olmuştur.

Bu gelişmelerin en çok etkilediği alan internet olmuştur. Dünya çapında birbirine bağlı milyonlarca bilgisayardan oluşan bu dev network ağı, bilgi alışverişi ve uluslar arası haberleşme için büyük kolaylıkları beraberinde getirmiştir. İnternetin hayatımızda çok fazla yayılması beraberinde internet üzerinden eğitimi ortaya çıkarmıştır. Bu sayede herkes istediği yerde, istediği bilgiye görsel ve multimedia(çoklu ortam) destekli bir şekilde ulaşabilmektedir. Bu çalışma içerisinde günümüzde yaygınlaşmaya başlayan Java programlama dilinin internet üzerinden anlatımı yapılmıştır. Bu çalışmanın amacı kişilerin Java dilini en kolay bir şekilde ve istediği her mekanda rahatlıkla öğrenebilmesidir. İnternet üzerinde Java ile ilgili Türkçe kaynaklar çok az bulunmaktadır. Olan bilgilerde ya çok ağır ya da gerektiği kadar açık değildir. Çalışmada Microsoft Frontpage, Macromedia Dreamveawer, Xara webstyle gibi programlar kullanılmıştır. Bu programları kullanarak kolay, hızlı ve anlaşılır bir şekilde Java ile ilgili bir web sitesi yapılmıştır. Bu web sitesi Java programlama dilini basit ve eğlenceli bir şekilde anlatmaktadır. Web sitesi dokuz bölümden oluşmaktadır. Her bölüm kendi arasında bağımsızdır. Bu şekilde istenilen konular kolaylıkla seçilebilmektedir. Her bölümde konular sol taraftan seçilmekte ve ilgili konu sağ tarafta açılmaktadır. Açılan konu tek sayfadan fazlaysa sayfa üzerindeki ileri geri butonlarıyla diğer sayfalara geçiş sağlanmaktadır. Bu şekilde bilgiye hızlı bir şekilde ulaşılabilmektedir.

Tezin ilk bölümünde giriş, ikinci bölümünde Java'nın genel yapısı, üçüncü bölümünde kontrol ifadeleri, dördüncü bölümünde diziler, beşinci bölümünde class yapısı, altıncı bölümünde applet'ler, yedinci bölümünde yerleştirme biçimleri, sekizinci bölümünde program hataları, dokuzuncu bölümünde Java paketleri, onuncu bölümünde awt araçları, onbirinci bölümünde sonuç ve son bölümünde de kullanılan programlar anlatılmıştır.

JAVA PROGRAMMING LANGUAGE'S PRESENTATION OVER WEB

SUMMARY

Key words: Java, Internet Supported Education, Web Site,

Progress of technology in the information age is extremely fast. Especially developments about computer and communication effected other areas. So technology controls our age. Developments in this areas has become beginning of a new age.

Internet effected these developments. This large network network that originated millions computers all around the world, simplfied information relations and international communication, because many people use internet, anda re exposed internet education. Thus everybody can reach information that he/she wants. In this application, Java programming language that begins to develop was explained up to internet. Aim of this application to teach Java programming language easily in all places. There is no enough Turkish source about Java. They are very difficult. In this application, Microsoft Frontpage, Macromedia Dreamweaver and Xara Webstyle programs were used. This website explain Java programming language simply and amusingly. Website is composed of 9 sections. Every section is independent. Thus subjects can be selected easily. In all sections, subjects are chosen from left side and opened right side. If subject are more than one page, you must press next button.

The first section contains introduction, second section Java's general structure, third section control statements, fourth section series, fifth section class structure, sixth section applets, seventh section layout managers, eighth section program errors, ninth section Java packages, tenth section awt, eleventh section conclusion and the last section contains programs that were used.

BÖLÜM 1. GİRİŞ

1.1. İnternet Nedir?

İnternet sözcüğü international Network sözcüğünden oluşmuş uluslararası ağ anlamına gelen bir terimdir. İnternet milyonlarca alt ağdan oluşan ve ağlar içerisinde aktif olarak bulunan insanların tümünü yazılı, görsel ve işitsel olarak bir araya getirip etkileşim halinde bulunmasını sağlayan bir ağ protokolüdür.

İnternet birçok bilgisayar sistemini TCP/IP(Transmission Control Protokol/Internet Protokol) protokolü ile birbirine bağlayan dünya çapında yaygın olan ve sürekli büyüyen bir iletişim ağıdır. İnternet, tüm dünyayı kapsayan, çoğu ülkeye dağılmış ve iki milyondan fazla bilgisayarı birbirine bağlayan bilgisayar ağının toplamıdır. Kısaca internet, birbiriyle tüm dünya üzerinde yayılmış bilgisayar ağlarının birleşiminden oluşan bir bilgisayar ağıdır. [1]

1.2. İnternet ve Eğitim

Etkileşim özelliği sayesinde internet, öğrenmeye katkı getirmekte ve doğrudan derslerde kullanılacak hemen her konu alanına yönelik kaynak ve materyaller sağlanmaktadır. Böylelikle öğrenme ortamlarının görünümü ve değerlendirme yöntemleri kökten değişmektedir.

İnternet kara tahta, tebeşir ve sıralara alternatif bir yol olarak düşünülebilir. Bu bağlamda, ABD'deki tüm okul ve kütüphanelerin internete bağlanma işlemleri büyük bir hızla tamamlanmaktadır.

Bir bilgi kaynağı olarak internetin kullanılması, geleneksel olarak tek bilgi kaynağı görülen öğretmenin rolünün bu konuda günümüze kadar geçerli olan geleneksel

fikrin değişmesine yol açmıştır. Böylece, öğrenen merkezli öğrenme ortamları ağırlık kazanmıştır.

İnterneti kullanma, öğrencilerin aktif katılımcılar haline gelmesini sağlamakta, kendi geleceklerini planlamakta ve öğrendikleri disiplinlerin uygulamaları içine girmelerinde yardımcı olmaktadır. Hem öğrencilerin hem de öğretim elemanlarının teknoloji ve bilgi okur-yazarlığını geliştirmelerini sağlamaktadır. Hem başlangıç hem de ileri düzeyde olanları, teknolojik araçları kullanmaları konusunda da cesaretlendirmektedir. Akademik araştırmalara duyulan ilgi artmakta, hem akademisyenlerin hem de öğrencilerin araştırma yapmaları web tarafından desteklenmektedir. Bilginin önemli olduğu kadar, onu kullanma süreçlerinin de eşit önemde olduğu bir çağa doğru ilerlenmektedir. İnternet bunların her ikisini de desteklemektedir. [2]

İnternet tabanlı eğitim gelişen bilgisayar ve İnternet teknolojisiyle birlikte hızla artış göstermiş ve çok kullanılan bir eğitim ortamı haline gelmiştir. Bir yandan eğitim kurumları, diğer yandan şirketler kendi internet tabanlı eğitim modellerini geliştirmektedirler. Çünkü hızla artan dünya nüfusunun eğitimi, iki yılda bir yarılanan teknolojinin takip edilmesi ancak internet tabanlı eğitimin kullanılmasıyla mümkün görünmektedir. [3]

İnternet tabanlı eğitim modellerinde kullanılan etkileşimli sayfalar, eğitim verilen dersin işlenişini kolaylaştırmakta, dersin eğitim yönünden kalitesini arttırmaktadır. [4]

Dersi etkileşimli sayfalardan izleyen öğrenciler gerçek hayatta gözle göremeyecekleri birçok olayı deneyerek görebilmekte ve konuyu daha kolay anlayabilmektedir. İnternet tabanlı eğitim için etkileşim, web sayfalarındaki gerekli yerlere canlandırma(animasyon), benzetim(simülasyon), ses, görüntü ve film yerleştirme ile sağlanabilir. Canlandırma(animasyon) ile ders içeriğine ait bir olayın nasıl olduğu canlandırılabilir. Simülasyonlar(benzetim) ile gerçek hayatta gözle görülemeyecek olaylar, tehlikeli deneyler veya pahalı araç gereç gerektiren laboratuar uygulamaları gerçekleştirilebilir.

1.3. İnternet Tabanlı Uzaktan Eğitim

Uzaktan eğitim, öğrenci ile eğitici arasında, basılı ve diğer teknolojik araçları da kullanan bağımsız bir eğitim şeklidir. Eğitsel kaynakları öğrenenle birleştiren eğitsel sisteme uzaktan öğrenme denir. Uzaktan öğrenme, öğrenenlerin eğitim kurumlarına yazılmadan eğitime girişlerini sağlamaktadır. Uzaktan eğitime geçişteki en önemli sorulardan biri "niçin uzaktan eğitim" sorusudur. Bu soruya verilecek cevaplar aşağıda listelenmiştir.

- Geniş bir öğrenen kitlesine ulaşmak.
- Konferans şeklinde farklı konuşmacılarla öğrenme ortamı oluşturmak.
- Monolog ortamlara hapis olmamak
- Derslere devam edemeyen ve özrü bulunan öğrencilere ulaşmak.
- Farklı kültürlerden öğrencilerle farklı konulara hakim alanlarda tanışmak

1.4. İnternet Ortamında Eğitim

İnternet Tabanlı Eğitim: Öğrenen ve öğretenin zaman ve mekan olarak birbirlerinden farklı durumlarda bulunduğu, altyapı olarak internet tabanlı yerlerin kullanıldığı gerçek zamanlı istendik davranışlar geliştirme sürecidir. Çağımızın en yaygın aracı olarak kullanılmakta olan internet eğitiminde de yaygın olarak kullanılmaktadır.

Web Tabanlı Eğitim: Türkiye'de internetin kullanımı her geçen gün belirli oranlarda artmaktadır. 1995 yılındaki tahminlere göre %6'dan %20'ye periyodik oranlarda artış meydana gelmiştir. 1995'li yıllardan sonra bu oran %100'lere varan bir artış göstermekte buda insanoğlunun internete olan ihtiyaç talebini göstermektedir. Bu şekilde artan kullanıcı yoğunluğuna ek olarak web üzerinden yapılan projelerin geliştirilmesi, web sunucuların yapılandırılması, ana sayfalar ve dijital kaynaklar ile ilgili yapılandırılmalar en üst düzeye çıkarılmıştır.

Posta Yoluyla Eğitim: Gerek yurtdışı, gerek yurtiçi öğretmen kanalıyla çeşitli yaş gruplarında posta grubu oluşturulabilmektedir. Öğrenciler derslerini, birbirlerinin deneyimlerini, becerilerini paylaşarak çalışabilmekte, eksiklerini

tamamlayabilmektedirler. Posta yoluyla eğitim, web tabanlı eğitimde olduğu gibi işitsel ve görsel olanaklar sağlamamakta, sadece metin(text) tabanlı bir etkileşim ortamı oluşturulmaktadır. [5]

1.5. Sanal Öğretimin Avantajları ve Dezavantajları

Uzaktan eğitim sisteminde yaşanılan aksaklıklar ve bunun yanında özel avantajlar da bulunmaktadır. Öğretimin, öğretmen ve öğrencinin öğrenme-öğretme yeteneğini belli düzeylere ayırmak mümkündür. Öğrenci ders materyalini kendi istediği zamanda ve mekânda izleyebilmektedir. Öğrenci anlamakta güçlük çektiği noktaları tekrar gözden geçirebilir. Böylece sınıf içindeki tüm öğrencilerin aynı zekâ ve istek düzeyinde kabul edildiği yüz yüze öğretim ortamındaki sıkıntılar aşılabilmektedir.

Sanal öğretimde ders materyali hızla güncelleştirilebilir. Basılı materyal dağıtımı hızlı bir şekilde aktarılabilir. Öğrencinin, öğretmen ve diğer öğrenci arkadaşları ile hızlı bir etkileşim içerisinde olması sağlanır. Sanal öğretim sistemleri ile dağıtılan bilgi süreklilik ve güncellik arz ettiğinden yaşam boyu öğrenme imkanı sağlar.

Bu avantajların yanında dezavantajlarda mevcuttur. Bunlardan biri hiç şüphesiz örgün öğretim sisteminde olduğu gibi öğrenciler arasında birliktelik, grup bilinci gelişimi ve kültürel etkileşim gibi bazı psikolojik ve sosyolojik unsurları sağlayamamasıdır.

Sanal öğretimde karşılaşılan en önemli sorunlardan biri de ders materyali hazırlama ve dağıtımıdır. Çünkü öğretmen ve öğrencilerin belirli seviyede internet teknolojilerini tanıması gerekmektedir. Aynı zamanda yüz yüze eğitimde, konu anlatımı sırasında öğrencilerin herhangi bir konu ile ilgili problemleri olduğunda bunu rahatlıkla öğretmenlerine sorabilirler, fakat sanal öğretimde böyle bir şansları yoktur.

1.6. Eğitimde İnternet Uygulamalarını Kullanmanın Faydaları

Öğretimin düzenlenmesine ve sunumuna yeni bakış açısı getiren internet, eğitimcilere birçok olanak sunmaktadır. Web sayfaları, haber ve tartışma gruplarına, Telnet sistemine, gopher'lara ve diğer web sayfalarına kolaylıkla bağlanabilmektedir. Bu şekilde hazırlanan sayfaları her alana genişletmek mümkündür. Fakat bu avantaj, kontrolün azalmasına yol açarak dezavantaja da dönüşebilmektedir, birçok bağlantı içerisinde, öğrenen kişi kaybolabilmektedir. Bu nedenle internet kullanılmalı ama sınırları da bilinmelidir. Örneğin değişiklik, güncelleme ihtiyacının yoğun olduğu konu alanlarında ve coğrafi olarak çok geniş boyutlu bir katılımcı ile öğretimi gerçekleştirmek gerektiğinde web kullanılabilir, ancak okul öncesinde çok sınırlı olarak yararlanılabilir. Eğitimde internetin sınırlılıklarını sıralayacak olursak;

- Teknolojinin sürekli ve çok hızlı gelişiyor olmasına karşın sınırlı band aralığı ve ses, video ve grafik iletiminde sorun çıkaran yavaş modellerin varlığı.
- Öğrenen kişinin insiyatifine dayalı olmasıyla daha yapılandırılmış ve aşamalı bir düzen isteyenler için uygun olmaması.
- Belli bir düzeyde bilgisayar ve teknik becerileri gerektirmesi.
- Gereğinden fazla bilgi yükü olması; okunacak ve üzerinde düşünülecek yüzlerce e-posta bulunmaktadır, bunları yanıtlamak da ayrıca zaman istemektedir, veri tabanları ve Web sitelerindeki bilgilere ulaşma için, bilgi yönetimi becerileri gerekmektedir.
- Kırsal bölgelerden internete girebilmenin hala bir sorun olması.
- Sosyal açıdan izolasyon yaratması ve sözel olmayan ipuçlarını eksikliği ile iletişim sorunlarına yol açması.
- Aktif öğrenmeyi desteklemesine karşın, televizyonda olduğu gibi pasif olarak izlemeye neden olabilmesi.
- Aile yaşantısını olumsuz etkileyebilme. [6]

1.7. Java Dilinin Önemi

Web tabanlı eğitim sayfalarında kullanılan canlandırma(animasyon) ve benzetimlerin(simülasyonların) Java programlama dili ile tasarlanmasının bazı

nedenleri vardır. Java dili bilgisayar ağları üzerindeki farklı bilgisayar platformlarında kullanılacak yazılımları geliştirmek amacıyla geliştirilmiştir. [7]

Java programlama dili bilgisayar ortamında kullanılabilecek en güvenli yazılımdır. Girilen web sayfasında bulunan Java appletinin bilgisayara virüs bulaştırmasını veya başka bir istenmeyen işlem yapmasını önlemek için içerisinde bir dizi mekanizma oluşturulmuştur. Java güvenlik mekanizması Java applet'inin kullanıldığı makinedeki veya İnternet üzerindeki bir başka makinedeki dosya sistemini okumasına veya bu dosya sistemine bir şey yazmasına, sunucu makinenin dışındaki bir makineye bağlanmasına ya da bunların dışındaki tehlikeli işlemlere izin vermez. İnternet ortamında bir web sayfasındaki applet'i çalıştırmak güvenlik açısından son derece zararsızdır. Diğer taraftan C++, Delphi gibi programların özel kütüphaneler kullanarak yaptıkları ağ işlemlerini Java programlama dili rahatlıkla yapabilmektedir. Java dili gün geçtikçe hızla kendini geliştirmektedir ve geleceğin belki de en gözde dili olmaya adaydır. [8]

Bu tezde günümüzde en çok kullanılan ve yeni yeni gelişmeye başlayan Java dili incelenecektir. Türkiye'ye baktığımızda Java dili ile yeterli Türkçe kaynak olmadığının farkına vardık. Bu tezde öğrencilerin, özellikle Java dilini hiç bilmeyen, fakat öğrenmek isteyen öğrencilere internet üzerinden hızlı ve kolay bir şekilde anlatmayı ve öğretmeyi hedefledik. Tez içerisinde Java dilini 10 ayrı bölümde inceleyeceğiz.

İkinci bölümde Java dilinin genel yapısı, Java'nın tarihi, Java'da kullanılan değişken tipleri ve kullanım biçimleri ve Java operatörleri hakkında bilgiler verilmiştir.

Üçüncü bölümde ise Java'da kullanılan kontrol ifadeleri anlatılmıştır. Bu bölümde if yapısı, switch yapısı, for döngüsü, while döngüsü, do-while döngüsü ve dallanma ifadeleri hakkında bilgiler verilmiştir.

Dördüncü bölümde Java dilindeki dizilerden bahsedilmiştir. Bu bölümde dizi değişkenleri, dizi elemanlarına erişim, dizi sıralama ve kopyalama ve çok boyutlu dizilerden bahsedilmiştir.

Beşinci bölümde Java dilindeki sınıf(class) yapısından bahsedilmiştir. Bu bölümde sınıf oluşturma, metot oluşturma, this anahtarı, sınıf metotları, sınıfın gizlenmesi, isim ve argüman oluşturma, yapı metotlarından bahsedilmiştir.

Altıncı bölümde Java dilindeki applet yapısından bahsedilmiştir. Bu bölümde appleti web sayfasına ekleme, applet etiketi ve parametreleri, graphics ve paint komutları, Mouse appleti, grafik appleti, hesap makinesi, blok oyunu, Mouse çizimi, tetris oyunu, ufo oyunu konularından bahsedilmiştir.

Yedinci bölümde Java dilindeki layout(yerleştirme) yapısından bahsedilmiştir. Bu bölümde borderlayout, gridlayout, flowlayout, cardlayout, gridbaglayout, boxlayout, verticalflowlayout konularından bahsedilmiştir.

Sekizinci bölümde Java dilindeki program hatalarından bahsedilmiştir. Bu bölümde program hatalarını yakalama, hatanın yakalanması, try-catch, çoklu hata yakalama, throw ve throws deyimleri ve finaly konularından bahsedilmiştir.

Dokuzuncu bölümde Java paketlerinden bahsedilmiştir. Bu bölümde Java paketleri, Java.lang, Java.io, Java.util, Java.net, Java.awt, Java.awt.image ve Java.awt.peer konularından bahsedilmiştir.

Onuncu bölümde Java dilindeki awt paketlerinden bahsedilmiştir. Bu bölümde awt, buton, checkbox, checkboxgroup, label, textfield, choice, menu, popupmenu, scrollbar, textarea ve panel konularından bahsedilmiştir.

Onbirinci bölümde uygulamayı hazırlarken kullanılan programlar anlatılmıştır.

BÖLÜM 2. JAVA'NIN GENEL YAPISI

2.1. Java'nın Tarihi

Belki de mikroişlemci devriminin en önemli tarafı dünya genelinde yüz milyonlarca sayıya ulaşan kişisel bilgisayarların yapılmasıdır. Kişisel bilgisayarlar insanların yaşamında derin bir etki bırakmıştır ve işlerini yapmada kolaylık sağlamıştır. [9]

Java programlama dili şu anda dünyadaki en popüler programlama dillerinden biri haline gelmiştir. Java, SUN bilgisayar şirketince elektrikli ev araçlarının birbirleriyle haberleşmesini amaçlayan bir proje içerisinde 1991 yılında geliştirilmeye başlandı. Orijinal adı dilin yaratıcıları James Gosling, Patrick Nauoght,Chis Wardh, ED Frank ve Mike Sheridian tarafından OAK olarak konulan programlama dili, daha sonra bu isimde başka bir programlama dili olduğu anlaşılınca kahve markasından esinlenerek Java olarak değiştirildi. Akıllı elektronik ev parçaları pazarı, SUN grubunun tahminlerinden çok daha yavaş bir gelişme gösteriyordu. Bu yüzden Java dili projesi, ticari bir geliştirme projesi olarak büyük olasılıkla iptal edilecekti. 1993 yılında internetin tüm dünyada yaygınlaşmaya başlaması ve Java'nın büyük bir atılım göstererek web sitelerinde kullanılması Java'nın önemini artırdı. Java'nın dinamik web sayfaları hazırlamadaki büyük başarısını gören SUN şirketi projeyi bu tarafa yönlendirdi.

Mayıs 1995'de SUN, Java'yı büyük bir konferansta tanıttı. Program, iş dünyası tarafından büyük bir ilgiyle karşılaştı. Java internet web sunucularının fonksiyonlarını çoğaltmak ve cep telefonu, sayfalayıcılar, PDA gibi birçok cihaz için program geliştirmek için çalışabilmekteydi; ayrıca modern bilgisayar dünyasının ses, grafik işlem, haberleşme gibi ihtiyaçlarına cevap verebilen ve ticari gayeler için hazırlanana bir programlama dili olarak daha önceki bilgisayar dillerinin hiç birinin kapsayamadığı özellikleri içermekteydi.

Bunun yanı sıra Java dilinin komut yapısı olarak C++ diline çok yakın olması, öğrenilmesini kolaylaştırıyordu. SUN, Java'yı internette kullanmak isteyen herkese ücretsiz olarak sundu. Java'nın internette yayınlanmasının ardından çok büyük bir patlama yaşandı. 1997'ye gelindiğinde dünyadaki bütün okullarda temel bilgisayar dili olarak gösterilmeye başlandı. Dünyada şuanda en çok kullanılan bilgisayar dili olan C++ dilinin, yapılan hataları tam olarak denetlenememesi, program çalışma hızını arttırma yönünde iyi bir özellik olsa da, profesyonel programcılar dışında kullanılmasını sınırlandırıcı bir etki yapıyordu. Java ise bütün hataları bildiren yapısı ve modern bilgisayarın bütün fonksiyonlarına ulaşabilen kütüphaneleriyle programcıların çok daha kolaylıkla öğrenebileceği bir dildir.

Java'nın diğer önemli bir temel özelliği, nesne temelli(object oriented) bir dil olmasıdır. Nesne temelli diller; nesneleri, gerçek dünyadakine daha benzer yapıda tanımlayarak anlaşılmalarını kolaylaştırırlar. Nesneleri gerçek dünyadaki gibi; masa, sandalye, bilgisayar, gerçek gaz gibi tanımlayarak programlamak insan beyninin anlaması açısından çok daha kolaydır. Bundan önceki nesne temelli programlama dilerinden hiçbiri çok yaygınlık kazanmamıştı. Bu tür dillere Smaltalk'u örnek verebiliriz. C++ nesne temelli programlama yapılabilen bir dildir. Fakat yapısal(structural) bir programlama dili olan C dilinin bir uzantısı olarak geliştirildiğinden, tam anlamıyla nesne temelli bir dil olduğu söylenemez.[10]

Java dilini geleceğin dili yapan diğer bir özelliği de, çok kullanımlı(multi tasking) ve paralel kullanımlı(multi threading) bir dil olmasıdır. Çok kullanımlılık birden fazla işlemin aynı anda yapılabilmesinin tanımıdır. Paralel kullanımlılık ise birden fazla programın aynı anda hafızayı beraber kullanılabilmesidir. Örnek olarak Word ve Excel programlarının Windows XP ortamında aynı anda kullanılmasıdır. Bazı eski programlama dillerinde çok kullanımlılık programlanabiliyordu. Paralel kullanım olanaklarını sunan ilk bilgisayar dili ise Java'dır. Paralel kullanım paralel programlama kavramından ayrıdır ve karıştırılmamalıdır. Paralel programlamada birden fazla merkezi işlem birimine (CPU), ayrı programlar veya bir programın ayrı parçaları gönderilir. Paralel kullanımda ise bir CPU'nun kullanım zamanı küçük parçalara ayrılarak, değişik program veya program parçacıkları bu zaman paketçiklerini paylaşarak kullanırlar.

Java'yı önemli bir program dili haline getiren en önemli özelliği ise, kullanılan bilgisayardan bağımsız olmasıdır. Java'da yazılan bir program Unix, Macintosh, Windows 98, Windows XP veya herhangi bir makinede hiç değiştirilmeden kullanılabilir. Java programlarının grafikleri, internet sayfalarının programlama dili olan HTML(Hypertext Markup Language) ile aktarılır. Bu yüzden HTML ve Java programlarını birlikte kullanmak ve Java programlarını gerçek zamanda internet sayfalarında göstermek mümkündür. [9]

Java'nın HTML diliyle kullanılabilmesi, bazı kişilerde HTML'nin bir parçası olduğu gibi bir kavram gelişmesine yol açmıştır. Programlamaya yeni başlayanlar sık sık HTML ile Java'yı birbirine karıştırırlar veya Java'nın, HTML'nin bir uzantısı olduğunu düşünürler. Bu temelde oldukça yanlış bir varsayımdır. HTML, internet belgelerini birbirine bağlamak amacıyla geliştirilmiş bir belge işlem sistemidir ve bir programlama dili değildir. HTML ile Java'nın tek gerçek ilgisi HTML'deki applet komutudur. Bu komut yardımıyla, Java dilinde yazılmış programların sonuçları HTML ortamına dinamik olarak aktarılabilir. [11]

Java programlama dili zaman geçtikçe hatalarını düzeltmeye devam etmiştir. Java 1.0, 1.1, 1.2, 1.3,1.4 ve son olarak 1.5 versiyonuyla devam etmiştir. Java 1.2 sürümü ile birlikte yapılan ciddi değişikliklerle Java 2 adını almıştır.

Özellikle Java 1.3 ve 1.4 ile gelen bazı ilerlemeler Java uygulamalarının daha hızlı çalışmasını sağladığı için Java son birkaç yıl içerisinde daha da popüler hale gelmiştir.

Bir bilgisayar programı, genel olarak belirlenen bir bilgisayar platformunda (İşletim sistemi) çalışan 1 ve 0'lardan oluşmaktadır. Bu yapılar insanların anlayabileceği şekilde değillerdir. Bu durumda programlama dilleri devreye girer. Bu dillerde yazılan programlar başka programlar tarafından çeşitli işlemlerden geçirilir ve 1 ve 0 haline getirilir. Tıpkı doğal diller gibi, programlama dillerinin de bir yapısı ve söz dizimi vardır. Bir programlama dilini öğrenmek de ilk olarak bu yapıyı anlamak ve sözdizimi hatası yapmadan komutlar yazabilmekle başlar.

Java, internet tabanlı ve veritabanlarına bağlantılar içeren, üstelik tarayıcıdan kullanılan bir görsel arabirim gerektiren uygulamalar yazmak için diğer dillerden daha gelişmiş özelliklere sahiptir. Servlet, JSP, JDBC, EJB, JMX gibi Java teknolojileri bu tarz uygulamaları yazarken ciddi kolaylıklar sağlamaktadır. [11]

2.2. Java Dilinin Temelleri

Java programı normalde 5 bölümde çalıştırılır. Bunlar Edit, Compile, Load, Verify ve Execute'dur. Java programlarını çalıştırabilmek için Java 2 Software Development Kit (J2SDK) versiyon 1.5.0 kurulmalıdır.

Birinci bölümde bir dosyanın düzenleme işlemi yapılmaktadır. Bu işlem bir editör programı ile yapılır. Programcı Java programını editör kullanarak yazar ve gerekli olduğunda düzeltmelerini yapar. Programı yazdıktan sonra, programı editör programı sayesinde diske kaydeder(Harddisk, Disket, USB Disk vb.). Java program dosyaları kaydedildiklerinde Java uzantısını alırlar. Java isimli uzantılar bu dosyanın Java programlama dilinin kaynak kodu olduğunu gösterir.

İkinci bölümde programcı programı derlemek için JAVAC komutunu kullanır. Java derleyicisi, Java programını Bytecode yapısına dönüştürür. Eğer program doğru bir şekilde derlenmişse, derleyici Welcome.class isminde bir dosya oluşturur. Bu dosya içerisinde komutların Bytecode hali bulunmaktadır.

Üçüncü bölüm yükleme diye adlandırılır. Program çalıştırılmadan önce ilk olarak hafızaya yerleştirilmelidir. Bu işlem Bytecode'ları içeren sınıf(class) dosyalarını alan Class Loader ile yapılır ve onları hafızaya aktarır. .Class dosyaları sistemdeki hard disk' ten veya bir ağ üzerinden yüklenebilir.

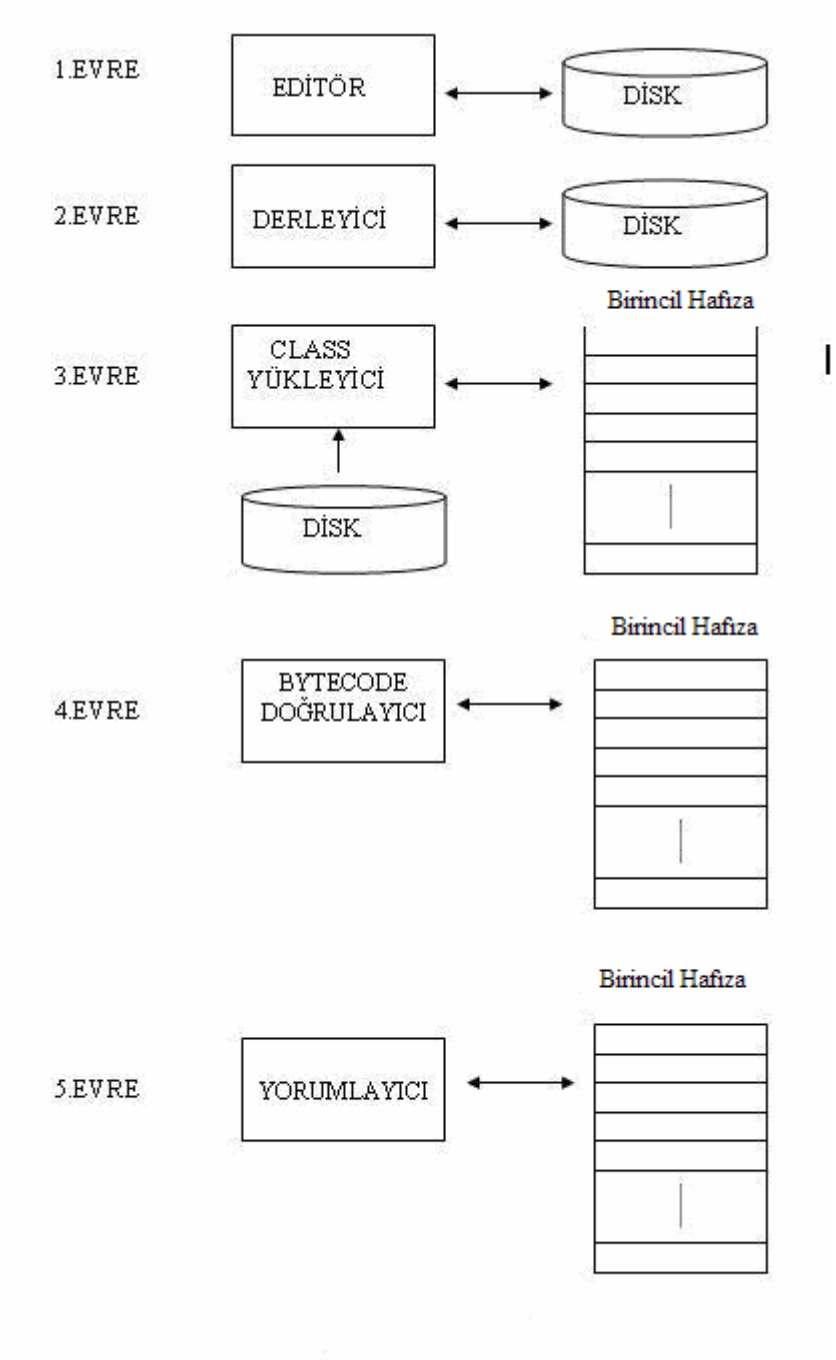
Dördüncü bölümde yüklenen sınıf(class) dosyalarının bytecode'ları Bytecode Verifier sayesinde doğrulanır. Bu doğrulama işlemi sınıf(class) Bytecode'larının geçerli olduğunu kesinleştirir ve Java'nın güvenlik sınırlamalarını bozmaz. Java kuvvetli bir güvenlik içermektedir. Çünkü Java programları ağ üzerinden dosyalara

ve sisteme zarar vermemelidir. Bytecode doğrulaması, aynı zamanda ağ üzerinden yüklenen uygulamalarda da olur.

Son olarak beşinci bölümde, işletim sisteminin kontrolündeki yapı programı bir Bytecode'a çevirir. Böylece program tarafından belirtilen işlemler gösterilir. Bu işlemin olması için iki tür program vardır. Bunlar uygulamalar ve applet'lerdir. Uygulama dediğimiz yapıya, kelime işlemci programı, çizim programı veya bir mail programı örnek gösterilebilir. Bu tür programlar kullanıcını kendi bilgisayarında depolanır ve çalıştırılır. Applet ise internet tarayıcılarına bağlı uzak bilgisayarlarda depolanan küçük programlara denir. Buradaki uzak bilgisayarlar web sunuculardır. Appletler uzaktaki web sunucu bilgisayarında yüklenir, tarayıcıda çalıştırılır ve çalıştırma işlemi bittiği zaman bilgisayardan boşaltılır. Uygulamalar hafızaya yüklenir ve Java derleyicisindeki Java komutu ile çalıştırılır.

2.3. Hoşgeldin Java Uygulaması

Java Hoşgeldin programı çalıştırıldığı zaman hoşgeldin uygulaması için derleyici çalışır ve hoş geldin programında kullanılan bilgiler Class Loader ile yüklenir. Şekil 2.1'de Java programının çalışma evreleri görünmektedir.



Şekil 2.1. Tipik Java evreleri [12]

Java platformu üç ana gruba ayrılır.

- Standart Java
- Enterprise Java
- Tüketici için ve gömülü cihazlar için Java (embedded devices)

Yukarıdaki kategoriler açılacak olursa 3 gruba ayrılır.

2.3.1. Standart java

- Java 2 SDK (J2SE)
- Java 2 Runtime Environment
- Java Plug-in
- Java Web Start
- Java HotSpot Server Virtual Machine
- Collections Framework
- Java Foundation Classes (JFC)
- Swing Components
- Pluggable Look & Feel
- Accessibility
- Drag and Drop
- Security
- Java IDL
- JDBC
- Java Beans
- Remote Method Invocation (RMI)
- Java 2D

2.3.2. Enterprise java

- Enterprise Java Beans (EJB) Architecture
- Java Server Pages (JSP)
- Java Servlet
- Java Naming and Directory Interface (JNDI)
- Java IDL
- JDBC
- Java Message Service (JMS)
- Java Transaction (JTA)
- Java Transaction Service (JTS)

- Java Mail
- RMI-IIOP
- Software Development Kit & Application Model
- Java 2 SDK, Enterprise Edition (J2EE)
- Sun BluePrints Design Guidelines for J2EE

2.3.3. Java micro edition

- Java 2 Platform, Micro Edition (J2ME technology)
- Connected Device Configuration (CDC)
- Connected Limited Device Configuration (CLDC)
- C Virtual Machine (CVM)
- K Virtual Machine (KVM)
- Personal Java
- Java Card
- Java Phone API
- Java TV API
- Jini Network Technology
- Mobile Information Device Profile (MIDP) [12]

2.4. Java Programlama Dili Temel Değişken Türleri

Programlama dillerinde rakamlar bilgisayar belleğinin temel depolama birimlerine yazılırlar. Temel bilgisayar bellek birimi bit olarak adlandırılır. Bir bilgisayar belleğindeki tek bir transistörden oluşmuştur. Bu transistörden akım geçiyorsa tansistörün veya bitin bellek değeri 1 veya true(doğru) olarak alınır. Eğer akım geçmiyor veya düşük düzeyde bir akım geçiyorsa transistörün bellek değeri 0 veya false(yanlış) olarak alınır. Bilgisayar bit birimleri bir araya gelerek, bilgisayar temel değişken türlerinin yazılabileceği bir sistem oluşturur. Değişken türü bilgisayar tarafından bilinmelidir, çünkü aynı bit topluluğu bir harfi simgeleyebileceği gibi bir rakamı da simgeleyebilir. Java dilinde temel değişken türleri mevcuttur. Aşağıda bu temel değişkenler anlatılmıştır.

2.4.1. Booelan değişken türü

Boolean değişken türü mantık işlemlerinde kullanılır. Sadece true(doğru) veya (false) değerleri alır. True doğru false yanlış anlamı taşır. Mantık değişkenlerine doğrudan true(doğru) veya false(yanlış) değerleri yüklenebileceği gibi diğer değişkenleri mantık işlemlerini kullanarak ve karşılaştırarak ta değerler programların içinde hesaplanabilir. Tablo 2.1'de temel değişkenler verilmiştir.

Tablo 2.1. Java Temel Değişken türleri

Değişken Türü	Türkçe karşılığı	Bit büyüklüğü	Sınır Değerleri
Boolean	Mantık değişkeni	1	true(doğru) , false(yanlış)
Char	Harf değişkeni	16	'\u0000' den '\uFFFF'
Byte	Tam sayı değişkeni	8	-128 den 127 e kadar
Short	Tam sayı değişkeni	16	-32768 den 32767 e kadar
Int	Tam sayı değişkeni	32	-2157483648 den 2147483647 e kadar
Long	Tam sayı değişkeni	64	-9223372036854775808 den 9223372036854775808 e kadar
Float	Gerçek sayı değişkeni	32	-3.40292347e+38 den 3.40292347e+38 e kadar
Double	Gerçek sayı değişkeni	64	-1.7976931348623157e+308 den 1.7976931348623157e+308 e kadar

Aşağıdaki küçük program parçası mantık değişkeni ilk mantık değişkenini tanımlamakta ve true(doğru) değerini bu değişkene yüklemektedir.

```
boolean ilkmantıkdegiskeni;
ilkmantikdegiskeni=true;
```

2.4.2. Char değişken türü

Char değişken türü karakterlerin tanımlanmasında kullanılır. Karakterler Java dilinde ISO Unicode kodu ile bilgisayara aktarılır. Unicode 4 hexadecimal (16 tabanlı) sayının bir araya gelmesiyle oluşur. Hexadecimal sayı sisteminin onlu ve ikili sayı sistemiyle eşitliği aşağıdaki tabloda gösterilmiştir. Tablo 2.2'de hexadecimal, onlu ve ikili sayı sistemleri listeleri verilmiştir.

Tablo 2.2. Hexadecimal(onaltılı), onlu ve ikili sayı sistemleri eşitlikleri

Hexadecimal	Onlu	İkili
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

ISO Unicode da tanımlanan '\u0041' kodu 'A' harfi anlamına gelir. Veya '\u03E1' kodu '□' karakterini tanımlar. ISO Unicode karakter setinin ilk iki rakamı sıfır

olduğunda ASCII karakter kodunu tanımlar. Aşağıdaki örnekte char tipi A1, A2 ve alpha1,alpha2 değişkenlerine A ve a harflerini yüklemektedir.

```
char A1,A2;
char alpha1,alpha2;
A1='\u0041';
A2='A';
alpha1='\u03E1';
alpha2='□';
```

2.4.4. Tam sayı değişken türleri

Tam sayı değişkenler hafızada işgal ettikleri yere göre byte(8 bit), short(16 bit), int(32 bit) ve long(64 bit) adını alırlar. Bir bitlik hafızaya sadece iki rakamın (0 veya 1) yazılabileceği göz önüne alınırsa örneğin sekiz bitlik byte türü tamsayı değişkenine 256 sayı (ikili sayı eşiti 11111111) yazılabileceği ortaya çıkar. Bitlerden biri + veya - işareti için kullanıldığından byte değişkeninin sınır değerleri -128den 127 e kadardır. Eğer bir tamsayı değişkenin sadece artı değerleri kullanılmak istenilirse unsigned terimi kullanılır. Örneğin unsigned byte tipi tamsayı değişkenin sınır değerleri 0 dan 256 ya kadardır. Tamsayı değişken türleri içinde en fazla kullanılan int türüdür. Aşağıdaki örnekte int türü ilktamsayi değişkenine 32 rakamı yüklenmektedir.

```
int ilktamsayi;
ilktamsayi=32;
```

bu iki satırlık örnek tek bir satır olarak

```
int ilktamsayi=32;
```

Şeklinde de yazılabilir.

Java’daki tamsayı değişken türleri + ve – değeri alabilen türlerdir. Örneğin byte değişken türünde –128 den +127 e kadar toplam 256 sayı kullanılabilir. Eğer – bölge kullanılmayacaksa tamsayı değişkenlerin kullanım bölgesini unsigned deyimini kullanarak tamamen artı bölgeye çekilebilir. Aşağıda bunla ilgili bir örnek bulunmaktadır.

unsigned byte artibolgetamsayisi;

Yukarıdaki örnekte tanımlanan artibolgetamsayisi değişkeni 0 ile 256 arasında değerler alabilir.

2.4.5. Gerçek sayı değişken türleri

Gerçek sayı sistemleri de 0 ve 1 bitleri kullanılarak oluşturulabilir. Gerçek sayıların tamsayıdan küçük kısımları ikilli tabanda eksi üstler kabul edilerek oluşturulur.

Gerçek sayı değişkende yeterli hassasiyeti sağlayabilmek için genelde 64 bit uzunluğundaki double değişken türü kullanılır. Java’daki Matematik kütüphaneleri de double değişken türü için tanımlanmıştır. Aşağıdaki örnekte double türü ilkgercekdegisken değişkenine 22.625e-7 sayısı yüklenmektedir.

double ilkgerçekdegisken;
ilkgerceldegisken=22.625e-7;

Eğer double sayı hassaslığı yetmezse long double kullanılabilir. Bu değişken türü 128 bit boyutundadır.

2.4.6. String nesne tipi değişkeni

String değişkeni yazı yazdırma işleri için kullanılır. Nesne türü değişkendir. Java dilinde char değişken türü kullanılarak tanımlanmış bir nesne tipi değişkendir. Aşağıdaki örnekte bu değişken türünün kullanılışı görülmektedir.

```
String a="ali";
String c="veli";
String d;
d=a+b; // d nin çıkış değeri "ali veli"
System.out.println(d); // bu satır ali veli çıktısı veri
```

String türü sabitler her zaman " " işaretleri arasına yazılırlar ve + işaretiyle bir araya getirilebilirler. Aşağıdaki örnekte bu değişken türünün kullanılışı görülmektedir.

```
String a=new String("ali");
String c=new String("veli");
String d=new String();
d=a+b; // d nin değeri "ali veli"
System.out.println(d); // bu satır ali veli çıktısı verir
```

2.4.7. Integer nesne tipi değişkeni

Integer, tam sayı işlemlerinde kullanılan bir değişken türüdür. Integer değişkeni

```
Integer i;
i=new Integer(3);
```

veya

```
Integer i=new Integer(3);
```

Şeklinde tanımlanabilir. Integer tanımı String değişkeni üzerinden de tanımlanabilir. Aşağıdaki örnekte bu yapının kullanılışı görülmektedir.

```
String s=”15”;
Integer i=new Integer(s);
```

Yukarıda 15 değeri s değişkenine yüklenir. Integer değeri int değerine çevrilebilir.

```
int x;
Integer  y=new Integer(3);
x=Integer.IntegerValue(y);
```

String değerini int değerine doğrudan çevirmek içinse

```
String s=”15”;
Int x=Integer.parseInt(s);
```

kullanılır.
int tipi değişkeni String değişkenine dönüştürmek için

```
int x=3;
String s=Integer.toString(x);
```

Komutları kullanılır.

2.4.8. Double nesne tipi değişkeni

Double değişkeni kullanım olarak Integer değişkeninden bir farkı yoktur. Aynı tanımlamalar Integer yerine Double kullanılarak yapılabilir. Değişkenler

```
Double  x;
X=new Double(3.66e5); Veya

Double x=new Double(3.66e5);
```

Şeklinde tanımlanabilir.

String değerini double (temel değişken) değerine doğrudan çevirmek içinse

```
String s=”15”;
Int x=Double.parseDouble(s);
```

İşlemi kullanılabilir.

Double tipi (temel) değişkeni String değişkenine değiştirmek için

```
Double x=3.75;
String s=Double.toString(x);
```

kullanılır.

2.4.9. Diğer nesne temelli değişken türleri

Object, Long, Float, Boolean, Character, Vector gibi diğer nesne tipi değişken türleri de mevcuttur. Object nesne tipi tüm bu değişken nesne tiplerini içinde barındıran genel bir tiptir. Tüm nesne tipi değişkenlerin bizim için temel avantajı, alt metotlarını kullanarak işlemler gerçekleştirme ihtimalidir. Fakat bu tür değişkenler temel değişken türlerine göre daha çok hafıza yeri işgal ettiklerinden mecbur kalınmadıkça da kullanılmamalıdırlar. Ayrıca nesne tipi BigDecimal ve BigInteger türleri de hassasiyeti kullanıcı tarafından belirlenen nesne tipi değişkenler olarak kullanılabilirler.

2.4.10. Final terimi ve sabitler

Java dilinde değişken yerine sabit değer kullanılmak istenilirse tanımın başına final sözcüğü getirilir. Final olarak tanımlanan sabitlerin bir kere değerleri verildikten sonra değiştirilemez. Aşağıdaki örnekte bu değişken türünün kullanılışı görülmektedir.

```
final double pi=3.14159;
```

Yukarıdaki örnekte kullanılan yapı Pi sabitini tanımlar.

2.4.11. Primitive tipler

Tablo 2.3'de Java dilindeki 8 tane primitive tip listelenmektedir. Önceki C ve C++ dillerinde olduğu gibi, Java dili de tüm değişkenlerin bir tipi olmasını istemektedir. Bu düşünce ışığında Java kuvvetli tipli dilleri kapsamaktadır.

C ve C++ dillerinde, programcılar sık sık değişik bilgisayar sistemlerini destekleyen programların ayrı versiyonlarını yazmak zorunda kalırlar. Çünkü primitive tiplerin bilgisayardan bilgisayara aynı olduğu garanti edilemez. Örneğin bir makinedeki tamsayı değeri hafızanın 32 biti ile temsil edilirken, başka bir makinede hafızanın 16 biti ile temsil edilebilir. Java'da ise tamsayı değerler daima 32 bittir.

Tablo 2.3. Java dilindeki primitive tipler

OPERATÖRLER	ÇAĞRIŞIMI	TİPİ
++ --	Sağdan sola	Unary Postfix
++ -- + -	Sağdan sola	Unary
* / %	Soldan sağa	Çarpımlı
+ -	Soldan sağa	Katkılı
< <= > >=	Soldan sağa	Bağlantılı
== !=	Soldan sağa	Eşitlik
?:	Sağdan sola	Durumsal
= += -= *= /= %=	Sağdan sola	Atama

2.5. Nesneler ve Atamalar

Nesneler için atama işlemleri, temel tiplere göre biraz daha karmaşıktır. Nesneleri yönetmek için referanslar kullanılır; eğer, nesneler için bir atama işlemi söz konusu ise, akla gelmesi gereken ilk şey, bu nesnelere bağlı olan referansın gösterdiği hedeflerde bir değişiklik olacağıdır. Aşağıda bu özellik ile ilgili örnek bir program bulunmaktadır.

```
class Sayi {
    int i;
}
public class NesnelerdeAtamaislemi {
  public static void main(String[] args)
{
    Sayi s1 = new Sayi();
    Sayi s2 = new Sayi();
    s1.i = 9;
    s2.i = 47;
    System.out.println("1: s1.i: " + s1.i +", s2.i: " + s2.i);
    s1 = s2; // referanslar kopyalaniyor.
    System.out.println("2: s1.i: " + s1.i +", s2.i: " + s2.i);
    s1.i = 27;
    System.out.println("3: s1.i: " + s1.i +", s2.i: " + s2.i);
  }
}
```

Program 2.1. Nesnelerde atama örneği

Yukarıda verilen uygulama şu şekilde çalışmaktadır. Önce 2 adet Sayi nesnesi oluşturulmaktadır. Bunlar Sayi tipindeki referanslara bağlı olan s1 ve s2'dir. Bu referanslar artık 2 ayrı Sayi nesnesini göstermektedirler. Daha sonra s1 referansının işaret ettiği Sayi nesnesinin i alanına 9 sayısı atanmaktadır. Benzer şekilde s2 referansının işaret ettiği Sayi nesnesinin i alanına da 47 sayısı atanmaktadır. Yapılan işlemlerin düzgün olup olmadıklarının görülebilmesi için ekrana yazdırıldığında aşağıdaki sonuç ile karşılaşılır.

1: s1.i: 9, s2.i: 47

2.6. Java Operatörleri

Operatörler programlama dillerinin en temel işlem yapma yeteneğine sahip simgesel isimlerdir. Tüm programlama dillerinde önemli bir yere sahip olup, bir işlem operatör ile gerçekleştirilebiliyorsa en hızlı ve verimli ancak bu şekilde yapılır denilebilir. Yalnızca bir operatör ile gerçekleştirilemeyen işlemler, ya bir grup operatörün bir araya getirilmesiyle ya da o işlemi gerçekleştirecek bir metot yazılmasıyla sağlanır. Java dili oldukça zengin ve esnek operatör kümesine sahiptir. Örneğin matematiksel, mantıksal, koşulsal, bit düzeyinde gibi birçok operatör kümesi vardır ve bunlar içerisinde çeşitli operatörler bulunmaktadır. Aşağıda operatör çeşitleri gösterilmiştir.

- Aritmetik Operatör
- İlişkisel Operatör
- Mantıksal Operatörler
- Bit düzeyinde Operatörler

Operatörler, genel olarak, üzerinde işlem yaptığı değişken/sabit sayısına göre tekli operatör veya ikili operatör olarak sınıflanmaktadır. 3 adet değişken/sabite ihtiyaç duyan operatörlere de üçlü operatör denilir. Tekli operatörler hem önek(prefix) hem de sonek(postfix) işlemlerini desteklerler. Önekten kastedilen anlam operatörün değişkenden önce gelmesi, sonek'te operatörden sonra gelmesidir.

→ operatör değişken //önek ifadesi

Sonek işlemlerine örnek olarak,

→ değişken operatör // sonek ifadesi

İkili operatörlerde operatör simgesi ara ek(infix) olarak iki değişkenin ortasında bulunur.

→ değişken1 operatör değişken2 //ara ek

Üçlü operatörlerde ara ek(infix) işlemlerde kullanılır. Java'da üçlü operatör bir tanedir.

→ değişken1 ? değişken2 **:** değişken3 //ara ek

2.6.1. Aritmetik operatörler

Java programlama dili kayan noktalı(floating-point) ve tamsayılar için birçok aritmetik işlemi destekleyen çeşitli operatörlere sahiptir. Bu işlemler toplama operatörü(+), çıkartma operatörü(-), çarpma operatörü(*****), bölme operatörü (/) ve son olarak da artık bölme(%) operatörüdür. Tablo 2.4'de aritmetik operatörler verilmiştir.

Tablo 2.4. Java'da aritmetik operatörler

Operatör	Kullanılış	Açıklama
+	değişken1 + değişken2	değişken1 ile değişken2'yi toplar
-	değişken1 - değişken2	değişken1 ile değişken2'yi çıkarır
*	değişken1 * değişken2	değişken1 ile değişken2'yi çarpar
/	değişken1 / değişken2	değişken1, değişken2 tarafından bölünür
%	değişken1 % değişken2	değişken1'in değişken2 tarafından bölümünden kalan hesaplanır.

Verilenler bir Java uygulamasında aşağıdaki gibi gösterilebilir.

```
 public class AritmetikOrnek
{
 public static void main(String[] args)
{  //  Değişkenler atanan değerler
    int a = 57, b = 42;
   double c = 27.475,  d = 7.22;
   System.out.println("Değişken Değerleri...");
   System.out.println(" a = " + a);
   System.out.println(" b = " + b);
```

```
        System.out.println(“ c = “ + c);
        System.out.println(“ d = “ + d);

        // Sayılar toplanıyor
        System.out.println(“Toplama...”);
        System.out.println(“ a + b = “ + (a + b));
        System.out.println(“ c + d = “ + (c + d));

        //  Sayılar çıkarılıyor
        System.out.println(“Çıkartma...”);
        System.out.println(“ a - b = “ + (a - b));
        System.out.println(“ c - d = “ + (c - d));

        //  Sayılar çarpılıyor
        System.out.println(“Çarpma...”);
        System.out.println(“ a * b = “ + (a * b));
        System.out.println(“ c * d = “ + (c * d));

        //  Sayılar bölünüyor
        System.out.println(“Bölme...”);
        System.out.println(“ a / b = “ + (a / b));
        System.out.println(“ c / d = “ + (c / d));

        //  Bölme işlemlerinden kalan sayı hesaplanıyor
        System.out.println(“Kalan sayıyı hesaplama...”);
        System.out.println(“ a % b = “ + (a % b));
        System.out.println(“ c % d = “ + (c % d));

        //  double ve int tipleri karışık şekilde kullanılıyor.
        System.out.println(“Karışık tipler...”);
        System.out.println(“ b + d = “ + (b + d));
        System.out.println(“ a * c = “ + (a * c));
    }
}
```

Program 2.2. Aritmetik işlemler

Uygulamanın sonucu aşağıdaki gibi olur.

Değişken Değerleri...
a = 57
b = 42
c = 27.475
d = 7.22
Toplama...
a + b = 99
c + d = 34.695
Çıkartma...
a - b = 15
c - d = 20.255000000000003
Çarpma...
a * b = 2394
c * d = 198.36950000000002
Bölme...
a / b = 1
c / d = 3.805401662049862
Kalan sayıyı hesaplama...
a % b = 15
c % d = 5.815000000000002
Karışık tipler...
b + d = 49.22
a * c = 1566.075

Verilen örnek dikkatlice incelenirse, tamsayı ile kayan noktalı sayılar bir operatörün değişkenleri olursa sonuç kayan noktalı sayı olmaktadır. Bu işlemde tamsayı, kendiliğinden kayan noktalı sayıya çevrilir. Tablo 2.5'te dönüştürme işleminde izlenen yol verilmiştir.

Tablo 2.5. Operatörlerin veri tipini etkilemesi

Sonuç Veri Tipi	Değişkenlerin Veri Tipleri
long	Değişkelerin float veya double tipinden farklı olması ve en az bir değişkenin long tipinde olması
int	Değişkelerin float veya double tipinden farklı olması ve değişkenlerin long tipinden farklı olması
double	En az bir değişkenin double tipinde olması
float	Değişkenlerin hiçbirinin double tipinde olmaması ve değişkenlerden en az birinin float tipinde olması

+ ve **–** operatörlerinin, aynı zamanda, karakter tipindeki verileri sayısal tipe dönüştürme görevleri de vardır. Tablo 2.6'da bu operatörler verilmiştir.

Tablo 2.6. Toplama ve Çıkartma operatörlerinin tip etkilemesi

Operatör	Kullanılış Şekli	Açıklama
+	+ değişken	Eğer değişken char, byte veya short tipinde ise int tipine dönüştürür
-	- değişken	Değişkenin değerini eksi yapar (-1 ile çarpar).

Aşağıda bu özelliğin daha rahat anlaşılabilmesi için örnek bir program bulunmaktadır.

```
public class OperatorTest
{
public static void main(String args[] )
{
   char kr = 'a' ;
   int b = +kr ;      // otomatik olarak int temel tipine çevrildi
```

```
    int c = -b ;       // değeri eksi yaptı
    System.out.println("kr = " + kr );
    System.out.println("b = " + b );
    System.out.println("c = " + c );
  }
}
```

Program 2.3. Operatör test örneği

Char temel(primitive) bir tiptir ve bu tiplere değer atanırken veri tek tırnak içerisinde verilmelidir. Bu örnekte girilen değer a harfidir. Daha sonra + operatörü kullanılarak char değerini int tipine dönüştürülüyor ve son olarak ta bu int değeri - operatörüyle eksi hale getiriliyor. Uygulamanın sonucu aşağıdaki gibi olur.

```
kr = a
b = 97
c = -97
```

2.6.2. İlişkisel operatörler

İlişkisel operatörler iki değeri karşılaştırarak bunlar arasındaki mantıksal ilişkiyi belirlemeye yararlar. Örneğin iki değer birbirine eşit değilse, == operatörüyle bu ilişki sonucu yanlış(false) olur. Tablo 2.7'de ilişkisel operatörler ve anlamları verilmiştir.

Tablo 2.7. İlişkisel operatörler

Operatör	Kullanılış Şekli	True değeri döner eğer ki.....
>	değişken1 > değişken2	değişken1, değişken2'den büyükse
>=	değişken1 >= değişken2	değişken1, değişken2'den büyükse veya eşitse
<	değişken1 < değişken2	değişken1, değişken2'den küçükse
<=	değişken1 <= değişken2	değişken1, değişken2'den küçükse veya eşitse
==	değişken1 == değişken2	değişken1, değişken2'ye eşitse
!=	değişken1 != değişken2	değişken1, değişken2'ye eşit değilse

Aşağıda ilişkisel operatörlerin kullanılması ile ilgili örnek bir program bulunmaktadır.

```
Public class IliskiselDeneme
  {
  public static void main(String[] args)
  {
   // değişken bildirimleri
   int i = 37, j = 42, k = 42;
   System.out.println("Değişken değerleri...");
   System.out.println(" i = " + i);
   System.out.println(" j = " + j);
   System.out.println(" k = " + k);
   //Büyüktür
   System.out.println("Büyüktür...");
   System.out.println(" i > j = " + (i > j));  //false - i, j  den küçüktür
   System.out.println(" j > i = " + (j > i));  //true - j, i den Büyüktür
   System.out.println(" k > j = " + (k > j));  //false - k, j ye eşit
   //Büyüktür veya eşittir
```

```
        System.out.println("Büyüktür veya eşittir...");
        System.out.println(" i >= j = " + (i >= j)); //false - i, j  den küçüktür
        System.out.println(" j >= i = " + (j >= i)); //true - j, i den büyüktür
        System.out.println(" k >= j = " + (k >= j)); //true - k, j'ye eşit
        //Küçüktür
        System.out.println("Küçüktür...");
        System.out.println(" i < j = " + (i < j));    //true - i, j'den küçüktür
        System.out.println(" j < i = " + (j < i));    //false - j, i' den büyüktür
        System.out.println(" k < j = " + (k < j));    //false - k, j'ye eşit
        //Küçüktür veya eşittir
        System.out.println("Küçüktür veya eşittir...");
        System.out.println(" i <= j = " + (i <= j)); //true - i, j'den küçüktür
        System.out.println(" j <= i = " + (j <= i)); //false - j, i  den büyüktür
        System.out.println(" k <= j = " + (k <= j)); //true - k, j  ye eşit
            //Eşittir
        System.out.println("Eşittir...");
        System.out.println("i == j =" + (i == j)); //false - i, j'den küçüktür
        System.out.println(" k == j = " + (k == j)); //true - k, j'ye eşit
        System.out.println("Eşit değil..."); //Eşit değil
        System.out.println(" i != j = " + (i != j)); //true - i, den küçüktür
        System.out.println(" k != j = " + (k != j)); //false - k, ye eşit
      }
    }
```

Program 2.4. İlişkisel operatörler

Uygulamanın sonucu aşağıdaki gibi olur.

Değişken değerleri...
i = 37
j = 42
k = 42
Büyüktür...

i > j = false

j > i = true

k > j = false

Büyüktür veya eşittir...

i >= j = false

j >= i = true

k >= j = true

Küçüktür...

i < j = true

j < i = false

k < j = false

Küçüktür veya eşittir...

i <= j = true

j <= i = false

k <= j = true

Equal to...

i == j = false

k == j = true

Not equal to...

i != j = true

k != j = false

2.6.3. Mantıksal operatörler

Mantıksal operatörler birden çok karşılaştırma işlemini birleştirip tek bir koşul ifadesi haline getirilmesi için kullanılır. Örneğin bir koşul sağlanması için hem a'nın 10'dan büyük olması hem de b'nin 55 küçük olması gerekiyorsa iki karşılaştırma koşulu VE mantıksal operatörüyle birleştirilip a>10 && b<55 şeklinde yazılabilir. Aşağıda, Tablo 2.8'de mantıksal operatörlerin listesi ve anlamları verilmiştir.

Tablo 2.8. Mantıksal operatörlerin kullanımı

Operatör	Kullanılış Şekli	İşlevi/Anlamı
&&	değişken1&& değişken2	VE operatörü
||	değişken1|| değişken2	VEYA operatörü
^	değişken1 ^ değişken2	YA DA operatörü
!	! değişken	DEĞİL'ini alma operatörü

Aşağıda birden çok ilişkisel karşılaştırmanın mantıksal operatörler birleştirilmesi için örnekler verilmiştir. Buradaki m,r,z,a,b isimleri değişkenlerdir.

m>10 && m<55
(m>0 && r<55) && z==10
a>10 && b<55 || r<99

Not: Mantıksal operatörlerden && (VE), || (VEYA) operatörleri sırasıyla tek karakterli olarak ta kullanılabilmektedir. Örneğin a&&b şeklinde bir ifade a&b şeklinde de yazılabilir. Aralarında fark, eğer tek karakterli, yani & veya | şeklinde ise, operatörün her iki yanındaki işlemler/karşılaştırmalar kesinlikle yapılır. Ancak, çift karakterli kullanılırsa, yani && veya || şeklinde ise, işleme soldan başlanır; eğer tüm ifade bitmeden kesin sonuca ulaşılırsa ifadenin geri kalan kısmı göz ardı edilir. Örneğin VE işleminde sol taraf yanlış(false) ise sonuç kesin yanlış olacaktır ve ifadenin sağına bakmaya gerek yoktur. Aşağıda mantıksal operatörlerle ilgili örnek bir program bulunmaktadır.

```
public class KosulOp
{
public static void main( String args[] )
 {
  int a = 2 ;
  int b = 3 ;
  int c = 6 ;
```

```
    int d = 1 ;
    /*  (a < b) = bu ifadenin doğru (true) olduğu bilinmektedir
     (c < d) = bu ifadenin yanlış (false) olduğu bilinmektedir */
    System.out.println(“ (a<b)&&(c<d) --> “ + ((a<b)&&(c<d)) );
    System.out.println(“ (a<b)||(c<d) --> “ + ((a<b)||(c<d)) );
    System.out.println(“ ! (a<b) --> “ + ( ! (a<b)) );
    System.out.println(“ (a<b)&(c<d) --> “ + ((a<b)&(c<d)) );
    System.out.println(“ (a<b)|(c<d) --> “ + ((a<b)|(c<d)) );
    System.out.println(“ (a<b)^(c<d) --> “ + ((a<b)^(c<d)) );
  }
}
```

Program 2.5. Mantıksal operatörler

Uygulamanın çıktısı aşağıdaki gibidir.

```
(a < b) && (c < d) --> false
(a < b) || (c < d) --> true
! (a < b) --> false
(a < b) & (c < d) --> false
(a < b) | (c < d) --> true
(a < b) ^ (c < d) --> true
```

2.6.4. Bit düzeyinde(Bitwise) operatörler

Bit düzeyinde operatörler, değişkenlerin/sabitlerin tuttuğu değerlerin doğrudan ikili kodlarının bitleri koşuluyla bir sola kaydırılırsa 1100, tümleyeni alınırsa 1001 olur. Görüldüğü gibi bit düzeyinde operatörler veriyi bit düzeyde etkilemektedir. Bit düzeyinde operatörlerin listesi Tablo 2.9’da, doğruluk tabloları da sırasıyla tablo 2.10, tablo 2.11 ve üzerinde işlem yaparlar. Örneğin 6 sayısının iki karşılığı 0110’dır. Bu değer sonuç 4 bit üzerinde kalmak tablo 2.12’de gösterilmiştir.

Tablo 2.9. Bit düzeyinde operatörler

Operatör	Kullanılış Şekli	Açıklama
&	değişken1 & değişken2	bit düzeyinde VE
|	değişken1 | değişken2	bit düzeyinde VEYA
^	değişken1 ^ değişken2	bit düzeyinde YA DA
~	~değişken	bit düzeyinde tümleme
>>	değişken1 >> değişken2	bit düzeyinde sağa öteleme
<<	değişken1 << değişken2	bit düzeyinde sağa öteleme
>>>	değişken1 >>> değişken2	bit düzeyinde sağa öteleme

2.6.4.1. Ve(and) operatörü

VE işleminde her iki taraf ta doğru(true) ise sonuç doğru diğer durumlarda sonuç yanlış(false) olur. VE işlemi doğruluk tablosu Tablo 2.10'da verilmiştir. Bit düzeyinde VE işleminde operatörün hemen sağ ve sol yanında bulunan parametrelerin ikili karşılıkları bit bit VE işlemine sokulur. İşlem en sağdaki bitten başlatılır. Örneğin 10 ve 9 sayılarının ikili karşılıkları sırasıyla 1010 ve 1011'dir. Her ikisi bit düzeyinde VE işlemine sokulursa, sonuç 1000 çıkar. Aşağıda bu örneğin çalışması görünmektedir.

```
  1010   ⇨ 10
& 1001   ⇨ 9
---------
  1000
```

Tablo 2.10. VE(AND) işlemi doğruluk tablosu

değişken1	değişken2	Sonuç
0	0	0
0	1	0
1	0	0
1	1	1

2.6.4.2. Veya(or) operatörü

VEYA işleminde her iki taraftan birinin doğru(true) olması sonucun doğru çıkması için yeterlidir. VEYA işlemi doğruluk tablosu Tablo 2.11'de verilmiştir. Bit düzeyinde VEYA işleminde operatörün hemen sağ ve sol yanında bulunan parametrelerin ikili karşılıkları bit bit VEYA işlemine sokulur. İşlem en sağdaki bitten başlatılır. Örneğin 10 ve 9 sayılarının ikili karşılıkları sırasıyla 1010 ve 1011'dir. Her ikisi bit düzeyinde VE işlemine sokulursa, sonuç 1011 çıkar. Aşağıda bu örneğin çalışması görünmektedir.

1010 ⇨ 10
1001 ⇨ 9
--------- 1011

Tablo 2.11. VEYA(OR) işlemi doğruluk tablosu

değişken1	değişken2	Sonuç
0	0	0
0	1	1
1	0	1
1	1	1

2.6.4.3. Yada(Exclusive or) operatörü

YADA işleminde her iki taraftan yalnızca birinin doğru(true) olması sonucu doğru yapar. Her iki tarafın aynı olması durumunda sonuç yanlış(false) çıkar. YADA işleminin doğruluk tablosu Tablo 2.12'de verilmiştir.

Tablo 2.12. Dışlayan YADA işlemi doğruluk tablosu

Değişken1	değişken2	Sonuç
0	0	0
0	1	1
1	0	1
1	1	0

VE ve VEYA işlemlerinde kullanılan örnek sayıları YADA için de gösterilirse, sonuç aşağıdaki gibi olur.

```
  1010    ⇨  10
^ 1001    ⇨  9
---------
  0011
```

2.6.4.4. Tümleme(Not) operatörü

a bir değişken adı ise ~a ifadesi açılımı : ~a = (-a) - 1, yani, ~10= (-10) -1 = -11 sonucunu verir. Aşağıda tümleme operatörü ile ilgili bir örnek bulunmaktadır.

```
public class BitwiseOrnek2 {
public static void main( String args[] )
 {
 int a = 10, b = 9, c = 8 ;
```

```
    System.out.println(“ (a & b) --> “ + (a & b) );
    System.out.println(“ (a | b) --> “ + (a | b ) );
    System.out.println(“ (a ^ b) --> “ + (a ^ b) );
     System.out.println(“ ( ~a ) --> “ + ( ~a ) );
    System.out.println(“ ( ~b ) --> “ + ( ~b ) );
    System.out.println(“ ( ~c ) --> “ + ( ~c ) );
  }
}
```

Program 2.6. Bitwise örneği

Uygulamanın sonucu aşağıdaki gibi olur.

(a & b) --> 8
(a | b) --> 11
(a ^ b) --> 3
(~a) --> -11
(~b) --> -10
(~c) --> -9

VE, VEYA ve YADA(Exclusive Or) operatörleri birden çok mantıksal sonuç içeren ifadelerde kullanılabilir.

2.6.5. Öteleme(Shift) operatörleri

Bit düzeyinde işlem yapan bir grup operatörün adı öteleme operatörleri olarak adlandırılırlar. Bunlar >>, >> ve >>> simgeleriyle gösterilmektedir. Öteleme operatörleri veri üzerindeki bitlerin sağa veya sola kaydırılması için kullanılır. Aşağıdaki örnekte bu operatörlerin Java uygulamalarında nasıl kullanılacakları görülebilir.

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

```
    int a = 9 ;
    System.out.println(“ (a >> 1) -->” + (a >> 1) );
    System.out.println(“ (a >> 2) -->” + (a >> 2) );
    System.out.println(“ (a << 1) -->” + (a << 1) );
    System.out.println(“ (a << 2) -->” + (a << 2) );
    System.out.println(“ (a >>> 2) -->” + (a >>> 2) );
  }
 }
```

Program 2.7. Öteleme operatörleri örneği

Verilen örnekte a değişkenine 9 sayısı atanmıştır; bu sayının ikili karşılığı aşağıdaki gibi bulunur.

$$9 = (1001)_2 = 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0$$

Yani, 9_{10} sayısının ikili tabandaki karşılığı 1001 olmaktadır. Buna göre a değişkeni üzerinde öteleme operatörünün etkisi aşağıda açıklandığı gibi olur.

(a >> 1) şeklinde ifade ile, 9 sayısının ikilik karşılığı olan 1001 bitleri sağa doğru 1 basamak kaydırılır. Boşalan yere 0 yerleştirildiğinde sonuç elde edilir; dolayısıyla 0100 elde edilir ve bunun ondalık karşılığı 4 çıkar.

(a >> 2) şeklinde ifade ile 9 sayısının ikilik karşılığı olan 1001 bitlerini sağa doğru 2 basamak kaydırılır; __10, boşalan yerlere 0 yerleştirildiğinde sonuç elde edilir. Dolayısıyla 0010 elde edilir ve bunun ondalık karşılığı 2 çıkar.

(a << 1) şeklinde ifade ile 9 sayısının ikilik karşılığı olan 1001 bitlerini sola doğru 1 basamak kaydırılır; 1001_, boşalan yere 0 yerleştirildiğinde sonuç elde edilir. Dolayısıyla 10010 elde edilir ve bunun ondalık karşılığı 18 çıkar.

(a << 2) şeklinde ifade ile 9 sayısının ikilik karşılığı olan 1001 bitleri sola doğru 2 basamak kaydırılır. 1001_ _, boşalan yerlere 0 yerleştirildiğinde sonuç elde edilir. Dolayısıyla 100100 elde edilir ve bunun ondalık karşılığı 36 çıkar.
(a >>> 2) şeklinde verilen ifadenin (a >> 2) ile arasında sonuç olarak bir fark yoktur. Yine 2 elde edilir. Arasındaki fark ">>>" operatörü işaretsiz (unsigned) sağa doğru kaydırma yapar.

Eğer char, byte, veya short tiplerinde kaydırma işlemi yapılacaksa bu tipler ilk önce int tipine dönüştürülürler. Eğer long tipinde kaydırma işlemi yapılıyorsa o zaman yine long tipinde bir sonuç elde edilir.

Uygulamanın sonucu aşağıdaki gibi olur.

(a >> 1) -->4
(a >> 2) -->2
(a << 1) -->18
(a << 2) -->36
(a >>> 2) -->2

2.6.6. Atama operatörleri

Atama operatörü en temel operatördür. Atama işlemi, bir değeri veya değişkenin içeriğini bir başka değişkene yerleştirmektir. Hemen hem tüm programlama dillerinde atama operatörü olarak = simgesi kullanılır. Yalnızca Pascal ve benzeri dillerde := karakter çifti kullanılır. Aşağıda atama operatörleri ile ilgili bir örnek program bulunmaktadır.

```
public class EnBuyukSayilar {
public static void ekranaBas(String deger) {
System.out.println(deger);
}
public static void main( String args[] ) {
// tamsayılar
```

```
byte enbuyukByte = Byte.MAX_VALUE;
short enbuyukShort = Short.MAX_VALUE;
int enbuyukInteger = Integer.MAX_VALUE;
long enbuyukLong = Long.MAX_VALUE;
ekranaBas("enbuyukByte-->" + enbuyukByte );
ekranaBas("enbuyukShort-->" + enbuyukShort );
ekranaBas("enbuyukInteger-->" + enbuyukInteger );
ekranaBas("enbuyukLong-->" + enbuyukLong );
ekranaBas("");
// gerçek sayılar
float enbuyukFloat = Float.MAX_VALUE;
double enbuyukDouble = Double.MAX_VALUE;
ekranaBas("enbuyukFloat-->" + enbuyukFloat );
ekranaBas("enbuyukDouble-->" + enbuyukDouble );
ekranaBas("");
// diğer temel (primitive) tipler
char birChar = 'S';
boolean birBoolean = true;
ekranaBas("birChar-->" + birChar );
ekranaBas("birBoolean-->" + birBoolean );
 }
}
```

Program 2.8. Değişkenlere değer atama

Java'da C dilinde olduğu gibi bitişik atama operatörleri de vardır. Bunlar atama operatörüyle diğer operatörlerden birinin birleştirilmesinden oluşurlar. Böylece kısa bir yazılımla hem aritmetik, öteleme gibi işlemler yaptırılır hem de atama yapılır. Örneğin, int tipinde olan toplam değişkeninin değeri 1 arttırmak için aşağıda gibi bir ifade kullanılabilir.

```
toplam = toplam + 1;
```

Bu ifade bitişik atama operatörüyle aşağıdaki gibi yazılabilir. Görüldüğü gibi değişken adı yukarıdaki yazımda 2, aşağıda yazımda ise 1 kez yazılmıştır.[5]

toplam += 1;

Tablo 2.13'de bitişik atama operatörlerinin listesi verilmiştir. Bu operatör, özellikle, uzun değişken kullanıldığı durumlarda yazım kolaylığı sağlar.

Tablo 2.13. Java'daki bitişik atama operatörleri

Operatör	Kullanılış Şekli	Eşittir
+=	değişken1 += değişken2	değişken1 = değişken1 + değişken2
-=	değişken1-= değişken2	değişken1 = değişken1 – değişken2
*=	değişken1*= değişken2	değişken1 = değişken1 * değişken2
/=	değişken1/= değişken2	değişken1 = değişken1 / değişken2
%=	değişken1%= değişken2	değişken1 = değişken1 % değişken2
&=	değişken1&= değişken2	değişken1 = değişken1 & değişken2
|=	değişken1|= değişken2	değişken1 = değişken1 | değişken2
^=	değişken1^= değişken2	değişken1 = değişken1 ^ değişken2
<<=	değişken1<<= değişken2	değişken1 = değişken1 << değişken2
>>=	değişken1>>= değişken2	değişken1 = değişken1 >> değişken2
>>>=	değişken1>>>= değişken2	değişken1 = değişken1 >>> değişken2

2.6.7. String operatörü

"+" operatörü String verilerde birleştirme görevi görür. Eğer ifade String ile başlarsa, onu izleyen veri tipleri de kendiliğinden String'e dönüştürülür. Aşağıda string operatörü ile ilgili örnek bir program bulunmaktadır.

```
 public class OtomatikCevirim
{
 public static void main(String args[ ])
{
  int x = 0, y = 1, z = 2;
```

```
    System.out.println(“Sonuç =” + x + y + z);
  }
}
```

Program 2.9. Otomatik string’e çevirme

Uygulamanın sonucu aşağıdaki gibi olur.

Sonuç =012

Görüldüğü gibi String bir ifadeden sonra gelen tamsayılar toplanmadı. Doğrudan String nesnesine çevrilip ekrana çıktı olarak gönderildiler.[13]

2.7. Dönüştürme(Casting) İşlemi

Temel bir veri tipi diğer bir temel tipe dönüştürebilir; fakat oluşacak değer kayıplarından programcı sorumlu olacaktır. Aşağıda değişken dönüştürme ile ilgili örnek bir program bulunmaktadır.

```
public class IlkelDonusum {
public static void main(String args[]) {
  int a = 5;
  double b = (double) a;
  double x = 4.15;
  int y = (int) x;
  long z = (long) y;
  System.out.println(“b = “ + b + “ y = “ + y + “ z = “ + z);
  }
}
```

Program 2.10. Değişken dönüştürme

Uygulamanın sonucu aşağıdaki gibi olur.

b = 5.0 y = 4 z = 4

2.8. Bir Artırma ve Azaltma

Java dilinde, aynı C dilinde olduğu gibi, birçok kısaltmalar vardır. En çok kullanılan kısaltmalardan iki tanesi arttırma ve azaltma operatörleridir; bu operatörler değişkenin içeriğini bir arttırmak veya azaltmak için kullanılır.

Arttırma ve azaltma operatörleri iki farklı konumda kullanılabilirler. Birincisi önek'tir ve bu -- veya ++ operatörünün kullanılan değişkenin önüne gelmesi anlamını taşır. Diğeri ise sonek'tir. Bu ekte -- veya ++ operatörünün değişkenin sonuna gelmesi anlamına gelir. Aşağıda bu operatörleri gösteren tablo bulunmaktadır. [13]

Tablo 2.14. Arttırma ve azaltma

Operatör	Kullanılış Şekli	Açıklama
++	değişken++	Önce değişkenin değerini hesaplar sonra değişkenin değerini bir arttırır.
++	++değişken	Önce değişkenin değerini arttırır sonra değişkenin değerini hesaplar.
--	değişken--	Önce değişkenin değerini hesaplar sonra değişkenin değerini bir azaltır.
--	--değişken	Önce değişkenin değerini azaltır sonra değişkenin değerini hesaplar.

Örneğin ++a veya --a şeklinde verilmesinde, önce matematiksel toplama/çıkartma işlemi gerçekleşir; daha sonra değer üretilir. Ancak, a++ veya a-- şeklinde verilmesi durumunda ise, önce değer üretilir, daha sonra matematiksel toplama/çıkartma işlemi gerçekleşir. Aşağıda verilen kısa program bunu güzel bir şekilde ifade etmektedir.

```
 public class OtomatikArtveAz
{
static void ekranaYaz(String s)
{
  System.out.println(s);
 }
 public static void main(String[] args) {
  int i = 1;
  ekranaYaz("i : " + i);
  ekranaYaz("++i : " + ++i);     // önek artırım
  ekranaYaz("i++ : " + i++);     // sonek artırım
  ekranaYaz("i : " + i);
  ekranaYaz("--i : " + --i);     // önek azaltma
  ekranaYaz("i-- : " + i--);     // sonek azaltma
  ekranaYaz("i : " + i);
 }
}
```

Program 2.11. Otomatik artırma örneği

Uygulamanın sonucu aşağıdaki gibi olur.

```
i : 1
++i : 2
i++ : 2
i : 3
--i : 2
i-- : 2
i : 1
```

2.9. Nesnelerin Karşılaştırılması

Nesnelerin eşit olup olmadığı = = veya ! = operatörleriyle belirlenebilir. Aşağıda nesnelerin karşılaştırması ile ilgili bir örnek bulunmaktadır.

```
public class Denklik {
public static void main(String[ ] args)   {
  Integer a1 = new Integer(47);
  Integer a2 = new Integer(47);
  System.out.println(a1 == a2);
  System.out.println(a1 != a2);   }
}
```

Program 2.12. Nesnelerin karşılaştırılması örneği_1

Önce Integer sınıfı tipinde olan n1 ve n2 referansları, içlerinde 47 sayısını tutan Integer nesnelerine bağlı durumdadırlar. Uygulamanın sonucu aşağıdaki gibi olacaktır.

```
True
False
```

Ancak ne yazık ki, sonuç yukarıdaki gibi değildir! Nedeni ise, elimizde iki adet farklı Integer nesnesi bulunmaktadır. Bu nesnelerin taşıdıkları değerler birbirlerine eşittir; ancak, a1==a2 ifadesinin kullanılması şu anlama gelir. “a1 ve a2 referanslarının işaret etmiş oldukları nesneler aynı mı?”. Yanıt tahmin edilebileceği gibi hayırdır. Yani, false’dur. a1 ve a2 ayrı Integer nesnelerini işaret etmektedirler. Eşit olan tek şey, bu iki ayrı nesnenin tuttukları değerin 47 olmasıdır. Programın çıktısı aşağıdaki gibidir.

```
False
True
```

Verilen örnekteki Integer nesneleri yerine temel tip olan int tipi kullanılsaydı sonuç farklı olurdu. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
public class IntIcinDenklik {
public static void main(String[] args) {
  int s1 = 47;
  int s2 = 47;
  System.out.println(s1 == s2);
  System.out.println(s1 != s2);
 }
}
```

Program 2.13 Nesnelerin karşılaştırılması örneği_2

Bu uygulamanın sonucu aşağıdaki gibi olur.

True
False

Temel(primitive) tipler değerleri doğrudan kendi üzerlerinde taşıdıkları için == operatörüyle s1 ve s2 değişkeleri değerleri karşılaştırıldı ve doğru(true) yanıtı döndürüldü. Benzer şekilde != operatörü de s1 ve s2 değişkenleri değerlerini karşılaştırdı ve yanlış(false) döndürüldü. [13]

BÖLÜM 3. KONTROL İFADELERİ

3.1. Karşılaştırma Deyimleri

Kullanıcıdan alınan girdilere göre, bazı işlemlerin daha önceden belirlenmiş şartlara göre yapılması istenebilir. Örnek olarak gelire göre vergi hesaplayan uygulama yazılımı olabilir.

- 0–6 bin YTL arası gelir diliminden %15
- 6–12 bin YTL arası gelir diliminden %20
- 12–18 bin YTL arası gelir diliminden %15
- 18 bin YTL üzerindeki gelir diliminden %20

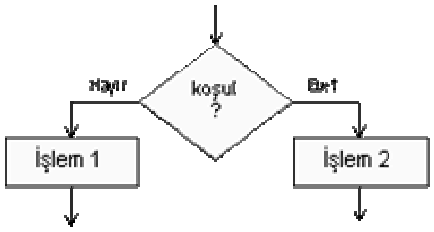
oranında vergi alınacaktır. Bu durumda yazılan uygulama, kullanıcıdan o yıl için vergilendirilecek geliri soracak ve daha sonra ne kadar vergi ödemesi gerektiğini ekranda gösterecektir. İşte bu tür bir uygulamanın yapılabilmesi için bazı yapıların kullanılması gerekir. Bu yapılara kontrol deyimleri denir. Kontrol deyimleri kullanmadan bir program yazmak hiçbir programlama dilinde mümkün olmadığı gibi Java dilinde de mümkün değildir. Aşağıda Java dilinde kullanılan kontrol ifadeleri bulunmaktadır.

3.1.1. If yapısı

Programcılar program yazarken çoğunlukla şart yapılarını kullanılırlar. Bunların içerisinde ise en yaygın olanı da if yapısıdır. If yapısının genel yazılımı şu şekildedir.

```
if (koşul) {
 koşul doğru(true) olduğunda çalışması istenen kod bloğu
} else
{  Koşul yanlış(false) olduğunda çalışması istenen kod bloğu }
```

Şekil 3.1'de if yapısının algoritma mantığı görünmektedir.



Şekil 3.1. If yapısının algoritması [13]

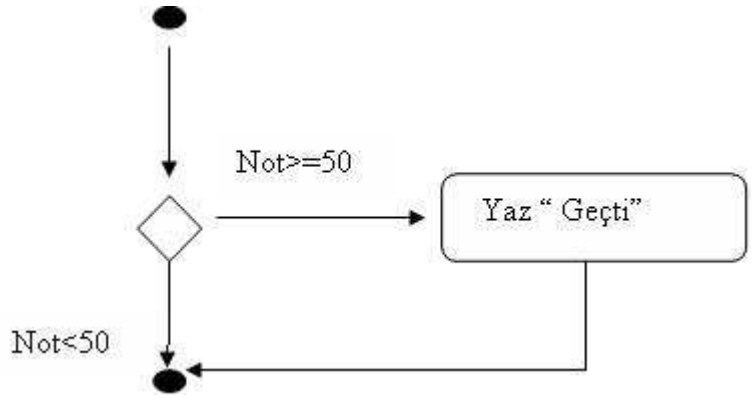
Programlar, birden fazla şart içerisinden bazılarını seçmek için seçme yapıları kullanırlar. Bu yapı bir örnekle açıklanabilir. Bir öğrencinin sınav geçme notu 50 olarak düşünülürse aşağıdaki mantığı kullanmak gerekir.

Eğer öğrencinin notu 50'e eşit veya fazlaysa
Yaz "Geçti"

Yukarıdaki ifade öğrencinin notunun 50 veya üstü olup olmadığını belirlemek için yazılmış bir ifadedir. Bu ifadede eğer sonuç true(doğru) ise "Geçti" ifadesini ekrana yazacaktır. Sonuç false(yanlış) ise, yaz ifadesinin bulunduğu satır işleme konulmayacak ve bundan sonraki var olan satırlar işlenecektir. Yukarıdaki ifadenin Java dilinde yazımı aşağıdaki gibidir.

```
If (ogrencinot>=50)
System.out.println("Geçti");
```

Aşağıdaki şekilde bu basit örneğin tek if yapısı gösterilmektedir. If yapısı tek giriş ve tek çıkışlı bir kontrol yapısıdır. Şekil 3.2'de if yapısı bazı geometrik şekillerle ayrıntılı olarak ifade edilmektedir. Burada if yapısının kullanımı daha rahat bir şekilde anlaşılmaktadır.



Şekil 3.2. If yapısı [13]

3.1.2. If-else yapısı

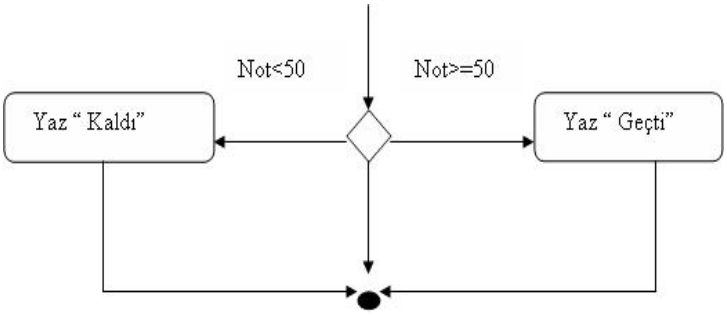
Tek if yapısında sonuç true(doğru) ise komutlar işlenir, false(yanlış) ise komutlar işleme sokulmaz program diğer satırları işleme sokar. If..Else yapısında ise sonuç true(doğru) ise komutlar işlenir, false(yanlış) ise else ifadesinden sonraki komutlar işlenir. Aşağıda bununla ilgili bir örnek bulunmaktadır. [13]

Eğer öğrencinin notu 50'e eşit veya fazlaysa
Yaz "Geçti"
Değilse
Yaz "Kaldı"

Yukarıdaki örnekte öğrencinin notu 50 ve üstüyse ekrana "Geçti", 50'nin altında ise ekrana "Kaldı" ifadesini yazacaktır. Yukarıdaki ifadenin Java dilindeki karşılığı ise şu şekildedir.

```
If (ogrencinot>=50)
System.out.println("Geçti");
Else System.out.println("Kaldı");
```

Şekil 3.3'de If…Else yapısı görünmektedir.



Şekil 3.3. If-Else yapısı [13]

Aşağıda if-else yapısı ile ilgili bir örnek bulunmaktadır.

```
public class IfElseTest {
public static void main(String[ ] args) {
  int puan = 76;
  char sonuc;
  if (puan >= 90)
{
    sonuc = 'A';
  } else if (puan >= 80)
{
    sonuc = 'B';
  } else if (puan >= 70)
{
    sonuc = 'C';
  } else if (puan >= 60) {
    sonuc = 'D';
  } else {
    sonuc = 'F';
  }
  System.out.println("Sonuc = " + sonuc);
 }
}
```

Program 3.1. if-Else örneği_1

Yukarıdaki örnekte else if yapısı kullanılmıştır. Bu programda puan değişkenine tamsayı 76 sayısı yüklenmiştir. Programın diğer satırlarında puan değişkenindeki sayı ile ilgili if yapıları bulunmaktadır. Bu programda puan 90 ve üstüyse sonuç değişkenine A karakteri yüklenecektir. Puan 80 ve üstüyse sonuç değişkenine B karakteri yüklenecektir. Puan 70 ve üstüyse sonuç değişkenine C karakteri yüklenecektir. Puan değişkeninde 76 sayısı olduğundan sonuç değişkenine C karakteri yüklenecektir. Diğer satırlar şart sağlanmadığından dolayı işleme konulmayacaktır. Aşağıda if-else yapısı ile ilgili örnek bir program bulunmaktadır.

```
public class Kestirme {
public static boolean hesaplaBir(int a) {
 System.out.println("hesaplaBir sınıfına girildi");
 return a > 1;
        }
 public static boolean hesaplaIki(int a) {
 System.out.println("hesaplaIki sınıfına girildi");
 return a > 2;
        }
  public static void main(String[] args) {
 System.out.println("Başlangıç");
 //hesaplaBir(0) --> false deger doner
 //hesaplaIki(3) --> true deger doner
 System.out.println("hesaplaBir(0) && hesaplaIki(3)");
 if ( hesaplaBir(0) && hesaplaIki(3) ) {
  System.out.println(" 1 -true ");
 } else {
  System.out.println(" 1 -false ");
 }
 System.out.println("--------");
 System.out.println("hesaplaBir(0) || hesaplaIki(3)");
 if (hesaplaBir(0) || hesaplaIki(3)) {
  System.out.println(" 2 -true ");
 } else {
```

```
 System.out.println(" 2 -false ");       }
System.out.println("--------");
System.out.println("hesaplaBir(0) & hesaplaIki(3)");
if (hesaplaBir(0) & hesaplaIki(3)) {
  System.out.println(" 3 -true ");
} else {
  System.out.println(" 3 -false ");      }
System.out.println("--------");
System.out.println("hesaplaBir(0) | hesaplaIki(3)");
if (hesaplaBir(0) | hesaplaIki(3)) {
  System.out.println(" 4 -true ");       }
else {
  System.out.println(" 4 -false ");      }
System.out.println("--------");
System.out.println("hesaplaBir(0) ^ hesaplaIki(3)");
if (hesaplaBir(0) ^ hesaplaIki(3)) {
  System.out.println(" 5 -true ");
} else {
  System.out.println(" 5 -true ");
}
System.out.println("Son..");
 }
}
```

Program 3.2. if-Else örneği_2

Bu uygulamada hesaplaBir() ve hesaplaiki() adında iki adet sınıf bulunmaktadır. Bunlar int tipinde parametre kabul edip mantıksal sonuç döndürmektedirler. Bu sınıflara girildiği zaman ekrana kendilerini tanıtan mesaj çıkmaktadır.

HesaplaBir() sınıfı kendisine gelen int tipindeki parametreyi alıp 1'den büyük veya küçük olduğunu kontrol etmektedir. Burada hesaplaBir() sınıfına parametre olarak sıfır sayısı gönderildiğinden dönecek değerin olumsuz olacağı bilinmektedir.

HesaplaIki() sınıfına da aynı şekilde 3 sayısı gönderilerek bunun olumlu bir yanıt vermesi beklenir.

Öncelikle, sınıflardan geri dönen değerler ve işlemine tabii tutulur. Burada yalnızca hesaplaBir() sınıfına girilmektedir. Çünkü hesaplaBir() sınıfından olumsuz değer dönmektedir. Ve işleminde olumlu değer dönebilmesi için iki değerinde olumlu olması gerektiğinden, hesaplaIki() sınıfı çağrılmayarak kestirme özelliği kullanılmıştır.

Daha sonra gelen değerler veya işlemine tabii tutulmaktadır. Hem hesaplaBir() hem de hesaplaIki() sınıfları beraber çağrılmaktadır. Veya işlemine göre, sonucun olumsuz olması için iki değerinde olumsuz olması gerekmektedir. Burada ilk değerden olumsuz değer dönmüştür; ancak, ikinci değerin de hesaplanması gerekmektedir. Aksi durumda sonucun öğrenilmesi imkansız olur. Bu nedenden dolayı, burada kestirme işlemi gerçekleşmemiştir. Ancak, ilk değer olumlu dönseydi, o zaman, ikinci sınıf olan hesaplaIki() hiç çağrılmayacaktı. Çünkü veya işlemleri sonucunun olumlu olabilmesi için parametrelerden birisinin olumlu olması gereklidir.

Bir sonraki aşamada, değerler yine ve işlemine tabii tutulmaktadır. Ancak, burada & operatörü kullanıldığı için kestirme işlemi ortadan kalkmaktadır.

Bir sonraki aşamada, değerler veya işlemine tutulmaktadır. Son olarak, değerler ya da (Exclusive Or) işlemine tutulmaktadır. Bu işlemde kesinlikle iki değere de bakılma zorunluluğu olduğundan kestirme işlemi söz konusu olmaz.

Uygulamanın sonucu aşağıdaki gibi olur.

```
Başlangıç
hesaplaBir(0) && hesaplaIki(3)
hesaplaBir sınıfına girildi
1 -false
--------
```

```
hesaplaBir(0) || hesaplaIki(3)
hesaplaBir sınıfına girildi
hesaplaIki sınıfına girildi
2 -true
--------
hesaplaBir(0) & hesaplaIki(3)
hesaplaBir sınıfına girildi
hesaplaIki sınıfına girildi
3 -false
--------
hesaplaBir(0) | hesaplaIki(3)
hesaplaBir sınıfına girildi
hesaplaIki sınıfına girildi
4 -true
--------
hesaplaBir(0) ^ hesaplaIki(3)
hesaplaBir sınıfına girildi
hesaplaIki sınıfına girildi
5 -true
Son..
```

Aşağıda if yapısı ile ilgili başka bir örnek program bulunmaktadır.

```
import java.io.*; // giris çikis
class ifyapisi
{
    public static void main (String args[ ]) throws IOException
    {
    double not;
        System.out.println("Öğrencinin notunu giriniz : ");
      if( not >= 90)
        { System.out.println("A"); }
      else if(not >=75)
```

```
            { System.out.println("B"); }
        else if(not >=60)
            { System.out.println("C"); }
        else if(not >=50)
            { System.out.println("D"); }
        else if(not >=40)
            { System.out.println("E"); }
        else
            { System.out.println("F"); }
    }
}
```

Program 3.3. if-Else Örneği_3

Yukarıdaki örnekte girilen notun harf olarak karşılığı ekrana yazdırılmaktır. System.out.println komutuyla kullanıcıya not girmesi gerektiği bildirilmektedir. Girilen bu değer double tipindeki not değişkenine aktarılmaktadır. Program, kullanıcı 90 veya üstü bir not girmiş ise ekrana A, 75 veya üstü bir not girmiş ise ekrana B, 60 veya üstü bir not girmiş ise ekrana C, 50 veya üstü bir not girmiş ise ekrana D, 40 veya üstü bir not girmiş ise ekrana E, bunların dışında bir not girmiş ise ekrana F harfini yazdırmaktadır.

3.1.3. Switch ifadesi

Tüm dillerdeki genel programlama deneyimi bir değişkendeki değeri başka değerlerle test etmektir ve bunu değişik değerlerle karşılaştırmaktadır. Bazen If…Else yapısını kullanmak elverişli olmayabilir.

Switch deyimi tamsayıların karşılaştırılması ile doğru koşulların elde edilmesini sağlayan mekanizmadır. Switch deyiminin genel yazım biçimi aşağıdaki gibidir.

switch(tamsayı) {
Case uygun-tamsayı-deger1 : çalışması istenen kod bloğu; break;

Case uygun-tamsayı-deger2 : çalışması istenen kod bloğu; break;
Case uygun-tamsayı-deger3 : çalışması istenen kod bloğu; break;
Case uygun-tamsayı-deger4 : çalışması istenen kod bloğu; break;
Case uygun-tamsayı-deger5 : çalışması istenen kod bloğu; break;
default: çalışması istenen kod bloğu ;
}

Switch deyimi içerisindeki tamsayı ile bu tamsayıya karşılık gelen koşul girilir ve istenen kod bloğu çalıştırılır. Kod bloklarından sonra break ifadesini koymak gerekir. Aksi takdirde uygun koşul bulunduktan sonraki her koşula girilecektir. Eğer tamsayı koşullardan hiçbirine uymuyorsa default yapısındaki kod bloğu çalıştırılarak program bloğu tamamlanır.

Switch deyimi birçok case yapılarından ve isteğe bağlı default ifadesinden meydana gelir. Bu yüzden default yapısının kullanılmasına gerek olmayabilir. Default yapısı kullanılmak istenmiyorsa, yani şartlardan hiçbiri sağlanmadığı zaman switch yapısından çıkılması isteniyorsa

Default: /* do nothing */

İfadesi yazılması daha iyi olacaktır. Aşağıda switch yapısı ile ilgili örnek bir program bulunmaktadır.

```
 public class AylarSwitchTestNoBreak
{
 public static void main(String[] args)
{
   int ay = 8;
   switch (ay)
        {
        case 1: System.out.println("Ocak"); break;
        case 2: System.out.println("Şubat"); break;
        case 3: System.out.println("Mart"); break;
```

```
            case 4: System.out.println("Nisan"); break;
            case 5: System.out.println("Mayıs"); break;
            case 6: System.out.println("Haziran"); break;
            case 7: System.out.println("Temmuz"); break;
            case 8: System.out.println("Ağustos"); break;
            case 9: System.out.println("Eylül"); break;
            case 10: System.out.println("Ekim"); break;
            case 11: System.out.println("Kasım"); break;
            case 12: System.out.println("Aralık"); break;
        }
      }
    }
```

Program 3.4. Switch Yapısı Örneği_1

Yukarıdaki örnekte ay değişkenindeki sayısal değere göre ekrana ay ismi yazılmaktadır. Programın başlangıcında int tipindeki ay değişkenine sabit 8 değeri yüklenmektedir. Girilen bu değere göre switch yapısı kullanılmış ve Case ifadesi 1'den 12'ye kadar oluşturulmuştur. Ay değişkenine 8 sayısı yüklendiği için Case 8: System.out.println("Ağustos"); break; satırı çalışacaktır ve ekrana Ağustos mesajı çıkacaktır. Diğer case ifadeleri şartı sağlamadığı için hiçbiri çalıştırılmayacaktır. Programın çıktısı şu şekilde olacaktır.

Ağustos

Aşağıda verilen uygulama, switch deyiminde default yapısının kullanımını göstermektedir.

```
public class AylarSwitchDefaultTest {
public static void main(String[] args) {
  int ay = 25;
  switch (ay) {
    case 1: System.out.println("Ocak"); break;
```

```
      case 2: System.out.println("Şubat"); break;
      case 3: System.out.println("Mart"); break;
      case 4: System.out.println("Nisan"); break;
      case 5: System.out.println("Mayıs"); break;
      case 6: System.out.println("Haziran"); break;
      case 7: System.out.println("Temmuz"); break;
      case 8: System.out.println("Ağustos"); break;
      case 9: System.out.println("Eylül"); break;
      case 10: System.out.println("Ekim"); break;
      case 11: System.out.println("Kasım"); break;
      case 12: System.out.println("Aralık"); break;
      default: System.out.println("Aranılan Şart Bulunamadı!!");
    }
  }
}
```

Program 3.5. Switch Yapısı Örneği_2

Bu örnekte istenen durum hiçbir koşula uymadığı için default koşulundaki kod bölümü çalışmaktadır.

Uygulamanın sonucu aşağıdaki gibi olur.

Aranılan Şart Bulunamadı !!

Aşağıda switch yapısı ile ilgili başka bir örnek program bulunmaktadır.

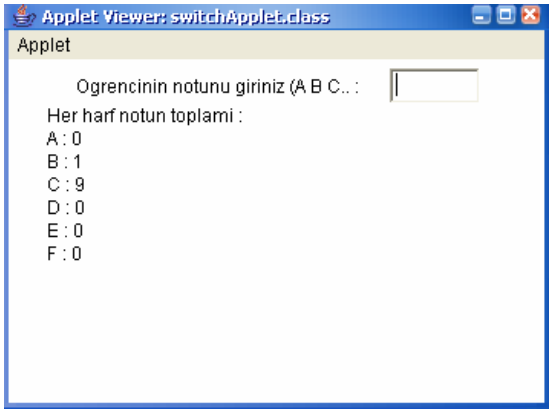
```
import java.applet.Applet;   // java applet sinifini cagir
import java.awt.*;           // java pencere kullanma sinifini cagir
import java.awt.event.*;     //  java pencereyi dinleme sinifini cagir
public class switchApplet extends Applet implements ActionListener
{
   //sinif degiskenleri
```

```
Label kutubasligi; //Label sinifi degiskeni (nesnesi) kutubasligi
TextField kutugirdisi;//Textfield sinifi degiskeni (nesnesi) kutugirdisi
char not;
int Asayisi=0,Bsayisi=0,Csayisi=0,Dsayisi=0,Esayisi=0,Fsayisi=0;
// pencereyi başlatma metodu
public void init()    {
kutubasligi=new Label("Öğrencinin notunu giriniz (A B C.. : ");
add(kutubasligi);//kutu başlığını pencereye yaz
kutugirdisi=new TextField(5);
add(kutugirdisi);//kutuyu pencereye yerleştir
// kutuya yeni ilave edilecek komutları bekle
// her yeni komutta actionPerformed metodunu çalıştır.
kutugirdisi.addActionListener(this);    }
public void paint(Graphics g)     {
    g.drawString("Her harf notun toplamı : ",25,40);
    g.drawString("A : "+Asayisi,25,55);
    g.drawString("B : "+Bsayisi,25,70);
    g.drawString("C : "+Csayisi,25,85);
    g.drawString("D : "+Dsayisi,25,100);
    g.drawString("E : "+Esayisi,25,115);
    g.drawString("F : "+Fsayisi,25,130);    }
// girdi alanındaki olan olayları dinleme metodu
public void actionPerformed(ActionEvent e)
{
//ogrencinin notunu pencereden oku
String not1=e.getActionCommand();
not=not1.charAt(0);
showStatus(""); // sonuç bölgesindeki yazıyı sil
kutugirdisi.setText(""); //kutudaki harfi sil
switch(not)
{
  case 'A': case 'a':
    ++Asayisi;
```

```
            break;
          case 'B': case 'b':
            ++Bsayisi;
            break;
          case 'C': case 'c':
            ++Csayisi;
            break;
          case 'D': case 'd':
            ++Dsayisi;
            break;
          case 'E': case 'e':
            ++Esayisi;
            break;
          case 'F': case 'f':
            ++Fsayisi;
            break;
          default:
            showStatus("yanlış not tanımlandı yeni bir not giriniz.");
            break;
        }  // switch deyiminin sonu
        repaint();//pencereyi yeniden paint metoduna göre çiz
        }
    }
```

Program 3.6. Switch Yapısı Örneği_3

Yukarıdaki programda öğrencini notunun A,B,C,D,E,F olarak text kutusuna girilmesi istenilmektedir ve her enter'e basıldığında o harfin sayısı bir artmaktadır. A,B,C,D,E,F değerlerinden başka bir değer girilirse applet'in alt tarafında "yanlış not tamamlandı yeni bir not giriniz" mesajı ekrana gelmektedir. Programın ekran çıktısı şekil 3.4'de görünmektedir.



Şekil 3.4. Switch örneği

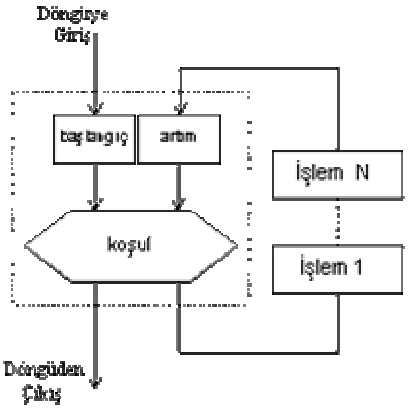
3.2. Döngüler

3.2.1. For döngüsü

FOR döngüsü var olan bir durumu test eder ve eğer durumun sonucu true ise durum false(yanlış) oluncaya kadar blok halindeki komutları devamlı olarak çalıştırır. Bu döngü yapısı sık sık kullanılan ve tüm blokların istenilen sayı kadar tekrar edilmesini sağlayan basit bir yapıdır. FOR döngüsünün yazım şekli şu şekildedir.

For (Başlangıç değeri;Bitiş değeri;Artırım miktarı)
Komutlar(s);

Şekil 3.5'de For döngüsünün algoritma yapısı görünmektedir.



Şekil 3.5. For döngüsünün algoritması [13]

For döngüsünün başlangıcında 3 tane bölüm vardır.

- Başlangıç değeri For döngüsünün başlangıç değerinin belirlenmesini sağlayan ifadedir. For döngüsünde bir index varsa, bu ifade tanımlanmalı ve başlatılmalıdır. Örneğin int i=0;. For döngüsünün bu bölümünde tanımlanan değişkenler döngünün kendisine özeldir.
- Bitiş değeri döngünün her dönüşünde kontrol edilmektedir. Bunun için döngü kendi içerisinde bir test mekanizmasına sahiptir. Buradaki test değer, boolean ifadesidir ya da boolean değer dönen bir fonksiyondur. Örneğin i<10. Eğer test sonucunda true(doğru) değer dönerse, döngü çalışmaya devam edecektir. Bu şekilde test sonucunda false(yanlış) değer dönene kadar döngü çalışmaya devam edecektir. False(yanlış) değer döndüğünde program döngüden çıkarak, programa kaldığı yerden devam edecektir.
- Artırım miktarı herhangi bir ifade veya bir fonksiyondur. Genellikle artırım miktarı döngünün indeks değerini değiştirmek için kullanılır. İstenilirse burada artırım miktarı birden fazla olarak ayarlanabilir.

For döngüsünün herhangi bir bölümünde boş ifadeler olabilir. Kolaylıkla bir ifade olmadan bile noktalı virgül eklenebilir. Aşağıda For yapısı ile ilgili örnek bir program bulunmaktadır.

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

```
    for (int i=0 ; i < 5 ; i ++) {
      System.out.println("i = " + i);    }
   }
}
```

Program 3.7. FOR yapısı örneği_1

Yukarıdaki örnekte for yapısı ile ilgili bir program bulunmaktadır ve ForOrnek isminde bir sınıf(class) yapısı oluşturulmuştur. İ değişkeni tamsayı tipinde tanımlanmış ve For yapısının başlangıç değeri 0 olarak ayarlanmıştır. Döngünün sona erme şartı ise i değişkenindeki değerin 5'ten küçük olmasıdır. Her döngü döndüğü zaman i değişkeni birer birer artacağından i sayısı 5 olana kadar döngüdeki komutlar işlenmeye devam edecektir. İ değişkeninin değeri 5 olduğu zaman döngüden çıkılıp program kaldığı yerden devam edecektir. Uygulamanın çıktısı aşağıdaki gibi olur.

```
i = 0
i = 1
i = 2
i = 3
i = 4
```

Aşağıda for yapısı ile ilgili başka bir örnek program bulunmaktadır.

```
public class ForOrnekVersiyon2 {
public static void main(String args[]) {
  for ( int i = 0, j = 0 ; i < 20 ; i++, j++ ) {
    i *= j ;        System.out.println("i = " + i + " j = " + j);
    }
  }
}
```

Program 3.8. For yapısı örneği_2

ForOrnekVersiyon2 programında değişik yapıda bir for yapısı kullanılmıştır. Burada başlangıç değeri olarak 2 tane değişken ve artırım miktarı olarak ta 2 tane değişken kullanılmıştır. Döngünün başlangıcında i ve j değişkenleri tamsayı tipinde ve değeri 0 olarak ayarlanmıştır. For döngüsünün bitiş test değeri ise i değişkenindeki değerin 20'den küçük olması olarak ayarlanmıştır. Artırım miktarında ise i ve j değişkenleri döngünün her dönüşünde birer artacak şekilde ayarlanmıştır. Uygulamanın çıktısı aşağıdaki gibidir.

i = 0 j = 0
i = 1 j = 1
i = 4 j = 2
i = 15 j = 3
i = 64 j = 4

Aşağıda for yapısı ile ilgili başka bir örnek program bulunmaktadır.

```
import java.io.*;
class foryapisi
{
  public static void main(String args[])
  {
  int toplam=0;
  for(int sayi=1;sayi<=100;sayi++)
  { toplam+=sayi;}
  System.out.println("1 den 100 e sayilarin toplami : "+toplam);
  }
}
```

Program 3.9. For yapısı örneği_3

Programın Çıktısı aşağıdaki gibidir.

1 den 100 e sayilarin toplami : 5050

Aşağıda for yapısı ile ilgili başka bir örnek program bulunmaktadır.

```
import java.awt.Graphics;
import java.applet.Applet;
public class forornek extends Applet {
public void paint(Graphics g)
{
int y=25;
for (int sayac=1;sayac<=20;sayac++){
g.drawString("SAYAÇ DEĞERİ : "+sayac,25,y);
y=y+25;
     }
  }
}
```

Program 3.10. For yapısı örneği_4

Program çalıştırıldığında, çıktı olarak elde edilen applet incelenirse sayaç kontrol değişkeninin ilk olarak 1 değeri aldığı görülür. Daha sonra döngü içerisinde birer birer artırılır ve 20 değerine ulaşınca döngünün son bulduğu görülmektedir. Böylece birinci ifade diye tanımlanan sayac=1; komutu ile kontrol değişkenine ilk değer atanır.

İkinci ifade diye tanımlanan sayaç<=20; komutu ile döngü uzunluğu belirlenmektedir. Üçüncü ifade diye tanımlanan sayaç++; komutu ile kontrol değişkeninin her dönüşü için değişim miktarı belirlenir.

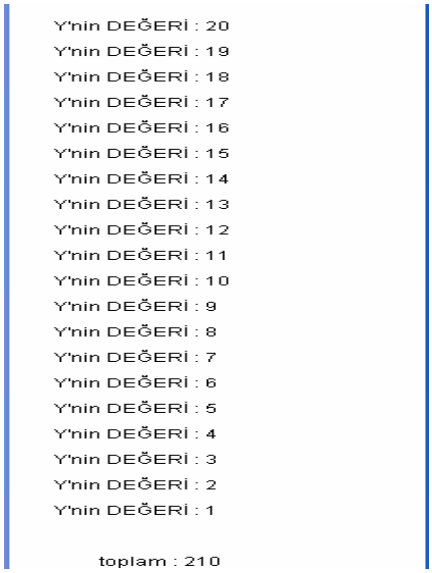
Java programlama dilinde for döngü yapıları için kullanılan 3 ifadenin hepsi isteğe bağlı ifadelerdir. Eğer programcı isterse bazı bölümleri yazmayabilir. Ama eğer birinci ifade verilmezse bu, kontrol değişkeninin daha önceki komutlarda tanımlanmış ve değer atanmış olmasını gerektirir. İkinci ifade verilmezse, for döngüsünün sonsuz bir döngü olacağı anlamına gelir. Üçüncü ifade verilmezse, döngü değişkeninin döngü içerisinde arttırılıp, azaltılacağı anlaşılır. Aşağıda döngü

değeri azalarak oluşturulmuş bir for döngüsü programı bulunmaktadır. Program 20 den 1' e kadar i kontrol değişkeninin döngü içinde aldığı değerleri applet'e yazıp, döngü değişkeninin aldığı değerlerin toplamını çıkış olarak vermektedir.

```
import java.awt.Graphics;
import java.applet.Applet;
public class forornek02 extends Applet {
public void paint(Graphics g) {
int toplam=0;
int y=25;
for (int i=20;i>=1;i--){
g.drawString("Ý'nin DEĞERİ : "+i,25,y);
y=y+25;
toplam=toplam+i; }
g.drawString("toplam : "+toplam,50,y+25);  }
}
```

Program 3.11. For yapısı örneği_5

Bu örnekte azalarak giden döngü kontrolü değişkeni incelenmektedir. I değişkeni ilk değeri olan 20 değerinden sonra, döngü her dönüşünde bu ilk değerinden birer birer azalarak 1 değerine ulaşmaktadır. 1 değerine ulaşılınca döngü son bulmaktadır ve değerlerin toplamı applet'e yazılıp program sonlandırılmaktadır. Şekil 3.6’da programın ekran çıktısı görünmektedir.



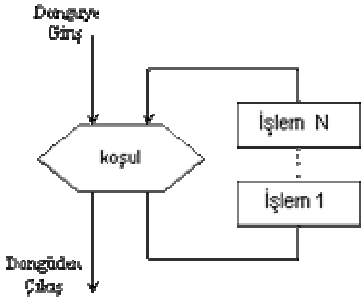
Şekil 3.6. For döngüsü ekran çıktısı

3.2.2. While döngüsü

While döngüsü bir blok ya da blok halindeki ifadelerin durumu true(doğru) oluncaya kadar komutları tekrarlayan yapıdır. While döngüsünün yazım şekli aşağıdadır.

While (Durum) Komutlar;

Şekil 3.7'de for döngüsünün algoritma yapısı görünmektedir.



Şekil 3.7. While Döngüsünün yapısı

Durum boolean sonuç döndüren boolean bir ifadedir. Eğer true(doğru) değer üretilirse, While döngüsü, döngü içerisine yazılmış olan komutları çalıştırır ve daha

sonra durumu tekrar test eder. Eğer döngü sonucunda false(yanlış) değer üretilirse while döngüsünde bulunan komutlar çalışmayacaktır.

Aşağıda tamsayı bir dizinin elamanlarını float bir diziye kopyalanmasını sağlayan bir while döngüsü bulunmaktadır. Bu ifadeleri daha ilgi çekici yapabilmek için iki durumun sonucu olarak true(doğru) değerin dönmesi gerekir.

- Count değişkeni dizinin uzunluğundan daha az olmalıdır.
- Tamsayı değeri 0 olmamalıdır.

Bunları başarabilmek için bir bileşik test uygulanır ve bir durumdan daha fazlası kontrol edilir. && operatörü kullanıldığı zaman, true(doğru) değer dönebilmesi için her iki ifadenin de true(doğru) olması gerekir. Bu döngü her sefer çalıştığında sayma değerini artırmak için aynı zamanda artırma operatörünü(++) kullanır. Aşağıda bununla ilgili bir örnek bulunmaktadır.

```
İnt count = 0;
While ( (count < arrInt.length && (arrIntcount[count] !=0) ) {
 arrFloat [count] = (float) arrInt [count];
 count++;
}
```

Program 3.12. While döngüsü örneği

ArrInt dizisinin uzunluğunun 20 olduğu düşünüldüğünde, eğer arrInt dizisinin değerlerinin hiçbiri 0 değilse, bu while döngüsünün 20 defa çalışacağını belirtir. Çünkü sayma değerleri 0 ile 19 arasında olduğunda test değeri true(doğru) olacaktır. Diğer bir yandan, eğer ArrInt dizisinin değerlerinden herhangi birinin değeri 0 olursa, döngü 0 kereden 19 kereye kadar herhangi bir yere kadar çalışacaktır. Aşağıda while döngüsü ile ilgili örnek bir program bulunmaktadır.

```
public class WhileOrnek {
int i = 0 ; //döngü kontrol değişkeni
 while (i < 5 ) {
```

```
            System.out.println("i = " + i);
            i++ ;
        }
        System.out.println("Sayma işlemi tamamlandı.");
    }
}
```

Program 3.13. While döngüsü örneği_1

Yukarıdaki programda WhileOrnek isminde bir sınıf(class) yapısı oluşturulmuştur. Başlangıçta i tamsayı değişkenine o değeri sabit olarak atanmıştır. while döngüsü içerisinde i değeri 5'ten küçük olduğu sürece döngü dönmeye devam edecektir. Döngü içerisinde System.out.println komutuyla i değişkeninin o anki değeri ekrana yazdırılmaktadır. I++ komutuyla da i değeri 1 artırılmaktadır. I değişkenindeki değer 5 olana kadar döngü bu şekilde işlemeye devam edecektir. I değeri 5 olduğunda baş taraftaki şart yerine getirilmediğinden döngüye girilmeyecek ve daha sonraki komut işlenmeye başlanacaktır. Son olarak ta ekrana "Sayma işlemi tamamlandı" mesajı gelecektir. Uygulamanın sonucu aşağıdaki gibi olur.

```
i = 0
i = 1
i = 2
i = 3
i = 4
Sayma işlemi tamamlandı.
```

Aşağıda while döngüsü ile ilgili başka bir örnek program bulunmaktadır.

```
import java.io.*; //java girdi cikti sinifini cagir
class whileyapisi
 {
     public static void main(String args[])
     {
```

```
    int sayi=2;
    while(sayi<=1000)
      {
        sayi*=2;
        System.out.println("sayı = "+sayi);
    }
}
```

Program 3.14. While döngüsü örneği_2

Yukarıdaki örnekte while yapısı isminde bir sınıf(class) oluşturulmaktadır. Ana programda sayi değişkeni tamsayı olarak tanımlanmakta ve sayi değişkenine 2 değeri atanmaktadır. While döngüsü, sayi değişkenindeki değer 1000 ve 1000'den küçük olduğu değerlerde çalışmaktadır. Şart doğru olduğunda döngü içerisindeki komutlar işlenmeye başlanacaktır. Sayı değeri 2 ile çarpılarak tekrar sayı değişkenine aktarılmaktadır. System.out.println komutuyla da ekrana sayı ifadesi ve o anki değeri yazdırılmaktadır. Sayı değişkenindeki değer 1000 sayısını geçtiği zaman program döngüden çıkarak diğer komutları işlemeye devam edecektir. Uygulamanın çıktısı aşağıdaki gibi olur.

```
sayı = 4
sayı = 8
sayı = 16
sayı = 32
sayı = 64
sayı = 128
sayı = 256
sayı = 512
sayı = 1024
```

3.2.3. Do-while döngüsü

While döngüsünde eğer şart true(doğru) ise döngünün çalışacağı, eğer şart false(yanlış) ise döngü içerisindeki komutların hiç çalışmayacağı incelenmişti. Eğer döngünün şarta bağlı olmadan en az bir defa çalışması isteniyorsa, bu durumda do-while döngüsü kullanılmalıdır. Çalışma prensibi olarak while-do döngüsüne benzer bir yapıdadır. While döngüsünde bulunan şart sağlandığında döngü içerisinde bulunan komutlar çalıştırılacaktır. Java dilindeki do-while döngüsü Pascal programlama dilindeki repeat-until yapısının mantık olarak aynısıdır.

Aşağıda do-while döngüsünün yazım şekli bulunmaktadır.

```
do {
 çalışması istenen kod bloğu
} while(koşul);
```

Bu ifadede döngü içerisindeki komutlar çalıştırılır ve daha sonra durum testi yapılır. Eğer döngüden true(doğru) değer dönerse, komutlar tekrar çalıştırılır. Eğer false(yanlış) değer dönerse, döngü orada sona erecektir. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
Int x = 1;
Do {
        System.out.println("Döngü, dönüyor" + x);
        X++;
}  while (x <=20);
```

Program 3.15. Do-While örneği

Yukarıdaki örnekte basit bir do-while döngüsü bulunmaktadır. Do-while döngüsünde komutlar şarta bakılmaksızın en az bir kere çalıştırıldığı için ekrana "Döngü dönüyor" yazısı çıkartılır. X++ komutuyla da x değişkenindeki değer 1 artırılır. X

değeri 20'ye eşit veya küçük olana kadar döngüdeki komutlar işlemeye devam edilir. Uygulamanın çıktısı aşağıdaki gibi olur.

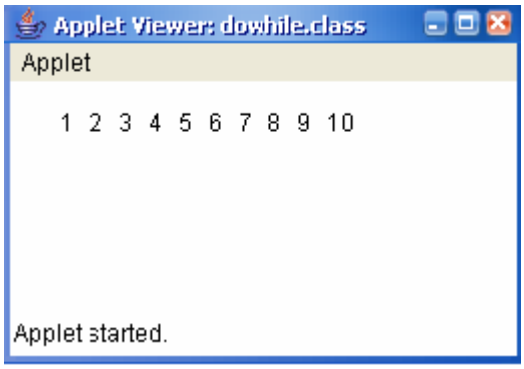
Döngü, dönüyor1
Döngü, dönüyor2
Döngü, dönüyor3
Döngü, dönüyor4
Döngü, dönüyor5
Döngü, dönüyor6
Döngü, dönüyor7
Döngü, dönüyor8
Döngü, dönüyor9
Döngü, dönüyor10
Döngü, dönüyor11
Döngü, dönüyor12
Döngü, dönüyor13
Döngü, dönüyor14
Döngü, dönüyor15
Döngü, dönüyor16
Döngü, dönüyor17
Döngü, dönüyor18
Döngü, dönüyor19
Döngü, dönüyor20

Aşağıda do-while ile ilgili başka bir örnek program bulunmaktadır.

```
import java.awt.Graphics;
import javax.swing.*;
public class dowhileApplet extends JApplet
{
   public void paint(Graphics g)    {
   int saydirici=1;
   int x=25;
   do {
       g.drawString(Integer.toString(saydirici),x,25);
        x += 15;
       } while(++saydirici <= 10);
   }
}
```

Program 3.16. Do-While applet örneği

Yukarıdaki örnekte do-while yapısını gösteren bir applet bulunmaktadır. Saydirici değişkeni tamsayı tipinde tanımlanmış ve 1 değeri atanmıştır. Aynı şekilde x değişkeni de tamsayı tipinde tanımlanmış ve 25 değeri değişkene aktarılmıştır. Do-while döngüsü içerisinde, drawString komutuyla ekranın belirlenmiş koordinatlarına saydirici değişkenindeki değer yazdırılmaktadır. Döngü içerisinde x+=15 komutuyla x değişkenindeki değer 15 artırılıp tekrar x değişkenine aktarılmaktadır. Test kısmında ise saydirici değişkenindeki değer 1 arttırılıp değer 10 ve 10'dan küçük olduğu sürece döngü dönmeye devam etmektedir. Şekil 3.8'de programın ekran çıktısı görünmektedir.



Şekil 3.8. Do-While applet örneği

3.3. Dallanma İfadeleri

Dallanma ifadeleri döngüler içerisinde kullanılmaktadır. Java programlama dilinde 3 tane dallanma ifadeleri mevcuttur.

- Break
- Continue
- Return

3.3.1. Break deyimi

Break deyiminin 2 farklı uyarlaması bulunmaktadır. Birisi etiketli(labeled), diğeri ise etiketsiz(unlabeled) break deyimidir. Etiketsiz break sistemi switch deyiminde kullanılan deyimin kullanım olarak aynısıdır. Etiketsiz break yapısında koşul

true(doğru) duruma geldiğinde switch deyimini sonlandırılmasını sağlamaktadır. Break deyimi aynı şekilde while, do-while veya for deyimlerinden çıkılması için de kullanılmaktadır. Aşağıda break deyimi ile ilgili örnek bir program bulunmaktadır.

```
public class BreakTest {
public static void main(String[] args) {
  for ( int i = 0; i < 100; i++ ) {
    if ( i ==9 ) {     // for döngüsünü kesiyor
       break;
    }
System.out.println("i =" +i);
  }
  System.out.println("Döngüden çıktı");
 }
}
```

Program 3.17. Break deyimi örneği

Programın normal çalışmasına göre kod bloğunun 0'dan 99'a kadar dönmesi gerekmektedir. Fakat i değişkenin 9 değerine gelmesiyle break komutu sayesinde for döngüsünün dışına çıkılmaktadır. Uygulamanın çıktısı aşağıdaki gibidir.

```
i =0
i =1
i =2
i =3
i =4
i =5
i =6
i =7
i =8
Döngüden çıktı
```

Etiketsiz break deyimleri en içteki while, do-while veya for döngüsü ifadelerini sona erdirirken, etiketli break ifadeleri, etiket(label) hangi döngünün başına konulmuş ise o döngü sistemini sona erdirmektedir. Aşağıda break deyiminin etiket ile kullanılmasıyla ilgili örnek bir program bulunmaktadır.

```
public class BreakTestEtiketli {
public static void main(String[] args) {
  kes :
  for ( int j = 0 ; j < 10 ; j ++ ) {
    for ( int i = 0; i < 100; i++ ) {
  if ( i ==9 ) {// for döngüsünü kesiyor
        break kes;
  }
  System.out.println("i =" +i);
}
System.out.println("Döngüden çıktı");
System.out.println("j =" +j);
  }
 }
}
```

Program 3.18. Etiketli Break Örneği [13]

Yukarıdaki örnekte etiket kullanılarak, daha geniş çaplı bir döngü sisteminden çıkılmaktadır. I değeri 9'a eşit olduğunda break deyimi sayesinde program kiril etiketine gitmektedir. Uygulamanın çıktısı aşağıdaki gibidir.

```
i =0
i =1
i =2
i =3
i =4
i =5
```

```
i =6
i =7
i =8
```

3.3.2. Continue deyimi

Continue ifadesi, döngü içersinde o anki döngü işleminin çalıştırılmamasını ve bir sonraki döngü işleminin başlamasını sağlayan bir mekanizmadır. Continue ifadeleri de break ifadeleri gibi etiketsiz continue ve etiketli continue olmak üzere iki çeşide ayrılır. Etiketsiz continue en içteki döngü içersinde etkili olurken, etiketli continue ise başına konulduğu döngü sisteminde etkili olmaktadır. Aşağıda continue deyimi ile ilgili örnek bir program bulunmaktadır.

```
public class ContinueTest {
public static void main(String[] args) {
  for ( int i = 0; i < 10; i++ ) {
if ( i == 5 ) {       // for döngüsünü kesiyor
 continue;
  }
System.out.println("i =" +i);
  }
  System.out.println("Döngüden çıktı");
 }
}
```

Program 3.19. Continue deyimi örneği [13]

Uygulamanın sonucu aşağıdaki gibi olur:

```
i =0
i =1
i =2
i =3
```

```
i =4
i =6
i =7
i =8
i =9
Döngüden çıktı
```

Ekrana yazılan sonuca dikkatli bakıldığında 5 değerinin olmadığı görülür. Continue deyimi break gibi döngüleri iptal etmez, yalnızca belli durumlardaki döngü işleminin atlanmasını sağlar.

Aşağıda continue deyiminin etiket ile beraber kullanılmasıyla ilgili örnek bir program bulunmaktadır.

```
public class ContinueTestEtiketli {
public static void main(String[] args) {
  pas :
  for ( int j = 0 ; j < 6 ; j ++ ) {
 for ( int i = 0; i < 5; i++ ) {
  if ( i ==3 ) {   // for döngüsünü kesiyor
        continue pas;
  }
  System.out.println("i =" +i);
 }
 System.out.println("Döngüden çıktı");
 System.out.println("j =" +j);
  }
 }
}
```

Program 3.20. Etiketli continue örneği

Bu uygulamada, pas etiketi kullanılarak continue işleminin en dıştaki döngüsel sistemden tekrardan başlaması sağlanmaktadır. Uygulamanın sonucu aşağıdaki gibi olur.

```
i =0
i =1
i =2
i =0
i =1
i =2
i =0
i =1
i =2
i =0
i =1
i =2
i =0
i =1
i =2
i =0
i =1
i =2
```

i değişkeninin her seferinde yeniden 0'dan başladığı ve 2 de kesildiği görülmektedir. Bu işlem toplam 6 kez oluşmaktadır.

3.3.3. Return deyimi

Return deyiminin iki tür kullanım şekli vardır. Birinci kullanım şekli değer döndürmek içindir ve deyimlerden üretilen değerler bu şekilde geri döndürülmektedir. İkincisi ise eğer deyimin dönüş tipi buna izin vermiyorsa return ifadesi yazılıp ilgili deyim bırakılabilmektedir. Aşağıda return deyimi ile ilgili örnek bir program bulunmaktadır.

```
public class ReturnTest {
public double toplamaYap(double a, double b) {
  double sonuc = a + b ;
  return sonuc ;   // normal return kullanımı
}
public void biseyYapma(double a) {
  if (a == 0) {
return ; // sınıfı acilen terk et
  } else {
System.out.println("-->" + a);
  }
}
}
```

Program 3.21. Return deyimi örneği

Yukarıdaki örnekte toplamayap adında bir sınıf(class) oluşturulmuştur. Return deyimiyle toplama sonucunda elde edilen değer geri döndürülmektedir. Biseyyapma sınıfında ise a değişkenindeki değer 0'a eşit olduğunda return ile bırakılmaktadır.[13]

BÖLÜM 4. DİZİLER

4.1. Dizilerin Tanımı

Java dilinde yazılmış programlarda birkaç değişken kullanılabilir. Programlarda eğer 20 tane bilgi tutulmak isteniyorsa bunun için 20 tane farklı değişken kullanılması gerekir. Bu da programın çok genişlemesine yol açacaktır. Eğer programda 100, 1000 hatta daha fazla bilgi için değişken kullanılması gerekiyorsa bu programın daha da fazla büyümesine sebep olacaktır.

Bu gibi durumlar için her programlama dilinde olduğu gibi Java dilinde de dizi dediğimiz yapılar kullanılır. Diziler benzer yapıdaki değişken bilgilerini depolamanın bir yoludur. Her bilgi numaralandırılmış başka bir bölümde saklanır. Kullanıcı kolay bir şekilde bu numaralanmış bilgilere erişebilir.

Diziler değişkenlerde depolanan her türlü bilgiyi içerebilir. Örneğin kullanıcı tamsayı tipinde bir dizi veya metinsel bilgi(string) bilgi içeren bir dizi oluşturabilir. Fakat her iki bilgi aynı dizi içerisinde kullanılamaz. Dizilerin eleman sayıları belirlendikten sonra bir daha değiştirilemez.

Dizi tanımlamanın aşamaları aşağıdaki gibidir.

- Diziyi tutacak bir değişken belirlemek.
- Yeni bir dizi nesnesi oluşturmak ve onu dizi değişkenine atamak.
- Dizi içerisine bilgi depolamak.

4.2. Diziyi Tutacak Bir Değişken Belirlemek.

Dizi oluşturmanın ilk adımı, diziyi tutacak değişkenin belirlenmesidir. Dizi değişkenleri dizinin nesne ya da veri tipini belirler. Normal değişken tanımlamaktan

biraz farklıdır. İki parantez ([ ]) içerisine nesne, veri tipi ya da değişken ismi eklenir. Aşağıda dizi tanımlamayla ilgili örnekler bulunmaktadır.

```
String kelimeler[ ]; // String tipinde bir dizi
Point sayılar[ ]; // Nokta tipinde bir dizi
Int kitaplar[ ]; // Tamsayı tipinde bir dizi
double dd[ ] ;     // double tipindeki dizi
```

İstenilirse parantezler, değişken ismi yerine bilgi tipinden sonra da konulabilir.

```
String [ ] kelimeler;
Point [ ] sayılar;
Int [ ]  kitaplar;
```

4.3. Dizi Nesneleri Oluşturmak

Dizi değişkeni belirlendikten sonra bir sonraki adım dizi nesnesi belirlemektir ve değişkene atamaktır. Bununla ilgili örnek aşağıdadır.

```
String oyuncular = new String [10];
```

Yukarıdaki örnekte string nesneleri içeren 10 elemanlık bilgiyi alabilecek bir dizi oluşturulmuştur. New komutu kullanılarak bir dizi nesnesi oluşturulduğu zaman, dizinin ne kadar bilgiyi tutacağı belirtilmelidir. [14]

```
double[ ] d = new double[30];   //   30 elemanlı double tipindeki dizi
double dd[ ]= new double[10];   //   10 elemanlı double tipindeki dizi
float [ ]fd = new float [24];   //   24 elemanlı float tipindeki dizi
Object[ ]ab = new Object[19];   //   19 elemanlı Object tipindeki dizi
String[ ] s = new String[25];   //   25 elemanlı String tipindeki dizi [13]
```

Dizi nesneleri nesneler içerebildiği gibi, tamsayı ya da boolean tipi bilgiler de içerebilir.

Int[ ] temps = new int[99];

Dizi oluşturulduğu gibi hemen kullanılabilir. New komutu ile dizi nesnesi oluşturmak yerine, dizi içerisine doğrudan, bilgiler araya virgül işareti konularak da oluşturulabilir.

String[ ] oyuncular = { “ Ahmet”, “Ayhan”, “Adnan”, Metin”, “Zeynep”};

Parantez içerisindeki her eleman dizide tanımlanan değişken tipi ile aynı tipte olmalıdır ve aynı zamanda dizideki elemanların sayısı tanımlanan sayı ile aynı olmalıdır.

4.4. Dizi Elemanlarına Erişim

Java dilinde dizi kullanımı sırasında tanımlanan dizinin sınırları aşılırsa, çalışma anında (runtime) java kullanıcıyı hata mesajı ile uyarır. Örneğin 20 elemanlı bir double dizi tanımlanmışsa ve bu dizinin 78. elemanına ulaşılmak istenilirse Java ArrayIndexOutOfBounds Exception hata mesajını çıkaracaktır. Bu hatanın çalışma anında alınması güvenlik açısından daha iyi olacaktır. Böylece dizi için ayrılmış bellek alanından dışarı çıkılıp başka verilere müdahale edilmesi engellenmiş olunur.

Aşağıda bellek alanından çıkılarak dışarıdaki verilere müdahele etmekle ilgili örnek bir program bulunmaktadır.

```
public class DiziElemanlariGosterim {
public static void main(String args[]) {
  double[ ] d = { 2.1, 3.4, 4.6, 1.1, 0.11 } ;
  String[ ] s = { "defter", "kalem", "silgi", "Kedi", "Köpek" };
  // double tipindeki dizi ekrana yazdırılıyor
  for (int i = 0 ; i < d.length ; i ++) {
    System.out.println("d["+i+"] = " + d[i] );
    // System.out.println("d["+78+"] = " + d[78] );
  }
```

```
    System.out.println("----------------");

    // String tipindeki dizi ekrana yazdırılıyor.
    for (int x = 0 ; x < s.length ; x ++) {
      System.out.println("s["+x+"] = " + s[x] );
      // System.out.println("s["+78+"]=" + s[78] );     //   Hata !
    }
  }
}
```

Program 4.1. Dizi elemanları örneği_1 [13]

Length ifadesiyle bir dizinin içerisindeki eleman sayısı öğrenilebilir. Bu örnekte iki adet dizi tanımlanmıştır. Double ve String türündeki dizilerin içerisine 5'er adet eleman yerleştirilmiştir ve daha sonra dizi içerisindeki bilgiler for döngüsü ile ekrana yazdırılmıştır.

Eğer 5 elemana sahip olan dizinin 78. elemanına erişilmeye kalkışılırsa, derleme anında bir hata ile karşılaşılmaz. Ancak, uygulama çalıştırıldığı zaman hata mesajı ile karşılaşılır. Uygulamanın sonucu aşağıdaki gibi olur.

```
d[0] = 2.1
d[1] = 3.4
d[2] = 4.6
d[3] = 1.1
d[4] = 0.11
----------
s[0] = defter
s[1] = kalem
s[2] = silgi
s[3] = Kedi
s[4] = Köpek
```

Bir önceki uygulamanın çalışma sırasında hata vermesi istenmiyorsa, yorum satırı olan yerlerin açılması ve uygulamanın baştan derlenip çalıştırılması gerekmektedir.

4.5. Dizi Elemanlarını Sıralama

Dizi elemanlarını büyükten küçüğe doğru sıralamak için java.util paketi altındaki Arrays sınıfı kullanılır. Bu sınıfın içindeki statik sort() ifadesi sayesinde dizilerin içerisindeki elemanlar sıralanır.

Aşağıda dizilerde sıralama ile ilgili örnek bir program bulunmaktadır.

```
import java.util.*;
public class DiziSiralamaornegi {
  public static void ekranaBas(double[ ] d) {
     for (int i = 0 ; i < d.length ; i++) {
       System.out.println("d["+i+"] = " + d[i]);          }
  }
  public static void main(String args[]) {
     double d[ ] = new double[9];
     d[0] = 2.45;          d[1] = 3.45 ; d[2] = 4.78;
     d[3] = 1.45;          d[4] = 15.12; d[5] = 1;
    d[6] = 9;  d[7] = 15.32 ; d[8] = 78.17;
     System.out.println("Karışık ");  ekranaBas(d);
     Arrays.sort( d ) ;
     System.out.println("Sıralanmış olarak"); ekranaBas(d);
  }
}
```

Program 4.2. Dizi sıralama örneği

Uygulama sonucu aşağıdaki gibi olur.

Karışık
d[0] = 2.45
d[1] = 3.45
d[2] = 4.78
d[3] = 1.45
d[4] = 15.12
d[5] = 1.0
d[6] = 9.0
d[7] = 15.32
d[8] = 78.17
Sıralanmış olarak
d[0] = 1.0
d[1] = 1.45
d[2] = 2.45
d[3] = 3.45
d[4] = 4.78
d[5] = 9.0
d[6] = 15.12
d[7] = 15.32
d[8] = 78.17

4.6. Dizilerin Dizilere Kopyalanması

Kullanıcı eğer isterse bir dizideki tüm elemanları başka bir diziye kopyalayabilir. Bu özellik aşağıdaki örnek program ile daha iyi anlaşılabilir.

```
public class DiziKopyalanmasi {
public static void main(String args[ ])  {
   int[ ] dizi1 = { 1,2,3,4 };  // ilk dizi
   int[ ] dizi2 = { 100,90,78,45,40,30,20,10}; // daha geniş olan 2. dizi
    // kopyalama işlemi başlamaktadır.
   // 0. indisinden dizi1 uzunluğu kadar kopyalama yapar
   System.arraycopy(dizi1,0,dizi2,0,dizi1.length);
```

```
        for (int i = 0 ; i < dizi2.length ; i++) {
          System.out.println("dizi2["+i+"] = "+ dizi2[i] );
           }
        }
    }
```

Program 4.3. Dizilerin kopyalanması örneği

Yukarıdaki örnekte System sınıfının statik ifadesi olan arraycopy() sayesinde dizi1 tamamiyle dizi2'ye kopyalanmaktadır. Sonuç aşağıdaki gibi olur.

```
dizi2[0] = 1
dizi2[1] = 2
dizi2[2] = 3
dizi2[3] = 4
dizi2[4] = 40
dizi2[5] = 30
dizi2[6] = 20
dizi2[7] = 10
```

4.7. Çok Boyutlu Diziler

Java dilinde çok boyutlu dizilerin yapısı, diğer programlama dillerinden farklıdır. Tanımlanan dizinin mutlaka tek türde olması gerekir. Aşağıda basit bir çok boyutlu dizi tanımlaması gösterilmiştir.

```
Int [ ][ ] t1 = {
    { 1, 2, 3, },
    { 4, 5, 6, },
            };
```

Yukarıda ifade edildiği gibi temel türden oluşmuş iki boyutlu diziden çok boyutlu dizi oluşturulabilir. Çok boyutlu dizileri oluşturmanın diğer bir yolu ise, new komutunu kullanmaktır. Aşağıda new yapısı ile ilgili bir örnek bulunmaktadır

```
Int [ ] [ ] t1 = new int [3][4] ;
Int [ ] [ ] t1 = new int [ ][4] ; // Hata meydana gelir
```

Aşağıda çok boyutlu diziler ile ilgili örnek bir program bulunmaktadır.

```
public class CokBoyutluDiziler {
public static void main(String args[ ]) {
   int ikiboyutlu[ ][ ] = new int[3][4] ;
   ikiboyutlu[0][0] = 60 ;
   ikiboyutlu[0][1] = 90 ;
   ikiboyutlu[0][2] = 11 ;
   ikiboyutlu[0][3] = 18 ;
   ikiboyutlu[1][0] = 17 ;
   ikiboyutlu[1][1] = 56 ;
   ikiboyutlu[1][2] = 26 ;
   ikiboyutlu[1][3] = 79 ;
   ikiboyutlu[2][0] = 3 ;
   ikiboyutlu[2][1] = 93 ;
   ikiboyutlu[2][2] = 43 ;
   ikiboyutlu[2][3] = 12 ;
   // ekrana yazdırılıyor
   for (int i = 0 ; i<ikiboyutlu.length ; i++ ) {
   for (int j = 0 ; j < ikiboyutlu[i].length ; j++ ) {
   System.out.println(" ikiboyutlu["+i+"]["+j+"] = "
                                + ikiboyutlu[i][j] );    }  }
  }
}
```

Program 4.4. Çok boyutlu diziler örneği

Yukarıdaki örnekte int türünde 3’e 4’lük (3x4) çok boyutlu dizi oluşturulmuştur. Bu dizi 3’e 4'lük bir matris gibi de düşünülebilir. Uygulamanın sonucu aşağıdaki gibidir.

```
ikiboyutlu[0][0] =60
ikiboyutlu[0][1] =90
ikiboyutlu[0][2] =11
ikiboyutlu[0][3] =18
ikiboyutlu[1][0] =17
ikiboyutlu[1][1] =56
ikiboyutlu[1][2] =26
ikiboyutlu[1][3] =79
ikiboyutlu[2][0] =3
ikiboyutlu[2][1] =93
ikiboyutlu[2][2] =43
ikiboyutlu[2][3] =12
```

Uygulama sonucu matris gibi düşünülürse aşağıdaki gibi olur.

```
60 90 11 18
17 56 26 79
 3 93 43 12
```

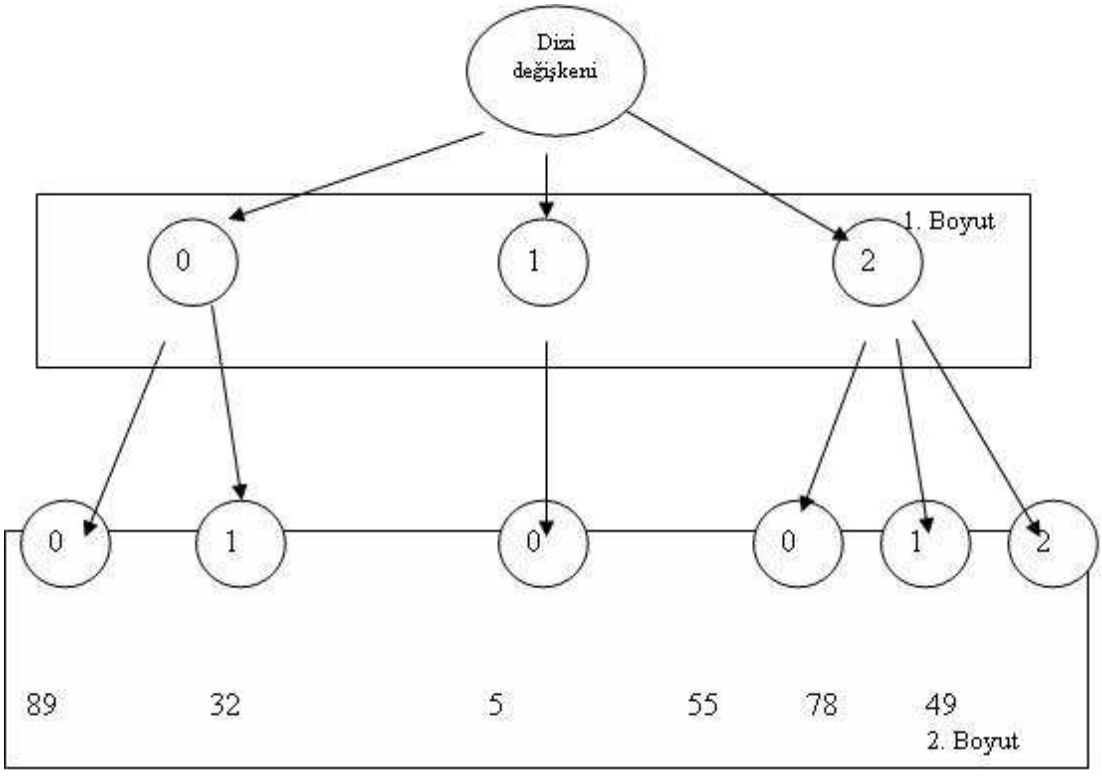
Dizilere bağlı diğer dizilerin aynı boyutta olma zorunluluğu yoktur. Dizilerin boyutları farklı olabilir. Aşağıda bağlı dizilerin aynı boyutta olması gerekmediği ile ilgili örnek bir program bulunmaktadır.

```
public class CokBoyutluDizi {
public static void main(String args[ ])
  {
    int ikiboy[ ][ ] = new int[3][ ];
    ikiboy[0] = new int[2] ;
    ikiboy[1] = new int[1] ;
    ikiboy[2] = new int[3] ;
```

```
        ikiboy[0][0] = 80 ;
        ikiboy[0][1] = 25 ;
        ikiboy[1][0] = 10 ;
        ikiboy[1][1] = 3 ;
        ikiboy[2][0] = 55 ;
        ikiboy[2][1] = 78 ;
        ikiboy[2][2] = 49 ;
    }
}
```

Program 4.5. Çok boyutlu diziler örneği_2

Yukarıda örnekte görüldüğü gibi, dizilere bağlı her farklı dizinin boyutları birbirinden farklı olmuştur. Şekil 4.1 bu ifadeyi göstermektedir.



Şekil 4.1. Farklı elemana sahip çok boyutlu dizi

Çok boyutlu dizilerin içerisine sınıf tipleri de yerleştirilebilir. Örneğin String sınıfı tipinde olan çok boyutlu bir dizi oluşturulabilir. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
public class Hayvanlar {
String isimler[ ][ ][ ] ;
public Hayvanlar() {
   isimler = new String[2][2][3] ;
   veriAta();
}
public void veriAta() {
   isimler[0][0][0] = "aslan" ;
   isimler[0][0][1] = "Ayı" ;
   isimler[0][0][2] = "ceylan";
   isimler[0][1][0] = "at" ;
   isimler[0][1][1] = "essek" ;
   isimler[0][1][2] = "fare" ;
   isimler[1][0][0] = "geyik" ;
   isimler[1][0][1] = "hamsi" ;
   isimler[1][0][2] = "inek" ;
   isimler[1][1][0] = "balık" ;
   isimler[1][1][1] = "kedi" ;
   isimler[1][1][2] = "lama" ;
ekranaBas() ;   }
public void ekranaBas() {
   for (int x = 0 ; x < isimler.length ; x++) {
     for (int y = 0 ; y < isimler[x].length ; y++) {
       for (int z = 0 ; z < isimler[x][y].length ; z ++) {
                System.out.println("isimler["+x+"]["+y+"]["+z+"] =" +
                                    isimler[x][y][z]);
       }
     }
   }
```

```
    }
    public static void main(String args[ ]) {
       HayvanlarAlemi ha = new HayvanlarAlemi();
    }
}
```

Program 4.6. Çok boyutlu diziler örneği_3

Yukarıdaki örnekte HayvanlarAlemi nesnesinin oluşturulmasıyla olaylar başlatılmıştır. İster tek boyutlu olsun ister çok boyutlu olsun, diziler üzerinde işlem yapmak isteniliyorsa, onların mutlaka new kelimesi ile tanımlanması gerekmektedir. Bu örnekte olduğu gibi, dizi ilk oluşturulduğunda, dizi içerisindeki String tipindeki referansların ilk değeri boş(null)'dur. Uygulamanın sonucu aşağıdaki gibi olur. [13]

```
isimler[0][0][0] =aslan
isimler[0][0][1] =Ayı
isimler[0][0][2] =ceylan
isimler[0][1][0] =at
isimler[0][1][1] =essek
isimler[0][1][2] =fare
isimler[1][0][0] =geyik
isimler[1][0][1] =hamsi
isimler[1][0][2] =inek
isimler[1][1][0] =balık
isimler[1][1][1] =kedi
isimler[1][1][2] =lama
```

BÖLÜM 5. SINIF YAPISI

5.1. Sınıf(Class) Oluşturma

Java'da bulunan class(Sınıf) terimi program terimiyle eşanlamlı kullanılır. Java dilinde program bir main sınıfından meydana gelir ve diğer sınıfların da main sınıfını desteklemesi gerekir. Bu sınıf yapıları Java'nın sınıf kütüphanesindeki gereken bazı sınıfları desteklerler.

5.2. Sınıf Tanımlama

Program içerisinde sınıf class kelimesi ile tanımlanır ve daha sonra sınıfa verilecek olan isim belirlenir. Aşağıda bununla ilgili bir örnek bulunmaktadır.

```
Class Ticker  {
//Sınıftaki komutlar
}
```

Bununla beraber sınıflar Java sınıf hiyerarşisindeki tüm sınıfların superclass'ı (supersınıf) olan nesne sınıflarının bir uzantısıdır. Eğer sınıf bir altsınıf(Subclass) ise extends kelimesi ile yeni sınıfın supersınıf'a ait olduğu belirtilir. Aşağıda bununla ilgili bir örnek bulunmaktadır.

```
Class SportsTicker extends Ticker  {
//Sınıftaki komutlar
}
```

5.3. Sabitler

Değişkenler program çalıştırıldığı zaman değişen bilgi depolamak için kullanılan yapılardır. Eğer değişkenin değeri program içerisinde hiçbir zaman değişmiyorsa, sabit(Constant) adı verilen değişken tipleri kullanılabilir.

Bir sabiti tanımlayabilmek için, değişken tanımlamasından önce Final ifadesi kullanılır ve değişken için uygun bir değer içerir. Aşağıda bazı sabit tanımlamaları bulunmaktadır.

```
Final float pi=3.141592;
Final boolean debug=false;
Final int number=8675309;
```

Sabitler, bir nesnenin değişik durumlarını ifade etmede ve nesneleri test etmede yararlı olabilirler. Örneğin sol, sağ, ya da orta özelliklerinin kullanılacağı text elemanı düşünülürse, sabit değişkenler aşağıdaki gibi tanımlanmalıdır.

```
Final int SOL=0;
Final int SAG=1;
Final int ORTA=2;
```

Bir sonraki işlemde text elemanının özellikleri aşağıdaki gibi ayarlanabilir.

```
This.alignment=ORTA;
 Switch (this.alignment)
{
Case SOL: //Sola dayalı
        Break;
Case SAG: //Sağa dayalı
        Break;
Case ORTA: //Ortalanmış
        Break; }
```

Aynı zamanda sabitleri kullanmak programın daha kolay anlaşılmasını da sağlar. Aşağıda bunla ilgili bir örnek bulunmaktadır.

```
This.alignment=ORTA;
This.alignment=2;
```

Örneğe bakılacak olursa, ilk ifadenin daha anlaşılır olduğu görülmektedir.

5.4. Metot Oluşturma

Bu bölümde metot tanımı ve tanımlanan metotların nasıl çalışacağı açıklanacaktır.

5.4.1. Metot tanımlama

Metot tanımlamanın 4 temel bölümü vardır.

- Metodun ismi
- Nesnenin tipi veya metottan dönen değerin tipi
- Parametre listesi
- Metodun ana kısmı

İlk metodun metot ismi olduğu söylenmişti. Java dilinde aynı sınıf içerisinde aynı isimde fakat dönüş tipi yada argüman listesi farklı olan birkaç metot vardır. Basit metot tanımlaması bu şekilde olmaktadır.

```
Dönüştipi metotismi (tip arg1, tip2 arg2….) {
//Metod komutları
}
```

Dönüş değeri primitive tip ya da metottan döndürülen değerin sınıfıdır. Eğer metot bir dönüş değeri içermiyorsa void ifadesi kullanılır. Bu metot bir dizi nesnesi döndürürse, dizi parantezleri dönüş tipinden sonra ya da parametre listesinden sonra konulabilir. Aşağıda bununla ilgili bir örnek bulunmaktadır. [15]

```
Int[ ] degeryap(int dusuk, int yuksek) {
//Metot komutları
}
```

Metot parametresi değişken tanımlamalarının ayar listesidir. Parantez virgülle ayrılırlar ve bu parametreler metodun ana kısmındaki yerel değişkenler olabilirler. Aynı zamanda metot çağrıldığı zaman bazı değerleri de alabilirler.

Eğer bir metodun dönüş değeri void kelimesi ile tanımlanmamışsa, metot birkaç değişik değer döndürür. Bu değer return deyimi kullanılarak metodun çıkış noktasında döndürülmelidir. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
Class RangeClass {
 İnt[] makeRange(int lower, int upper) {
        İnt arr[]=new int[ (upper-lower)+1];
        For (int i=0;i<arr.length; i++)
        {
              Arr[i]=lower++;              }
        return arr;              }
 public static void main(String arguments[])
        {
        int theArray[];
        RangeClass theRange=new RangeClass();
        theArray=theRange.makeRange(1,10);
        System.out.print("The array: [");
        For (int i=0;i<theArray.length; i++) {
              System.out.print(theArray[i] + " ");  }
        System.out.println("]");
        }
}
```

Program 5.1. Metot tanımlama örneği

Bu sınıf içerisinde bulunan main metodu 1 ile 10 arasında düşük ya da yüksek değerler oluşturarak makeRange metodunu test etmektedir ve daha sonra yeni dizi değerlerini ekranda görüntüleyebilmek için bir for döngüsü kullanmaktadır. Programın çıktısı aşağıdaki gibi olur.

The Array: [1 2 3 4 5 6 7 8 9 10] [15]

5.4.2. This anahtar sözcüğü

Metot tanımlamanın ana kısmında kullanılmakta olan nesneye, erişim sağlanmak istenebilir. This anahtar sözcüğü, içinde bulunulan nesneye ait bir referans değeri döndürür. Bu sayede nesnelere ait global alanlara erişme fırsatı bulunur. Aşağıdaki örnekte this yapısının kullanımı daha iyi anlaşılmaktadır.

```
t=this.x          // Bu nesnenin x değişkenini temsil eder
this.Data(this)  // Bu sınıf içerisinde tanımlanan Data metodunu  çağırır
return this;        // Bulunan nesneyi döndürür.
```

Aşağıda this anahtar sözcüğünü ile ilgili örnek bir program bulunmaktadır.

```
public class TarihHesaplama {
int gun, ay, yil;
public void gunEkle(int gun) {
   this.gun += gun ;  }
public void gunuEkranaBas() {
System.out.println("Gun = " + gun);  }
public static void main(String[] args) {
TarihHesaplama th = new TarihHesaplama();
th.gunEkle(2);  th.gunEkle(3);  th.gunuEkranaBas();
      }
}
```

Program 5.2. This anahtar sözcüğü örneği_1

gunEkle() prosedürü sayesinde parametre olarak gönderilen değer, global olan temel int tipindeki gun alanının değerini arttırmaktadır.

Nesnelere ait global alanlar, içinde bulundukları nesnelere ait alanlardır ve nesne içerisindeki her statik olmayan prosedür tarafından doğrudan erişilebilirler. Yerel değişkenler ise prosedürlerin içerisinde tanımlanırlar ve sadece tanımlandığı prosedür içerisinde geçerlidir.

gunEkle() prosedüründe, gun ismini hem TarihHesaplama nesnesine ait global bir alanının adı olarak hem de yerel değişken adı olarak kullanıldığı görülmektedir. Gönderilen değer, gunEkle() prosedürünün yerel değişkeni sayesinde TarihHesaplama nesnesinin global olan alanına eklenmektedir. Programın sonunda gunuEkranaBas() prosedürü ile global olan gun alanının değeri görülmektedir.

Özet olarak, gunEkle() prosedürünün içerisinde kullanılan this.gun ifadesiyle TarihHesaplama nesnesinin global olan gun alanına erişebilmektedir.

Programın çıktısı aşağıdaki gibi olur.

Gun = 5

Programda gunEkle() prosedürünün içerisindeki this.gun ifadesi yerine sadece gun ifadesi kullanılsaydı neler olurdu? Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
public class TarihHesaplama2 {
int gun, ay, yil;
public void gunEkle(int gun) {
   gun += gun ;
}
 public void gunuEkranaBas() {
System.out.println("Gun = " + gun);
}
```

```
    public static void main(String[] args) {
    TarihHesaplama2 th = new TarihHesaplama2();
    th.gunEkle(2);
    th.gunEkle(3);
    th.gunuEkranaBas();
    }
}
```

Program 5.3. This anahtar sözcüğü örneği_2

Uygulamanın çıktısı aşağıdaki gibi olur.

Gun = 0

TarihHesaplama nesnesine ait olan global gun alanına herhangi bir değer ulaşmadığı için sonuç sıfır olacaktır.[5]

5.5. Sınıf(Class) Metodları

Sınıf ve örnek değişkenler arasındaki ilişki direk olarak sınıfın ve örnek metotların nasıl çalıştığını karşılaştırabilmektedir.
Sınıf metotları kendi sınıfının örneğine uygundur ve diğer sınıflara da uygun hale getirilebilir. Örneğin Java sınıf kütüphaneleri Math olarak isimlendiren bir sınıf içerirler. Math sınıfı birçok programda kullanılan matematik işlemlerini tanımlamakta kullanılır. Aşağıda bununla ilgili bir örnek bulunmaktadır.

```
Float root=Math.sqrt(453.0);
System.out.print("X ve Y'nin en büyüğü "+ Math.max(x,y));
```

Sınıf metotlarını tanımlamada, tanımlama işleminden önce static anahtar sözcüğü kullanılır. Önceki örnekte kullanılan max() sınıf metodu aşağıdaki gibi olmalıdır.

```
Static int max(int arg1, int arg2) {
```

```
// Metot komutları
}
```

Başka sınıflar içerisinde tanımlanan sınıf metotları kullanılarak, nesneler primitive tiplere veya primitive tipler nesnelere çevrilebilir. Örneğin tamsayı sınıfındaki parseint() sınıf metodu string ile kullanılabilir. String, metoda bir argüman gibi gönderilir ve bu int gibi geri döndürülen dönüş değerini hesaplamak için kullanılır. Aşağıda parseInt() metodunun kullanımı görülmektedir.

```
int count =Integer.parseInt(“42”);
```

Önceki ifadede string “42” değeri tamsayı olan 42 değerinin parseInt() metodundan dönmesiyle olmuştur. [14]

5.6. Java Uygulamaları Oluşturmak

Java uygulamaları applet’lardan farklıdır. Applet’lar tarayıcı ile çalışabilmeleri için biraz daha fazla altyapı isterler.

Bir Java uygulaması bir veya daha fazla sınıftan oluşur ve istenildiği kadar büyük ya da küçük olabilir. Java uygulamaları pencereler, grafikler ve kullanıcı ara yüzü elemanları kullanarak oluşturulur. Uygulama için gereken başlangıç noktası sınıfı main() metodudur. Uygulama çalıştırıldığı zaman, ilk olarak main() metodu çağrılır. Main() metodu sürekli aşağıdaki gibi olur.

```
Public static void main(>String arguments[]) {
// Metot komutları
}
```

Aşağıda main() metodunun parçaları bulunmaktadır.

- Public deyimi, diğer sınıf ve nesnelere ulaşabilme anlamına gelir. Main() metot public olarak ifade edilmelidir.
- Static deyimi, main() metodunun bir sınıf metodu olduğu anlamına gelir.

- Void deyimi, main() metodunun geriye bir değer döndürmeyeceği anlamına gelir.
- Main() sınıfı string dizileri içeren bir parametre alır. Bu argüman program argümanları için kullanılır.

Main() metodu, uygulamayı başlatmak için gereken değişkenlerin başlatılması ya da sınıf örneklerinin oluşturulması gibi bir kod içerir. [14]

5.7. Sınıf Değişkenlerinin Dış Dünyadan Gizlenmesi

Private olarak tanımlanan değişkenlere dışarıdan doğrudan,

Kutular x=new Kutular(3.0,5.0,7.0);
x.en=8.0;

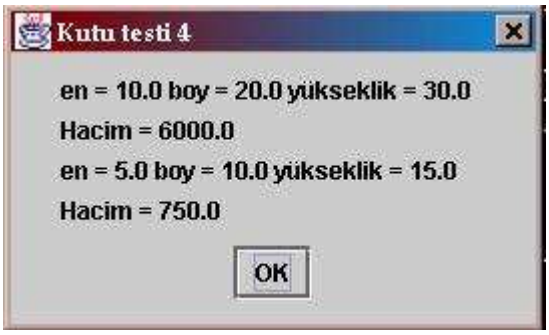
gibi doğrudan ulaşım mevcut değildir. Bu değişkenlere ulaşım ancak ulaşım metotları adı verilen metotlar aracılığıyla yapılabilir. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
import javax.swing.JOptionPane; // giriş çıkış
 class Kutular   {
private double en;
private double boy;
private double yukseklik;
 //kurucu metod
Kutular(double e,double b,double y) {
en=e;
boy=b;
yukseklik=y;
}
 //sınıf değişkenlerini değiştirme set metodu
public void setKutular(double e,double b,double y) {
en=e;
```

```
boy=b;
yukseklik=y;
}
//sınıf değişkenlerine ulaşma metotları
public double getEn()
{return en;}
 ublic double getBoy()
{return boy;}
 public double getTukseklik()
{return yukseklik;}
 //bu metot kutunun hacmini hesaplar
public double hacim() {
return en*boy*yukseklik; }
 //string çıktı metodu
public String toString() {
return("en = "+en+" boy = "+boy+" yükseklik = "+yukseklik+"\n"+
        "Hacim = "+hacim()+"\n");  }
 }
 class kututesti4      {
   public static void main(String args[])       {
   double hacim;
   String s="";
   Kutular kutu1=new Kutular(10.0,20.0,30.0);
   s+=kutu1.toString();
   Kutular kutu2=new Kutu(5.0,10.0,15.0);
   s+=kutu2.toString();
      JOptionPane.showMessageDialog(null,s,
     "Kutu testi 4",JOptionPane.PLAIN_MESSAGE);
   System.exit(0);
   }
}
```

Program 5.4. Sınıf değişkenlerinin gizlenmesi[10]

Yukarıdaki örnekte kutular adında bir sınıf oluşturulmuştur. Bu örnek girilen boy ve yükseklik değerlerine göre hacim değerini vermektedir. Şekil 5.1'de programın ekran çıktısı görünmektedir.



Şekil 5.1. Sınıf değişkenleri örneği

5.8. Metotları Aynı İsimle Ve Değişik Argümanla Oluşturma

Java sınıf kütüphanesinde çalışılırken, sık sık aynı isimde birçok metot sınıflarıyla karşılaşılır. Örneğin java.lang.String sınıfında birkaç değişik valueOf() metodu vardır.

Aynı isimdeki metotlar birbirlerinden iki noktada farklılaşırlar.

- Aldıkları argümanların sayıları
- Her argümanın nesneleri veya veri tipi

Bu iki nesne metodun imzasını oluşturur ve aynı isimdeki birkaç metot ve değişik imzalar kullanmak overloading olarak isimlendirilir.

String örneğinde değişik overload valueof() metotları, parametre gibi değişik veri tipleri alırlar. Overloading aynı zamanda metotların değişik bir şekilde davranmalarını mümkün kılar.

Overload metodu oluşturmak için sınıf içerisinde, her biri aynı isimde fakat argümanları farklı metot tanımlamaları oluşturulmalıdır. Metotlar arasındaki fark, sayılar, argümanların tipi ya da her ikisi beraber olabilir. Eğer aynı isimde ve değişik

dönüş değerlerinde iki metodun oluşturulması denenirse, sınıf derlenmeyecektir. Buna ek olarak her argüman için seçilen değişken isimleri önemsiz olacaktır. Aşağıda MyRectangle olarak adlandırılan sınıf için basit bir overload metot örneği bulunmaktadır. Burada MyRectangle, dikdörtgen şeklini tanımlayan bir sınıf yapısıdır.

```
Class MyRectangle {
int x1=0;
int y1=0;
int x2=0;
int y2=0;  }
```

MyRectangle sınıfı oluşturulduğunda tüm değişken değerleri 0 olarak başlatılmıştır. BuildRectangle metodu da, dikdörtgenin 2 köşesinin doğru değerlerini ayarlamak için kullanılır. Bu metotta 4 tamsayı argümanı vardır ve dikdörtgen nesnesinin sonucu dönmektedir. Çünkü argümanlar örnek değişkenler gibi aynı isimlere sahiptir. Burada değerleri değiştirebilmek için this anahtar sözcüğü kullanılmıştır.

```
MyRect buildRectangle (int x1, int y1, int x2, int y2) {
this.x1=x1;
this.y1=y1;
this.x2=x2;
this.y2=y2;
return this;      }
```

Bu metot dikdörtgen oluşturmakta kullanılabilir. Başka bir alternatif yolda Point nesnelerini kullanmaktır. Bu alternatifi yerini getirmekle, buildRectangle() sınıfını aşırı yüklemektedir. Bu yüzden argüman listesinde 2 tane Point nesnesi vardır. Aşağıda bunla ilgili bir örnek bulunmaktadır.

```
MyRect buildRectangle(Point topleft, Point bottomRight) {
x1=topleft.x;
y1=topleft.y;
```

```
x2=bottomRight.x;
y2=bottomRight.y;     }
```

Bir önceki metodun çalışabilmesi için, Point sınıfı, kaynak dosyanın üst tarafına eklenmelidir. Bu durumda Java onu bulacaktır.

5.9. Yapı Metotları

Düzgün metotlara ek olarak, sınıf tanımlamada yapıcı metotlar tanımlanabilir. Yapıcı metot, bir nesne oluşturulduğu zaman çağrılan bir metottur. Diğer bir deyişle yapılırken çağrılan bir metottur. Diğer metotlar da olduğu gibi yapıcı metot doğrudan çağrılamaz. Bunun yerine Java yapıcı metotları otomatik olarak çağırır.

New komutunun kullanıldığı bir sınıf oluşturulması için zaman 3 nesne kullanılır.

- Nesne için hafıza ayırmak.
- Nesnelerin örnek değişkenlerini başlatmak
- Birkaç metottan oluşan sınıfın yapıcı metodunu çağırmak.

Eğer bir sınıfta hiç yapıcı metot tanımlanmamışsa, nesne hala new deyimi, sınıfla beraber kullanıldığı zaman oluşturulur.

Sınıflardaki yapıcı metotların tanımlanmasıyla, örnek değişkenlerin ilk değerleri ayarlanabilir. Aynı zamanda düzgün metotlarla yapılabildiği gibi, yapıcı metotlarla da yapılabilir.

Yapıcılar, iki farklılık dışında düzgün metotlar gibi davranırlar.

- Yapıcı metotlar sürekli sınıf ile aynı isme sahip olurlar.
- Yapıcı metotlar da dönüş değeri dönmez.

Aşağıda bununla ilgili basit bir örnek program bulunmaktadır.

```
Class Person {
  String name;
```

```
Int age;
Person(String n, int a) {
        name=n;
        age=a;
}
void  printPerson() {
        System.out.print("Merhaba benima dım" +name);
        System.out.println("  Ben"+age +"yaşındayım");
}
public static void main (String arguments[]) {
        Person p;
        P=new Person("Luke", 50);
        p.printPerson();
        System.out.println("-----");
P=new Person("Laura", 35);
p.printPerson();

System.out.println("-----");
        }
}
```

Program 5.5. Yapıcı metod örneği

Aşağıda programın çıktısı görülmektedir.

Merhaba, benim adım Luke. Ben 50 yaşındayım.

Merhaba, benim adım Laura. Ben 35 yaşındayım.

Person sınıfında 3 tane metot bulunmaktadır. Birincisi 5 ve 8. satırlar arasında bulunan Person yapısı üzerine bina edilmiş yapıcı metottur. Person sınıfı aynı zamanda printPerson() adında bir metot içerir. Böylece nesne kendini tanımlar ve main() metodu bu nesnelerin her birini test eder. [15]

BÖLÜM 6. APPLETLAR

6.1. Applet Nedir?

JAVA applet'leri Web tarayıcı sayfalarında çalışan küçük programlardır. Aslında bu cümlenin çok az bir kısmı doğrudur. Applet tam olarak bir program değildir. Applet'lerin birçoğu web sayfalarında kullanılmasına rağmen, başka yerlerde de kullanılabilir. Applet, Java.applet.Applet sınıfına veya onun alt sınıflarına ait bir nesnedir.

Applet grafiksel kullanıcı arabiriminin bir parçasıdır. Aynı zamanda bir pencere içerisinde görüntülenebilen bir grafiksel bileşen tipidir. Applet bir pencere içerisinde görüntülendiğinde, içerisinde buton, textbox gibi bileşenler bulunan dikdörtgensel bir yapıda gözükür.

Applet sınıfı, Java.applet paketi içerisinde tanımlanmıştır. Yararlı bir applet oluşturmak için, programcı applet sınıfının uzantısı olan altsınıfı tanımlamalıdır. Applet sınıfı içerisinde bir şey yapmamak için tanımlanmış birkaç metot vardır. Programcı en azından bu metotların birkaçını başka işler için kullanmalıdır.

Java applet'inin main() ana programına ihtiyacı yoktur. Bununla beraber, applet'teki metotların birçoğu sistem tarafından isimlendirilen main() yapısına benzerdir ve programcının görevi, sistem çağırmalarına verilen cevabın ne olduğunu söylemektir.

Applet sınıfında tanımlanan metotların bir tanesi de paint() metodudur. Paint() metodu applet'in çizme işlemini yapması gerektiği zaman, sistem tarafından çağrılır. Applet'in alt sınıfında, paint() metodu, applet üzerindeki dikdörtgen, çizgi ve yazı gibi çeşitli grafiksel elemanları çizmek için tekrar tanımlanabilir. Bu metodun tanımı aşağıdaki gibi olmalıdır.

```
public void paint(Graphics g) {
// Çizim komutları
 }
```

Graphics tipinin g parametresi, paint() metodunu çağırdığı zaman sistem tarafından sağlanır. Java dilinde, tüm çizim çeşitleri Graphics nesnesi tarafından sağlanan metotlar kullanılarak yapılır.

Appletin Paint() metodu aslında buton, textbox, input kutuları gibi GUI bileşenlerini çizmek için kullanılır. Tüm bileşen nesnelerinin paint() metodu vardır. Her bileşen kendi paint() metoduyla kendisini çizmekle sorumludur.[16] Tablo 6.1'de applet'in metotları gösterilmiştir.

Tablo 6.1 Applet metotları [17]

public void init ( )	Bu metot appletviewer veya browser tarafından applet yüklendiği zaman çağrılır. Appletin çalışması ile çalışır. Örnek değişkenleri ve bir applete ait GUI bileşenlerini başlatmak, gösterilecek resim gibi nesneleri yüklemek için kullanılır.
Public void start ( )	Bu metot başlangıç metodu çalışıp, işlenmesi bittikten sonra tekrar kullanıcının appletin yüklü olduğu HTML sayfasına dönmesiyle birlikte çalışmaya baslar.
public void paint (Graphics g)	Bu metot, init() metodu çalışmasını bitirdikten ve start() metotu çalışmaya başladıktan sonra appletin üzerine bir şeyler çizmek için çalışmaya baslar. Aplet tekrar boyanacak olduğu zaman tekrar otomatik olarak aktif hale gelir.
public void stop ( )	Applet çalışmasını durdurduğunda veya kullanıcı HTML sayfasını terk ettiğinde veya kullanıcı HTML sayfasını terk ettiği zaman çalışmaya baslar.
public void destroy ( )	Applet bellekten ayrıldığı zaman çalışmaya başlar. Applet için ayrılmış kaynakları yok eder.

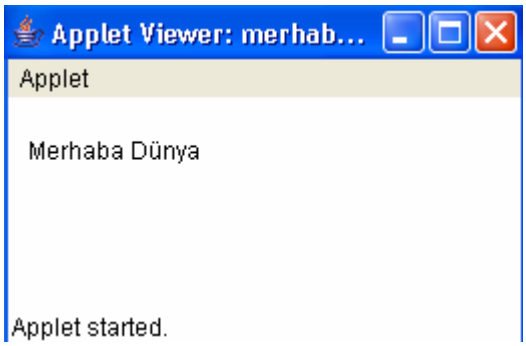
Aşağıda bulunan ilk applet ekrana Merhaba Dünya yazısını çıkarmaktadır. Bu ifadeyi ekranda göstermek için paint() metodu kullanılmaktadır. Başlangıçta bulunan import ifadeleri, java.applet.applet ve java.awt sınıflarındaki tam isimlerin yerine applet ve

graphics kısa isimlerini kullanmaya izin verir. [15] Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
import java.awt.*;
import java.applet.*;
 public class MerhabaDunyaApplet extends Applet {
// Merhaba Dünya yazısını gösteren applet
 public void paint(Graphics g) {
 g.drawString("Merhaba Dünya!", 10, 30);
     }
  }  // MerhabaDunyaApplet sınıfı sonu
```

Program 6.1.merhaba Dünya Appleti

Şekil 6.1'de programın ekran çıktısı şu görünmektedir.



Şekil 6.1. Merhaba dünya applet'i

Graphics sınıfında tanımlanmış olan drawString() metodu aslında çizme işlemini yapar. Bu metodun parametreleri çizilecek metini ve metinin yerleştirildiği yerdeki applet'in noktasını açıkça belirtir.

6.2. Appleti Web Sayfasına Gömme

Applet bir nesne ya da sınıf değildir. Applet oluştururken applet'in daha sağlıklı bir şekilde kullanılabilmesi için tanımlanması gerekir. Aşağıda bununla ilgili bir örnek bulunmaktadır.

Applet hw = new HelloWorldApplet();

Bir program yazılıyorsa ve oluşturulan pencereye applet eklenmek isteniyorsa, bu yapı çok yararlı olabilir. Bununla birlikte, applet nesneleri sistem tarafından oluşturulur. Örneğin web tarayıcısındaki sayfada bir applet göründüğü zaman, sistem web tarayıcı demektir. Burada tarayıcı program applet nesnesini oluşturur ve onu web sayfasına ekler. Web tarayıcı, sayfanın içerisinde applet nesnesinin nasıl göründüğü ile ilgili bilgileri alır. Web sayfasında bir applet'in görüntülenmesi için, sayfanın kaynağı applet'in ismini ve boyutunu açıkça belirtmelidir. Bu ifadeler HTML kodu içerisine yazılır. Aşağıda MerhabaDunyaApplet applet'ini web tarayıcısında görüntülemek için kullanılan HTML kodları gözükmektedir.

```
<center>
<applet code=" MerhabaDunyaApplet.class" width=200 height=50>
</applet>
</center>
```

Eğer tarayıcı Java destekli değilse veya Java desteği kapatılmışsa, herhangi bir şey görünmeyecektir. Bunun haricinde "Merhaba Dünya" mesajı görünecektir. Mesaj 200 piksel genişliğinde ve 50 piksel yüksekliğinde bir dikdörtgen içinde görüntülenir.

Applet nesnesi içerisinde öncelikle sistem init() metodunu aşağıdaki şekilde çağırır.

public void init() { . . . }

Bu metot genellikle bir yapı tarafından yapılan, appletin örnek değişkenlerinin başlatılması gibi işleri gerçekleştirebilir. Init() metodu aynı zamanda applet'e eklenen buton gibi diğer bileşenlerin bulunduğu yerdir.

Kullanıcı applet içerisindeki butona tıkladığı zaman, bir olay oluşturur. Applet, bu olaya yanıt vermeye programlanabilir. Bu durumda buton tarafından meydana gelen olayın tipi ActionEvent çağrılır. Appletin bu olaya cevap vermesi için public void actionPerformed(ActionEvent evt) { . . . } şeklinde tanımlanması gerekir.

Ayrıca, butondan meydana gelecek olaylar için applet listening modunda olmalıdır. Bu işlem buton nesnesinin örnek metotlarının birinin çağrılmasıyla yapılır. Bu metotlara örnek olarak addActionListener ve applet içerisindeki init() metodu verilebilir.

Paint() metot sürekli çağrılabilir ve her seferinde yeniden adlandırılır. Applet içerisinde bir mesaj yazılmak istenildiğinde, bu mesaj doğru renklerde çizimle oluşturulur. Çizimi oluşturmak için kullanılacak renkler nesne içerisindeki değişkenlere depolanmaktadır. Paint() metodu, metot mesajı oluşturmak için kullanılacak olan renklere karar vermek için, değişkenin değerini kontrol eder. Gerçekte, actionPerformed() metodu çizimin hiçbirini yapamaz. Ancak örnek değişkeni ayarlar ve repaint() metodunu çağırır. Aşağıda bununla ilgili örnek bir program bulunmaktadır. [15]

```
import java.awt.*; // GUI programlama için temel //sınıfları tanımlar.
import java.awt.event.*;  // Olaylarla çalışmak için
//kullanılacak sınıfları tanımlar.
    import java.applet.*;     // Applet sınıfını tanımlar
    public class ColoredMerhabaDunyaApplet
                extends Applet implements ActionListener
    int colorNum;     // Hangi rengin görüntülendiğini belirtir.
                //     Kırmızı için 1, mavi için 2, yeşil için 3.
    Font textFont;    // Görüntülenen mesajın fontunu belirler.
                      // Fontun gerçek boyutunu belirler.
    public void init()       // Bu sistem tarafından başlatılır.
      setBackground(Color.lightGray);
      // Applet Paint() metotu çağrılmadan önce zemin rengi ile doldurulur.
     // Buton ve applet içerisindeki mesaj gri zemin üzerinde görünecektir.
      colorNum = 1;     // Mesajın rengi kırmızı olarak ayarlanmaktadır.
```

```
        textFont = new Font("Serif",Font.BOLD,24);
        Button bttn = new Button("Renk değiştir.");
            // Yeni bir buton oluşturulmaktadır.
        bttn.addActionListener(this);
            // Kullanıcı butona tıkladığı zaman Buton applete olay
         // yollamaktadır.
        add(bttn);  // Applete buton eklemektedir.  Bu şekilde ekranda
              //görünecektir.
    }  // init() sonu
    public void paint(Graphics g) {
         // Burada Merhaba Dünya yazısı görüntülenmektedir.
      switch (colorNum) {         //Renk ayarlanmaktadır.
        case 1:
          g.setColor(Color.red);
          break;
        case 2:
          g.setColor(Color.blue);
          break;
        case 3:
          g.setColor(Color.green);
          break;          }
      g.setFont(textFont);       // Font ayarlanmaktadır.
      g.drawString("Merhaba Dünya!", 20,70);    // Mesaj çizilmektedir
    }  // paint() sonu
    public void actionPerformed(ActionEvent evt)
{
            // Kullanıcı butona tıkladığı zaman Buton applete olay
            // yollamaktadır
      if (colorNum == 1)       // Colornum değiştirilmektedir.
        colorNum = 2;
      else if (colorNum == 2)
        colorNum = 3;
      else
```

```
            colorNum = 1;
          repaint();
        }  // init() sonu
      } // Sınıfın sonu
```

Program 6.2. Renk değiştirme appleti

Şekil 6.2'de programın ekran çıktısı görünmektedir.

Şekil 6.2. Renk değiştirme applet'i

6.3. Applet Etiketi ve Applet Parametreleri

Applet'ler genellikle herhangi bir web tarayıcı programındaki sayfalarda görüntülenir. Web sayfaları HTML(HyperText Markup Language) olarak adlandırılan bir dilde yazılır. Bir HTML belgesi sayfanın içeriğini tanımlar. Web tarayıcısı HTML kodlarını yorumlayarak, ekranda ne görüntüleneceğine karar verir. HTML belgesi sayfa üzerinde görünen tüm yazıları içerir.

<APPLET> etiketi Java applet'ini web sayfasına eklemek için kullanılır. Bu etiketin sonunda </APPLET> yapısı bulunmalıdır. HTML kodları arasında bulunan APPLET CODE deyimiyle de derlenecek olan sınıf tanımlanmaktadır. HEIGHT ve WIDTH deyimleri de applet'in boyutunu belirtmek için gereklidir. Eğer applet'in sayfanın ortasında olması isteniyorsa, appleti CENTER deyimiyle kullanmak gerekir. Aşağıda HTML kodları arasında bir applet tanımlaması gözükmektedir.

```
<P ALIGN=CENTER>
 <APPLET CODE="MerhabaDunyaApplet.class" HEIGHT=50
WIDTH=150>
</APPLET>
</P>
```

Burada MerhabaDunyaApplet ve HTML dosyası aynı konumda olmalıdır. Eğer sınıf dosyası ile HTML dosyası farklı yerlerde olursa, CODEBASE adında başka bir deyim kullanmak gerekir.

Eğer bir applet birçok sınıf dosyası kullanıyorsa, dosyaların hepsini tek bir. zip veya. jar dosyasına çevirmek daha iyi olacaktır. Java derleme programları genellikle bunu kendileri oluştururlar. Eğer sınıf dosyaları arşiv içerisindeyse, arşivin ismi <APPLET> deyimi içerisinde ARCHIVE deyimiyle açıkça belirtilmelidir. Arşiv dosyalar eski tarayıcılarda çalışmayacaktır, fakat Java 1.1 desteği olan diğer tüm tarayıcılarda çalışacaktır. Applet'ler davranışlarını ayarlamak için bazı parametreler kullanılırlar. Applet parametreleri <PARAM> deyimi kullanılarak belirtilmelidir. <PARAM> deyimi <APPLET> deyimi ile </APPLET> deyimi arasına yazılmalıdır. PARAM deyimi NAME ve VALUE deyimlerine ihtiyaç duyar. Aşağıda PARAM ifadesine bir örnek bulunmaktadır.

```
<PARAM  NAME="param-name"  VALUE="param-value">
```

Applet çalıştığı zaman parametreler kullanılabilir. Applet, PARAM deyiminde belirtilen parametreleri kontrol etmek için getParameter() metodunu kullanabilir. Parametrenin kullanımı aşağıdaki gibi olmaktadır.

```
String getParameter(String paramName)
```

ParamName parametresi, PARAM deyimindeki param-name ile uyuşmaktadır. Eğer belirtilen paramName gerçekten PARAM deyimlerinin birinde bulunursa, getParameterden ortak param-value değeri döner. Eğer belirtilen paramName herhangi bir PARAM deyiminde bulunmazsa, getParameterden null değeri döner.

Parametre isimleri büyük-küçük harf duyarlıdır. Bu yüzden ifadeler kullanılırken çok dikkat edilmelidir. Eğer <APPLET> ve <APPLET/> arasındaki PARAM deyimleri arasına bazı deyimler yazılırsa, bu, Java destekleyen herhangi web tarayıcı tarafından görülmeyecektir. Diğer bir deyişle, Java desteği olmayan web tarayıcıları APPLET ve PARAM deyimlerini görmezden gelecektir. Bu deyimlerin Arasına "Web tarayıcınız Java'yı desteklemiyor" mesajı yazılırsa, bu mesaj Java desteği olmayan web tarayıcılarında görünecektir. Aşağıda bununla ilgili bir örnek bulunmaktadır.

```
<APPLET code="ShowMessage.class" WIDTH=200 HEIGHT=50
<PARAM NAME="message" VALUE="Hoşça kal Dünya!">
<PARAM NAME="font" VALUE="Serif">
 <PARAM NAME="size" VALUE="36">
 <p align=center> Web tarayıcınız JAVA'yı desteklemiyor </p>
 </APPLET>
```

Mesajgöster appleti, init() metodu içerisinde bu parametreleri okur. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
String display;
String fontName;
public void init() {
   String value;
   value = getParameter("mesaj");
   if (value == null)
     display = "Merhaba Dünya!";
     display = value;
     value = getParameter("font");
   if (value == null)      fontName = "SansSerif"
   else
     fontName = value;
    ---
```

Program 6.3 Param deyimi örneği [18]

6.4. Graphics ve Paint Metotları

Bilgisayar ekranında görülen her şey, text dahi ekrana çizilmelidir. Java apisi sınıf dizilerini ve çizime bağlı metotları içermektedir. Program applet'in içerisinde bulunan birkaç bileşenden biri olan canvas içerisinde çizilir. Canvas sınıfı, Applet sınıfı ve tam olarak GUI bileşenlerini temsil eden sınıfların hepsi bileşen olarak adlandırılan diğer sınıfı alt sınıflarıdır. Bileşen sınıfı ekranda görünen GUI bileşenini genel olarak temsil eder.

Java dilinde herhangi bir çizim yapabilmek için, bir grafik yapısına ihtiyaç vardır. Grafik yapısı, graphics sınıfa uygun olan bir nesnedir. Örnek metotlar, bu sınıfta resim, metin ve şekillerin çizimi için kullanılır. Verilen Graphics nesnesi sadece bir yer çizebilir. Graphics sınıfı, direkt olarak bir yapıcıyla grafik yapısı oluşturmanın imkansız olacağı anlamına gelen soyut bir sınıftır. Bir bileşenin çizilebilmesi için iki yöntem vardır. İlk olarak, bir bileşenin paint() metodu çağrıldığı zaman, bu parametre metodu, bileşenin çizimini yapar. İkincisi ise paint() metodu olmadan çizimi yapmaktır. Her bileşenin getGraphics() olarak adlandırılan bir örnek metodu vardır. Bu metot bileşen için grafik şeklini döndüren bir fonksiyondur.

Örnek metot, getGraphics() bileşen sınıfında tanımlanır. Buradan çizim için kullanabilen bir grafik yapısı döndürür. Eğer comp herhangi bir bileşen ve eğer g Graphics tipi değişkeni ise bu ifade aşağıdaki gibi yazılabilir.

```
g = comp.getGraphics();
```

Bu atamadan sonra, g comp bileşenini temsil eden ekranın dikdörtgen alanına çizim yapmak için kullanılabilir. Basit bir şekilde g=getGraphics() olarak yazılabilir. Bu ifade yazılan bileşendeki çizim için grafik yapısı verir.

Eğer g getGraphics() metoduyla elde edilen bir grafik yapısı ise, onu kullanmayı bitirdikten sonra g.dispose() metodunu çağırmak iyi bir fikir olacaktır. Bu kaynakların sınırlı olduğu çoğu sistemde iyi bir fikirdir. Bunun yanında, paint() metoduyla sağlanan grafik bağlamı için hiçbir zaman dispose() çağrılmaz. [15]

Aşağıda appletler ile ilgili örnek programlar bulunmaktadır.

6.5. Rastgele Şekil Çizimi

```
package scroll;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class grafik extends Applet {
  private boolean isStandalone = false;
  public void paint(Graphics g) {
Font ff=new Font("Helvetica",Font.PLAIN,10); g.setFont(ff);
g.drawRect(10,10,70,70);  g.fillRect(10,100,70,70);
g.drawString("drawRect",10,90); g.drawString("fillRect",10,180);
g.setColor(new Color(230,56,2));
g.drawRoundRect(100,10,70,70,20,30);
g.fillRoundRect(100,100,70,70,40,10);
g.drawString("drawRoundRect",100,90);
g.drawString("fillRoundRect",100,180);
g.setColor(Color.red);
g.draw3DRect(180,10,70,70,true);  g.draw3DRect(180,100,70,70,false);
g.drawString("draw3DRect",180,90);  g.drawString("fill3DRect",180,180);
int xler[]= {289, 344, 347, 392, 303, 308, 276 };
int yler[]= {3, 44, 6, 40, 78, 50, 76 };
int yl[]= {103, 144, 106, 140, 178, 150, 176 };
int pts=xler.length;
Color c=new Color(0,250,0);  g.setColor(c);
g.drawPolygon(xler,yler,pts);      g.fillPolygon(xler,yl,pts);
g.drawString("drawPolygon",300,100);
g.drawString("fillPolygon",300,190);
g.setColor(Color.gray);
g.drawArc(400,10,80,60,30,330);   g.fillArc(400,100,80,60,100,300);
g.drawString("drawArc",400,90);  g.drawString("fillArc",400,180);
```

```
g.setColor(new Color(30,56,250));
g.drawOval(500,10,80,60);        g.fillOval(500,100,60,80);
g.drawString("drawOval",500,90);    g.drawString("fillOval",500,200);
Font f=new Font("Arial",Font.PLAIN,14);
Font fB=new Font("TimesRoman",1,16);
Font fI=new Font("Helvetica",Font.ITALIC,18);
Font fBI=new Font("Tahoma",Font.BOLD+Font.ITALIC,20);
g.setColor(Color.blue);g.setFont(f);
 g.drawString("Bu düz Arial düz fontla yazılmıştır. 14 punto ",10,200);
g.setColor(new Color(0,255,0));g.setFont(fB);
g.drawString("Bu koyu TimesRoman fontla yazılmıştır. 16 punto",10,230);
g.setColor(Color.yellow);g.setFont(fI);
g.drawString("Bu italik Helvetica fontla yazılmıştır. 18 punto ",10,260);
g.setColor(Color.gray);g.setFont(fBI);
g.drawString("Bu koyu ve italik Tahoma fontla yazılmıştır. 20 punto
",10,290);
setBackground(new Color(230,250,230));
g.setColor(Color.red);
g.fillRect(50,300,200,140);
g.setColor(Color.white);
g.fillOval(70,320,100,100);
g.setColor(Color.red);
g.fillOval(100,330,80,80);
// TURK BAYRAGI CIZIMI
int i,faz,aci,icfaz,xm,ym,dr,ir  ;
int koordx[]=new int[10];
int koordy[]=new int[10];
 faz=36;  aci=(int) (360/5);  icfaz=(int)(aci/2); xm=185; ym=370; dr=30; ir=10;
for(i=0;i<koordx.length;i+=2)  {
koordx[i] = xm + (int) ( dr * Math.cos(Math.PI * faz / 180));
koordx[i+1] = xm + (int) ( ir * Math.cos(Math.PI * (icfaz+faz) / 180));
koordy[i] = ym + (int) ( dr * Math.sin(Math.PI * faz / 180));
koordy[i+1] = ym + (int) ( ir * Math.sin(Math.PI * (icfaz+faz) / 180));
```

```
faz+=aci;
}
g.setColor(Color.white);   g.fillPolygon(koordx,koordy,koordx.length);
g.setFont(new Font("Arial",0,20)); g.setColor(Color.black);
g.drawString("TURK BAYRAGI",270,350);
}
}
```

Program 6.4. Rasgele çizim appleti[19]

Yukarıdaki applet örneğinde çeşitli grafikler çizilmektedir. Çizimlerin altında çizimle ilgili bilgi verilmektedir. G.drawString komutuyla metinsel bilgiler yazdırılmaktadır. g.setColor komutuyla da çizimin rengi ayarlanmaktadır. Şekil 6.3'te bu applet'in ekran görüntüsü görünmektedir.

Şekil 6.3. Rasgele çizim applet'i

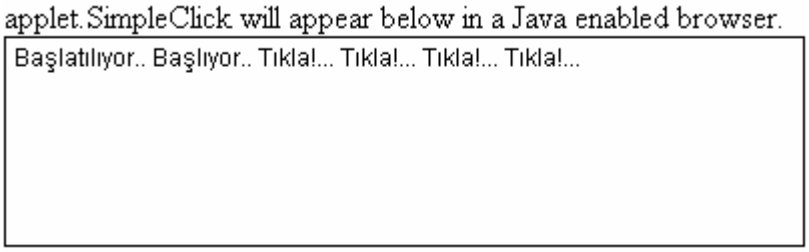
6.6. Mouse Applet'i

```
package applet;
import java.awt.*;
```

```
import java.awt.event.*;
import java.applet.*;
Public class SimpleClick extends Applet {
  private boolean isStandalone = false;
  StringBuffer buffer;
  public void init() {
     buffer = new StringBuffer();
     addItem("Başlatılıyor.. ");
  }
  public void start() {
     addItem("Başlıyor.. "); }
  public void stop() {
     addItem("Duruyor... "); }
  public void destroy() {
     addItem("Çıkarılmaya hazırlanıyor..."); }
  void addItem(String newWord) {
     System.out.println(newWord);
     buffer.append(newWord);
     repaint(); }
  public void paint(Graphics g) {
     g.drawRect(0, 0, size().width - 1, size().height - 1);
     g.drawString(buffer.toString(), 5, 15);
  }
  public boolean mouseDown(Event event, int x, int y) {
     addItem("Tıkla!... ");
     return true;
  }
}
```

Program 6.5. Mouse applet'i [20]

Yukarıdaki applet örneğinde applet ekranı üzerine tıklandığında tıkla mesajını yazmaktadır. Bu programın amacı Mouse tıklama olayının anlaşılmasıdır. Şekil 6.4'te bu appletin ekran görüntüsü görünmektedir.



Şekil 6.4. Mouse appleti

6.7. Grafik Applet'i

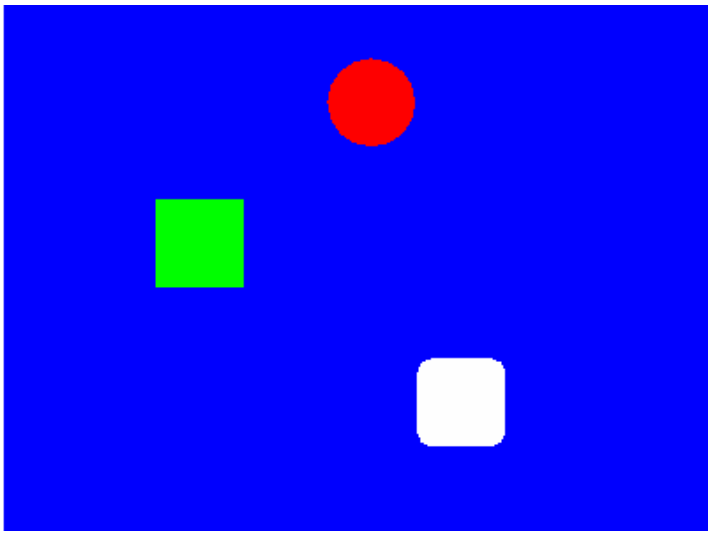
```
import java.awt.*;
import java.applet.*;
public class Project9 extends Applet implements Runnable
{
   Thread runner;
   private Image Buffer;
   private Graphics gBuffer;
   int x, x1, x2=100;
   boolean moveRight=true;
   //init() metotu ilk olarak çalıştırılır
   public void init()
   {
      //Applet için grafik bufferını hazırla
      Buffer=createImage(size().width,size().height);
      gBuffer=Buffer.getGraphics();
   }
   public void start()
   {
      if (runner == null)
```

```
        {
            runner = new Thread (this);
            runner.start();
        }
    }
    public void stop()
    {
        if (runner != null)
        {
            runner.stop();
            runner = null;
        }
    }
    public void run()
    {
        while(true)
        {
            //15 ms beke
            try {runner.sleep(15);}
            catch (Exception e) { }
            //paint background blue
            gBuffer.setColor(Color.blue);
            gBuffer.fillRect(0,0,size().width,size().height);
            x++;
            if(x>size().width)
                x=-50;
            x1-=2;
            if(x1<-50)
                x1=size().width
if(moveRight==true)
            {
                if(x2<size().width-50)
                    x2+=2;
```

```
                else
                    moveRight=false;
                }
               if(moveRight==false)
                {
                   if(x2>0)
                      x2-=2;
                   else
                      moveRight=true;
                }
                gBuffer.setColor(Color.red);
                gBuffer.fillOval(x,30,50,50);
                gBuffer.setColor(Color.green);
                gBuffer.fillRect(x1,110,50,50);
                gBuffer.setColor(Color.pink);
                gBuffer.fillRoundRect(x2,200,50,50,16,16);
                repaint();
            }
        }
        public void update(Graphics g)
        {
           paint(g);
        }
        public void paint(Graphics g)
        {
           g.drawImage (Buffer,0,0, this);
        }
    }
```

Program 6.6. Grafik appleti örneği [21]

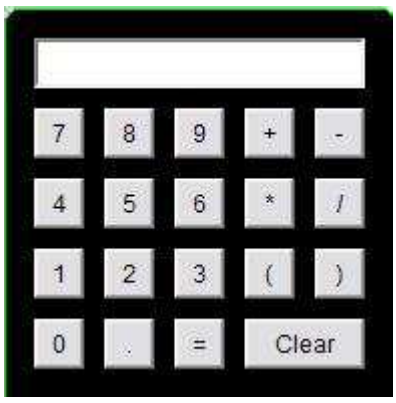
Yukarıdaki örnek daire, oval kare ve kare animasyonlarını içermektedir. Şekiller kendi başlarına hareket etmektedir. Programın kodları yukarıda görünmektedir. Şekil 6.5'te bu applet'in ekran görüntüsü görünmektedir.



Şekil 6.5. Grafik applet'i

6.8. Hesap Makinesi Applet'i

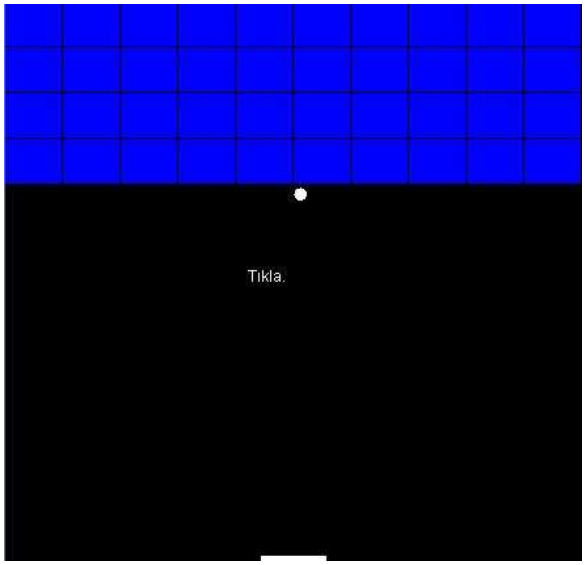
Şekil 6.6'da hesap makinesi applet programının ekran görüntüsü görünmektedir. Bu applet programı bir hesap makinesinde bulunan tüm işlemleri rahatlıkla yapabilmektedir. Sayılara Mouse ile tıklayarak sayılar gerekli yere yazılır. Daha sonra işlem seçilerek eşittir düğmesine tıklanır. Böylece sonuca varılmış olur.



Şekil 6.6. Hesap makinesi applet'i [22]

6.9. Blok Oyunu Applet'i

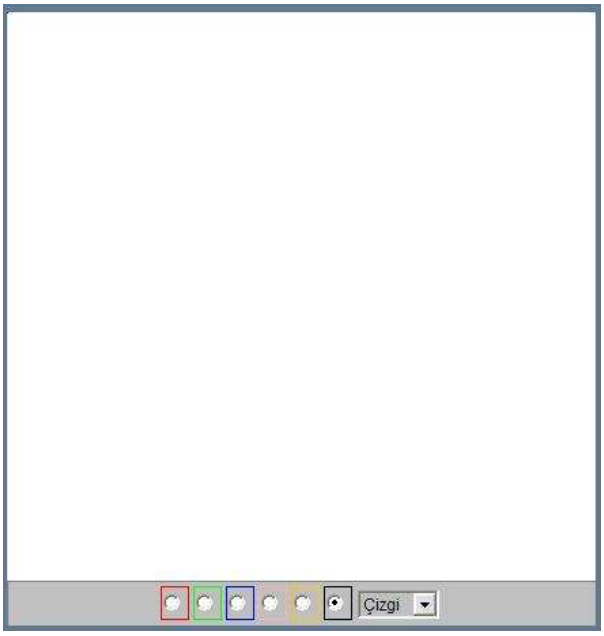
Bu oyun applet'i eskiden beri oynanan blok patlatma oyunun Java dilinde hazırlanmış şeklidir. Mouse ile tıklanır ve daha sonra boşluk tuşuna basılarak oyun başlatılır. Şekil 6.7'de blok oyununun ekran görüntüsü görülmektedir.



Şekil 6.7. Blok oyunu applet'i [23]

6.10. Rasgele Çizim Applet'i

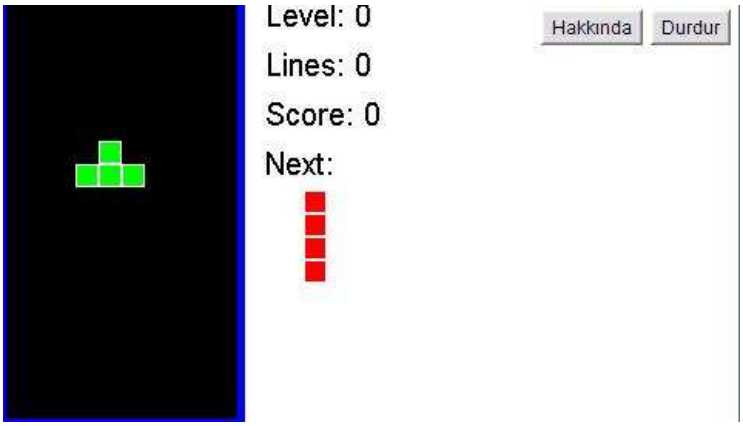
Bu applet programı basit bir şekilde düz çizgi ve noktasal çizgi oluşturmaktadır. Çizgi şekli kısmından çizgi veya nokta seçilerek çizimler rahatlıkla yapılabilir. Şekil 6.8'de çizim applet programının ekran görüntüsü görünmektedir.



Şekil 6.8. Çizim applet'i [24]

6.11. Tetris Oyunu Applet'i

Bu oyun applet'i eskiden beri bilinen tetris oyunun Java dilinde hazırlanmış şeklidir. Programda değişik biçimlerde şekiller gelir. Gelen bu şekilleri düzenli bir şekilde sıraya dizerek bir sıra oluşturulmalıdır. Şekil 6.9'da tetris oyununun ekran görüntüsü görünmektedir.



Şekil 6.9. Tetris oyunu applet'i [25]

6.12. Ufo Oyunu Applet'i

Bu oyun applet'i bilinen basit UFO oyunun Java dilinde hazırlanmış şeklidir. Gelen bombaları Mouse ile yönlendirerek vurulması gerekir. Bu şekilde ne kadar fazla bomba vurulursa o kadar fazla puan alınır. Şekil 6.10'da UFO oyununun ekran görüntüsü görünmektedir.

Şekil 6.10. UFO oyunu applet'i [26]

BÖLÜM 7. LAYOUT MANAGER

Java'da bileşenlerin nereye ve nasıl konacağını belirleyen bazı yapılar vardır. Bu yapılara Layout Manager(yerleştirme biçimi) denir. Her Container'ın bir layout manager'ı bulunmaktadır. Layout manager'lar sayesinde her bileşen için ayrı ayrı ayar yapmak gerekmez. Hangi bileşenin nerde ne nasıl görüneceğine layout managerlar karar verir. Java'da çok sayıda layout manager bulunmaktadır.

7.1. Borderlayout

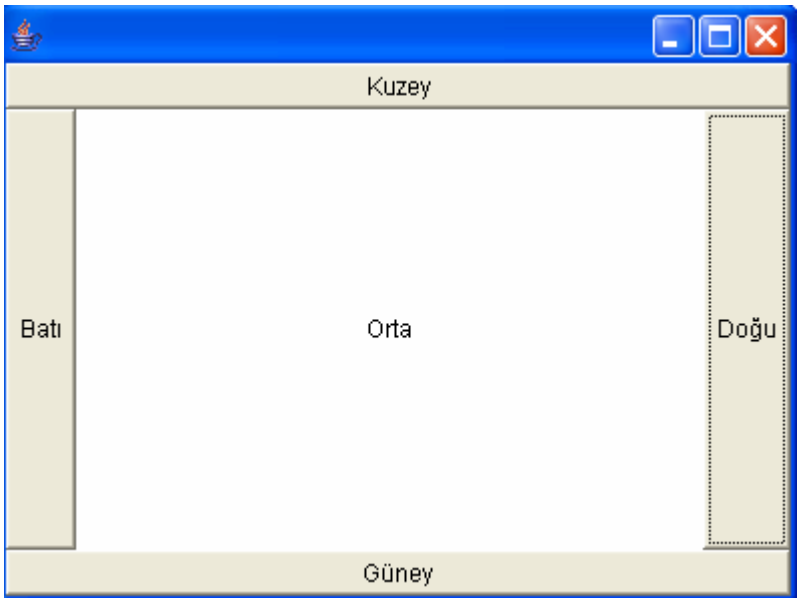
En fazla kullanılan layout'lardan biri BorderLayout'tur. Bileşenleri North, South, East, West ve Center gibi konumlara ayarlayabilir. Kullanıcı elemanları yerleştirirken yerleştirme işlerini sadece kenarlara yapabilir. Aşağıda bu özellik ile ilgili örnek bir program bulunmaktadır.

```
import java.awt.*;
public class BorderLayoutSample{
  public static void main(String[] args){
    Frame f=new Frame();
    Button e=new Button("Doğu");
    Button w=new Button("Batı");
    Button n=new Button("Kuzey");
    Button s=new Button("Güney");
    Label c=new Label("Merkez",Label.CENTER);
    f.setLayout(new BorderLayout());
    f.add(e,BorderLayout.EAST);
    f.add(w,BorderLayout.WEST);
    f.add(n,BorderLayout.NORTH);
    f.add(s,BorderLayout.SOUTH);
    f.add(c,BorderLayout.CENTER);
```

```
        f.setBounds(100,100,400,300);
        f.setVisible(true);
    }
}
```

Program 7.1. BorderLayout örneği [27]

Yukarıdaki örnekte BorderLayout adında bir sınıf oluşturulmuştur. İçerisine east, west, South ve North etiketleri olan 5 tane buton eklenmiştir. Daha sonra f.setLayout(new BorderLayout()); komut satırıyla bileşenlerin layout özelliği BorderLayout olarak ayarlanmıştır. Şekil 7.1'de programın ekran çıktısı görünmektedir.



Şekil 7.1 Borderlayout örneği

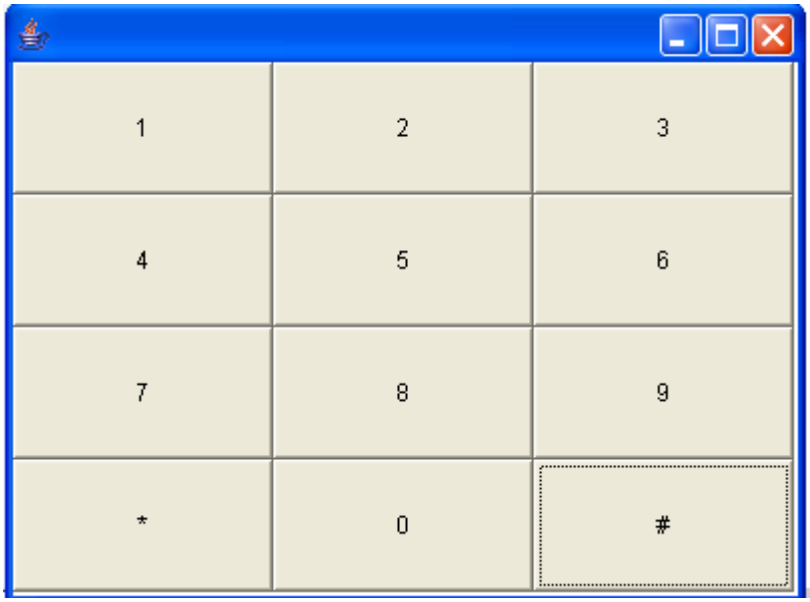
7.2. Gridlayout

Yatay ve düşey kenarlar ile container, istenen sayıda dikdörtgen parçalara ayrılabilir. Her dikdörtgen içine bir düğme konumlandırılır. Aşağıda GridLayout ile ilgili örnek bir program bulunmaktadır.

```
import java.awt.*;
public class GridLayoutSample{
  public static void main(String[] args){
    Frame f=new Frame();
    f.setLayout(new GridLayout(4,3));
    f.add(new Button("1"));
    f.add(new Button("2"));
    f.add(new Button("3"));
    f.add(new Button("4"));
    f.add(new Button("5"));
    f.add(new Button("6"));
    f.add(new Button("7"));
    f.add(new Button("8"));
    f.add(new Button("9"));
    f.add(new Button("*"));
    f.add(new Button("0"));
    f.add(new Button("#"));
    f.setBounds(100,100,400,300);
    f.setVisible(true);
  }
}
```

Program 7.2. GridLayout örneği [16]

Yukarıdaki örnekte GridLayoutSample adında bir sınıf oluşturulmuştur. İçerisine üzerlerinde sayıların yazdığı 12 buton eklenmiştir. Daha sonra f.setLayout(new GridLayout(4,3)); komut satırıyla bileşenlerin layout özelliği Gridlayout olarak ayarlanmıştır. Şekil 7.2'de programın ekran çıktısı görünmektedir.



Şekil 7.2. GridLayout örneği [16]

7.3. Flowlayout

FlowLayout, bileşenleri soldan sağa doğru sırayla dizen bir yerleşim şeklidir. Bir sıra bitince sol altından yeniden başlar. Aşağıda FlowLayout ile ilgili örnek bir program bulunmaktadır.

```
import java.awt.*;
public class FlowLayoutSample{
  public static void main(String[] args){
    Frame f=new Frame();
    f.setBackground(SystemColor.control);
    f.setLayout(new FlowLayout());
    f.add(new Button("Button"));
    f.add(new TextArea("Area"));
    f.add(new TextField("TextField"));
    f.add(new Label("Label"));
    f.add(new Checkbox("Checkbox"));
    List list=new List();
    list.add("List Item 1");
```

```
        list.add("List Item 2");
        list.add("List Item 3");
        f.add(list);
        f.setBounds(100,100,600,300);
        f.setVisible(true);
         }
}
```

Program 7.3. FlowLayout örneği

Yukarıdaki örnekte FlowLayoutSample adında bir sınıf oluşturulmuştur. İçerisine Buton, TextArea, TextField, Label ve CheckBox bileşenleri eklenmiştir. Daha sonra f.setLayout(new FlowLayout()); komut satırıyla bileşenlerin layout özelliği Flowlayout olarak ayarlanmıştır. Şekil 7.3'te programın ekran çıktısı görünmektedir.

Şekil 7.3. FlowLayout örneği [16]

7.4. Cardlayout

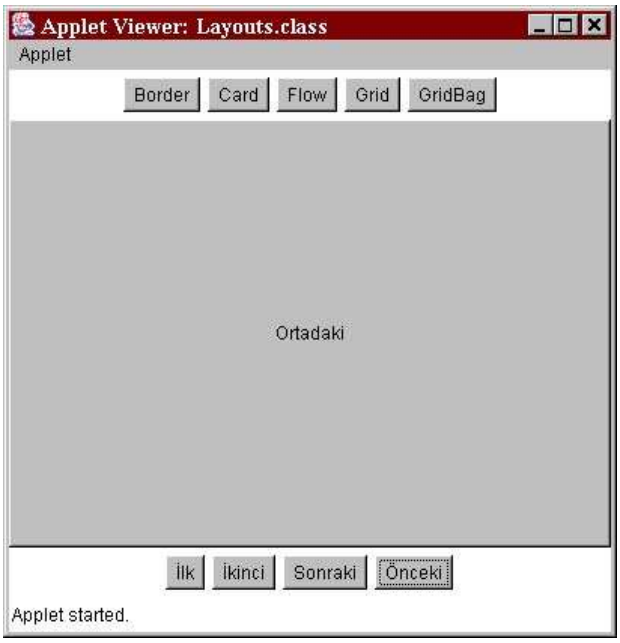
CardLayout birkaç layout düzeninin üst üste konmasıyla oluşan bir kartlar destesidir. İskambil destesine benzetilebilir. Destedeki her kart kendi başına bir layout düzenidir. Genellikle, destedeki her kart bir Panel'dir. Hangi kartın üste geleceği bir düğme tıklamasıyla kontrol edilebilir. Üste gelen kart görünür; alttakiler gizli tutulur. Aşağıda CardLayout ile ilgili örnek bir program bulunmaktadır.

```
import java.awt.*;
import java.applet.*;
public class CardLayoutExample extends Applet
 {
    Button back , next;
    Label lb1 , lb2 , lb3 , lb4;
    TextField other1;
    Panel p1 , first , second ,third , fourth;
    CardLayout c1;
   public void init()
{
      back = new Button("Back");
      next = new Button("Next");
       add(back);
       add(next);
       c1 = new CardLayout();
       p1 = new Panel();
    p1.setLayout(c1);
    lb1 = new Label("First");
    lb2 = new Label("Second");
    lb3 = new Label("Third");
    lb4 = new Label("Fourth");
    first = new Panel();
    first.add(lb1);
    second = new Panel();
    second.add(lb2);
    first = new Panel();
    third.add(lb3);
    fourth = new Panel();
    fourth.add(lb4);
    p1.add("1", first );
    p1.add("2", second);
    p1.add("3", third );
```

```
p1.add("4", fourth);
    add(p1);
   }
}
```

Program 7.4. CardLayout örneği [28]

Yukarıdaki örnekte CardLayoutExample adında bir sınıf oluşturulmuştur. İçerisine Buton bileşenleri eklenmiştir. Daha sonra p1.setLayout(c1); komut satırıyla bileşenlerin layout özelliği Cardlayout olarak ayarlanmıştır. Şekil 7.4'te programın ekran çıktısı görünmektedir.



Şekil 7.4. Cardlayout Örneği

7.5. Gridbaglayout

GridLayout'a benzer, ama ondan daha karmaşıktır. Düğmelerin büyüklüğü birbirlerine eşit değildir; istenildiği gibi ayarlanabilir. Ayrıca, düğmelerin konumları da istenildiği gibi belirlenebilir. Soldan sağa doğru otomatik konumlandırma yoktur. Bu layout düzeninin komut satırı aşağıdaki gibidir.

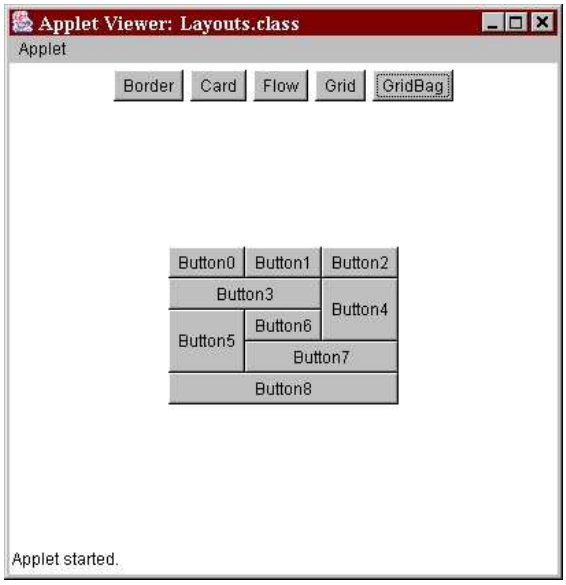
GridBagLayout gb = new GridBagLayout(); ContainerAdı.setLayout(gb);

Aşağıda GridBagLayout ile ilgili örnek bir program bulunmaktadır.

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Layouts extends Applet
{
 Panel[] panels;
 Panel currentPanel;
 static int border=0;
 static int card=1;
 static int flow=2;
 static int grid=3;
 static int gridBag=4;
 String[] layouts = {"Border","Card","Flow","Grid","GridBag"};
 String[] cards = {"İlk","İkinci","Sonraki","Önceki"};
 Button[] layoutButtons = new Button[layouts.length];
 Button[] navigateButtons = new Button[cards.length];
 Panel layoutButtonPanel = new Panel();
 Panel navigateButtonPanel = new Panel();
 public void init()
{
  setLayout(new BorderLayout());
    setupButtons();
  add("North",layoutButtonPanel);
  setupDisplayPanels();
 }
```

Program 7.5. GridBagLayout örneği

Yukarıdaki örnekte layouts adında bir sınıf oluşturulmuştur. İçerisine Buton bileşenleri eklenmiştir. Daha sonra GridBagLayout gb = new GridBagLayout(); komut satırıyla bileşenlerin layout özelliği GridBaglayout olarak ayarlanmıştır. Şekil 7.5'te programın ekran çıktısı görünmektedir.



Şekil 7.5. GridBagLayout örneği [28]

7.6. Boxlayout

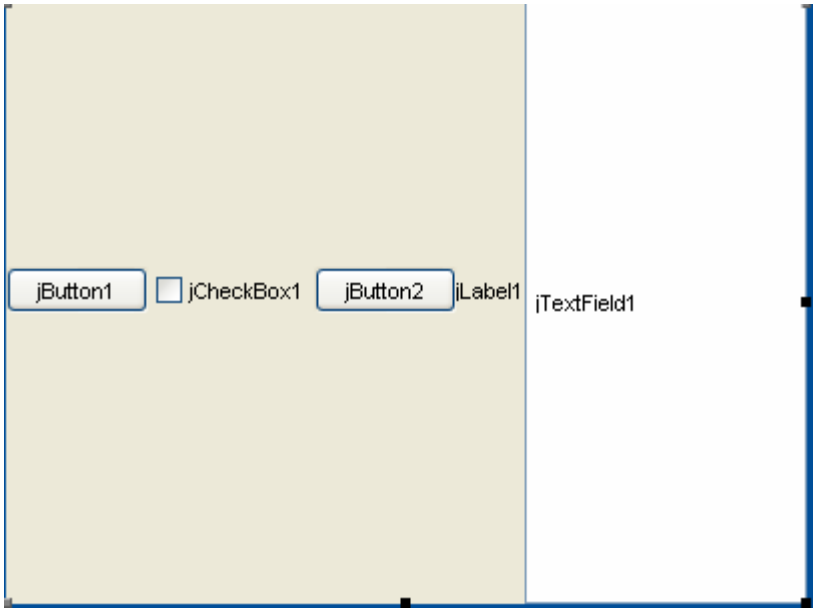
Verilen bileşenleri, seçime göre, yatay ya da dikey sıralanmasını sağlar. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
import java.awt.*;
import javax.swing.*;
import com.borland.jbcl.layout.*;
public class boxlayout extends Frame {
  JButton jButton1 = new JButton();
  BoxLayout2 boxLayout21 = new BoxLayout2();
  JCheckBox jCheckBox1 = new JCheckBox();
  JButton jButton2 = new JButton();
  JLabel jLabel1 = new JLabel();
  JTextField jTextField1 = new JTextField();
```

```
public boxlayout() {
  try {
    jbInit();
  }
  catch(Exception ex) {
    ex.printStackTrace();
  }
}
void jbInit() throws Exception {
  jButton1.setText("jButton1");
  this.setLayout(boxLayout21);
  jCheckBox1.setText("jCheckBox1");
  jButton2.setText("jButton2");
  jLabel1.setText("jLabel1");
  jTextField1.setText("jTextField1");
  this.add(jButton1, null);
  this.add(jCheckBox1, null);
  this.add(jButton2, null);
  this.add(jLabel1, null);
  this.add(jTextField1, null);
 }
}
```

Program 7.6. BoxLayout örneği

Yukarıdaki programda frame üzerine buton, checkbox, label ve textfield bileşenleri eklenmiştir. this.setLayout(boxLayout21); komut satırıyla bileşenlerin layout özelliği boxLayout olarak ayarlanmıştır. Şekil 7.6'da programın ekran çıktısı görünmektedir.



Şekil 7.6. BoxLayout örneği

7.7. Verticalflowlayout

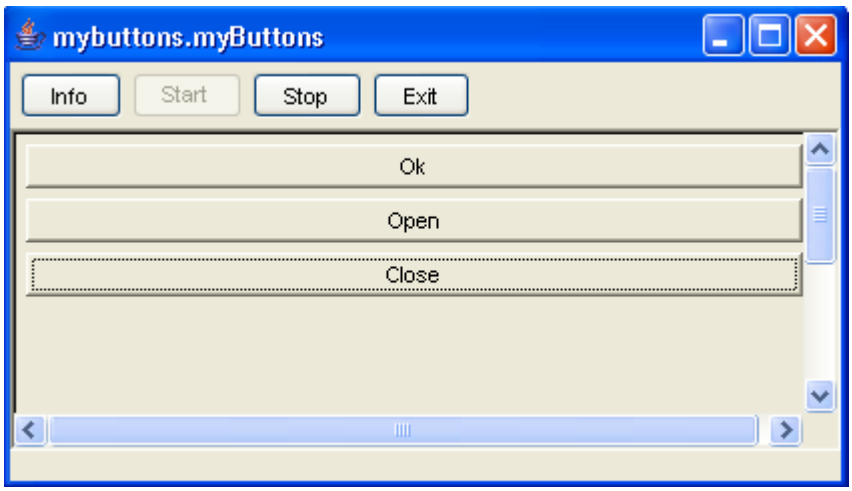
VerticalFlowLayout, frame üzerindeki bileşenlerin yukarıdan aşağıya doğru sanki paragraftaki satırlar gibi yerleşmesini sağlar. Flowlayout özelliği genellikle paneldeki butonları düzenlemek için kullanılır. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
import java.awt.*;
import java.applet.Applet;
import ta.awt.*;
public class myButtons extends Applet
   {
   Button button1, button2, button3;
   public void init()
   {
      setLayout(new VerticalFlowLayout());
      button1 = new Button("Ok");
      button2 = new Button("Open");
      button3 = new Button("Close");
```

```
            add(button1);
            add(button2);
            add(button3);
        }
    }
```

Program 7.7. VerticalFlowLayout örneği

Yukarıdaki programda frame üzerine üç tane buton eklenmiştir. this.setLayout(boxLayout21); komut satırıyla bileşenlerin layout özelliği VerticalFlowLayout olarak ayarlanmıştır. Şekil 7.7'de programın ekran çıktısı görünmektedir.



Şekil 7.7. VerticalFlowLayout örneği

BÖLÜM 8. PROGRAM HATALARINI YAKALAMA

Programlamanın altın bir kuralı vardır. Program, ya doğru sonuçlar vermeli ya da hiçbir sonuç vermemelidir. Çünkü yanlış sonuçlar, ciddi sorunlar meydana getirebilir. Program hataları üç gruba ayrılabilir.

8.1. Sözdizimi Yanlışları

Sözdizimi yanlışları program tarafından tanınmazlar çünkü derlenemezler, çalıştırılamazlar. Dolayısıyla yanlış sonuç vermeleri olanaksızdır. Bu gruptakiler tehlikesiz yanlışlardır.

8.2. Mantıksal Yanlışlar

Programın sözdizimi doğrudur, derlenebilir, çalıştırılabilir. Fakat yapılmak istenen işler için kullanılan deyimler yanlıştır. Yanlış işlemler, yanlış hesaplar yapar. Programın denenme aşamasında bu tür yanlışlar ortaya çıkmazsa, programın kullanılması ciddi sorunlar oluşturabilir. Örneğin, bir bankada hesaplar arasında para akışını kaydeden bir program parçasının eksik veya fazla toplama yaptığı düşünülürse, üstesinden gelinmesi zor sorunlar oluşturabilir. Bu tür yanlışlar, programcılıkta en tehlikeli sayılan yanlışlardır.

8.3. Çalışma Zamanı Yanlışları

Program sözdizimi ya da mantıksal yanlış içermiyor, ama bazı nedenlerle çalışmıyor olabilir. Örneğin, gerekli verileri bir giriş biriminden okuyamaması, verileri bir çıkış birimine gönderememesi, işlemlerde sıfıra bölme gibi olanaksız bir durumla karşılaşılması gibi.

Çalışma sırasında beklenmedik bir hata ile karşılaştığında, kaynak program başka bir önlem almamışsa, sistem program akışını durdurur. Bu durum bazen zararlı sonuçlar da meydana getirebilir. Örneğin, bir I/O işleminde, gerekli veri dönüşümünün bir nedenle yapılamaması istisnai bir haldir ve bu noktadan sonra programın çalışması aniden durmak zorundadır. Ama programın çalışması durunca, veri alış-verişi yaptığı dosyaları kapatamaz. Böylece o dosyalarda veri kaybına yol açabilir.

Programlamanın ilk yıllarında, bu tür yanlışların üstesinden gelebilmek için hayli zorluk çekilirdi. Java, çalışma sırasında oluşan birçok yanlışı kendiliğinden belirler, durumu kullanıcıya bildirir ve programı kapatıp işletim sistemine döner. Buna ek olarak, bu tür yanlışlar oluştuğunda, programcı ne yapmak istiyorsa, onu yapmasını sağlayan araçlara da sahiptir.

Java nesne yönelimli bir programlama dili olduğuna göre, istisnai halleri bir sınıf olarak düşünmüş olması doğaldır. Java.lang paketi içinde Throwable sınıfı bu iş için oluşturulmuştur. Throwable sınıfı, karşılaşılabilecek bütün istisnai halleri ortaya koyabilecek durumdadır. Çok sayıda alt sınıflara ve metotlara sahiptir. Throwable sınıfı, önce iki büyük alt sınıfa ayrılır. Bunlar Exceptions ve Error sınıflarıdır. Programın yakalamasını istediğimiz istisnai halleri, Exceptions sınıfının alt sınıfları ve onlar içindeki metotlarla hallederiz. Ayrıca, programcı, kendisine gerekli olan sınıfları da buradan türetebilir. Throwable sınıfının 60'dan fazla alt sınıfı vardır ve her bir alt sınıfın da başka alt sınıfları, yapıcıları(constructor) ve metotları vardır. Örneğin, çok kullanılan alt sınıflardan birisi RunTimeException sınıfı ile onun alt sınıfı olan ArithmeticException sınıfıdır.

Error sınıfı ise, normal koşullarda programın yakalayamayacağı hataları, sistemin yakalaması içindir. Stack-overflow gibi durumlar buna örnektir. Bu gruba giren hataları, normal koşullarda, program içinde çözümleme olanağı yoktur. Bunlar, programdan çok, sistemle ilgili istisnalardır. Çoğunlukla, programın çalışmasının durmasıyla sonuçlanır.

Yukarıda sözü geçen sınıfların hiyerarşik yapıları şöyledir.

Java.lang.Object
└Java.lang.Throwable

Java.lang.Object
└Java.lang.Throwable
└Java.lang.Error

Java.lang.Object
└Java.lang.Throwable
└Java.lang.Exception
└Java.lang.RuntimeException
└Java.lang.ArithmeticException

8.4. Hatanın Yakalanması

Program çalışırken oluşabilecek hataları yakalamanın ne kadar önemli olduğunu anlamak için aşağıdaki örneğe bakılabilir.

```
class Hata01
{    public static void main(String args[])
   {    int payda = 0;
int pay   = 38/payda ;
    }
}
```

Bu program çalıştırıldığında, sıfıra bölme(divide by zero) hatasıyla karşılaşılır. Matematiksel olarak bu işlem olanaksızdır. Bu durumda, Java run time sistemi, programın çalışmasını iptal eder ve kullanıcıya

```
Exception in thread "main" Java.lang.ArithmeticException:
/ by zero     at Hata01.main(Hata01.Java:4)
```

mesajını verir. Yakalanan hata, Exceptions sınıfının ArithmeticException adlı alt-sınıfı ile belirlenen hatadır. Bu hatanın açıklaması olarak yazılan by zero kısa mesajı, divide by zero(sıfıra bölme) hatası yakalandı anlamındadır. Hata01.main(Hata01.Java:4 dizisi ise, Hata01, main, Hata01.Java, 4 verilerinin programı yöneten stack'tan, yazıldığı sıra ile alındığını gösterir.

Aynı program, başka bir metot kullanılarak aşağıdaki gibi oluşturulabilir.

```
class Hata02 {
static void kesir() {
int d = 0;
int a = 10 / d;
 }
public static void main(String args[]) {
Hata02.kesir();
  }
}
```

Bu kez de aynı hata oluşacak, programın çalışması kesilecek ve aşağıdaki mesaj

```
Exception in thread "main" Java.lang.ArithmeticException: / by zero
at Hata02.subroutine(Hata02.Java:4)
at Exc1.main(Exc1.Java:7)
```

ekrana gelecektir. Bu mesaj, diğer mesajın anlamını taşır. Ancak, programın yapısı farklı olduğundan, yönetici stack'tan çıkanlar farklıdır.

Yedinci satırda main'in çağırdığı kesir() metodunun dördüncü satırında ArithmeticException sınıfının belirlediği divide by zero hatası yakalanmıştır. Yönetici stack'ın bu biçimde hataları listelemesi, programın düzeltilebilmesi için çok yararlıdır. Çünkü bu şekilde hatanın nerede oluştuğu görülür.

8.5. Hata Yakalama

Java'da çalışma zamanı oluşan hatalara exceptions denir. Exceptions beş anahtar sözcükle işlenir.

- try
- catch
- throw
- throws
- finally

8.5.1. Try-catch

Önceki iki programda görüldüğü gibi, Java run-time sisteminin, çalıştırma anında oluşan hatayı yakalayıp, çalışmayı kesmesi ve hatayı belirten bir mesaj vermesi çok faydalıdır.

Ama çoğunlukla, program çalışırken oluşabilecek hataları, programın çalışmasını kesmeden giderme olanağı mümkün olabilir. Java, Throwable sınıfını bunun için oluşturmuştur. İstisnai halleri çözümlemek için, Java try/catch bloklarını kullanır.

Programda, istisnai hallerin oluşabileceği yerler try{} bloğu içine alınır. Bu blokta bir hata oluşursa, sistem o hatayı ortaya atar. Hatanın ortaya atılması demek, hatayı temsil eden bir nesnenin oluşturulması demektir. Bu nesne, Throwable sınıfının alt sınıflarından birisine aittir. Hangi alt sınıfa ait olacağı konusu, oluşan hataya bağlıdır. Nesne, oluşan hata ile ilgili bütün ayrıntılara sahiptir. Catch{} bloğu atılan hatayı yakalar, hatanın ne olduğunu belirlemeye çalışır ve gerekli önlemleri alır. Hata giderilemeyecek türden ise, program kapatılır ve kullanıcıya açıklama gönderilir. Her durumda, program akışının durmasıyla sistemin kilitlenmesi ve diğer kaynakların zarar görmesi kesinlikle önlenebilir. Eğer, programcı, yakalanan hata için bir çözüm öngörmüyorsa, Java run time sisteminin öngördüğü işlemler default olarak etkin olacaktır. Java run time sistemi, istisnai hal oluştuğunda, sistem kaynaklarına zarar vermeden program akışını durdurur ve oluşan istisnai durum hakkında kullanıcıya bir açıklama gönderir.

Try/catch bloklarının sözdizimi aşağıdaki gibidir.

```
try {
// istisna yaratıp yaratmadığı denenecek kodlar
   }
 catch(ExceptionType1 e1){
 // ExceptionType1 için yapılacak işler
   }
catch(ExceptionType2 e2){
 // ExceptionType2 için yapılacak işler
}
…

catch(ExceptionTypeN eN){
// ExceptionTypeN için yapılacak işler
}

finally{
// blok bitmeden yapılması istenen işler
                    }
```

Try bloğunda atılan hatayı catch bloğu yakalar ve onun hangi tip olduğunu belirlemeye çalışır. Önce, hatanın ExceptionType1 tipinden olup olmadığına bakar. O tipten ise, istenen önlemleri alacak kodları çalıştırır. Değilse, hatayı kendisinden sonra gelen catch bloğuna yollar. Bu süreç hata tipi belirlenene kadar arka arkaya devam eder. Eğer, hiçbir catch bloğu hata ile uyuşmazsa, finally bloğu gerekeni yapar. Catch{} bloğu bir tek olabileceği gibi, programcının istediği sayıda olabilir. Bu blok içinde uygun Java deyimleri yer alır. ExceptinType Java'da Throwable sınıfının herhangi bir alt sınıfı olabileceği gibi, programcının türettiği bir sınıf da olabilir.[28]

Aşağıdaki program, önceki iki programın yaptığı işi yapar. Hata oluşabilecek kodlar try{} bloğu içine alınmıştır. Bu blokta bir aritmetik işlem yapılmaktadır. Oluşan

istisna sıfıra bölme hatasıdır. Bu tür hatalar ArithmeticException sınıfına aittir. O nedenle, oluşabilecek hatanın ArithmeticException tipinden olacağını öngörüp, o tipten bir nesneyi işaret etmek üzere e referans(pointer) değişkeni tanımlanmaktadır. Böylece catch(ArithmeticExceptione){} bloğu oluşturulmaktadır. Bu blok hatayı yakalayacaktır. e referans değişkeni, yakalanan hatayı işaret ediyor olacaktır. Aşağıdaki program, hatayı yakaladıktan sonra kullanıcıya sıfıra bölme hatası mesajını gönderecek, ama program akışını kesmeden geride kalan kodları çalıştırmayı sürdürecektir. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
class Hata03 {
  public static void main(String args[]) {
    int d, a;
    try {  // denenecek blok.
      d = 0;
      a = 38 / d;
      System.out.println("Bu satır yazılmaz.");
    } catch (ArithmeticException e) { // catch divide-by-zero error
      System.out.println("Sıfıra bölme hatası.");     }
    System.out.println("catch bloğundan sonraki kodlar.");
  }
}
```

Program 8.1. Try-Catch yapısı

Programın çıktısı aşağıdaki gibidir. Dikkat edilirse, yedinci satır işlevini görmemiştir. Çünkü altıncı satırda hata oluşmuş ve programın çalışması kesilmiştir.

```
Sıfıra bölme hatası.
After catch statement.
```

Programda hatayı yakalayınca, hata giderilip, programın sonuna kadar çalışmasını sürdürmesi istenir. Aşağıdaki program bunun nasıl yapılacağını gösteren basit bir örnektir. Program iki tane rasgele(random) sayı alıp, onun birisini diğerine

bölmektedir. 56789 sayısı, çıkan sayıya bölünmektedir. Bu işlem bir for döngüsü altında 12000 kez tekrarlanmaktadır; yani 12000 tane rasgele(random) sayı oluşturulur. Catch bloğuna önlem alıcı kodlar yazılmazsa, bölenlerden birisi sıfır olduğunda run time sistemi program akışını kesecektir. Ama aşağıdaki catch bloğunda, sıfıra bölme hatası oluştuğunda, a=0 ataması yapılarak, programın çalışmaya devam etmesi sağlanmıştır. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
import Java.util.Random;
class Hata04 {
  public static void main(String args[]) {
    int a=0, b=0, c=0;
    Random r = new Random();
    for(int i=0; i<22000; i++) {
      try {
        b = r.nextInt();
        c = r.nextInt();
        a = 56789 / (b/c);
      } catch (ArithmeticException e) {
        System.out.println("Sıfıra bölme hatası.");
        a = 0; } // devam etmek için 0 atandı
            System.out.println("a: " + a);
    }
  }
}
```

Program 8.2. Try-catch örneği_2

Throwable sınıfı toString() metodunu override ederek, hatanın açıklamasını yapacak hale getirmiştir. Bu özelliği kullanarak, yukarıdaki program aşağıdaki programa dönüşebilir.

```
import Java.util.Random;
class Hata05 {
  public static void main(String args[]) {
    int a=0, b=0, c=0;
    Random r = new Random();
    for(int i=0; i<12000; i++) {
      try {
        b = r.nextInt();
        c = r.nextInt();
        a = 56789 / (b/c);
  }catch (ArithmeticException e) {
                    System.out.println("Exception: " + e);
                    a = 0; // devam etmek için 0 atandı
            }
      System.out.println("a: " + a);
    }
  }
}
```

Program 8.3. Try-catch örneği_3

Ayrıca, Throwable sınıfının ve alt sınıflarının çok sayıda metotları vardır. Örneğin printStackTrace() metodu, ilk iki programdakine benzer açıklamaları yazar.

8.5.2. Çoklu hata yakalama(multiple catch)

Bazen bir try bloğu içinde birden çok hata oluşabilir. Bu durumlarda, catch bloğu içine birden çok hata atıcı deyim konabilir. Program çalışırken, hangi hata oluşmuş ise, ona ait deyim atılır. Birden fazla deyim atılamaz. İlk hataya karşılık gelen deyim işlerlik kazanır. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
// Çoklu catch deyimleri.
class Hata06 {
  public static void main(String args[]) {
    try {
      int a = args.length;
      System.out.println("a = " + a);
      int b = 38 / a;
      int c[] = { 1 };
      c[38] = 99;
    } catch(ArithmeticException e) {
      System.out.println("0 ile bölme : " + e);
    } catch(ArrayIndexOutOfBoundsException e) {
      System.out.println("Array index oob: " + e);    }
    System.out.println("try/catch bloğundan sonraki kodlar.");
  }
}
```

Program 8.4. Çoklu hata yakalama örneği

Bu program çalıştırılmak üzere çağrılırken komut satırından hiç parametre girilmezse, a=0 olur. Dolayısıyla, sıfıra bölme hatası oluşur. Öte yandan, çalıştırılmak üzere çağrılırken komut satırından parametreler girilirse a>0 olur, dolayısıyla sıfıra bölme hatası oluşmaz. Ancak, hemen arkasından şu hatayla karşılaşır. Yedinci satırdaki tanımı uyarınca c[ ] diziminin bir tek bileşeni vardır, o da c[0] = 1'dir. 8-inci satırdaki deyim ise c[37] = 89 atamasını yapmaktadır. Bu olanaksızdır ve hata oluşur. Bu hata ArrayIndexOutOfBoundsException adıyla bilinen hatadır. Öyleyse, bu program mutlaka hata verecektir. Bu program çalıştırılmak üzere çağrılırken komut satırından hiç parametre girilmezse, aşağıdaki bölme hatasını atar.

```
a = 0
Divide by 0: Java.lang.ArithmeticException: / by zero
After try/catch blocks.
```

Bu program çalıştırılmak üzere çağrılırken komut satırından parametreler girilirse, sıfıra bölme hatasını atlar. Aşağıda bununla ilgili bir örnek bulunmaktadır.

```
a = 1
Array index oob: Java.lang. Java.lang.ArithmeticException:38
try/catch bloğundan sonraki kodlar.
```

Catch bloğunda alt sınıflardaki hatalar daima üst sınıftaki hatalardan önce yazılmalıdır. Çünkü catch deyimi, üst sınıftaki hatalardan başlayarak alt sınıflara doğru iner. Eğer üstlerde bir hata yakalanmışsa, o hatayı atar ve alttaki hatalara ulaşamaz.

Java'da erişilemeyen kodlar da hata atımına neden olur. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
/*
Bu program hatalıdır.
Bir dizi catch deyimi varsa, bu deyimler alt sınıftakilerden başlayarak üst
sınıflara doğru sıralanmalıdır. Aksi halde, ulaşılamayan deyimler derleme
zamanı hatası oluşturur.
*/
class Hata07 {
  public static void main(String args[]) {
    try {
      int a = 0;
      int b = 38 / a;
    } catch(Exception e) {
      System.out.println("Generic Exception catch.");
    }
  /*
          Aşağıdaki koda ulaşılamaz, çünkü
          ArithmeticException sınıfı Exception sınıfının bir alt sınıfıdır.
  */
```

```
    catch(ArithmeticException e) { // HATA - ulaşılamaz
      System.out.println("Bu koda ulaşılamaz.");
    }
  }
}
```

Program 8.5. Erişilemeyen kodlar örneği

8.5.3. Throw

Buraya kadar yazılan programlarda hataları Java run time sistemi yakalıyordu. Bunun yerine, istenirse hata belli bir nesne olarak atılabilir. Atılan nesne api'deki bir throwable nesne olacağı gibi, programcının onlardan türettiği bir nesne de olabilir. Aşağıda throw yapısının sözdizimi görülmektedir.

```
try {
if (flag < 0)
{
throw  new ThrowableNesne();
    }
}
```

Aşağıda throw yapısı ile ilgili örnek bir program bulunmaktadır.

```
// throw.
class ThrowDemo {
  static void demoyap()
  {
    try {
      throw new NullPointerException("demo");
        }
catch(NullPointerException e) {
      System.out.println("hata, demoyap içinde yakalandı.");
```

```
      throw e; // hata tekrar atılıyor
           }
     }
  public static void main(String args[]) {
     try
{
      demoyap();
     }
catch(NullPointerException e) {
      System.out.println("Tekrar yakalanıyor: " + e);
     }
   }
}
throw new NullPointerException("demo");
```

Program 8.6. Throw yapısı

8.5.4. Throws

Throws deyimi throw'dan farklıdır. Bir metodun atması olasılığı olan istisnaları listeler. Aşağıda throws yapısının sözdizimi görülmektedir.

```
 public void metod_adı() throws Exception1, Exception2,…,ExceptionN
 {
          try
            {
                  deyimler
            }
          catch(Exception1 ex1)
            {       deyimler
            }
          …
          catch(ExceptionN exN)
```

```
            {
             deyimler
            }
        }
```

Aşağıda throws deyimi ile ilgili örnek bir program bulunmaktadır.

```
        class ThrowsDemo {
        static void throwBir() {
          System.out.println("Inside throwBir.");
          throw new IllegalAccessException("Örnek");
        }
        public static void main(String args[]) {
          throwBir();
        }
      } // Bu program doğrudur, derlenebilir.
      class ThrowsDemo {
        static void throwBir() throws IllegalAccessException {
          System.out.println("throwBir içinde.");
          throw new IllegalAccessException("demo");
        }
        public static void main(String args[]) {
          try {
            throwBir();
          } catch (IllegalAccessException e) {
            System.out.println("Yakalanıyor … " + e);
          }
        }
      }
```

Program 8.7. Throws yapısı

8.5.5. Finally

Bir hata oluştuğunda, onun oluştuğu kodu içeren metot duracak ve işlemi sürdüremeyecektir. Bu sırada metot bir I/O işlemi yapıyorsa, açık dosyaları kapatamayacaktır. Finally bloğu bu gibi durumların üstesinden gelmek için kullanılır. Başka bir deyişle, temizleme işlemi yapar. Finally bloğu şu işlerde çok kullanışlıdır.

- I/O için açılmış dosyaları kapatma,
- Veri tabanı programlarında açılan resultset'i kapatma,
- Veri tabanına yapılan bağlantıyı kapatma

Finally bloğunun kullanılması isteğe bağlıdır. Hiç kullanılmayabilir. Ama kullanılacaksa, son catch bloğunun hemen arkasına konulmalıdır. [28]

Aşağıda finally deyimiyle ilgili örnek bir program bulunmaktadır.

```
SomeFileClass  f = new SomeFileClass();
if (f.open("/a/file/name/path")) {
   try {
      someReallyExceptionalMethod();
       }
catch (Throwable t)
  {
      f.close();
      throw t;
   }
}
SomeFileClass  f = new SomeFileClass();
if (f.open("/a/file/name/path")) {
   try    {
      someReallyExceptionalMethod();
   }
finally
```

```
{
  f.close();
   }
}
```

Program 8.8. Finally yapısı örneği_1

Finally bloğunu yalnızca exception'dan sonra değil, return, break ve continue deyimlerinden sonra da kullanılabilir. Aşağıda bununla ilgili örnek bir program bulunmaktadır.

```
public class  MyFinalExceptionalClass extends ContextClass {
public static void  main(String argv[]) {
   int  mysteriousState = getContext();
   while (true) {
      System.out.print("Who ");
      try {
         System.out.print("is ");
         if (mysteriousState == 1)
            return;
         System.out.print("that ");
         if (mysteriousState == 2)
            break;
         System.out.print("strange ");
         if (mysteriousState == 3)
            continue;
         System.out.print("but kindly ");
         if (mysteriousState == 4)
            throw new UncaughtException();
         System.out.print("not at all ");
      } finally {
         System.out.print("clever man?\n");          }
      System.out.print("I'd like to meet the man ");
```

```
            }
            System.out.print("Please tell me.\n");
        }
    }
```

Program 8.9. Finally yapısı örneği_2

MysteriousState değerlerine bağlı olarak bu programın çıktısı şöyledir.

```
1   Who is clever man?
2   Who is that clever man?
3   Please tell me
4   Who is that strange clever man?      ...
5   Who is that strange but kindly clever man?
6   Who is that strange but kindly not at all clever man?
7   I'd like to meet the man Who is that strange...?
```

BÖLÜM 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 arabirimler ayrıntılı olarak anlatılacaktır.

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

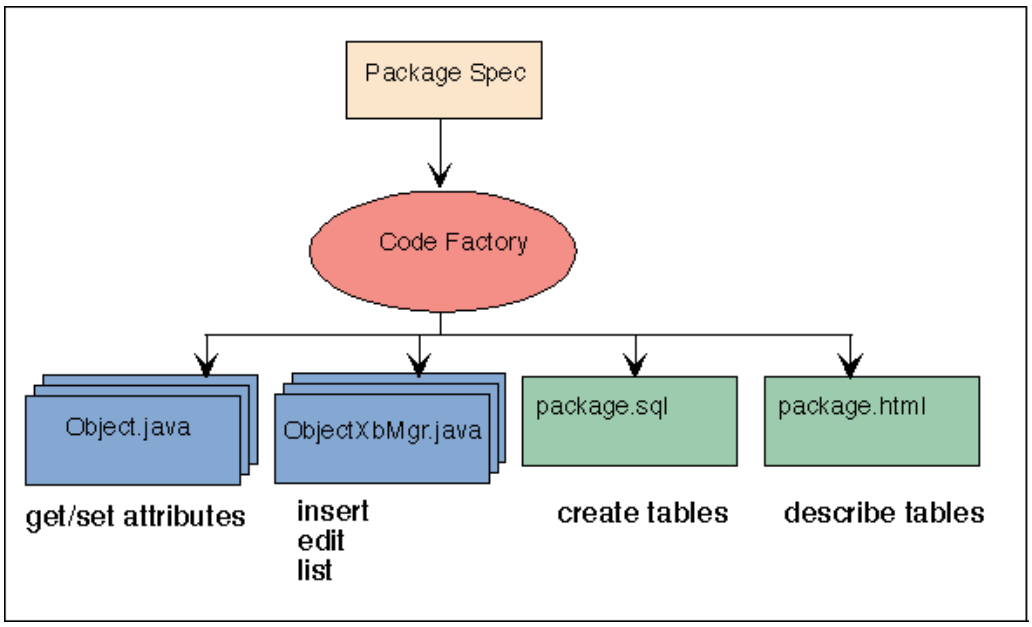
```
class sınıfınadı extends üstsınıfınadı
{
tür örnek-değişken;
tür  ve yöntem(parametre)
{
yöntem-gövde
}
}
```

Burada adı geçen extends anahtar sözcüğü, sınıfadı sınıfının üstsınıfınadı sınıfının 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 aşağıdaki gibidir.

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

Aslında kurucular da birer yöntemdir. Bir kurucu, sınıfı ile aynı isme sahiptir. Nesne hazırlandıktan sonra kurucu otomatik olarak çağrılır. Şekil 9.1'de bazı java paketleri görünmektedir.[15]



Şekil 9.1. Bazı Java paketleri [17]

9.1. 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 gruplandırılabilir.

Temel sınıflar arasında yer alan Object, Class, String ve StringBuffer sınıfları hemen her program tarafından kullanılır. Boolean, Character, Double Float, Integer, Long sınıfları container sınıflarıdır. Bunlar daha önceki türleri korumak için kullanılır.

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

9.2. 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 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, iletişim ağına veya bir girdi aygıtına başvurulduğu zaman, ihtiyaç duyulacak tek şey InputStream ve OutputStream nesnelerini 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.

9.3. 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 nesneler, değişikliğe uğradıklarını ve bir başka nesneye 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.

9.4. Java.Net Paketi

Java.net paketinde iletişim ağı çalışmalarını sağlayan sınıf ve arabirimler yer almaktadır. Java internet'in TCP/IP protokolünü hem önceden kurulmuş olan akış I/O arabirimini çıkararak hem de I/O nesneler kurmak için gereken özellikleri ekleyerek desteklemektedir. Bunun yanı sıra bir 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. İnternetteki bir bilgiye tekdüze kaynak yerleştiriciler kullanılarak kolayca erişilebilir.

9.5. 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, containerlar ile kütüklerin açılması ve saklanması için kullanılan diyalog

pencereleri gibi yüksek düzeyde bileşenleri de içermektedir. AWT serisi içerisinde java.awt.image ve java.awt.peer olmak üzere iki paket daha yer almaktadır.

AWT Java'nın paketleri arasında en ağır olanıdır. Bu sınıflar temel Machintosh 84, Windows 95, X/Motif 88 ve Xerox 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 görünmesine ve hareket etmesine ihtiyaç duyulan uygulamalarda ise, java.awt.peer paketinde bulunan tanımlı arabirimlerle birlikte toolkit sınıfı da kullanılabilir.

9.6. 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 ihtiyaç duyulan uygulamalar tarafından kullanılabilir. Daha karmaşık resim işlemleri için kullanılan sınıf ve arabirimleri kapsamaktadır.

9.7. Java.Awt.Peer Paketi

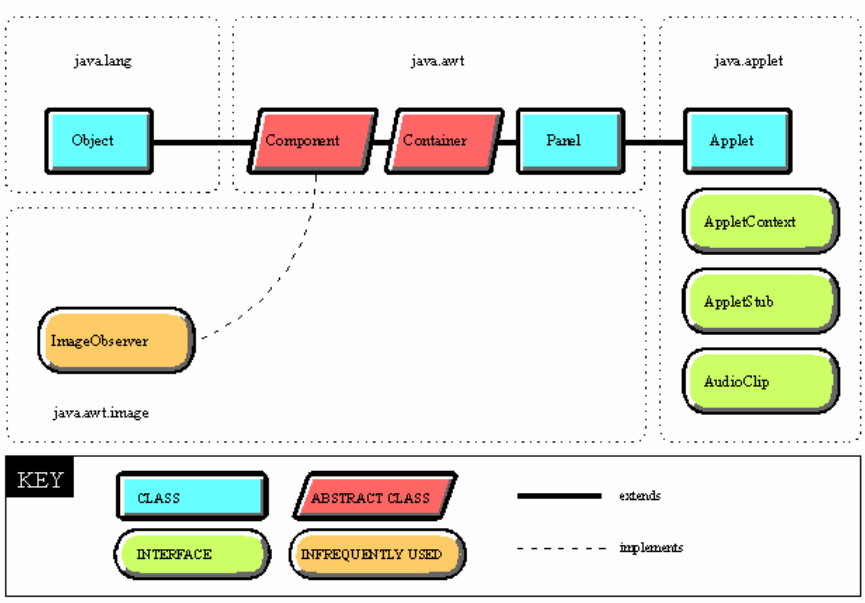
Java.awt.peer paketi, AWT bileşenlerini pencere sistemine ve özel çalışmalara bağlamak için kullanılan arabirimleri kapsamaktadır. AWT'nin pencere sistemine özel çalışmalar hazırlanmazsa, java.awt.peer paketindeki arabirimlerin kullanılması gerekir.

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

9.8. 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 arabirimler kullanılır. Bundan sonra applet'lerin çalışması ile ilgili temel bilgilere sahip olunduğu kabul edilecektir[15]. Şekil 9.2'de applet sınıfının yapısı görünmektedir.



Şekil 9.2. Java.Applet paketi [18]

BÖLÜM 10. AWT

10.1. AWT Bileşenleri

AWT, Java'nın genel Windows bileşenleri ve jawa.awt paketi içerisinde tanımlanmış olan grafik kontrol elemanlarından oluşan yapısıdır. Java AWT, JavaBeans oluşturulmadan önce grafiksel kullanıcı ara yüzü oluşturmaya imkan veren ve programcıya kolaylık sağlamak için tasarlanan bir yapıdır. Jbuilder gibi görsel program geliştirme programları olmadan önce, AWT nesneleri kullanılabiliyordu, özellikleri ayarlanabiliyordu ve el ile kaynak kodları yazılarak olaylar oluşturulabiliyordu.

Jbuilder gibi programlarda bileşen paketi içerisinden sürükle bırak mantığı ile istenilen bileşen frame üzerine eklenebilir ve program üzerinden o bileşene ait özellikler kolay bir şekilde değiştirilebilir. Java AWT'nin temel düşüncesi, küçük bir frame içerisinde temel Windows bileşenlerini kullanmaktır. Bileşenler buton, panel, checkbox gibi görülebilen görsel yapılar olabileceği gibi, aynı zamanda menü ve diyalog kutuları gibi görünmeyen bileşenler de olabilir. AWT bileşenleri iki temel bölüme ayrılır.

Containerlar: AWT bileşenleri diğer bileşenleri ve diğer containerleri de içerebilir. Containerin en genel şekli de paneldir. Panel ekranda görülebilen yapıyı temsil eder. Applet'in çizim alanı bir panel örneğidir ve gerçekte, applet sınıfı Panel sınıfının bir alt sınıfıdır.

UI Bileşenleri: Buton, list, checkbox, textfield ve diğer grafiksel kullanıcı ara yüzü elemanlarını temsil etmektedir.

Jawa.awt paketindeki sınıflar yazılır ve genel container yapısına bir ayna olacak şekilde ayarlanır.

10.2. UI Bileşenlerini Kullanmak

AWT bileşeninin en basit şekli temel UI bileşenidir. Bu bileşenler oluşturularak applet içerisine eklenebilir. Applet zaten bir AWT container'idir ve kendi içerisinde başka container'leri de içerir. Çünkü applet bir container'dir ve kullanıcı diğer AWT bileşenlerini bu container üzerine istediği şekilde yerleştirebilir.

Bileşenlerin tasarım ara yüzüne eklenme işlemi çok basittir. Bileşen paleti içerisinden hangi bileşen yerleştirilmek isteniyorsa, üzerine tıklanır ve kullanıcı ara yüzünde bileşenin nerede görünmesi isteniyorsa oraya tıklanır. Ara yüz üzerine yerleştirilen bileşen başka bir yere taşınmak isteniyorsa, Mouse sol tuşuna basılı tutulur ve istenilen tarafa doğru sürüklenir ve Mouse sol tuşu serbest bırakılır. Eğer bileşenin boyutu değiştirilmek isteniyorsa, bileşenin uçlarını Mouse ile çekerek boyutları istenilen şekilde değiştirilebilir. Bu bölümde UI bileşenleri ile ilgili konular üzerinde durulacaktır. Bileşenlerin özelliklerinin nasıl değiştirileceği, metotların kullanımı, değerlerin alınması ve atanması gibi işlemlerin nasıl yapılacağı hakkında bilgiler üzerinde durulacaktır.

Özellikler kısmında getXxxx() metodu özellik değerini elde etmek için kullanılır ve setXxxx(dataType) metodu da özellik değerini ayarlamak için kullanılır. Bu metotlar genellikle çalışma sırasında sıkça kullanılır. Çünkü tasarım sırasında özellikler kolaylıkla ayarlanabilir. Bu işlemler aynı zamanda olaylar kısmına kod yazarak da yapılabilir. [14]

10.3. Button

Java.awt.Button bileşeni, üzerine tıklandığı zaman belirlenmiş olan bir olayı çalıştırmak için tasarlanmıştır. Örneğin bir hesap makinesi applet'inde her numara ve işlemler için birer tane buton olmalıdır. Kullanıcı her butona tıkladığında o sayıları program içerisinde okuyarak işlemleri yapabilir. AWT paketinde Toolbar(araç

çubuğu) yoktur, ancak bir panele bir kaç düğme eklenerek aynı işlevsellik elde edilebilir. Buton bileşeninin özellikleri aşağıda belirtilmiştir.

Label özelliği: Butonun üzerinde görünecek olan metini belirtir. getLabel() özelliği butonun üzerindeki etiket bilgisini almak için kullanılır. setLabel() özelliği ise butonun üzerine etiket bilgisini yazmak için kullanılır.

actionPermormed(): Butonun üzerine tıklandığı zaman çalışan olaydır. Bu olay altına hangi komutlar yazılmışsa o komutlar çalışır. [14]

Aşağıda buton ile ilgili örnek bir program bulunmaktadır.

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class Buttons extends Applet {
    Label label = new Label("Düğmelere Etiket Yapıştırma");
    Button button1 = new Button("Birinci");
    Button button2 = new Button("İkinci");
    Button button3 = new Button("Üçüncü");
    Panel panel1 = new Panel();
    Panel panel2 = new Panel();
    public void init() {
     setLayout(new BorderLayout());
     panel1.add(label);
     button1.addActionListener(new ButtonHandler());
     button2.addActionListener(new ButtonHandler());
     button3.addActionListener(new ButtonHandler());
     panel2.add(button1);
     panel2.add(button2);
     panel2.add(button3);
     add("North",panel1);
     add("Center",panel2);
```

```
    }
 class ButtonHandler implements ActionListener {
  public void actionPerformed(ActionEvent e){
   String s = e.getActionCommand();
   label.setText(s);
  }
 }
}
```

Program 10.1. Buton bileşeni

Bu programda Buttons.class içerisinde kullanılacak olan butonlar belirlenmektedir. Button button1 = new Button("Birinci"); komutu ile etiketi Birinci olan ismi Button1 olan buton oluşturulmaktadır. Diğer butonlar da aynı şekilde oluşturulmaktadır. Daha sonra init() kısmında butonlar panel2.add(button1); komutu ile panel üzerine eklenmiştir. Şekil 10.1'de bu programın ekran çıktısı görünmektedir.

Şekil 10.1. Buton applet örneği[28]

10.4. Checkbox

Java.awt.Checkbox bileşeni seçenekleri oluşturmak için yada seçenekleri belirtmek için kullanılmaktadır. Checkbox bileşeninin iki durumu vardır. Bunlar on, off ya da

işaretli, işaretsizdir. Butonlarda olduğu gibi checkbox'lar normalde ara yüzdeki olayları etkilemezler. Fakat kullanıcının seçimli olayların içerisinden bir şeyler seçmesine izin verir. Checkbox'lar içerisinde iki yöntem kullanılır.

Bağlantısız paylaşılmayan: Birkaç tane checkbox içerisinde seçilmiş ya da seçilmemiş olması.

Bağlantılı paylaşılmayan: Grup içerisinde bulunan checkbox otomatik olarak seçilir. Checkbox bileşenini paylaşımsız yapmak için, checkboxGroup özelliğinin ayarlanması gerekir. Grup içindeki tüm checkbox'ların özellik değeri radio butonu gibi görünür.

Checkbox bileşeninin özellikleri aşağıda belirtilmiştir.

checkboxGroup özelliği: Checkbox'a ait olan checkboxGroup bileşeni açılır kutu şeklinde olan bir liste içerisinden seçilir. Eğer ayarlanmışsa, checkbox paylaşılmaz. Diğer olaylar getCheckboxGroup(), setCheckboxGroup()'tır.

Label özelliği: Checkbox bileşeninin üzerinde bulunan etiket değerini temsil eder. getLabel() özelliği checkbox'ın üzerindeki etiket bilgisini almak için kullanılır. setLabel(string) özelliği ise checkbox'ın üzerine etiket bilgisini yazmak için kullanılır.

State özelliği: Boolean bir ifadedir. Checkbox bileşeninin işaretli veya işaretsiz olacağını belirler. Buradaki getState() özelliği sayesinde checkbox bileşeninin işaretli veya işaretsiz olduğu öğrenilir. setState(boolean) özeliğiyle de checkbox bileşeninin durumu istenildiği gibi ayarlanır.

itemStateChanged olayı: Checkbox bileşeninin durumu değiştiği zaman bu metot çağrılır. Yani kullanıcı checkbox bileşenini işaretli veya işaretsiz duruma getirdiğinde bu olay altına yazılmış olan tüm komutlar çalışır. [14]

Aşağıda checkbox bileşeni ile ilgili bir örnek program bulunmaktadır.

```
import java.awt.*;
public class CheckboxTest extends java.applet.Applet {
  public void init() {
    setLayout(new FlowLayout(FlowLayout.LEFT));
    add(new Checkbox("Ayakkabı"));
    add(new Checkbox("Çoraplar"));
    add(new Checkbox("Kazak"));
    add(new Checkbox("Etek", null, true));
    add(new Checkbox("Gömlek"));
  }
}
```

Program 10.2. Checkbox bileşeni

Yukarıdaki örnekte CheckboxTest adında bir sınıf oluşturulmuştur. Bu sınıfın init() metodunda add komutu ile beş tane checkbox bileşeni eklenmiştir. add(new Checkbox("Ayakkabı")); satırında üzerinde etiketi “Ayakkabı” olan bir checkbox oluşturulmuştur. Diğer bileşenlerin yapısı da bu şekildedir. Şekil 10.2’de programın ekran çıktısı görünmektedir.

Şekil 10.2. Checkbox bileşeni örneği

10.5. Checkboxgroup

Bazı hallerde çok seçenekten yalnızca birisinin seçilmesi istenebilir. Bu durumlarda Checkbox kutucuklarının radyo düğmeleri gibi işlev yapması gerekir. Bilindiği gibi, bir radyoda kanal seçici düğmelerden yalnızca birisi etkin olur; aynı anda birden çok kanal çalamaz.

Radyo düğmeleri oluşturmak için, önce CheckboxGroup nesnesinin oluşturulması gerekir. Bunun için CheckboxGroup cg = new CheckboxGroup(); yapısı kullanılır. Bundan sonra, grup içinde yer alacak her düğmenin ayrı ayrı oluşturulması gerekir. Örneğin, kadın-erkek ayrımını yapmak için oluşturulacak iki radyo düğmesi için kullanılacak kodlar aşağıdaki gibi olacaktır.[14]

Checkbox erkek= new Checkbox("erkek", cg, true);
Checkbox kadin= new Checkbox("kadin", cg, false);

Checkbox bileşeninin özellikleri aşağıda belirtilmiştir.

SelectedCheckbox özelliği: Grup içerisindeki seçilmiş checkbox değerini döndürür. Bu özellik sayesinde grup içerisindeki hangi checkbox bileşeninin seçili olduğu anlaşılır. getSelectedCheckbox() özelliği ile seçili olan checkbox bileşeninin hangisi olduğu bilgisi alınır. setSelectedCheckbox(Checkbox) özelliği ise istenilen checkbox bileşenini seçili hale getirmeyi sağlar.

Bu bileşenin diğer bileşenlerden farklı olarak herhangi biri olayı yoktur.

Aşağıda CheckboxGroup içerisinde seçili olan bileşeni gösteren örnek bir program bulunmaktadır.

```
import java.awt.*;
public class CheckboxGroupTest extends java.applet.Applet
{
  public void init()
```

```
{
    setLayout(new FlowLayout(FlowLayout.LEFT));
    CheckboxGroup cbg = new CheckboxGroup();
    add(new Checkbox("Red", cbg, false));
    add(new Checkbox("Blue", cbg, false));
    add(new Checkbox("Yellow", cbg, false));
    add(new Checkbox("Green", cbg, true));

   add(new Checkbox("Orange", cbg, false));
    add(new Checkbox("Purple", cbg, false));
  }
}
```

Program 10.3. CheckboxGroup bileşeni örneği

Yukarıdaki örnekte değişik etiketleri olan 6 tane checkbox bileşeni bulunmaktadır. Bu bileşenler CheckboxGroup cbg = new CheckboxGroup(); komutu ile bir checkboxGroup haline getirilmiştir. Her bir bileşen add(new Checkbox("Yellow", cbg, false)); komutlarıyla tanımlanmıştır. Şekil 10.3'te programın ekran çıktısı görünmektedir.

Şekil 10.3. CheckboxGroup bileşeni

10.6. Label

Label bileşeni ekranda görülen ve kullanıcı tarafından seçilemeyen, değiştirilemeyen bir yapıya sahiptir. Bununla beraber, program içerisinde değiştirilebilir. Label bileşeni genellikle diğer bileşenlerle beraber kullanılır ve diğer bileşenlerin kullanılma amacını belirtir. Label aynı zamanda bileşen gruplarını da tanımlamak için kullanılır.

Label bileşeninin özellikleri aşağıda belirtilmiştir.

Alignment özelliği: Label bileşeninin üzerine yazılan yazının yatay olarak durumunu ayarlar. Label.LEFT labeldeki yazıyı sola dayalı hale getirir. Label.CENTER labeldeki yazıyı ortalanmış hale getirir. Label.RIGHT labeldeki yazıyı sağa dayalı hale getirir. getAlignment() özelliğiyle label bileşeninin hizalama bilgisi alınır. setAlignment(int) özelliğiyle de label bileşeninin hizalama işlemi yapılır.

Text özelliği: Label bileşeninin, program çalıştırıldığında üzerinde yazacak metni ayarlamayı sağlayan özelliktir. getText() özelliği label etiketinin üzerinde yazan bilginin alınmasını sağlar. setText(String) özelliği, label etiketinin üzerindeki yazının değiştirilmesini sağlar. [14]

Aşağıda label bileşeni ile ilgili bir örnek program bulunmaktadır.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import com.borland.jbcl.layout.*;
public class label extends JFrame {
  JPanel contentPane;
  XYLayout xYLayout1 = new XYLayout();
  Label label1 = new Label();
  public label() {
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
```

```
    try {
      jbInit();
        }
    catch(Exception e) {
      e.printStackTrace();
       }
  }
private void jbInit() throws Exception  {
    contentPane = (JPanel) this.getContentPane();
    label1.setFont(new java.awt.Font("Dialog", 1, 14));
    label1.setForeground(Color.red);
    label1.setText("Bu Bir Label Örneğidir");
    contentPane.setLayout(xYLayout1);
    this.setSize(new Dimension(400, 300));
    this.setTitle("Frame Title");
    contentPane.add(label1, new XYConstraints(91, 42, 184, 39));  }
  protected void processWindowEvent(WindowEvent e)
    {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      System.exit(0);
    }
  }
}
```

Program 10.4. Label örneği

Yukarıdaki örnekte label bileşeninin çalışma şekli ve özelliklerinin değiştirilmesi gösterilmiştir. Şekil 10.4'te programın ekran çıktısı görünmektedir.

Şekil 10.4. Label örneği

10.7. Textfield

Java.awt.TextField ,kullanıcının tek bir satır halinde bilgi girişi yapmasını sağlayan, içeriği değiştirilebilir bir bileşendir. Aynı zamanda textField bileşeni parola işlemleri için de kullanılmaktadır.[28]

TextField dört yöntemle oluşturulabilir.

TextField(): Yeni bir text alanı (TextField) oluşturur.

TextField(int kolon_sayısı): TextField'in kaç kolon genişliğinde olacağını belirtir.
TextField(String s): TextField içerisinde s metni görüntülenir. Kullanıcı kendi metnini girince, s metni silinir.

TextField(String s , int kolon_sayısı): TextField içinde s metni görüntülenir; giriş alanının uzunluğu kolon sayısı kadar olur. Kullanıcı kendi metnini girince, s metni silinir.

TextField bileşeninin özellikleri aşağıda belirtilmiştir.

Colounms özelliği: Bu özellik textField bileşeni içerisindeki karakter sayısına sınır getirmek için kullanılır. getColumns() özelliği ile textField bileşenindeki kolon sayısı öğrenilir. setColumns(int) özelliğiyle de textField bileşenindeki kolon sayısı ayarlanır.

echoChar özelliği: textField içerisindeki karakterlere maske uygulamak için kullanılır. Örnek olarak textField bileşeni parola girilmesi için kullanılacaksa, bu özellik sayesinde kullanıcının girdiği şifre görünmeyecektir. Girilen şifre yerine "*" gibi karakterler görünecektir. Maske kullanılacaksa echoCharIsSet() özelliği true(doğru) değer yapılır, kullanılmayacaksa false(yanlış) değer yapılır. getEchoChar() özelliği ile textField içerisinde belirlenmiş olan maske karakteri alınır. setEchoChar(Char) özelliğiyle de textfield içerisinde uygulanacak olan maske karakteri belirtilir.
Editable özelliği: TextField bileşeni içerisindeki bilginin değiştirilmesine izin verilip verilmeyeceğini belirtir. isEditable() TexField bileşeninin içindeki metnin değiştirilip değiştirilemeyeceği durumlarını belirler. Değiştirilebilecekse true(doğru) değerini alır, değiştirilemeyecekse false(yanlış) değerini alır.

Text özelliği: Program çalıştırıldığında textFiled bileşeninin içindeki metni belirler. getText() özelliği ile textField içerisindeki metin bilgisi alınır. setText(String) özelliğiyle de textField içerisine istenen metni yazılması sağlanır.

textValueChanged() olayı: textField bileşeni içerisindeki metnin değiştirilmesiyle çalışan olaydır. Yani textField içindeki bilginin silinmesi, yeni bilgi eklenmesi veya bilginin düzeltilmesi gibi durumlarda çalışır. [14]

Aşağıda textField bileşeni ile ilgili örnek bir program bulunmaktadır.

```
import java.awt.*;
  class TextFieldTest extends Frame
        {
           TextField tf = new TextField(32);
           public TextFieldTest(String ad)
        {
           super(ad);
           setLayout( new FlowLayout());
           add(tf);                    }
  public static void main(String args[]) {
```

```
TextFieldTest frm = new TextFieldTest("TextField");
frm.setSize(310 ,210);
frm.show();
}
}
```

Program 10.5. TextField bileşeni

Şekil 10.5'te programın ekran çıktısı görünmektedir.

Şekil 10.5. TextField bileşeni

10.8. Choice

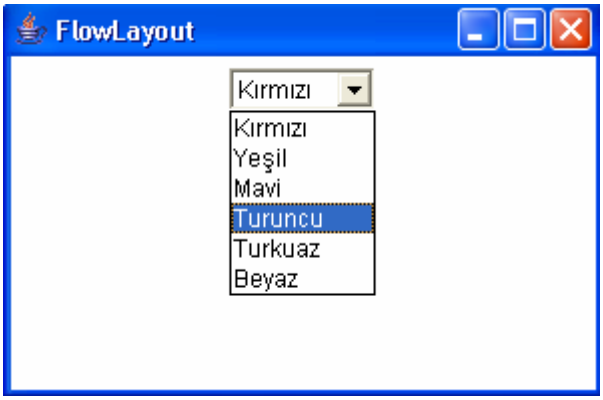
Java.awt.Choice textField ve List bileşenlerinin özelliklerini beraber kullanabilen bir bileşendir. Kullanıcı bileşene tıkladığı zaman aşağıya doğru açılan bir liste ekrana gelir. İçerisinde bulunan elemanlar aşağıya doğru listelenir. Aşağıda Choice bileşeni ile ilgili örnek bir program bulunmaktadır. [14]

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import com.borland.jbcl.layout.*;
public class checkbox extends Applet {
  private boolean isStandalone = false;
  Choice choice1 = new Choice();
```

```
  XYLayout xYLayout1 = new XYLayout();
  public String getParameter(String key, String def) {
    return isStandalone ? System.getProperty(key, def) :
      (getParameter(key) != null ? getParameter(key) : def);
  }
  public void init() {
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  private void jbInit() throws Exception {
    this.setLayout(xYLayout1);
    this.add(choice1,    new XYConstraints(112, 35, 143, 24));
    choice1.addItem("Kırmızı");
    choice1.addItem("Yeşil");
    choice1.addItem("Mavi");
    choice1.addItem("Turuncu");
    choice1.addItem("Turkuaz");
    choice1.addItem("Beyaz");
  }
}
```

Program 10.6. Choice bileşeni örneği

Yukarıdaki program bir choice bileşeni içerisine temel renk isimlerini eklemektedir. Bu programda sadece elemanların eklenmesi gösterilmiştir. Şekil 10.6'da programın ekran çıktısı görünmektedir.



Şekil 10.6. Choice bileşeni [28]

10.9. Menubar

Java.awt.MenuBar, alt menüleri ve hiyerarşik menü sistemi oluşturmayı sağlayan bileşendir. Ara yüzdeki her yeni pencerenin kendi menü sistemi olabilir. Kullanıcı eğer isterse, menü içerisindeki elemanları aktif ve pasif edebilir. Menü elemanları arasına çizgi koyabilir veya menü elemanlarına CTRL+C gibi kısayollar atayabilir.

MenuBar bileşeninin özellikleri aşağıda belirtilmiştir.

Label Özelliği: Menü elamanları üzerinde görünen metini belirtir. getLabel() özelliğiyle menü elemanlarının üzerindeki metin bilgisi alınır. setLabel(String) özelliğiyle de menü elemanlarının üzerindeki yazı değiştirilir.

Shortcut özelliği: Menü etiketinin sağ tarafında görünen ve kullanıcı tarafından bu menü elemanı için belirlenmiş kısayolu gösterir. getShortcut() özelliği menü elemanının kısayol tuş değerini almak için kullanılır. setShortcut(String) özelliği ise menü elemanlarına kısayol değeri atamak için kullanılır.

actionPerformed olayı: Menü elemanlarından herhangi biri seçildiği zaman çalışan metottur. [14]

Aşağıda menubar ile ilgili örnek bir program bulunmaktadır.

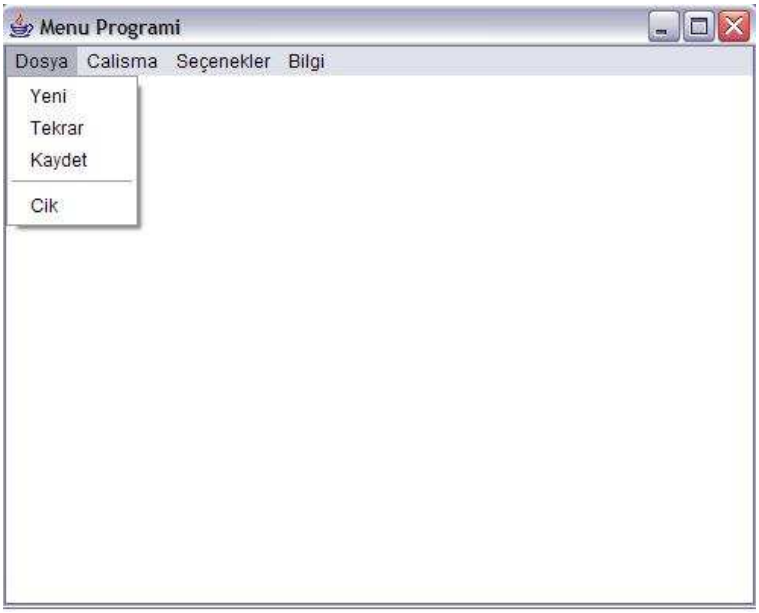
```
import java.awt.*;
import java.awt.event.*;
class AnaMenu
extends MenuBar
{
private MenuItem aGeri;
private CheckboxMenuItem aRenk;
public AnaMenu()
{
Menu m;
m = new Menu("Dosya");
m.add(new MenuItem("Yeni"));
m.add(new MenuItem("Tekrar"));
m.add(new MenuItem("Kaydet"));
m.addSeparator();
m.add(new MenuItem("Cik"));
add(m);
m = new Menu("Calismalar");
m.add((aGeri = new MenuItem("Geri")));
m.addSeparator();
m.add(new MenuItem("Kes"));
m.add(new MenuItem("Kopyala"));
m.add(new MenuItem("Yazdir"));
m.add(new MenuItem("Sil"));
add(m);
m = new Menu("Seçenekler");
m.add(new MenuItem("Düzen"));
m.add((aRenk = new CheckboxMenuItem("Renk")));
add(m);
//Bilgi
m=new Menu("Bilgi");
m.add(new MenuItem("Hakkinda"));
m.add(new MenuItem("Yardim"));
```

```
Menu m1=new Menu("YaziTipi");
m1.add(new MenuItem("Verdana"));
m1.add(new MenuItem("Arial"));
m1.add(new MenuItem("Impact"));
m.add(m1);
add(m);
enableMenuler(false);
setFarbe(true);
}
public void enableMenuler(boolean ena)
{
if (ena) {
aGeri.setEnabled(true);
} else {
aGeri.setEnabled(false);
}
}
public void setFarbe(boolean on)
{
aRenk.setState(on);
}
}
public class BabaMenu extends Frame {
public static void main(String[] args)
{
BabaMenu wnd=new BabaMenu(); }
public BabaMenu() {
super("Menu Programi");
setLocation(100,100);
setSize(500,400);
setBackground(Color.white);
setMenuBar(new AnaMenu());
setVisible(true);
```

```
addWindowListener (
new WindowAdapter()
{
public void windowClosing(WindowEvent event )
{
setVisible(false);
dispose();
System.exit(0);
        }
    } ;//dikkat ) ?
  }
}
```

Program 10.7. Menubar örneği [17]

Yukarıdaki örnek basit bir menü oluşturmanın kodlarını içermektedir. Bu yapıyı kullanarak menü istenilen şekilde genişletilebilir. Şekil 10.7'de bu menü programının ekran çıktısı görünmektedir.



Şekil 10.7. Menubar örneği

10.10. Popupmenu

Java.awt.PopupMenu ara yüz üzerinde menü oluşturmayı sağlayan bir bileşendir. Show() metodu popup menünün ekrandan nasıl ve ne şekilde görüneceğini belirtir. Örneğin kullanıcı popupMenu1'i mouse'un sağ tuşuna basarak ekrana çıkarmak istiyorsa, aşağıdaki gibi bir kod yazmalıdır.

popupMenu1.show(Frame1,10,30);

Bu kod button1 bileşeninin actionPerformed olayının altına yazılır. Kullanıcı mouse'un sağ tuşuna bastığında ekranın 10,30 koordinatlarında popup menü gözükür.

10.11. Scrollbar

Test alanları ve kaydırma listelerinin kendi kaydırma çubukları vardır. Bu tür kullanıcı ara yüzü elemanlarında kaydırma çubukları elemanlardan bağımsız parçalar değillerdir.

Bir kullanıcı, ara yüzü olarak kaydırma çubuklarını veya slider bileşenlerini tek başına oluşturabilir. Kaydırma çubukları maksimum ve minimum arasındaki değerleri seçmek için kullanılır.

Başlangıç ve scrollbar'ın sonundaki oklar değerleri azaltılır veya arttırılabilir. Normal artırım değeri 1'dir. Bir defalık, Page-Down hareketi sonucunda satır sayısı değiştirilebilir. Bunun normal artırım değeri de 10'dur. Scrollbars üzerinde bulunan kutuya asansör veya başparmak denir.

Mouse ile scrollbar'ı hareket ettirerek listeler içerisinde daha hızlı ilerlenebilir. Bir scrollbars oluşturmak için 3 tane constructor, 2 tane de değer almak ve değer ayarlamak için özellik bulunmaktadır. Bu özellikler aşağıda açıklanmıştır.

getMaximum(): Maximum değeri geriye döndürür.

getMinumum(): Minumum değeri geriye döndürür.

getOrientation(): Scrollbarın yönünü geriye döndürür: 0 > Scrollbar.HORIZONTAL
1 ->Scrollbar.VERTICAL
getValue(): Scrollbarın o andaki geçerli değerini döndürür.

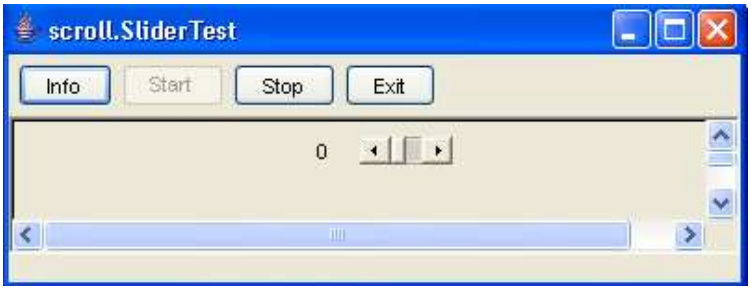
setValue(int): Scrollbarın geçerli değerini oluşturur. [14]

Aşağıda scrollbar bileşeni ilgili örnek bir program bulunmaktadır.

```
import java.awt.*;
public class SliderTest extends java.applet.Applet {
Label I;
public void init()     {
I=new Label("0");
add(I);
add(new Scrollbar(Scrollbar.HORIZONTAL,0,0,1,100));
   }
public boolean handleEvent(Event evt) {
if (evt.target instanceOf Scrollbar) {
int v=((Scrollbar)evt.target).getValue();
I.setText(String.valueOf(v));
  }
return true;
  }
}
```

Program 10.8. Scrollbars örneği

Yukarıdaki program basit bir scrollbar örneğidir. Programda scrollbar bileşenine her tıkladığımızda değeri artmaktadır. Aynı zamanda scrollbar bileşeninin ön tarafında elemanın o anki değeri görünmektedir. Şekil 10.8'de programın ekran çıktısı görünmektedir.



Şekil 10.8. Scrollbars örneği

10.12. Textarea

TextArea, textField bileşenine benzemektedir. Tek farkı, kullanıcı içerisine birden fazla satır bilgi girebilir. Yani TextArea için textField ve scrollBar'ın birleşimi denilebilir. Yan taraftaki ve aşağıdaki kaydırma çubuklarını kullanarak, textArea içinde görünmeyen kısımların görünmesi sağlanır. TextArea bileşeni özellik olarak TextField ve ScrollBar bileşenlerinin özelliklerini taşımaktadır. Sadece birkaç ek özelliği bulunmaktadır. Bu özellikler aşağıda belirtilmiştir.

Rows özelliği: TextArea bileşeninin satır sayısını belirlemek için kullanılır. getRows() özelliği ile satır sayısı öğrenilir. setRows(int) özelliği ile de textArea bileşeninin satır sayısı ayarlanır.

textValueChanged() olayı: TextArea bileşeni içerisindeki metinin değiştirilmesiyle çalışan olaydır. [14]

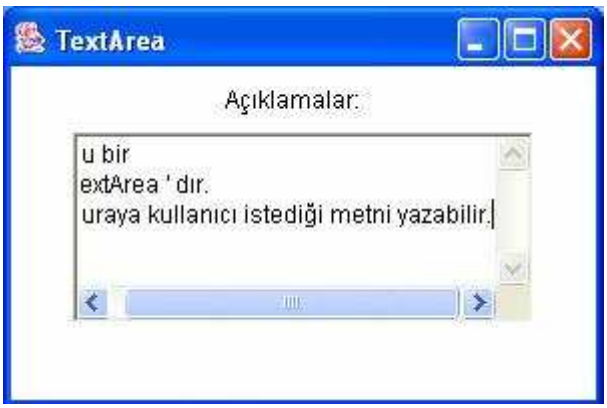
Aşağıda textArea bileşeni ile ilgili örnek bir program bulunmaktadır.

```
import java.awt.*;
class TextAreaTest2 extends Frame
    {
   Label lb = new Label("Açıklamalar:");
   TextArea ta = new TextArea(5,30);
   public TextAreaTest2(String ad)
     {
```

```
                super(ad);
                setLayout( new FlowLayout());
                add(lb);
                add(ta);
        }
        public static void main(String args[])
       {
                TextAreaTest2 t = new TextAreaTest2("TextArea");
                t.setSize(300 ,200);
                t.show();
        }
}
```

Program 10.9. TextArea bileşeni [28]

Şekil 10.9'da programın ekran çıktısı görünmektedir.



Şekil 10.9. TextArea bileşeni

10.13. Panel

Her frame ayrı ve bağımsız bir penceredir. Ama panel bağımsız bir pencere değildir. Her panel başka bir yapı içerisinde açılır. Bu yapı frame, applet, window olabilir. Panel'in kenarları yoktur. Tarayıcı onu bulunduğu yapı içinde gösterir. Panel bir container olduğundan, bunun içine istenilen ara yüz bileşenleri konulabilir.

Aşağıda panel bileşeninin özellikleri belirtilmiştir.

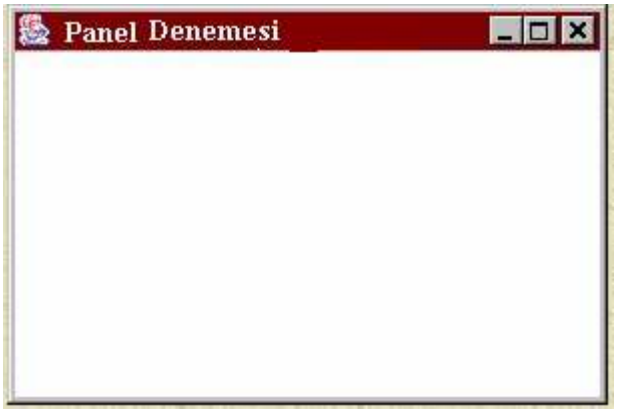
Layout özelliği: bu özellik panel üzerinde bulunan bileşenlerin yerleşim şeklini belirtir. [14]

Aşağıda panel oluşturma ile ilgili örnek bir program bulunmaktadır.

```
import java.awt.*; // Panel oluşturma
class PanelTest1 extends Panel {
  public static void main(String args[]) {
        PanelTest1 pt = new PanelTest1();
        Frame frm = new Frame(" Panel Yaratma Denemesi");
        frm.add(pt);
        frm.setSize(300 , 200);
        frm.setVisible(true);}    }
```

Program 10.10. Panel bileşeni

Şekil 10.10'da programın ekran çıktısı görünmektedir.



Şekil 10.10. Panel Bileşeni [28]

BÖLÜM 11. KULLANILAN PROGRAMLAR

11.1. Html Nedir?

HTML, kısaca bir yazı işaretleme dilidir. Yazının göze hoş gelmesi için şekillendirilmesi, HTML'nin amacıdır. Ancak yazıyı şekillendirirken arada resim, ses gibi öğelerin bulunması da iyi olacaktır.

HTML, belgelerin metinler, tablolar, listeler, fotoğraflar içerecek şekilde internette yayımına izin verir. Basit bir web sayfası hazırlamak için temel olarak HTML komutlarının bilinmesi yeterlidir.

Çeşitli web tasarım programlarında görsel olarak nesnelerle hazırlanan sayfaların HTML kodları da görüntülenebilir. Ayrıca web tarayıcıların kaynak kodu özelliği ile internetteki ilginç sayfaların HTML kodlarına ulaşılabilir.

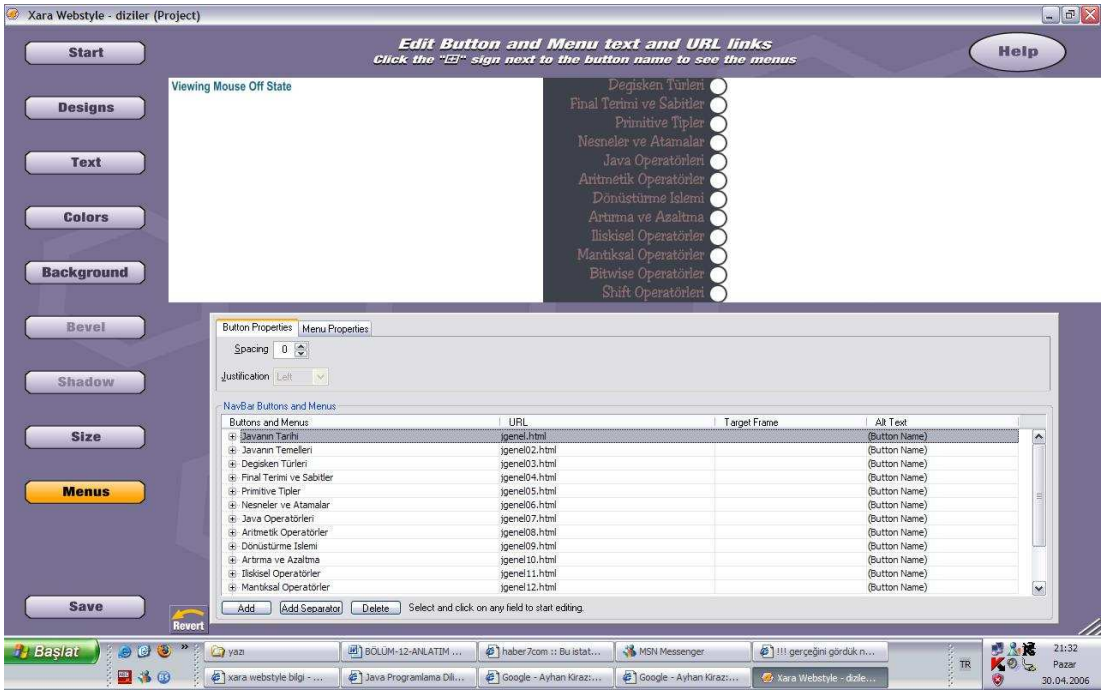
Bu tezin uygulama kısmı yapılırken bazı hazır paket programlar kullanıldı. Bu bölümde tezin yapılışı sırasında kullanılan paket programlar, programların kısa bir özeti ve programlar ile uygulamanın nasıl yapıldığı anlatılacaktır. Aşağıda uygulama sırasında kullanılan paket programların listesi verilmiştir.

- Xara WebStyle
- Microsoft Frontpage
- Macromedia Dreamveawer

11.2. Xara Webstyle

Xara Webstyle programı genellikle web siteleri için buton, navigation bar, banner, logo, başlık, 3 boyutlu başlık ve zemin rengi gibi ayarları yapabilmeyi sağlar. Bu

uygulamada Xara Webstyle programı genellikle web sitesinin butonlarını yapmakta kullanıldı. Şekil 11.1'de Java'nın genel yapısının buton yapımı görülmektedir.



Şekil 11.1. Xara Webstyle ile buton yapımı

Bu programda yapılan butonların özellikleri sol taraftaki menüden ayarlanmaktadır. Text kısmından butonların üzerindeki yazının font, punto gibi özellikleri ayarlanmaktadır. Colors kısmından butonların üzerindeki yazıların rengi, zemin rengi, Mouse üzerine gelindiğinde görünecek renk ve sayfanın zemin rengi gibi özellikler değiştirilir. Background kısmı menünün zemini için değişik biçimler belirlemeyi sağlar. Size kısmı butonların büyüklüğünü değiştirmeyi sağlar. Menus kısmından butonların tıklandığı zaman hangi bağlantıyı açacağı belirtilir. Save kısmından ise yapılan ayarların kaydedilmesi sağlanır. Uygulama içerisinde kullanılan tüm menüler Xara Webstyle kullanılarak bu şekilde yapılmıştır.

11.3. Macromedia Dreamweaver

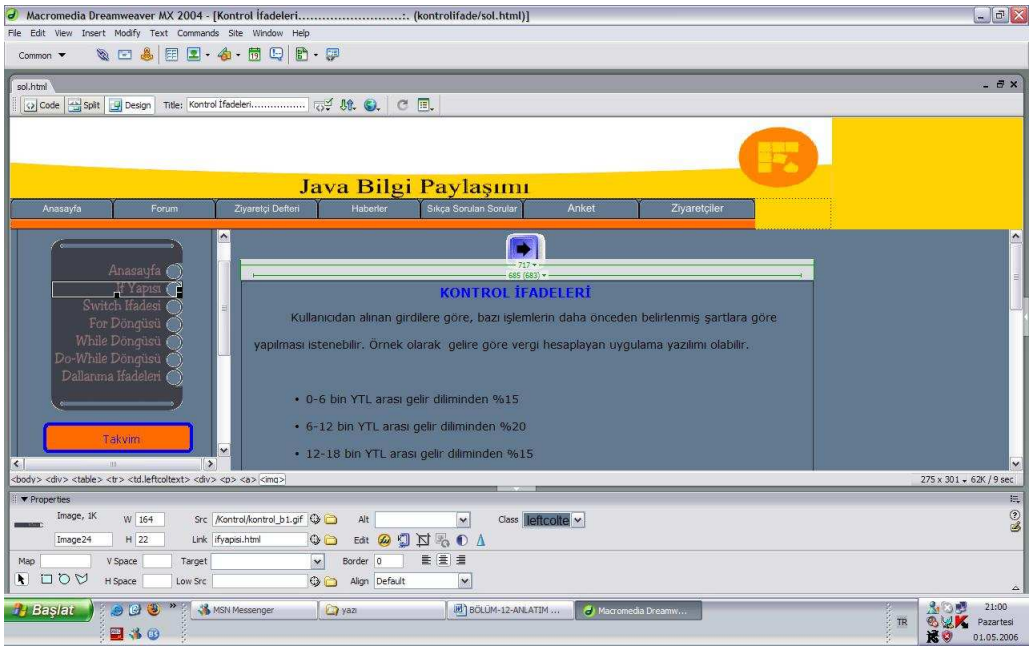
Programlama dillerinde olduğu gibi HTML için de, görsel(visual) programlama özelliği taşıyan ve hazır nesneler kullanan editör programları geliştirilmiştir. Bu

programlar kullanılarak HTML dili daha da basitleştirilmiş, sayfalar dolusu program kodu yazmak yerine hazır nesneler, görsel olarak kullanılmıştır. Macromedia firmasının yaptığı Dreamweaver, bu programların en iyileri arasında yer almaktadır. Bu programla büyük, karmaşık ve etkileşimli web sayfaları hazırlamak mümkündür. Dreamweaver web tasarımcıları için hazırlanmış olan profesyonel bir web tasarım paketidir. Basit bir dosya hazırlar gibi web sitesi hazırlanabilir. Dreamweaver, site tasarımını sadece kendi özellikleri ile sınırlamaz. Diğer programlarla bağlantı kurarak bunların çalıştırmasını sağlar. Oluşturulan web sayfalarının yayına hazırlanma işlemini kolaylaştırır ve site yönetimini en iyi düzeyde geliştirir.

Dreamweaver ile web sayfaları oluşturulurken sayfa yerleşimi hem görsel olarak düzenlenebilir hem de sayfayı temsil eden HTML kodları arka plânda takip edilebilir. Sayfa içerisinde değişiklik yapabilmek için ayrıca bir HTML editörü bulunmaktadır. Bu küçük editör ile birlikte birçok HTML komutu otomatik olarak sunulmaktadır. Hızlı bir şekilde web tasarımının sağlanması için geliştirilmiş bir diğer özellik ise HTML stilleridir. Stilleri kullanarak site içerisinde yazı tipi, yazı özellikleri, zemin renkleri, resimlerin tanımı, yerleşim işlemleri, çerçeveleme gibi birçok işlem ayni anda tanımlanıp bütün siteye uygulanabilir. Oluşturulan stil tanımı saklanarak bütün siteye uygulanabilir. Böylece siteyi oluşturan bütün sayfalar için ayrı ayrı tanımlar yapmak gerekmez ve bu şekilde özgün tasarım şablonları oluşturulabilir. Site içerisine eklenen görsel nesnelerin arka planda nasıl bir HTML kodu oluşturduğu takip edilebilir. Dreamweaver HTML kodlarını satir numarası vererek takip etme imkani da sağlar. Bu şekilde numaralara göre sayfa içerisindeki kodlar takip edilebilir. Ayrıca yerleşim ekranında seçilen bir nesnenin kodlarının, HTML kaynak kodları içerisinde de seçili olduğu görülebilir. Bu özellik yardımı ile nesneye ait kodlar işaretlenebilir. Oluşturulan her sayfa, diğer sayfalardan bağımsız olarak tasarlanır. Sayfaların birbirlerine bağlanması ile site meydana gelir. Dreamweaver sadece sayfaların oluşturulmasını değil aynı zamanda siteyi oluşturan bütün sayfaları kontrol eder ve yönetimini sağlar. Site içerisinde bulunan sayfaları kontrol eder ve sayfalar içerisinde tanımlanan kırık bağlantıları tespit eder. Ayrıca kullanılmayan herhangi bir dosyayı sitenin boyutunu büyütmemesi için silinmesini sağlar. Oluşturulan sitenin, yerel disk alanında saklanacağı konumu belirlenebilir. Ayrıca site içerisine eklenen ya da siteden silinen dosyalara göre, siteye ait dosya

listesi otomatik olarak güncelleştirilebilir. Dreamweaver, siteyi temsil eden dosyaları sadece disk alanında saklamakla yetinmez, aynı zamanda web ortamında bulunan herhangi bir FTP adresine de yönlendirebilir. Bu şekilde sitenin internet ortamında yayınlanması sağlanabilir. Ayrıca yayınlanan dosyaların kontrolünü de sağlamak mümkündür. Yani, daha önce yayınlanan dosyaların boyutları kontrol edilir, sadece içeriği değişen dosyalar ve yeni dosyaların yayınlanması sağlanır. Bu özellik kullanılarak sürekli güncellenen sitelerin yayınlanması hızlandırılabilir.[32]

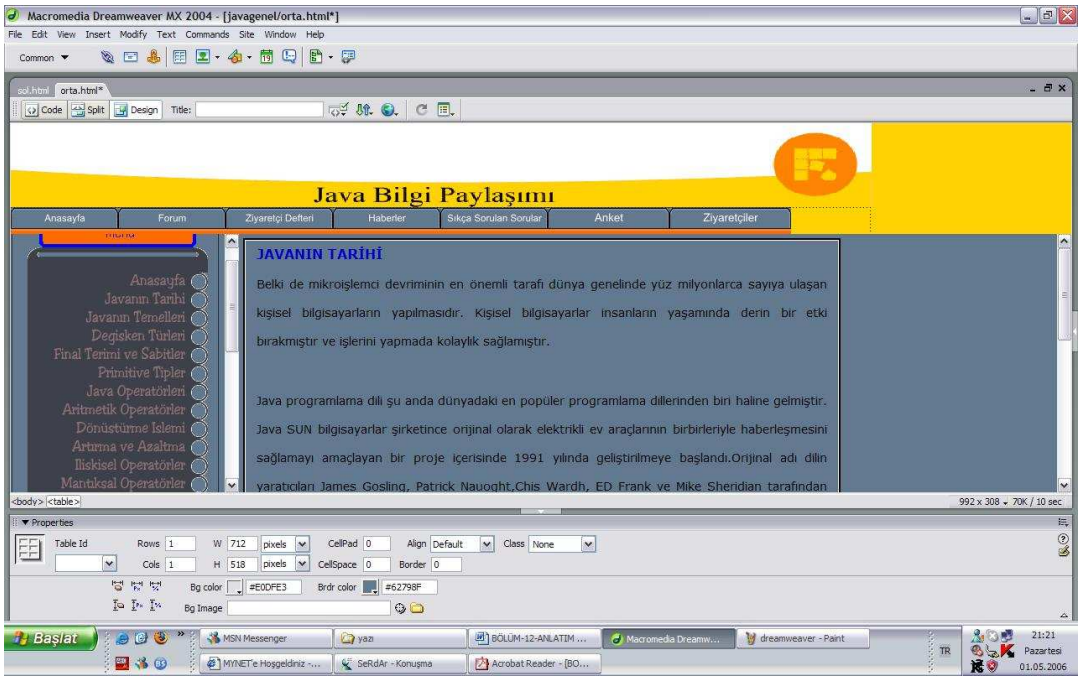
Dreamweaver genellikle uygulama içerisindeki Xara Webstyle programında yapılan butonları düzenlemek için kullanıldı. Butonların özellikleri bu program ile ayarlandı. Şekil 11.2'de Dreamweaver programında butonların nasıl ayarlandığı görünmektedir.



Şekil 11.2. Dreamweaver programında buton ayarları

Dreamweaver programında butonlara tıklandığı zaman hangi bağlantıyı nerede ve nasıl açacağı belirlenir. Bu işlemler aşağıdaki menüde bulunan link bölümünden yapılır. Yan tarafındaki klasör simgesine tıklanarak bağlantı dosyası seçilir. Target kısmından da verilen bağlantının ne şekilde ve nerede açılacağı belirtilir.

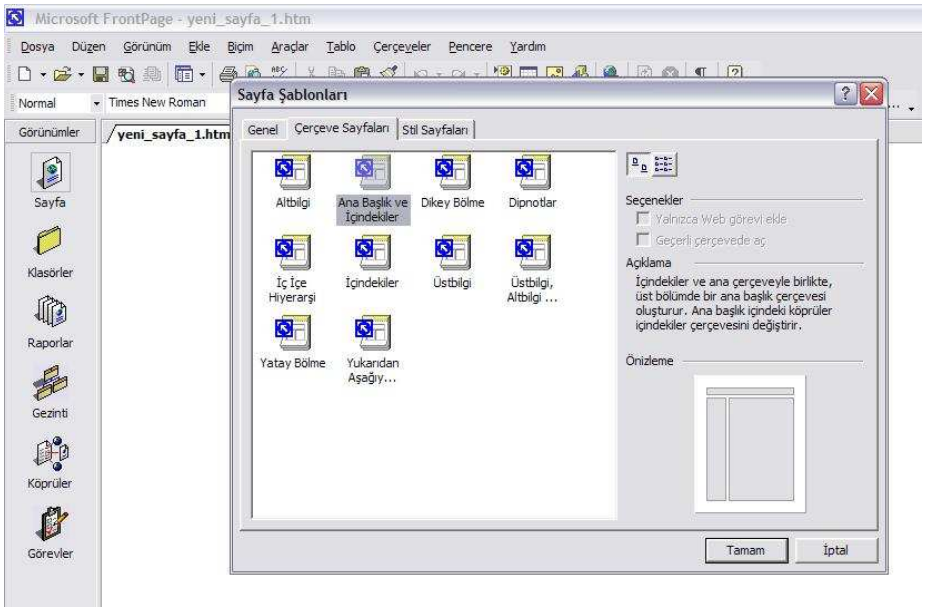
Şekil 11.3'de Dreamweaver programındaki tablo ayarları görünmektedir. Buradaki rows kısmından tablonun satır sayısı, cols kısmından da sütün sayısı belirlenir. W kısmından tablonun genişlik değeri, H kısmından da tablonun yükseklik değeri belirlenir. Align kısmından tablo içerisindeki bilgilerin yerleşim şekli belirlenir. Bgcolor kısmında tablonun zemin rengi, brdrcolor kısmından da tablonun kenar rengi belirlenir. Uygulamadaki her aşama bu şekilde yapıldı ve tasarım oluşturuldu.



Şekil 11.3. Dreamweaver programında tablo ayarları

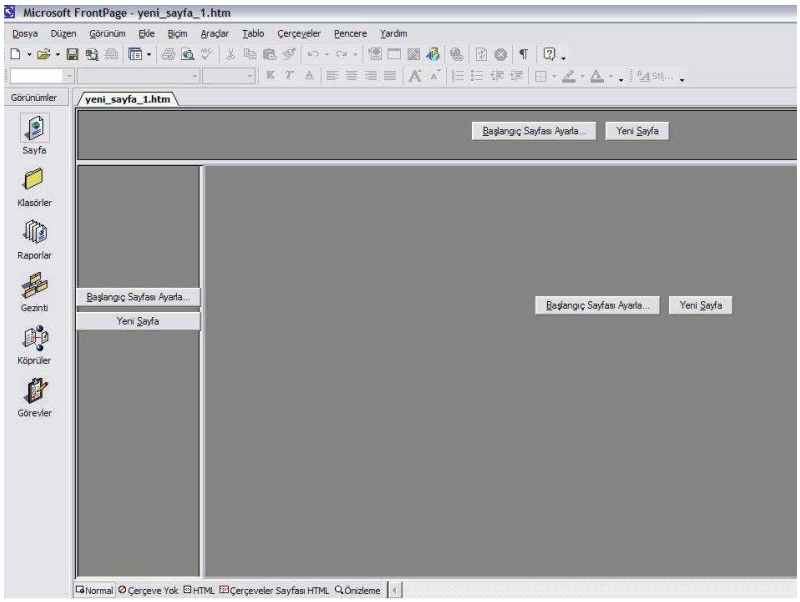
11.4. Microsoft Frontpage

Frontpage programı Microsoft Office içerisinde bulunan, web tasarımı yapmak için kullanılan paket programdır. Frontpage programının en büyük avantajı Türkçe olmasıdır. Yapılan uygulamanın bazı kısımları Frontpage programı kullanılarak yapılmıştır. Uygulamada kullanılan çerçeve ve frame yapısı Frontpage programı kullanılarak yapılmıştır. Kullanımı dreamweaver'a göre daha kolay ve basittir. Şekil 11.4'te frame yapısının nasıl kullanıldığı görünmektedir.



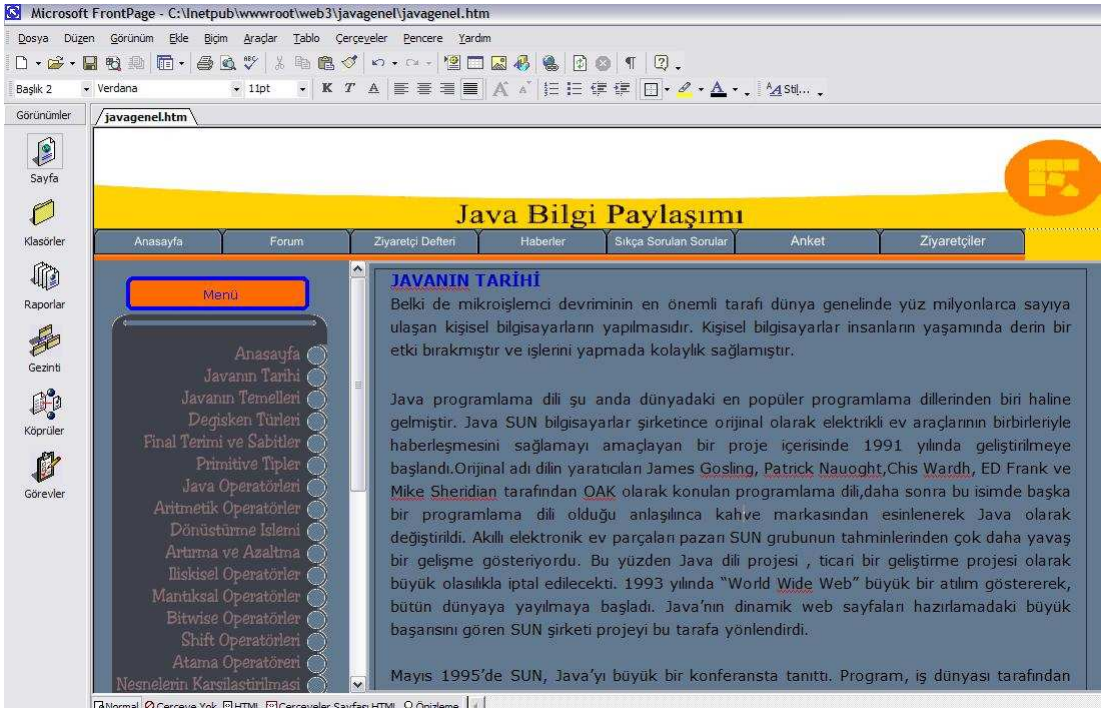
Şekil 11.4 Frontpage programında çerçeve sayfası oluşturma

Dosya menüsünden sayfa veya web kısmı tıklanır. Sağ tarafa gelen menüden sayfa şablonları kısmı seçilir. Gelen ekrandan çerçeve sayfaları seçilir. Buradan da ana başlık ve içindekiler kısmı seçilir ve tamam tuşuna basılarak çerçeve sayfası oluşturulmuş olur. Şekil 11.5'te oluşturulan çerçeve sayfasının ilk hali görünmektedir.



Şekil 11.5. Frontpage programında çerçeve sayfası

Kullanıcı eğer isterse varolan bir sayfayı çerçeve içerine ekleyebilir. Bunu yapabilmek için çerçeve içerisinde bulunan başlangıç sayfası ayarla butonuna tıklanır. Hazır bir sayfa yoksa çerçeve içerisindeki yeni sayfa butonuna basılır. Bu şekilde çerçeve sayfası hazırlanmış olur. Şekil 11.6'da çerçeve sayfasının son hali görünmektedir.



Şekil 11.6. Frontpage programında çerçeve sayfası örneği

BÖLÜM 12. SONUÇ VE ÖNERİLER

Bu çalışmada, genellikle zor olarak bilinen Java programlama dilini rahat, kolay ve eğlenceli bir şekilde internet üzerinden öğreten bir web sitesi yapılmıştır. Yapılan bu uygulamanın başka yöntemlerle daha da geliştirilmesi mümkündür. Buna örnek olarak ASP, PHP ve Flash gibi programlar verilebilir. Yapılacak uygulamada bu tür programlar kullanılırsa, daha ilgi çekici olabilir.

Yaşadığımız zamanda teknoloji giderek kendini geliştirmektedir. Özellikle bilgisayar ve iletişim alanındaki gelişmeler, diğer alanları da etkilemiş, dolayısı ile teknoloji yaşadığımız zamanı hakimiyeti altına almıştır. Bu gelişmelerin en çok etkilediği alan internet olmuştur. Dünyanın her tarafında birbirine bağlı bilgisayarlardan oluşan bu büyük ağ sistemi, bilgi alışverişini oldukça kolaylaştırmıştır. Bunun sonucu olarak internet üzerinden öğretim giderek yaygınlaşmaya başlamıştır. Böylece her insan istediği bilgiye istediği yerden ulaşabilmektedir. Görüldüğü gibi eğitim ve öğretimde internet çok önemli bir araçtır. Kişi istediği bilgiyi internet sayesinde kısa bir sürede elde edebilmektedir. Bunun sonucu olarak bu zamana kadar kitaplar üzerinden öğrenilen bilgiler internet üzerinden, yani web sitesi ile daha kolay bir şekilde öğrenilmektedir.

Yapılan web sitesi farklı tasarımlarda oluşturulabilir. Daha önce Java'yı anlatan İngilizce web siteleri yapılmıştır. Ancak yapılan siteler fazla görsellik içermediğinden dolayı fazla ilgi çekmemişlerdir. Bu yüzden bilgisayar başındaki kişinin sıkılma ihtimalini ortadan kaldırmak için görselliğe önem verilmesi faydalı olacaktır. Çünkü kişi görerek konuları daha iyi anlamaktadır. İşte bu konuda Java programlama dili gerçekten çok işe yaramaktadır. Java'daki applet'ler sayesinde yapılan uygulamalar, aynı anda web sitesi üzerinde çalıştırılarak kullanıcılara gösterilebilir. Applet'lerin web üzerinden çalışabilmesi, bazı uygulamaların gerçek zamanlı görülebilmesi için çok faydalıdır. Bu şekilde uygulamalar daha iyi anlaşılır.

Uygulamayı gerçekleştirirken bu noktalara önem verilmiştir. Buradaki amacımız kişilere Java dilinin genel yapısını interaktif olarak öğretmekti. Burada kullanıcının bilgilere kolay bir şekilde ulaşabilmesi için frame(çerçeve) yapısı kullanıldı. Kullanıcı sol taraftan butona tıkladığında ilgili bölüm sağ tarafa açılmaktadır. Bunun yanı sıra her sayfadaki anasayfa butonu ile kullanıcı istediği yerden anasayfaya dönebilmektedir. Ayrıca kullanıcının programları daha iyi anlayabilmesi için bazı programlar applet tarzında yapılmıştır. Bu sayede kullanıcı programın çalışmasını interaktif olarak görebilmektedir. Web sitesi içeisinde kullanıcının sıkılmaması için tasarıma önem verilmiştir. Aynı zamanda kullanıcıların aralarında bazı konularda birbirlerini danışabilmeleri için forum konulmuştur. Ayrıca Java ile ilgili son haberlere ulaşılabilmesi için de haberler bölümü konulmuştur. Yine kullanıcıların web sitesi ile ilgili fikirlerini yazabilmeleri için bir ziyaretçi defteri konulmuştur. Bunlar sayesinde web sitesinin daha da ilgi çekici olması sağlanmıştır.

Uygulamayı gerçekleştirirken karşılaşılan en büyük zorluk yapılan programların applet olarak çalıştırılması kısmıdır. Yapılan uygulamaların web sitesi üzerinde kolay bir şekilde çalışabilmesi için Java 2 Platform Enterprise Edition 1.4 SDK, Java 2 Runtime Enviroment SE v1.4.2_10 ve J2SE Runtime Enviroment 5.0 Update 6 paketlerinin kurulması gerekir. Bunları kurduktan sonra Java programları çalıştırılabilir.

Sonuç olarak bilgisayar başında Java programlama dilini kolay, anlaşılır ve zevkli bir şekilde öğrenmek isteyen herkes için yapılan Java programlama dilinin web üzerinden sunumu uygulaması istenilen hedeflere ulaşmıştır ve uygulama tamamıyla bitirilmiştir. Web sitesini tasarlayan kişi hayal gücünü kullanarak siteye istediği şekilde görsellik katarak daha eğlenceli hale getirebilir. Bu tamamıyla siteyi tasarlayan kişinin ufkuna bağlıdır.

KAYNAKLAR

[1] http://www.tk.gov.tr/internethaftasi/internet_nedir.htm

[2] http://cc.anadolu.edu.tr/Egitim/UEgitim/Uzak3.htm

[3] HORTON, W., "Designing Web-Based Training", New York, 2000

[4] HALL, B., Web-Based Training Cookbook, New York, 1997

[5] http://yunus.hacettepe.edu.tr/~umutal/publications/ Webbaseddistan ceeducation.pdf

[6] cet.boun.edu.tr/faculty/erkunt/papers/AUUE2002bildiri.pdf

[7] http://www.sun.com/java/

[8] http://www.ijee.dit.ie/articles/999971/article.html

[9] HARVEY M., Java How to Program, USA, 2005

[10] www.axtelsoft.com/turhan.coban/Java/

[11] GÜNGÖREN BORA, Java ile Temel Programlama, Ankara, 2003

[12] http://www.mutasyon.net/makaleler.asp?id=1

[13] http://www.kodcu.com/kodcu/

[14] LEMAY LAURA, CADENHEAD ROGERS, Sams Teach Yourself Java 2 in 21 Days, USA, 2001

[15] DOHERTY DON, MANNING MICHELLE, Teach Yourself Jbuilder 2 in 21 Days, USA 1998

[16] http://www.andamooka.org/reader.pl?section=javanotes

[17] http://www.dmry.net/java-ile-programlama-java-dersleri

[18] httpmath.hws.edueckcs124downloadsjavanotes4.pdf

[19] http://rapidshare.de/files/19388745/javadersnotuhafta5.doc.html

[20] gunaysoft_depo04.sitemynet.com/ java/java_egitim_seminerleri.pdf

[21] http://javaboutique.internet.com/tutorials/Java_by_Example/section3_4.html

[22] http://javaboutique.internet.com/Calculator

[23] http://people.cs.uchicago.edu/~stefanko/Teaching/CS102-Sum2001/ Applets/ Applets.html

[24] http://java.sun.com/applets/jdk/1.4/index.html

[25] http://www.bodo.com/Applets/Blocks/index.html

[26] http://www.rur.com/javacode/UFO_Attack/UFO_Attack.zip

[27] http://www.godoro.com/Divisions/Ehil/ Mahzen/Java/ TheJavaBook/ txt/html/document_BookIntroduction.html

[28] http://mail.baskent.edu.tr/~tkaracay/ders/java/exception/exception.htm

[29] ÇÖPOĞLU SERKAN, DOKUMACI MUHAMMET, Lisans Tezi, Java Programlama Dili, Fırat Üniversitesi Teknik Eğitim Fakültesi, Elazığ, 2000

[30] http://homepage.mac.com/iamnot/eden-pub/images/codeFac.gif

[31] http://support.novell.com/techcenter/articles/dnd19960801.html

[32] http://egitek.meb.gov.tr/dersdesmer/DersDestek/dersdestekmerkezi/ BilgKitap/pdf/ BOLUM9_webtasarim.pdf

ÖZGEÇMİŞ

Ayhan KİRAZ 1980 Sakarya-Pamukova doğumludur. İlkokulu ve ortaokulu Sakarya-Pamukova'da, lise eğitimini Sakarya-Adapazarı Fatih Anadolu Meslek Lisesi Elektronik bölümünde 1998 yılında tamamlamıştır. 1998 yılında girdiği Marmara Üniversitesi Teknik Eğitim Fakültesi Bilgisayar ve Kontrol Eğitimi bölümünden 2002 yılında mezun olmuş ve aynı yıl Bilgisayar öğretmeni olarak Sakarya Fatih Anadolu Teknik, Teknik ve Endüstri Meslek lisesine atanmıştır. 2003 yılından bu yana Sakarya Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Bilgisayar Eğitimi Bölümünde Yüksek Lisans öğrencisidir.