JAVA İLE PROGRAMLAMAYA GİRİŞ

Ekim 6 2008

Dr. Galip Aydın

İlkel Veri Türleri

Type Name	Kind of Value	Memory Used	Size Range
byte	integer	1 byte	-128 to 127
short	integer	2 bytes	-32768 to 32767
int	integer	4 bytes	-2,147,483,648 to 2,147,483,647
long	integer	8 bytes	-9,223,372,036,854,775,808 to 9,223,374,036,854,775,808
float	floating point	4 bytes	+/- 3.4028… x 10^{+38} to +/- 1.4023… x 0^{-45}
double	floating point	8 bytes	+/- 1.767… x 10^{+308} to +/- 4.940… x 0^{-324}
char	single character (Unicode)	2 bytes	all Unicode characters
boolean	*true* or *false*	1 bit	not applicable

İlkel Veri Türleri

- **`int`**
  - Tam sayilar
  - + veya - olabilir
  - Ondalik kismi yok
- **`char`**
  - Tek karakter
  - Tek tirnak kullanilir
  - mesela
    `char not = `A`;`
- **`double`**
  - Gercek sayilar, pozitif ve negatif
  - Ondalik kismi vardir
  - İki şekilde
    - Ondalikli gosterim, 514.061
    - e (or *bilimsel*, veya kayan nokta*)* gosterimi, mesela 5.14061 e2 yani 5.14061×10^2

İlk Java Programı

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

Java ve Javac komutları

- C:\Program Files\Java\jdk1.6.0_02\bin
- javac.exe, java compiler, derleme işlemi
- java.exe, programi calistirir
- Windows PATH, JDK/bin klasorunu icermelidir
- *Javac Test.java*, programı derler ve *Test.class* adında byte code dosyası oluşturur.
- *java Test* komutu bu class dosyasını çalıştırır

Değişkenler

- double maas;
- int ogrenciSayisi;
- long dunyaninNufusu;
- boolean bittimi;

```
int a,b=5;
a=3;
  System.out.println("a=  " + a + "\nb= " + b);
```

Escape (kaçış) Karakterleri

\b	Backspace
\t	Tab
\n	Yeni Satir
\r	Satirin basina git
\”	Cift Tirnak
\’	Tek tirnak
\\	Ters slash

```
public class Lab2_4 {
    public static void main(String[] args) {
        System.out.println(
                "        *\n" +
                "      *   *\n" +
                "    *       *\n" +
                "  *           *\n" +
                "    *       *\n" +
                "      *   *\n" +
                "        *\n");
    }
}
```

Kısayol Operatörleri

Bazı yaygın işlemler için kısayollar tanımlanmıştır

```
i = i + 1;          i += 1;          i++;

d = d - 1.0;        d -= 1.0;        d--;

f = f / 2.0;        f /= 2.0;
```

STRING CLASS

String class

- String karakterler dizisidir
- String kus = “mavi kanatli kus”;
- String değişkeni tanımlanması

```
String isim;
```

- Değişkene değer atanması

```
isim = “Şerafettin";
```

- Stringler metodlarda argüman olarak kullanılır

```
System.out.println(isim);
```

Stringlerin eklenmesi

Stringler birbirine “+” operatorü ile eklenir:

```
String ad = “Ali";
String soyad= “Demir";
System.out.println(“Calıskan ogrenci“ + ad +
  soyad);
```

Çıktısı :

```
> Calıskan ogrenciAliDemir
```

Boşlukları unutmayalım:

```
System.out.println(“Calıskan ogrenci “+ ad +
                    “ “ + soyad);
```

Çıktısı:

```
> Calıskan ogrenci Ali Demir
```

String karakterleri

- Bir string içindeki bir karakterin indeksi ilk karakter için 0 olmak üzere bir tam sayıdır.
- **`charAt(index)`** metodu verilen indeksteki karakteri döndürür
- **`substring(ilk, son)`** metodu *ilk* ve *son* indeksleri arasındaki stringi döndürür.
- Mesela:

```
String cumle = "Sali gunu odev gunu";
cumle.charAt(0)-> S
cumle.charAt(5)-> g
cumle.substring(5,8)-> gun
```

S	a	l	i		g	u	n	u
0	1	2	3	4	5	6	7	8

String İşlemleri

- String cumle = “kelimeler kifayet etmez”;
- int uzunluk = cumle.length();
- String buyukHarflerle = cumle.toUpperCase();
- String kucukHarflerle = cumle.toLowerCase();
- String altCumle = cumle.substring(0,8);
  - Kelimeler
- boolean ilkHarfTest = cumle.startsWith("A");

AKIŞ KONTROLÜ

- Dallanma
- Çoklu Dallanma
- Döngüler

Akış Kontrolü

- Akış kontrolü komutların bir programdaki komutların işletilmesi sırasıdır.
- Programlar üç tip akış kontrolü ile yazılabilir:
  1. **Sırayla**- sonraki komutu çalıştır
  2. **Dallanma** veya **Seçme** - en azından iki seçenek gerekir
     - Ya sonraki komutu işlet
     - Veya başka bir komuta atla
  3. **Döngü** veya **Tekrar** - döngü (bir blok kodu tekrar çalıştır)
     döngünün sonunda
     - Ya geri git ve kod bloğunu tekrar et
     - Veya bloktan sonraki komutu çalıştır

Javada Akış Kontrolü

Sırayla

- the default
- Java otomatik olarak sonraki komutu çalıştırır

Dallanma

- if
- if-else
- if-else if-else if- … - else
- switch

Döngü

- while
- do-while
- for

Javada `if` Yapısı

- Basit seçimler için
- Eğer test doğru ise komutu işlet, yanlışsa işletmeden atla
- Syntax:

```
if (Boolean_Test)
    komut;  //yalnızca test doğruysa işlet
Sonraki komut;  //her zaman işletilir
```

`if` Örnek

```
if(ogrenciSayisi >= 3){
  //if blogu baslangici
    System.out.println("Ders islemek icin sayı yeterli");
    System.out.println("Yoklama alalim");
  //if blogu sonu
    yoklamaAl();
}
islenenDersSayisi = islenenDersSayisi + 1;

System.out.println("İslediğimiz ders sayisi = " + dersSayisi);
```

- if bloğu ancak şartlı olarak yürütülür
- if bloğundan sonraki komutlar her zaman yürütülür

Çoklu seçim: `if-else`

- İki seçenekten birisini seç
- Testin sonucuna bağlı olarak ya işlem1 ya da işlem2
- Syntax:

```
if (Boolean_Test)
{
    İşlem1 //Sadece test doğruysa işlet
}
else
{
  İşlem2//sadece test yanlışsa işlet
}
İşlem3//her zaman yürütülür
```

if-else

```
if(toplamParaniz > kitapFiyati)
   System.out.println("Kitabi alabilirsin.");
else
   System.out.println("Biraz daha para bul.");
```

- Çoklu komutlar

```
if(toplamParaniz > kitapFiyati)
{
  System.out.println("Kitabi alabilirsin.");
  toplamParaniz = toplamParaniz - kitapFiyati;
}
else
{
  System.out.println("Biraz daha para bul.");
  gerekliPara = toplamParaniz + eksikMiktar;
}
```

Boolean değerler

- Doğru veya yanlış değerleri alan değişken veya ifadelere boolean değişkenler denir.
- boolean değişkenin değeri `true` veya `false olabilir`
- `Örnek:`

  `A sayısı B sayısından büyük mü`

  `A sayısı B sayısına eşit mi`

  `vb`

Java Karşılaştırma sembolleri

Matematiksel Notasyon	Ad	Java Notasyon	Java Örnekleri
=	eşittir	==	sayac == 0 cevap == 'y'
≠	eşit değil	!=	gelir != gider cevap != 'y'
>	büyüktür	>	gelir > gider
≥	büyük veya eşit	>=	puan >= 60
<	daha küçük	<	basınç < max
≤	küçük veya eşit	<=	gelir <= gider

Birleşik Boolean İfadeler

- Birden fazla şartı VE ile test etmek için `&&` kullanılır
  - İfade eğer her iki kısım da doğruysa doğru olur.
  - `A && B` ancak hem A hem de B doğruysa doğru olur

- Birden fazla şartı VEYA testine tabi tutmak için `||` kullanılır
  - İfade ya şartlardan biri veya her ikisi de doğruysa doğru olur.
  - `A || B` ifadesi A veya B nin doğru olduğu veya her ikisinin de doğru olduğu durumlarda doğru olur

Birleşik Boolean İfadeler

- Örnek: B'nin değerinin 0 veya A ile C arasında olup olmadığını test eden bir ifade yazınız

```
(B == 0) || (A <= B && B < C)
```

- A 3 veya 6'ya eşitse

```
(A == 3) || (A == 6)
```

- A 3'e ve B 6'ya eşitse

```
(A == 3) && (B == 6)
```

Çoklu dallanma seçimi: **`switch`**

```
switch(Kontrol ifadesi)
{
 case case_etiketi:
      komutlar
      …
      break;
   case case_etiketi :
      komutlar
      …
      break;

   default:
      komutlar
       …
     break;
}
```

- Çoklu dallanmaları programlamak için diğer bir yol.
- `Kontrol ifadesi` kullanılarak hangi ifadenin işletileceğine karar verilir.
- `Kontrol ifadesi` *`char`*, *`int`*, *`short`* veya *`byte`* türlerinden biri olmalıdır.
- `Kontrol ifadesi` ve*`Case Etiketi`* aynı türden olmalıdır.

Çoklu dallanma seçimi: `switch`

```
switch(Kontrol ifadesi)
{
 case case_etiketi:
      komutlar
      …
      break;
   case case_etiketi :
      komutlar
      …
      break;

   default:
      komutlar
        …
     break;
}
```

- `break` ile karşılaşınca blogun dışına çıkar.
- `break` ihmal edilebilir.
- Sınırsız sayıda case olabilir.
- `default` case kullanılması zorunlu değildir.

`switch` Örnek

```
switch(filmGunu)
{
 case 1:
    System.out.println("Pazartesi");
    biletFiyati  = 10;
    break;
 case 2:
    System.out.println("Sali");
    biletFiyati = 7;
    break;
 case 3:
    System.out.println("Carsamba");
    biletFiyati = 15;
    break;
 default:
    System.out.println("Gosterim    olmayan bir gun
  seçtiniz");
    break;
}
```

Tekrarlama: Döngüler (Loops)

- Yapı:
  - ***Genelde başlangıç şartları öncelikle tanımlanır***
  - ***loop gövdesi***
  - ***loop sonlandırma şartları***
- Çeşitli mantıksal türleri vardır
  - sayan döngüler
  - Gözcü kontrollü döngüler
  - sonsuz döngüler
  - minimum sıfır veya bir defa çalışma
- Çeşitli şekillerde programlanırlar
  - ***while***
  - ***do-while***
  - ***for***

`while` döngüsü

- Syntax:

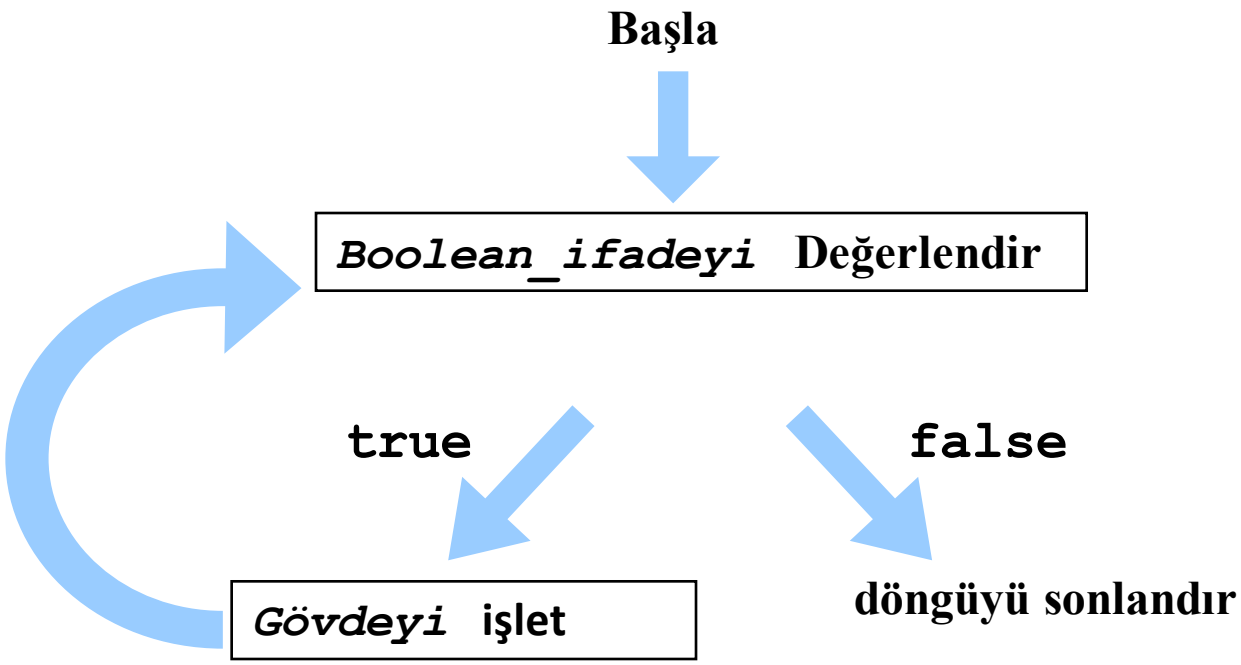
```
while(boolean_ifade)
{
    //döngü gövdesi
    birinci komut;
    ...
    son komut;
}
```

Döngü gövdesindeki herhangi bir şey mantıksal ifadenin mutlaka yanlış olmasına sebep olmalıdır.

- Başlangıç ifadeleri genelde döngüden önce yazılır.
- *boolean _ifade* döngü sonlandırma şartıdır.
- Döngü *boolean _ifade* doğru olduğu sürece çalışır.
- Sayan veya gözcü döngüleri olabilir

`while` döngüsü

```
while (Boolean_ifade)
        gövde
```



`while` : Sayan döngü örneği

- Kullanıcı tarafından girilen 10 sayıyı toplayan program

```
int sonraki;
//döngü başlangıcı
int sayac = 1;
int toplam =0;
while(sayac <= 10) //döngü sonlandırma
  koşulu
{ //döngü gövdesi
    sonraki = giris.nextInt();
    toplam = toplam + sonraki;
    sayac++; // döngü sonlandırma sayacı
}
```

`while`: Gözcü kontrollü döngü örneği

- `sonraki` **`gözcüdür`**
- döngü negatif bir sayı girilince sona erer

```
//Başlangıç
int sonraki = 0;
int toplam = 0;
while(sonraki >= 0) //sonlandırma şartı
{ //gövde
  toplam = toplam + sonraki;
   sonraki = giris.nextInt();
}
```

`while`: Minimum sıfır tekrar

- birinci giriş değeri döngüden önce okunup test edildiği için *`while`* döngüsünün gövdesi hiç çalıştırılmayabilir

```
int sonraki;
int toplam= 0;
sonraki = giris.nextInt();
while(sonraki >= 0)//sonlandırma şartı
{ //Body
  toplam = toplam + sonraki;
 sonraki = giris.nextInt();
}
```

- Eğer kullanıcının girdiği ilk numara negatif ise döngüye girilmez

`do-while` döngüsü

- Syntax

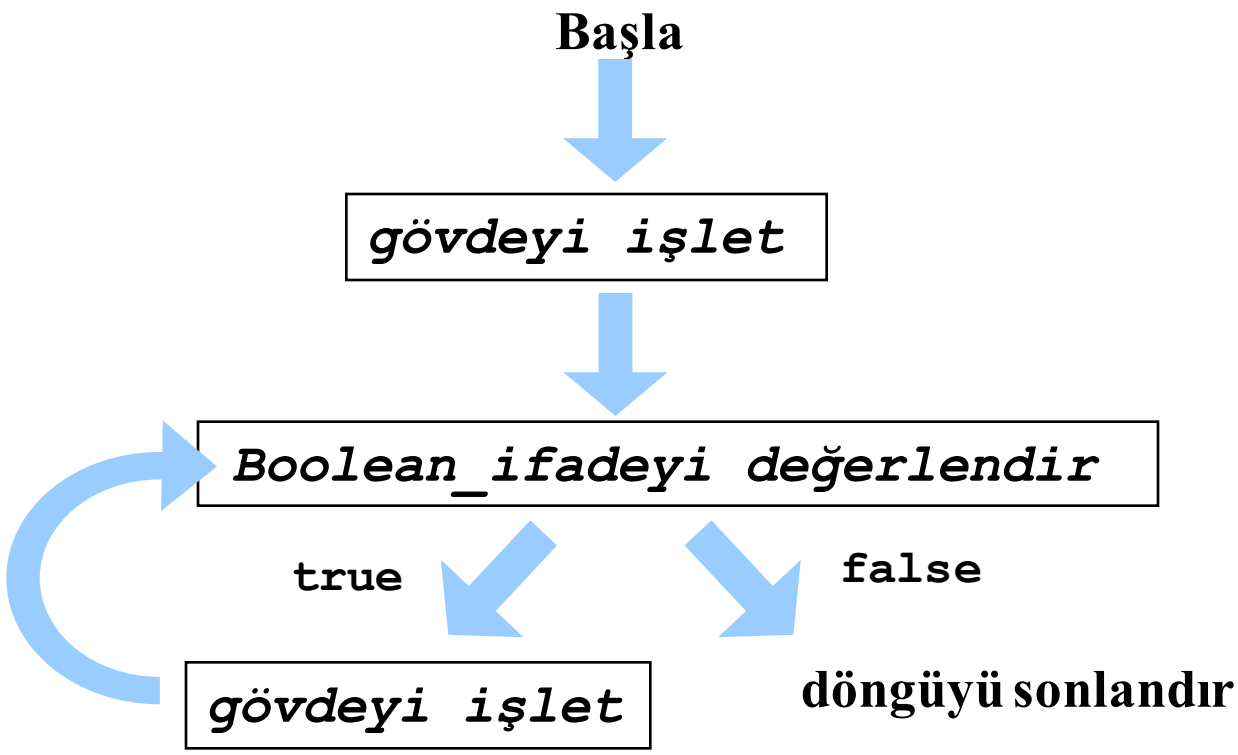
```
do
{   //döngü gövdesi
    ilk komut;
    ...
    son komut;
} while(Boolean_Ifade);
```

döngü gövdesinde bir ifade mutlaka *Boolean_ifadenin* yanlış olmasını sağlamalıdır

- Başlangıç kodu döngüden önce olabilir
- Döngü testi gövdeden sonra olduğu için gövde en az bir kere işletilir (minimum bir döngü)

do-while döngüsü

```
do
    gövde
while (Boolean_ifade);
```



do-while örnek

```
int sayac = 1;
int sayi = 5;
do //1'den 5'e kadar sayıları bir satırda göster
{
   System.out.print(sayac + " ");
   sayac++;
} while(sayac <= sayi);
```

Çıktı:

1 2 3 4 5

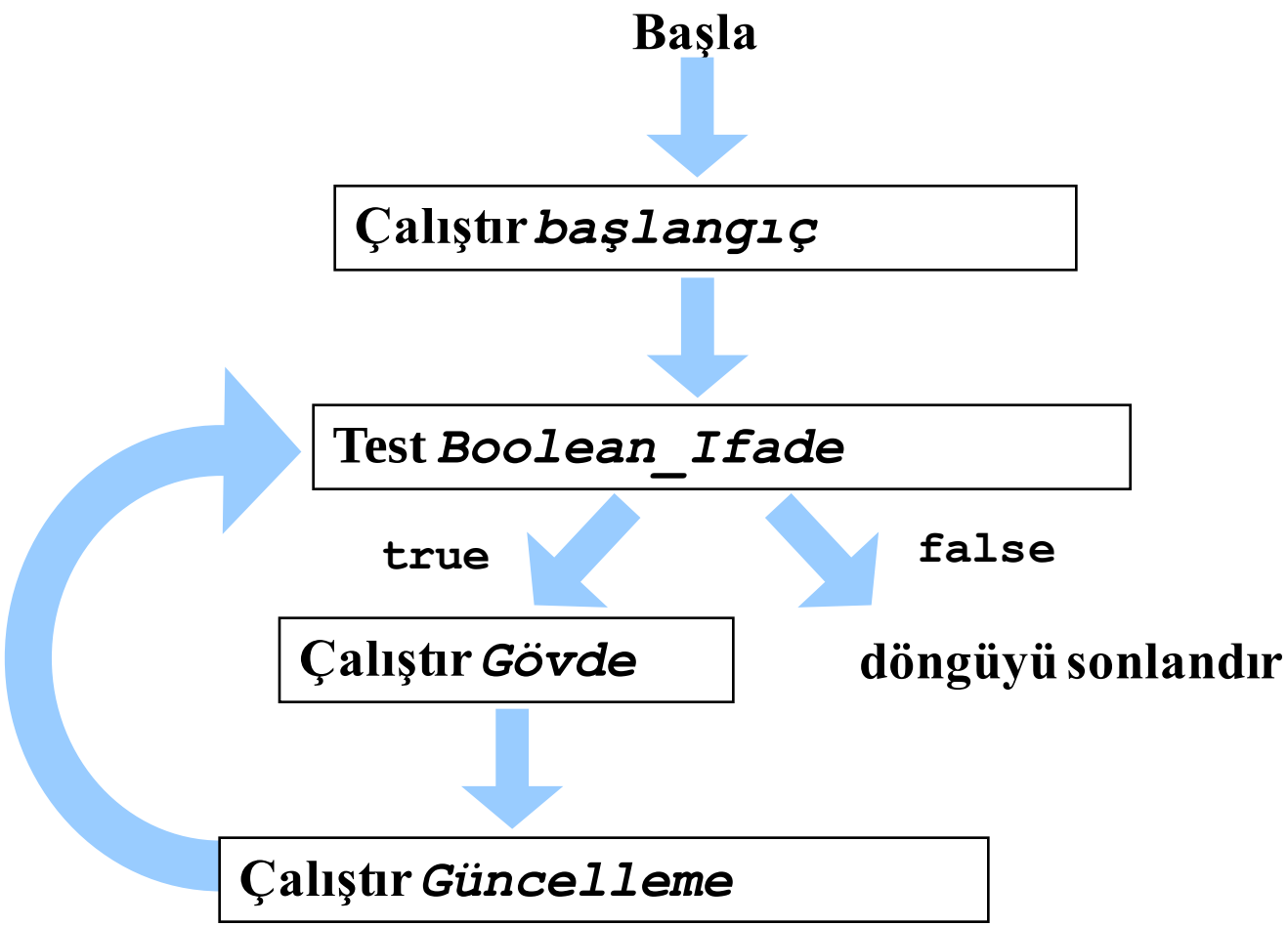
`for` döngüsü

- Sayan döngüler için iyi bir tercih
- Başlangıç kodu, döngü testi ve döngü sayacı döngünün parçasıdır
- Syntax:

```
for(başlangıç; Boolean_Ifade; güncelleme)
    döngü gövdesi;
```

`for` Döngüsü

```
for(başlangıç; Boolean_Ifade; güncelleme)
  döngü gövdesi;
```



`for` örnek

- 3den 1e kadar say

```
for(int sayac = 3; sayac >= 1; sayac--)
{
    System.out.print("T = " + sayac);
    System.out.println("ve sayiyor");
}
System.out.println("Son!");
```

Çıktı:

```
T = 3 ve sayiyor
T = 2 ve sayiyor
T = 1 ve sayiyor
Son!
```

`exit` Metodu

- Programın çalışmasına artık gerek olmadığı durumlarda, döngüden çıkmak veya programı durdurmak için *exit(n)* metodu kullanılır.
- *n* programın normal veya anormal yollardan sonlandırıldığını tanımlamak için kullanılır.
- *n* normal sonlandırmalar için genelde 0dır.

İç içe döngüler

- Bir döngü gövdesinde başka bir döngü de olmak üzere her çeşit komut olabilir.

```
for (satir = 0; satir < 4; satir++)
   for (yildiz = 0; yildiz < 5; yildiz ++)
      System.out.print('*');
   System.out.println();
```

Dış döngü gövdesi

İç döngü gövdesi

- Dış döngünün bir defa yürütülmesine karşılık, iç döngü 5 defa çalıştırılır.

Çıktı:

```
*****
*****
*****
*****
```