Java Platformunun Temel Özellikleri

Arakodlu Çalışma Sistemi: C ve C++ gibi programlama dillerinde programı derleyip çalışabilen dosya elde ettiğimizde bu dosyanın içerisinde microişlemcinin doğrudan çalıştırabileceği makine kodları bulunur. Halbuki Java programlama dili ile yazılmış bir programı derlediğimizde oluşan çalışabilen dosyanın içerisinde hiç bir mikroişlemcinin dili olmayan yapay bir arakod bulunur. Bu arakoda "byte code" denilmektedir. Bu arakod doğrudan çalışmaz. İsmine JVM (Java Virtual Machine) denilen program tarafından gerçek makine komutlarna dönüştürülerek çalışır. Arakod çalıştırılmak istendiğinde Java platformunun ismine JRE (Java Runtime Environment) denilen bir alt sitem devreye sokulur. Önce arakod arakod JRE tarafından gerçek makine komutlarına dönüştürülür ve çalıştırılır. Arakodun gerçek makine koduna dönüştürülmesi sürecine JIT derlemesi (Just In Time Compilation) denilmektedir. Şüphesiz bu süreçte ufakta olsa bir performans kaybı oluşmaktadır. Ama yüksek seviyeli uygulamalarda bu kayıp gözardı edilebilir.

Arakodlu çalışmanın ne avantajları vardır? Bu sayede programlar platform bağımsız bir biçimde yazılıp çalıştırılabilir. Bu kavrama WORA (Write Once Run Anywhere) denilmektedir. Derlenip oluşturulan byte kod JVM yüklü herhangi bir sistemde çalıştırılabilir. Byte code herhangi bir işletim sisteme bağımlı değildir.

Geniş Bir Sınıf Kütüphanesi: Java platformunun geniş bir sınıf kütüphanesi vardır. Veritabanı, web programlama, çeşitli GUI (Graphical User Interface) işlemleri için sınıflar vardır.

Hızlı Uygulama Geliştirme Ortamı: Java hızlı uygulama geliştirme ortamı rapid application development) sunar. Bunun için çeşitli görsel araçlardan faydalanılır. Öğrenilmesi nispeten daha kolaydır.

Güvenli Çalışma Ortamı: Java' da yazılmış programların sisteme zarar verme olasılıkları çok daha azdır. Yazılan program yüzünden makinada problem çıkma olasılığı (birtakım bilgilerin silinmesi vs.) çok daha azdır.

Programlama Dillerinin Tarihsel Gelişimi

Programlama dilleri birbirinden etkilenerek geliştirilmiştir. Bu etkilenme genel olarak bir dilden alıntı yaparak ona yeni özellikler ekleme şeklinde olmaktadır. Programlama dilleri bir ağaç olarak düşünülürse ağacın en tepesinde Fortran bulunmaktadır. (1954-1957). Fortran dünyanın ilk yüksek seviyeli programlama dilidir. Fortran'ı Algol ve Cobol dilleri izlemiştir. C programlama dili 1970 yılında Bell laboraturalarında Dennis Ritchie tarafından tasarlanmıştır. Java programlama dili Sun firmasında çalışan James Gosling tarafından tasarlanmıştır.

Programlama Dillerinin Sınıflandırılması

Programlama dillerini 3(üç) biçimde sınıflandırabiliriz:
1. Seviyelerine göre sınıflandırma
2. Kullanım alanlarına göre sınıflandırma
3. Programlama modeline göre sınıflandırma

Seviyelerine Göre Sınıflandırma

Seviye (level) bir programlama dilinin insan algısına yakınlığının bir ölçüsüdür. Yüksek seviyeli (high-level) diller hem dil özellikleri hem de uygulama geliştirme olarak kolay öğrenilebilen, insan algısına yakın dillerdir. Alçak seviyeli (low-level) diller makinenin doğal çalışmasına daha yakındır. Olabilecek en aşağı seviyeli dil saf makine dilidir.

Kullanım Alanlarına Göre Sınıflandırma

Genel Amaçlı diller, Bilimsel ve mühendislik diller, Oyun ve animasyon dilleri, Yapay zeka dilleri, Web dilleri, Sistem Programlama dilleri vs...

Programlama Modeline Göre Sınıflandırma

Bir programı yazarken kullanılan genel model (paradigm) önemlidir. Bazı dillerde sınıf yoktur. Program çeşitli alt programların (fonksiyon) birbirlerini çağırması yoluyla yazılır. Bu tekniğe prosedürel model denir. Bazı dillerde sınıflar vardır. Program sınıflar kullanılarak yazılmaktadır. Bu modele nesne yönelimli model (object oriented model) denilmektedir. Bazı dillerde programlar sanki matematiksel formül yazıyormuş gibi yazılmaktadır. Bu dillere fonksiyonel (functional) diller, programlama modeline ise fonksiyonel programlama modeli denilmektedir. Bazı diller birden fazla modeli destekler. Bunlara çok modelli (multiparadigm) diller denilmektedir.

Buna göre Java, genel amaçlı, yapay zeka, web programlama, bilimsel ve mühendislik alanlarında kullanılabilen nesne yöneimli, fonksiyonel özellikleri olan (özellikle Java 8 ile birlikte) yüksek seviyeli bir programlama dilidir.

Anahtar Notlar: *Hangi programlama dilinin yüzde kaç kullanıldığına ilişkin istatistikleri yapan Tiobe isimli bir şirket vardır. (http://www.tiobe.com/index.php/content/paperinfo/index.html). Daha geniş sınıflandırma için (http://en.wikipedia.org/wiki/List_of_programming_languages_by_type)*

Temel Kavramlar

Çevirici Program, Derleyici ve Yorumlayıcı: Bir programlama dilinde yazılmış olan programı eşdeğer olarak başka bir dile dönüştüren programlara çevirici programlar (translater) denir. Çevirici programlarda kaynak dil (source language) ve hedef dil (target language) söz konusudur. Örneğin, Fortran programlama dilini C programlama diline dönüştüren program çevirici programdır. Burada kaynak dil Fortran, hedef dil C dir. Bir çevirici programda hedef dil alçak seviyeli bir dilse (saf makine dili, sembolik makine dili ya da arakod) bu tür çevirici programlara özel olarak derleyici (compiler) denilmektedir. JVM nin byte kodu makine koduna dönüştürme faaliyeti de aslında bir derlemedir. Bu işlem çalıştırma ortamı içersinde bulunan JIT derleyici (Just In Time compiler) tarafından yapılmaktadır.

Bazı programlar hiç kod üretmeden doğrudan çalıştırılırlar. Bunları çalıştıran programlara yorumlayıcı (interpreter) denilmektedir. Bazı diller yalnızca derleyicilerle, bazı diller yalnızca yorumlayıcılarla, bazı diller de her ikisi ile birden çalışmaktadır. Java derleyici ile çalışan bir programlama dilidir.

İşletim Sistemi: İşletim sistemi makinenin donanımını yöneten, bilgisayar ile kullanıcı arasında köprü görevi gören bir sistem yazılımıdır. Bir işletim sisteminin bir çok görevi bulunmaktadır. Örneğin:

programları çalıştırmak, dosya organizasyonu, çeşitli donanımsal aygıtları yönetmek vs. İşletim sistemlerini günümüzde masaüstü ve mobil olmak üzere iki gruba ayırabiliriz. Popüler masaüstü ve mobil işletim sistemleri ve firmaları şunlardır:

Firma	**Masaüstü Sistem**	**Mobil Sistem**
Microsoft	Windows	Windows Mobile
Apple	Mac OS X	IOS
Google	-	Android
Açık Kaynak Kodlu (open source)	Linux	Embedded Linux

Mobil Sistemlerde Kullanılan Programlama Dilleri ve Ortamları: Her mobil sistemin kendine özgü bir geliştirme dili ve ortamı vardır. Her ne kadar tek bir ortamda geliştirilen ve tümünde çalışabilen platformlar üzerinde çalışılsa da henüz daha tam bir başarı sağlanamamıştır.

Java programlama dili özellikle Android işletim sisteminde çalışacak uygulamaların geliştirilmesinde yaygın olarak kullanılmaktadır.

Açık Kaynak Yazılım Özgür Yazılım ve Mülkiyete Sahip Yazılım: Özgür yazılım (free software) ve açık kaynak (open source) yazılımın aralarında bazı farklar olmasına karşın temel prensipleri aynıdır. Bu prensipler:

- Bedavadır. Kullanabilmek için herhangi bir lisans ücreti ödenmesi gerekmez
- Kaynak kod sahiplenilemez, Kaynak kod üzerine ekleme yapıldığında kodlar kapatılamaz. Onların da açılması gerekir.
- Çoğaltılabilir, izin almadan kullanılabilir.

Bunun tam tersi mülkiyete sahip (propriatery) yazılımlardır. Bu yazılımlar para verilerek kullanılır. Çoğaltılamazlar.

IDE (Integrated Development Environment): Derleyiciler komut satırından çalıştırılan basit bir arayüze sahip programlardır. Aslında teorik olarak Java ile geliştirilecek her uygulama notepad gibi basit bir editörle yazılıp, komut satırından derlenebilir. Fakat pratikte bu şekilde program yazmak oldukça fazla zaman alır. Yazılım geliştirmeyi kolaylaştırmak için ismine IDE denilen özel yazılımlar bulunmaktadır. IDE lerin içerisinde editörler, yardımcı araçlar, test araçları, çeşitli kodları otomatik olarak yazabilen araçlar vs bulunur. Java ile uygulama geliştirilirken kullanılabilecek çeşitli IDE ler bulunmaktadır. Bunlardan en yaygın olarak kullanılanları Eclipse, IntelliJIdea ve Netbeans IDE leridir.

Klavyede Kullanılan Karakterlerin İngilizce Karşılıkları

Sembol	**İngilizce Karşılığı**
+	plus
-	minus, hyphen, dash
*	asterisk
/	slash
\	back slash
.	period, dot
,	comma
:	colon
;	semicolon
“	double quote
'	single quote
(...)	paranthesis left, right, opening, closing
[...]	(square) paranthesis left, right, opening, closing
{...}	brace left, right, opening, closing
=	equal sign
&	ampersand
~	tilda
@	at
<...>	less than, greater than, angular bracket
^	caret
|	pipe
_	underscore
?	question mark
#	sharp, number sign
%	percent sign
!	exclamation mark
$	dollar sign
...	ellipsis

Sayı Sistemleri

Günlük hayatta 10'luk sayı sistemini kullanıyoruz. 10'luk sistemde bir sayının değeri aslında her bir basamak değerinin 10 sayısının ilgili kuvvetiyle çarpımlarının toplanmasıyla elde edilir.

Örneğin 1273 = (3 * 1) + (7 * 10) + (2 * 100) + (1 * 1000)

Ancak bilgisayar sistemlerinde bütün bilgiler ikilik sistemde ifade edilir.
Genel olarak sayı sistemi kaçlıksa o sayı sisteminde o kadar sembol bulunur.
Örneğin 10'luk sistemde 10 adet sembol vardır ve bu semboller 0, 1, 2, 3, 4, 5, 6, 7, 8, 9'dur.
Aynı şekilde ikilik sayı sisteminde yalnızca iki adet sembol bulunur. Yani yalnızca 0 ve 1.

Bir sayıyı başka bir sayı sisteminde ifade etmek o sayının değerini değiştirmez. Yalnızca sayının gösteriliş biçimi değişir. Sayısal değeri 2 olan büyüklüğü çeşitli farklı sayı sistemlerinde farklı biçimlerde gösterebiliriz ama sayının büyüklüğünü değiştirmiş olmayız.

İkilik sistemde her bir basamağa 1 bit denir. Bit kelimesi ***binary digit*** sözcüklerinden türetilmiştir.

Örneğin 1011 sayısı 4 bittir. (Ya da 4 bit uzunluğundadır).

11011001 sayısı 8 bittir.
8 bitlik bir büyüklük bir byte olarak isimlendirilir.

1 kilobyte 1K = 1024 byte dır. (yani 2^{10} byte)
1 mega byte 1 MB = 1024 Kilo byte dır. (yani 2^{20} byte)
1 gigabyte 1 GB = 1024 MB dır. (yani 2^{30} byte)
1 terabyte 1 TB = 1024 GB dır. (yani 2^{40} byte)
1 petabyte 1PB = 1024 TB dır. (yani 2^{50} byte)
1 exabyte 1EB = 1024 PB dır. (yani 2^{60} byte)
1 zettabyte 1ZB = 1024 EB dir.(yani 2^{70} byte)
1 yottabyte 1YB = 1024 ZB dır.(yani 2^{80} byte)

Kilo büyüklük olarak 1000 kat anlamına gelmektedir, ancak bilgisayar alanında Kilo 2'nin 1000'e en yakın kuvveti olan 2^{10} yani 1024 kat olarak kullanılır.

4 bit 1 Nybble (Nibble şeklinde de yazılır)
8 bit 1 byte
16 bit 1 word
32 bit 1 double word
64 bit 1 quadro word

olarak da isimlendirilmektedir.

8 bitlik bir alana yazılacak en büyük sayı nedir?
1111 1111 = 255 dir. (1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255)

8 bitlik bir alana yazılacak en küçük sayı nedir?
0000 0000 = 0'dır.

İkilik sisteme ilişkin genel işlemler

i. İkilik sistemdeki bir sayının 10'luk sistemde ifade edilmesi:
ikilik sayı sisteminde ifade edilen bir sayının 10'luk sistemdeki karşılığını hesaplamak için en sağdan başlayarak bütün basamakları tek tek 2'nin artan kuvvetleriyle çarpılır. Örneğin :

$1\ 0\ 1\ 1 = 1 * 2^0 + 1 * 2^1 + 0 * 2^2 + 1 * 2^3 = 11$
0010 1001 = (1 * 1) + (1 * 8) + (1 * 32) = 41
ii. 10'luk sistemdeki bir sayının 2'lik sistemde ifadesi:

Sayı sürekli olarak 2 ye bölünür. Her bölümden kalan oluşturulacak sayı sırasıyla sağdan sola yerleştirilir.
İkinci bir yöntem ise 10'luk sistemde ifade edilen sayıdan sürekli olarak 2'nin en büyük kuvvetini çıkarmaktır. 2'nin çıkarılan her bir kuvveti için ilgili basamağa 1 değeri yazılır.

iii. İkilik sistemde ifade edilen bir sayının 1'e tümleyeni. Sayının tüm bitlerinin tersinin alınmasıyla elde edilir. Yani sayıdaki 1'ler 0 ve 0'lar 1 yapılır.

Bir sayının 1'e tümleyeninin 1'e tümleyeni sayının yine kendisidir.

iv. İkilik sistemde ifade edilen bir sayının 2'ye tümleyeninin bulunması:

Önce sayının 1'e tümleyeni yukarıdaki gibi bulunur. Daha sonra elde edilen sayıya 1 eklenirse sayının 2'ye tümleyeni bulunmuş olur. 2'ye tümleyeni bulmak için daha daha pratik bir yol daha vardır: sayının en sağından başlayarak ilk defa 1 görene kadar (ilk görülen 1 dahil) sayının aynısı yazılır, daha sonraki tüm basamaklar için basamağın tersi yazılır. (Yani 1 için 0 ve 0 için 1) Örneğin:

1110 0100 sayısının ikiye tümleyeni 0001 1100 dır.
0101 1000 sayısının ikiye tümleyeni 1010 1000 dır.

Bir sayının ikiye tümleyeninin ikiye tümleyeni sayının kendisidir.

Negatif bir tamsayı ikilik sistemde nasıl gösterilir?

Negatif tamsayıların da ifade edildiği ikilik sayı sistemine işaretli ikilik sayı sistemi (***signed binary system***) denir. İşaretli ikilik sayı siteminde, negatif sayıları göstermek için hemen hemen tüm bilgisayar sisteminde aşağıdaki yol izlenir:

Sayının en soldaki biti, yani en yüksek anlamlı biti (***most significant bi***t) işaret biti (***sign bit***) olarak kabul edilir. Ve bu bit ***1*** ise sayı negatif, bu bit ***0*** ise sayı pozitif olarak değerlendirilir. İkilik sistemde bir negatif sayı aynı değerdeki pozitif sayının ikiye tümleyenidir. Örnek olarak, ikilik sistemde yazacağımız –27 sayısı yine ikilik sistemde yazılan 27 sayısının ikiye tümleyenidir.

Pozitif olan sayıların değerini tıpkı işaretsiz sayı sisteminde olduğu gibi elde ederiz:

0001 1110 işaretli sistemde pozitif bir sayıdır. (Decimal olarak 30 sayısına eşittir.)

Ancak negatif olan sayıların değerini ancak bir dönüşümle elde edebiliriz:

1001 1101 işaretli sistemde negatif bir sayıdır. (Çünkü işaret biti 1)

2'lik sistemde ifade edilen negatif bir sayının 10'luk sistemde hangi negatif sayıya eşit olduğunu nasıl bulunur?

Sayının en soldaki biti (the most significant bit) işaret bitidir. Bu bit 1 ise sayı negatifdir. Sayının kaça eşit olduğunu hesaplamak için ilk önce sayının 2'ye tümleyeni bulunur. Ve bu sayının hangi pozitif sayıya karşılık geldiğini hesap edilir. Elde etmek istenen sayı, bulunan pozitif sayı ile aynı değerdeki negatif sayı olacaktır.

Örneğin 1001 1101 sayısının 10'luk sistemde hangi sayıya karşılık geldiği bulunmak istenirse:

Sayının en soldaki biti 1 olduğuna göre bu sayı negatif bir sayı olacaktır. Hangi negatif sayı olduğunu bulmak için sayının 2'ye tümleyenini alınır.

1001 1101 sayısının ikiye tümleyeni 0110 0011 sayısıdır.

Bu sayının 10'luk sistemde hangi sayıya denk olduğu hesaplanırsa:
(1 * 1 + 1 * 2 + 0 * 4 + 0 * 8 + 0 * 16 + 1 * 32 + 1 * 64 = 99)
İlk yazılan sayının -99 olduğu anlaşılmış olur.

10'luk sistemde ifade edilen negatif sayıların işaretli ikilik sistemde yazılması:

Önce sayının aynı değerli fakat pozitif olanı ikilik sistemde ifade edilir: Daha sonra yazılan sayının ikiye tümleyenini alınarak, yazmak istenilen sayı elde edilir.

Örnek: İkilik sistemde –17 yazmak istenirse;

önce 17 yazılır.	0001 0001
bu sayının 2'ye tümleyeni alınırsa	1110 1111 sayısı elde edilir.

Sayı değeri aynı olan Negatif ve Pozitif sayılar birbirlerinin ikiye tümleyenleridir.
İkilik sistemde gösterilmiş olsa da aynı sayının negatifiyle pozitifinin toplamı 0 değerini verecektir. (Deneyiniz!)

Bir byte'lık (8 bitlik) bir alana yazabileceğimiz (işaret bitini dikkate almadan) en büyük sayı 255 (1111 1111) ve en küçük sayı ise 0'dır.(0000 0000). Peki işaret biti dikkate alındığında 1 byte'lık alana yazılabilecek en büyük ve en küçük sayılar ne olabilir?

En büyük sayı kolayca hesaplanabilir. İşaret biti 0 olacak (yani sayı pozitif olacak) ve sayı değerini en büyük hale getirmek için diğer bütün bit değerleri 1 olacak, bu sayı 0111 1111 sayısıdır. Bu sayıyı desimal sisteme dönüştürürsek 127 olduğunu görürüz. Peki ya en küçük negatif sayı kaçtır ve nasıl ifade edilir?

0111 1111 sayısının ikiye tümleyenini alındığında –127 sayısını elde edilir.
1000 0001 (127) Bu sayıdan hala 1 çıkartabilir.
1000 0000 (-128) 1 byte alana yazılabilecek en küçük negatif sayıdır.

Burada dikkat edilmesi gereken iki önemli nokta vardır:

1 byte alana yazılabilecek en büyük sayı sınırı aşıldığında negatif bölgeye geçilir.
0111 1111 (en büyük pozitif tamsayı)
1 (1 toplarsak)
1000 0000 (-128 yani en küçük tamsayı)
yani 1 byte alana yazılabilecek en büyük tamsayıya 1 eklendiğinde 1 byte alana yazılabilecek en küçük tamsayıyı elde ederiz.
1 byte alana yazılabilecek en küçük tamsayıdan 1 çıkardığımızda da 1 byte alana yazılabilecek en büyük tamsayıyı elde ederiz.

16'lık sayı sistemi (hexadecimal system) ve 8'lik sayı sistemi (octal system)

Bilgisayarların tamamen 2'lik sistemde çalıştığını söylemiştik, ama yukarıda görüldüğü gibi 2'lik sistemde sayıların ifade edilmesi hem çok uzun hem de zahmetli. Bu yüzden, yazım ve algılama kolaylığı sağlamak için 16'lık ve 8'lik sayı sistemleri de kullanılmaktadır.
16'lık ve 8'lik sayı sistemlerinde sayılar daha yoğun olarak kodlanıp kullanabilir.

Başta da söz edildiği gibi 10'luk sistemde 10, 2'lik sistemde ise 2 sembol bulunmaktadır. Bu durumda 16'lık sayı sisteminde de 16 sembol bulunur.

İlk 10 sembol 10'luk sistemde kullanılan sembollerle tamamen aynıdır :
1, 2, 3, 4, 5, 6, 7, 8, 9,

Daha sonraki semboller

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15
16'lık sayı sisteminde yazılmış bir sayıyı 10'luk sisteme çevirmek için, en sağdan başlayarak basamak değerleri 16'nın artan kuvvetleriyle çarpılır :

01AF = (15 * 1) + (10 * 16) + (1 * 256) + (0 * 4096) = 431

10'luk sistemde yazılmış bir sayıyı 16'lık sisteme çevirmek için 10 luk sistemden 2'lik sisteme yapılan dönüşümlerdekine benzer şekilde sayı sürekli 16 ya bölünerek, kalanlar soldan sağa doğru yazılır.

Pratikte 16 lık sayı sistemlerinin getirdiği önemli bir avantaj vardır. Bu avantaj 16 lık sayı sistemi ile 2'lik sayı sistemi arasındaki dönüşümlerin kolay bir şekilde yapılmasıdır.

16'lık sistemdeki her digit 2'lik sistemdeki 4 bit (1 Nibble) alan ile ifade edilebilir :

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Örnek : 2ADF Hex sayısının (en sondaki H sayının hexadecimal olarak gösterildiğini anlatır yani sayıya ilişkin bir sembol değildir) 16'lık sistemde ifadesi :

2 = 0010
A = 1010
D = 1101
F = 1111

Bu durumda 2ADFH = 0010 1010 1101 1111

2'lik sistemden 16'lık sisteme yapılacak dönüşümler de benzer şekilde yapılabilir :
Önce sayıları sağdan başlayarak dörder dörder ayırırız (en son dört eksik kalırsa sıfır ile tamamlarız.) Sonra her bir dörtlük grup için doğrudan Hexadecimal karşılığını yazarız.

1010 1110 1011 0001 = AEB1H
0010 1101 0011 1110 = 2D3EH

Soru : hexadecimal sayı sisteminde 2 byte'lık bir alanda yazılmış olan 81AC H sayısı pozitif mi negatif midir?
Cevap : Sayının yüksek anlamlı biti 1 olduğu için, işaretli sayıs sistemlerinde sayı negatif olarak değerlendirilecektir. (1000 0001 1010 1100)

8'lik sayı sistemi

Daha az kullanılan bir sayı sistemidir.

8 adet sembol vardır. (0 1 2 3 4 5 6 7)
8'lik sayı sisteminin her bir digiti 2'lik sistemde 3 bit ile ifade edilir.

000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

8'lik sayı sisteminin de kullanılma nedeni, 2'lik sayı sistemine göre daha yogun bir ifade tarzı olması, ve ikilik sayı sistemiyle, 8'lik sayı sistemi arasında yapılacak dönüşümlerin çok kolay bir biçimde yapılabilmesidir.

GERÇEK SAYILARIN BELLEKTE TUTULMASI

Sistemlerin çoğu gerçek sayıları ***IEEE 754*** standardına göre tutarlar. ***(Institute of Electrical and Electronics Engineers)*** Bu standarda göre gerçek sayılar için iki ayrı format belirlenmiştir:

single precision format (tek hassasiyetli gerçek sayı formatı)

Bu formatta gerçek sayı 32 bit (8 byte) ile ifade edilir.
32 bit üç ayrı kısma ayrılmıştır.

1. İşaret biti (sign bit) (1 bit)
Aşağıda S harfi ile gösterilmiştir.
İşaret biti 1 ise sayı negatif, işaret biti 0 ise sayı pozitiftir.

2. Üstel kısım (exponent) (8 bit)
Aşağıda E harfleriyle gösterilmiştir.

3. Ondalık kısım (fraction) (23 bit)
Aşağıda F harfleriyle gösterilmiştir.

```
S       EEEEEEEE            FFFFFFFFFFFFFFFFFFFFFFF
31      30----------23       22----------------------------------0
```

Aşağıdaki formüle göre sayının değeri hesaplanabilir :

V sayının değeri olmak üzere:

E = 255 ise ve F 0 dışı bir değer ise V = NaN (Not a number) bir gerçek sayı olarak kabul edilmez.
Örnek :

```
0 11111111 00001000000100000000000    = Sayı değil
1 11111111 00010101010001001010101    = Sayı değil
```

E = 255 ise ve F = 0 ise ve S = 1 ise V = -sonsuz
E = 255 ise ve F = 0 ise ve S = 1 ise V = +sonsuz

0 < E < 255 ise

$V = (-1)^S * 2^{(E-127)} * (1.F)$

Önce sayının fraction kısmının başına 1. eklenir. Daha sonra bu sayı $2^{(E-127)}$ ile çarpılarak noktanın yeri ayarlanır. Noktadan sonraki kısım 2'nin artan negatif kuvvetleriyle çarpılarak elde edilecektir.

Örnekler :

0 10000000 00000000000000000000000 = $+1 * 2^{(128-127)} * 1.0$
= $2 * 1.0$
= 10.00
= 2

0 10000001 10100000000000000000000 = $+1 * 2^{(129-127)} * 1.101$
= $2^2 * 1.101$
= 110.100000
= 6.5

1 10000001 10100000000000000000000 = $-1 * 2^{(129-127)} * 1.101$
= $-2^2 * 1.101$
= 110.100000
= -6.5

0 00000001 00000000000000000000000 = $+1 * 2^{(1-127)} * 1.0$
= 2^{-126}

E = 0 ve F sıfır dışı bir değer ise

$V = (-1)^S * 2^{(-126)} * (0.F)$

Örnekler :

0 00000000 10000000000000000000000 = $+1 * 2^{-126} * 0.1$
=

0 00000000 00000000000000000000001 = $+1 * 2^{-126}$ 0. 00000000000000000000001
= 2^{-149} (en küçük pozitif değer)

E = 0 ve F = 0 ve S = 1 ise V = -0

E = 0 ve F = 0 ve S = 0 ise V = 0

double precision format (çift hassasiyetli gerçek sayı formatı)

Bu formatta gerçek sayı 64 bit (8 byte) ile ifade edilir.
64 bit üç ayrı kısıma ayrılmıştır.

1. İşaret biti (sign bit) (1 bit)
Aşağıda S harfi ile gösterilmiştir.
İşaret biti 1 ise sayı negatif, işaret biti 0 ise sayı pozitiftir.

2. Üstel kısım (exponent) (11 bit)
Aşağıda E harfleriyle gösterilmiştir.

3. Ondalık kısım (fraction) (52 bit)
Aşağıda F harfleriyle gösterilmiştir.

S EEEEEEEEEEE FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
63 62----------52 51----------0

Aşağıdaki formüle göre sayının değeri hesaplanabilir :

Aşağıdaki formüle göre sayının değeri hesaplanabilir :

V sayının değeri olmak üzere:

E = 2047 ise ve F 0 dışı bir değer ise V = NaN (Not a number) bir gerçek sayı olarak kabul edilmez.

E = 2047 ise ve F = 0 ise ve S = 1 ise V = -sonsuz
E = 2047 ise ve F = 0 ise ve S = 1 ise V = +sonsuz

0 < E < 2047 ise

$V = (-1)^S * 2^{(E -1023)} * (1.F)$

Önce sayının fraction kısmının başına 1. eklenir. Daha sonra bu sayı $2^{(E-1023)}$ ile çarpılarak noktanın yeri ayarlanır. Noktadan sonraki kısım 2'nin artan negatif kuvvetleriyle çarpılarak elde edilecektir.

E = 0 ve F sıfır dışı bir değer ise

$V = (-1)^S * 2^{(-126)} * (0.F)$

E = 0 ve F = 0 ve S = 1 ise V = -0

E = 0 ve F = 0 ve S = 0 ise V = 0

İşletim Sisteminin Java ile Uygulama Geliştirmeye Hazır Hale Getirilmesi

Aşağıdaki sıralamaya ve açıklamalara göre işlemler yapılırsa çok fazla detay gerektirmeden sistem Java ile uygulama geliştirmeye hazır hale getirilebilmektedir.

Java Geliştirme Ortamının Kurulması: Java geliştirme ortamı için birden fazla alternatif bulunmaktadır. Oracle firmasnın ürünü olan JDK (son versiyonu JDK 8) istenilen işletim sitemi için indirilebilmetedir. (http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html). Windows işletim istemi için indirilen exe program doğrudan tüm yapılandırmayı yapmaktadır. Programcının herhangi bir şey yapmasına gerek olmayacaktır.
Linux işletim sistemi için Java ortamının kurulması dağıtımlara ilişkin paket yöneticileri kullanılarak yapılabilir. Örneğin Ubuntu Linux dağıtımı için terminal ekranından aşağıdaki gibi Java ortamı kurulabilmektedir. Komutlar sırasıyla terminal ekranında yazılmalıdır:

- sudo apt-get update
- sudo apt-get install oracle-java8-installer

Ayrıca açık kaynak kodlu olarak ismine OpenJDK denilen ayrı bir ürün de bulunmaktadır.

Uygulama Geliştirirken Kullanılacak IDE lerin yapılandırılması

Bilindiği gibi Java ile uygulama geliştirmede pek çok IDE kullanılabilmektedir. Burada yaygın olarak kullanılan Eclipse, IntelliJIDEA ve Netbeans IDE lerinin Java ile uygulama geliştirmeye hazır hale getirilmesi anlatılacaktır. Bu IDE lerin Windows, Linux ve MacOS X işletim sistemleri için versiyonları bulunmaktadır.

Eclipse IDE sinin kurulumu ve Yapılandırılması: Eclipse IDE si aslında genel amaçlı bir IDE olmasına karşın en çok Java ile geliştirilen uygulamalar için kullanılmaktadır. Ayrıca Eclipse IDE sinin MyEclipse isminde lisanslı versiyonu da bulunmaktadır. Eclipse IDE si çeşitli eklentilerle bir çok ortam için kullanılabilir duruma getirilebilmektedir. Eclipse IDE si işletim sistemine kurulum gerektirmeyen bir IDE dir. Çalıştırılabilir dosya ve diğer dosyalar da dahil tek bir klasör içerisinde bulunmaktadır. Eclipse IDE si nin çalışılan işletim istemi için bedava versiyonu www.eclipse.org sitesinden indirilebilir. JDK doğru şekilde kurulduktan sonra ayrıca bir yapılandırma gerektirmemektedir.

Dil Kavramı

Dil iletişimde kullanılan semboller topluluğudur. Bir dilin kurallarına gramer denir. Gramer kavramının çeşitli alt alanları vardır. Bunlardan en önemlileri sentaks (syntax)ve semantik (semantic) alanlarıdır. Bir olgunun dil olabilmesi için en azından sentaks ve semantiğe sahip olması gerekir.

Sentaks doğru yazıma ve doğru dizilime ilişkin kurallardır. Örneğin:

“I a student am”

cümlesinde sentaks hatası yapılmıştır. Öneğin:

System.out.println(“Merhaba Java”;
Java kodunda kapanan parantez konmadığından yazıma ilişkin hata yapılmıştır. Yani sentaks hatası yapılmıştır.

Dildeki doğru dizilmiş öğrelerin ne anlam ifade ettiğine ilişkin kurallara semantik denilmektedir.

Doğal diller ve programlama dillerinin her ikisinde de sentaks ve semantik kurallar vardır. Doğal diller de ayrıca morfoloji gibi bazı kavramlar da vardır.

Bilgisayar Dilleri ve Programlama Dilleri

Bilgisayar bilimlerinde kullanılmak üzere tasarlanmış dillere bilgisayar dilleri (computer languages) denir. Bir bilgisayar dilinde akış da varsa o zaman programlama dili (programming language) denilmektedir. Her programlama dili bir bilgisayar dilidir, ancak her bilgisayar dili bir programlama dili olmayabilir. Örneğin: HTML bir bilgisayar dilidir fakat akış olmadığı için programlama dili değildir. C, C++, Java, C# dillerinde akış olduğu için bu diller programlama dilleridir.

Merhaba Java Programı

İskelet java programı ile “Merhaba, Java” yazısını basan temel bir program şöyledir:

```
package csd;

class App {
	public static void main(String [] args)
	{
		System.out.println("Merhaba, Java");
	}
}
```

Programın Derlenmesi ve Çalıştırılması

Merhaba Java programı aşağıdaki adımlardan geçilerek derlenebilir:

1. Notepad gibi bir editörde yukarıdaki program yazılarak csd isimli bir dizin içerisinde App.java isimli bir dosya olarak kaydedilir. Java derleyicileri .java uzantılı dosyanın isminin .java dan önceki kısmı ile aynı isimli public bir sınıfın o dosyada bulunmasını zorunlu tutmaktadır. Bu sebeple yukarıdaki programın yazıldığı dosyanın ismi App.java olmalıdır. Ayrıca Java da bir sınıfı hangi paket içerisinde bildirilmişse üretilecek olan arakodun çalışıtırilması için o paket isminde bir klasör içerisinde bulunması zorunludur. Bu konu ileride ele alınacaktır.

2. Komut satırına geçilir. Oradan programın bulunduğu dizine geçilir. (Programın bulunduğu dizin Merhaba Java uygulamasında csd dizinidir)

3. Komut satırından aşağıdaki komut uygulanır.

javac <.java dosyası>

Örnek:

javac App.java

“javac” Java programlama dili ile yazılmış programların derlenmesi için kullanılan derleyici programın ismidir.

Anahtar Notlar: *Windows işletimde komut yorumlayıcı programın ismi cmd.exe programıdır. cmd.exe program için bazı komutlar şunladır:*
e: → *Sürücü değiştirme komutu (Burada e sürüc harfi)*
cd → *Dizin değiştime (Change Directory)*
dir → *İçerik görüntüleme*
cd .. → *Bir üst dizine git*
Derlenmiş ve arakodu üretilmiş olan program aşağıdaki gibi çalıştırılabilir:

1. Arakodun bulunduğu dizinin bir üst dizinine geçilir. (Burada App.class arakodunun csd dizininde olması gerektiğinden csd dizininin bulunduğu dizine geçilmelidir)
2. Komut satırından aşağıdaki komut uygulanır:

java <dizin ismi (paket ismi)>.<class dosyasınının .class uzantısı hariç adı>

Örneğin:

java csd.App

“java” programı ile, javac ile üretilen byte kod çalıştırılabilmektedir.

Atom Kavramı

Bir programlama dilinde kendi başına anlamlı olan en küçük yapı birimine atom (token) denir. Atomlar 6(altı) gruba ayrılır:

1. Anahtar Sözcükler (Keywords / Reserved Words): Dil için özel anlmı olan ve değişken ismi olarak kullanılması yasak olan atomlardır. Örneğin: public, if, while int gibi

2. Değişkenler (Identifiers /Variables): İsmini programcının belli bir kurala belirdiği atomlardır. Örneğin: “Merhaba, Java” programındaki App ismi gibi.

3. Operatörler (Operators): Bir işleme yol açan ve işlem sonucunda bir değer üretilmesini sağlayan atomlardır. Örneğin: a * b ifadesindeki * operatör atomdur.

4. Sabitler (Literals / Constants): Program kodu doğrudan yazılmış sayılara denir. Örneğin: a * 8 ifadesindeki 8 sabit atomudur.

5. Stringler (String literals): İki tırnak içerisindeki yazılara tırnaklarıyla birlikte string atomu denir.

6. Ayıraçlar (Delimiters/ Punctuators): Yukarıdaki gruplar dışında kalan tüm atomlara ayıraç atomlar denilmektedir.

Boşluk Karakterleri (white space): Klavyeden boşluk duygusu yaratmak için basılan karakterlere boşluk karakterleri (white spaces) denilmektedir. Boşluk karakterleri tipik olarak SPACE, TAB, ENTER karakterleridir.

Java Programlama Dilinin Yazım Kuralı

Java programlama dilininin yazım kuralı iki madde ile özetlenebilir:

1. Atomlar arasında istenildiği kadar boşluk bırakılabilir.
2. Anahtar sözcükler ve değişkenler dışında kalan atomlar istenildiği kadar bitişik yazılabilirler (Anahtar sözcüklerle değişken arasında en az bir boşluk karakteri olmalıdır).

Derleyicilerin Hata Mesajları

Derleyicilerin hata mesajları 3(üç) gruba ayrılmaktadır:

1. Uyarılar (Warnings): Byte kodun (.class uzantılı dosya) oluşmasına engel olmayan, olası mantık hatalarının programcıya bildirilmesi amacıyla kullanılan mesajlardır. Programcı uyarıları dikkate almalıdır. Bir programın özel bir durum yoksa hiç uyarı mesajı alınmadan derlenmesi hedeflenmelidir.

2. Gerçek Hatalar (Errors): Dilin sentaks ve semantik kurallarına uyulmamasından kaynaklanan hatalardır. Bir programda tek bir gerçek hata olsa bile byte kod üretilmez. Şüphesiz programcı tüm gerçek hataları düzelttiğinde programı derleyebilir ve ara kod üretilebilir.

3. Ölümcül Hatalar (Fatal Errors): Derleme işleminin bitirilmesini dahi engelleyecek ciddi hatalardır. Programcının kod olarak yapabileceği bir şey yoktur. Genellikle sistemdeki problemler yüzünden ortaya çıkmaktadır. Örneğin: diskte boş alan olmaması, işletim sisteminde bir problem, bellekte yer olmaması gibi.

Sentaks için Genel Biçimler

Programlama dillerinin standartlarında ya da spesifikasyonlarında iki anlamlılığa (ambiguity) izin vermeyen BNF (Backus-Naur Form) notasyonu ya da türevleri kullanılmaktadır. Fakat bu notasyon biraz karışıktır. Kurs içerisinde, açısal parantez, köşeli parantez tekniği kullanılacaktır. Açısal parantezler içerisinde yazılanlar mutlaka olması gereken öğeleri, köşeli parantez içerisinde yazılanlar ise isteğe bağlı öğreleri belirtecektir. Bunların dışındaki tüm atomlar aynı pozisyonda aynı biçimde bulundurulmak zorundadır. Ayrıca, "burada bir takım öğeler olabilir ancak bizi ilgilendirmiyor" anlamına gelen aşağıdaki gibi bir biçim de kullanılacaktır:

//...

Burada // atomu yorum satırı oluşturmak için kullanılmaktadır. Yorum satırı kavramı ileride açıklanacaktır.

Merhaba Java Programının Açıklaması

Genel olarak söylemek gerekirse bir Java programı paketlerden (package), paketler sınıflardan (class), sınıflar da metotlardan oluşur. Bir sınıfı bir paket içerisinde bildirmek için o sınıfın .java dosyasının en tepesinde aşağıdaki gibi bildirim yapmak gerekir:

package <paket ismi>;

Bununla birlikte bir sınıf bir paket içerisinde ise diskte o paket isminde bir dizinin (folder) içerisinde bulunması gerekmektedir. Derleme sonucunda üretilen arakodun da paket isminde bir dizin içerisinde olması zorunluluktur. Paket kavramı daha sonra detaylı olarak ele alınacaktır.

Sınıf bildiriminin genel biçimi şöyledir:

```
[erişim belirleyici] class <sınıf ismi> {
        //...
}
```

Burada görüldüğü gibi sınıf bildiriminin başındaki erişim belirleyici olmayabilir. Sınıf kavramı daha sonra detaylı olarak ele alınacaktır.

Metot bildiriminin genel biçimi şöyledir:

```
[erişim belirleyici] [static] <geri dönüş değerinin türü><metot ismi>([parametre bildirimi])
{
        //...
}
```

Java programlama dilinde alt programlara metot denir. Erişim belirleyici şunlardan biri olabilir:

public
protected
private

Erişim belirleyiciyi hiç yazılmayabilir (no modifier). Erişim belirleyiciler detaylı olarak ileride ele alınacaktır. Bir metot ayrıca static anahtar sözcüğü ile bildirilirse static metot olur.

Java programları main isimli bir metottan çalışmaya başlar.

Bir metodun bildirilmesi (declaration) demek onun programcı tarafından yazılması demektir. Çağrılması (call) ise çalıştırılması demektir. Static bir metodu çağırmanın genel biçimi şöyledir:

[paket ismi][.][sınıf ismi][.]<metot ismi>([argüman listesi]);

Örneğin:
csd.Sample.foo(10);

Ekrana yazdırmak için kullanılan metodlar static değildir. System standart bir sınıftır ve java.lang paketi içerisinde bulunmaktadır. out PrintStream sınıfı türünden bir referanstır. println ise PrintStream

sınıfının static olmayan bir metottur. println metodu kendisine gönderilen argümanı ekrana basar. Merhaba Java programında iki tırnak içerisindeki yazıyı ekrana basmaktadır. Ayrıca PrintStream sınıfının print isimli bir metodu daha vardır. Bu metot da ekrana yazmakiçin kullanılır ancak ekrana yazma işleminden sonra imleci (cursor) yazının sonunda bırakır, println ise aşağı satırın başında bırakır.

Anahtar Notlar: *Standart paketler ve sınıfların hepsi java isimli paketin altında bildirilmiştir. Çok kullanılan sınıflar da java isimli paketin içerisindeki lang isimili pakette bildirilmiştir. java.lang paketi içerisindeki sınıflar çok fazla kullanıldığından herhangi bir işlem yapmadan (import yada nitelikli kullanım vs) içerisindeki sınıflar doğrudan kullanılabilir. Bu konu ileride daha detaylı olarak anlatılacaktır.*

Anahtar Notlar: *Referans ve static olmayan metot kavramları daha sonra ele alınacaktır.*

Bir Java programı pek çok paketten, pek çok sınıftan ve pek çok metottan oluşabilir. main metodu herhangi bir paket içerisindeki herhangi bir sınıf içerisinde olabilir. main metodu public ve static olmak zorundadır.

Anahtar Notlar: *Kurs içerisinde metot ismi olarak foo, bar, tar, zar, baz gibi isimler sıklıkla kullanılacaktır. Bu isimlerin hiçbir anlamı yoktur. Tamamen uydurma isimlerdir. Pek çok kaynakta (kitap, makale vs.) bu isimler kullanılmaktadır.*

Eğer çağrılan metodun (callee) bildirimi çağıran metodun (caller) bildirimi ile aynı paket içerisindeyse (farklı sınıflarda olabilirler) paket ismi hiç yazılmayabilir (istenirse yazılabilir). Örneğin:

```
package csd;

class App {
	public static void main(String [] args)
	{
		Sample.foo();
	}
}

class Sample {
	public static void foo()
	{
		System.out.println("foo");
	}
}
```

Çağrılan metot ile çağıran metodun bildirimi aynı sınıf içerisindeyse sınıf ismi de yazılmayabilir. (istenirse yazılabilir). Örneğin:

```
package csd;

class App {

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

```
	{
		Sample.bar();
	}
}

class Sample {
	public static void bar()
	{
		System.out.println("bar");
		foo();
	}

	public static void foo()
	{
		System.out.println("foo");
	}
}
```

Yorum Satırları

Bir programın kaynak kodunda derleyici tarafından dikkate alınmayan kod bölümlerine yorum satırları (comment lines) denilmektedir. Yorum satırları program içerisinde çeşitli açıklamalar yapmak amacıyla kullanılabilir. Program kodu içerisinde gerektiğinden fazla yorum satırı kullanmak kodun okunabilirliğini bozabilmektedir. Buna dikkat edilmesi gerekir. Java programlama dilinde yorum satırları iki biçimde oluşturulabilir:

1. // atomu ile oluşturulan yorum satırları yalnız o satırın // atomundan sonraki kısmının derleyici tarafından yok sayılması amacıyla kullanılır.

2. /* ve */ atomları arasında kalan kod bölümleri derleyici tarafından dikkate alınmaz. Bu şekilde birden fazla satır için yorum satırları oluşturulabilir.

Örneğin:

```
package csd;

class App {
	public static void main(String [] args)
	{
		/*

			Bu yazı derleyici
			tarafından dikkate
		alınmayacak


			*/

		System.out.println("Merhaba, Java");
```

```
            //Bu satır derleyici tarafından dikkate alınmayacak
        }
}
```