



Wise Quarter  
first class IT courses

# Java

## Team 105 Ders-01

Genel Bilgilendirme  
Programlamaya Giriş

+1 912 888 1630  
[www.wisequarter.com](http://www.wisequarter.com)

 /wisequarter

 /wisequarter



[www.wisequarter.com](http://www.wisequarter.com)



# Onemli Hatirlatmalar

Bugun  
IT Dunyasinda  
Ilk Gununuz

- 1- Ders Tam Vaktinde Baslar
- 2- Ders Oncesi Hazirlik Yapin  
(Google en yakin arkadasiniz olmalii)
- 3- Derste Aktif Olun  
(Anlamadiklarinizi Mutlaka Sorun)
- 4- Derste Kodlari Kendiniz Yazin  
(Yetistiremiyorsaniz Onceligi Anlamaya Verin)
- 5- Ders Sonrasi Tekrar Yapin  
(En Iyi Tekrar Dersteki Kodlari Kendinizin yazmasidir)
- 6- Basari = Egitim + Calismak
- 7- Grup Calismalari Yapin  
(Derste Anlatilanlara Benzer Sorular Uretin)

# Onemli Hatirlatmalar



## Kullanimi

- 1- Ders esnasinda anlatilan konu disinda paylasim yapmayin.
- 2- Ders sirasinda ileri konulardan soru sormayin.
- 3- Dersle sirasinda veya sonrasinda, slack yazismalarinizda nezaket kurallarina uyun.
- 4- Kod paylasirken snippet kullanin
- 5- Kodla ilgili sorunlarinizda sorularinizi screenshot ile yollayin
- 6- Dersi takip edin ve daha once sorulmus konulari tekrar sormamaya ozen gösterin
- 7- Kurulumla ilgili problemleri teknik destege, dersle ilgili sorulari instructor'a sorun.



# Onemli Hatirlatmalar



- 1- Ders tam zamaninda baslar
- 2- Ders basinda bir onceki gunun kisa bir tekrari yapilir
- 3- Her konu bittiginde, o konu ile ilgili bir test yapilir.



## Programlama Dili Nedir?

Programlama dili, yazılımcının bir algoritmayı ifade etmek amacıyla, bir bilgisayara ne yapmasını istediği anlatmasının tektipleştirilmiş yoludur.



Programlama dilleri, yazılımcının bilgisayara hangi veri üzerinde işlem yapacağını, verinin nasıl depolanıp iletileceğini, hangi koşullarda hangi işlemlerin yapılacağını tam olarak anlatmasını sağlar.

Programlama dilleri sayesinde bir bilgisayarın hangi durumda ne çeşit çıktı verebileceği kontrol edilebilir.

Kısacası programlama dilleri sayesinde bilgisayarlar ve insanlar verimli bir iletişim sağlayabilirler.

## Bilgisayar'in bizim istedigimiz seyi yapabilmesi icin



### 1- Bilgisayar'in anlayacagi dili bizim bilmemiz



### 2- Yazdigimiz kodların bilgisayar tarafından anlasildigini bilmemiz



### 3- Bilgisayarin bizim icin urettigi sonucları anlamlendirmemiz gereklidir.

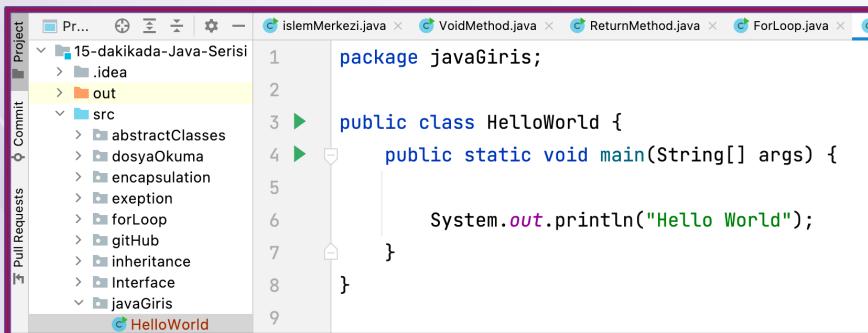
Kisaca ozetlersek, programlama dili bizimle bilgisayar arasindaki iletisimi saglayan dildir.

Peki..Bizimle bilgisayar arasindaki islemlerde patron kim ?

Kimin dedigi olur ?

Tabii ki bilgisayarlar bizim dedigimizi yaparlar

Ama bu istegimizi bilgisayarin anlayacagi ozelliklerde yazmamiz sartiyyla.



```
Pr... Project .idea out src abstractClasses dosyaOkuma encapsulation exception forLoop gitHub inheritance Interface javaGiris islemMerkezi.java VoidMethod.java ReturnMethod.java ForLoop.java
1 package javaGiris;
2
3 public class HelloWorld {
4     public static void main(String[] args) {
5         System.out.println("Hello World");
6     }
7 }
8
9
```

JDK Kodlari derler (Compile)

Ornegin;  
kodlarimizla, Hello World yazdirmak istiyorsak

```
110101010101010010100010101010
100101111000001000100101101010
101010010111000110
010001010010101110001000000101
0010101110001001010111
```



Binary kodlar

Islem

# Nicin Java ?

Java, 1995 Yılında ortaya çıkan high-level, Object Oriented bir program olarak ortaya çıkmıştır.

Java'yi ilk gunden itibaren populer programlama dilleri arasında one cikaran bazi ustunlukleri,

1- Öğrenmesi kolay

2- Dunyada en çok kullanılan programlama dili

3 milyar mobil cihazda Java kullanıyor.

USA'de şirket bilgisayarlarının %97'sinde, kişisel bilgisayarların %89'unda Java kullanılıyor

Linkedin, Uber, Netflix gibi pek çok popüler uygulama Java tabanlı çalışıyor.

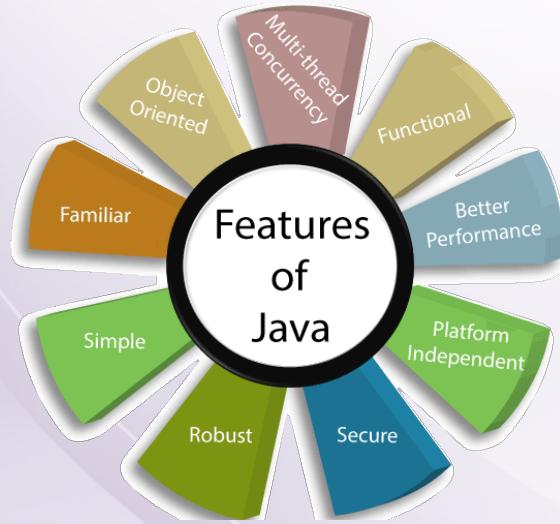
3- Güvenlidir

4- Platform independent'dir

5- Ücretsizdir.

Neden Java:

<https://youtu.be/RUYASEnT6pU>





# Nicin Java / OOP Concept



6- Java, Object Oriented Programming konseptinde calisir.

OOP Konsept, kompleks programlari yapmaya kucuk parcalardan baslayip, sonra bunlari birlestirerek istenen sonuca ulasmaya denir.



Java'da, once Object (Nesne) olusturmalıyız.

Her obje'nin 2 tur ozelligi olur.

- Feature (variable) – renk, pin sayisi vb...
- Functionality (method) – bizim girdigimiz degere gore degeri degisen ozellikler. Kanatli, tekerli vb..

# Object Nasıl Olusturulur

Objeleri olusturabilmek icin oncelikle kalip olusturmaliyiz.



Kalip varsa, bu kalıptan istedigimiz kadar obje uretebilir ve bu objeleri birlestirip istedigimiz uygulamayı elde edebiliriz.

Java'da obje olusturabilmemiz icin kullanmamız gereken kalip Class( Sinif )'dir.

Her bir obje bir Class'tan turetilmistir ve Class olmadan obje uretmek mumkun degildir.



# Class Hangi Bolumlerden Olusur ?

```
public class C2_MethodCreation2 {  
}  
} // Class sonu
```

Bir Class'da temel 3 bolum bulunur.

1- Class Declaration

2- Curly Braces : Suslu Parantez

3- Class Body : Suslu parantezler arasında kalan, kodlarimizi yazdigimiz bolum.



```
public class HelloWorld {  
  
    int ogrNo=1013;  
    String isim="Ali Can";  
    boolean ogrenciMi=true;  
    double notOrt=87.5;  
  
    public static void main(String[] args) {  
  
        double yazılıNotu=89;  
        double sozluNotu=92;  
  
    }  
  
    public void baskaMethod(){  
  
    }  
}
```

Bir Class Body'sinde variable ve method'lar bulunur.

1- Main Method

2- Variables

3- method'lar



# Main Method Nedir ?

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

Main Method, Java'nin calismaya basladigi giris noktasidir (Entry point)

Main method olusturulurken kullanilacak syntax (yazilacak keyword veya metin) sabittir, degistirilemez.

Parantez icine yazilan (**String[ ] args**) main method'un calismasi icin gerekli argumanların oldugu bir array'dir ve mutlaka yazilmalidir.

**NOT :** main method olmayan class'lar run edilemez (direk calistirilamaz).

```
3 ▶ public class HelloWorld {  
4  
5 ▶   public static void main(String[] k) {  
6  
7  
8  
9  ▶ }  
10 }  
11 }
```

```
3  
4  
5  
6  ▶   public void baskaMethod(){  
7  
8  
9  ▶ }  
10 }  
11 }
```

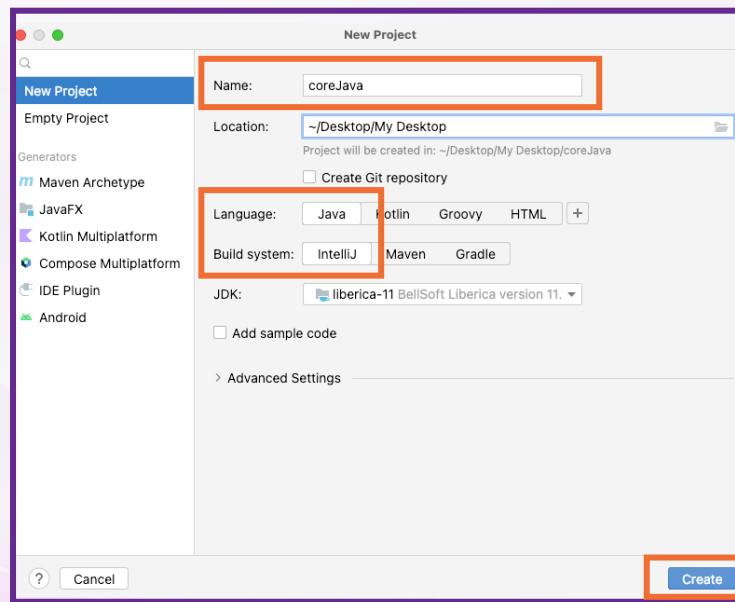
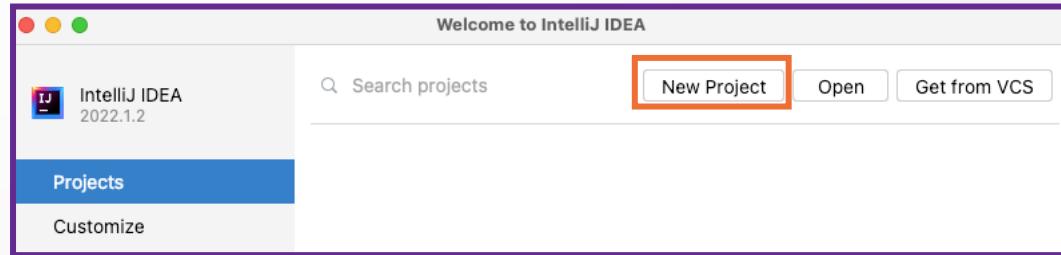
# IntelliJ'de Proje Olusturma

## IntelliJ Nedir?

Java gibi compiler programlar calismak icin ide ( Integrated Development Environment)'ye ihtiyac duyarlar.

Bircok ide olmakla birlikte, piyasada çok kullanildigi ve kod yazimini kolaylastirdigi icin biz intelliJ kullanacagiz.

IntelliJ'de Projects menusunde New Project'i secin,



Acilan menu'de

Name kismina projenin ismini yazin,

Language kisminda Java, Build system kisminda IntelliJ secili oldugunu kontrol edip,

Create butonuna basin.

# IntelliJ'de Proje Olusturma



## 2- Package(Class Dosyasi) olusturma

Acilan projede **src**'ye sag click yapip **New--Package**'i secip istedigimiz ismi yazin ve **finish**'e basin

## 3- Class olusturma

Acilan package ismine sag click yapip **New--Class**'i secip istedigimiz ismi yazin ve **finish**'e basin

## 4- Main method olusturma

Acilan Class icinde **main veya psvm** yazdigimizda, asagida cikan **main** yazisini sectiginizde IntelliJ bizim icin main method olusturacaktir.

## 5- Ilk kodumuzu yazdirma

Main method icerisinde **sout** yazip acilan menuden **system.out.println**'i secin.

```
5  public static void main(String[] k) {  
6      System.out.println("Hello World");  
7  }  
8 }
```

Parantez icine **"Hello World"** yazin

**Yesil run tusu**'na basip class'i calistirin

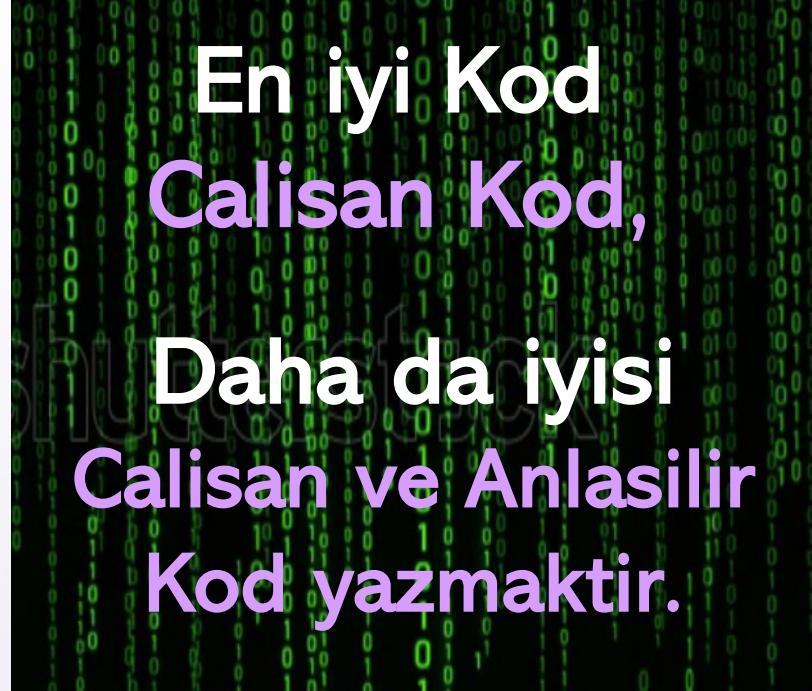
# Kod'a Comment Ekleme

Kod yazarken ilk hedef calisan bir kod yazmaktır.

Ama asil hedef calisan ve Anlasilabilir kod yazmaktır.

Kodlarimizi hem kendimiz hem de bizden sonra kodlari kullanacak kisilerin daha iyi anlayabilmesi icin, class'a aciklama cumleleri ekleyebiliriz .

```
5 ►   public static void main(String[] k) {  
6  
7     // Tek satiri comment yapmak icin
```



En iyi Kod  
Calisan Kod,  
  
Daha da iyisi  
Calisan ve Anlasilir  
Kod yazmaktir.

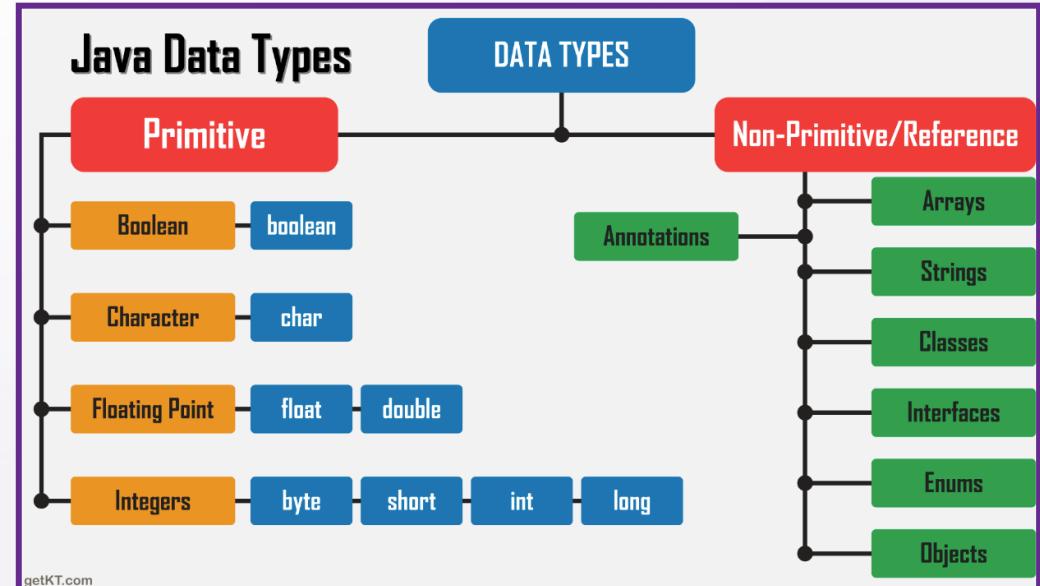
```
10 */  
11 Birden fazla  
12 satiri  
13 comment yapmak icin  
14 */
```

# Data Nedir ?

**Data** is a collection of facts, such as numbers, words, measurements, observations or just descriptions of things.

Data (**Veri**), sayilar, kelimeler, olcumler, gozlemler gibi bilgi iceren objelerin bir kolleksiyonudur.

Yazacagimiz her kod, yapacagimiz her program **data**'yi almak, **data**'yi islemek ve sonuc olarak bir **data** olusturmak icin kullanilir.



Yukarida da tanimlandigi gibi data'nin icerdigi bilgi çok farklı olabileceginden, tüm programlama dilleri farklı data turlerini kullanabilmek icin **kendi** kullanacakları **data turlerini** tanımlamışlardır.

Hangi programlama dilini kullanacaksak, oncelikle o dillerde kullanabilecegimiz data turlerini öğrenmeliyiz.



# Data Saklama(Store)

Her data hafizada(memory) bir yer kaplar.

Bir datanin hafizada saklanacagi en kucuk bolum bit'dir.

Her bir bit **1** veya **0** degerlerini icerir.

8 bit bir araya geldiginde bir **byte** olusur.

Her **byte**  $2^8 = 256$  farkli deger alabilir.

Sayı sistemimiz 10'luk sistem oldugu gibi hafiza da

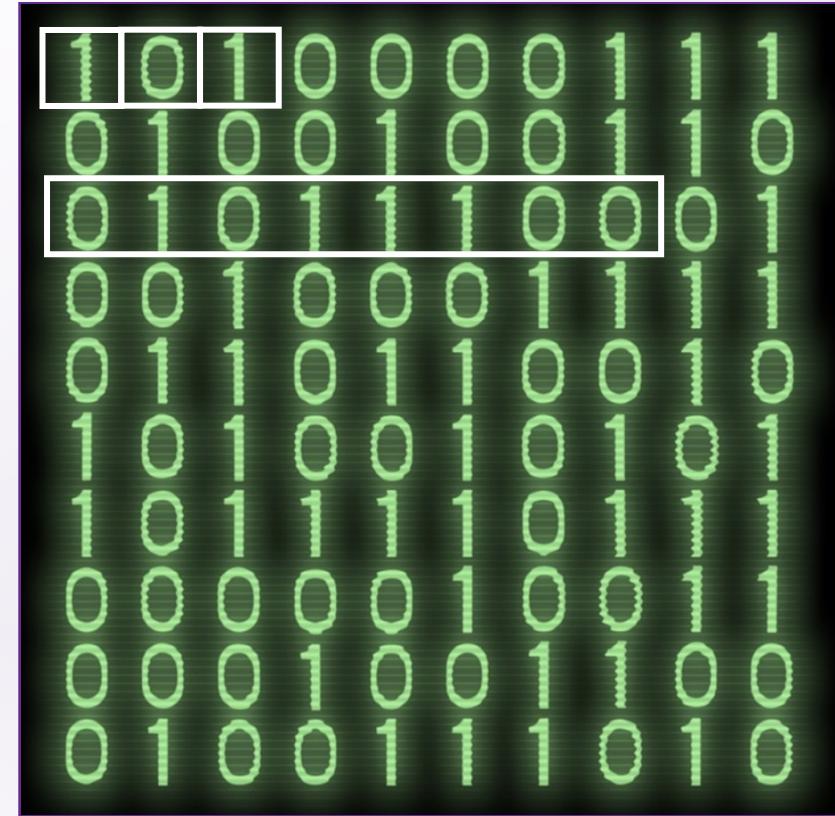
$2^{10} = 1024$ 'un katlari seklinde yapılandirilmistir.

1024 byte = 1 KB

1024 KB = 1 MB

1024 MB = 1 GB

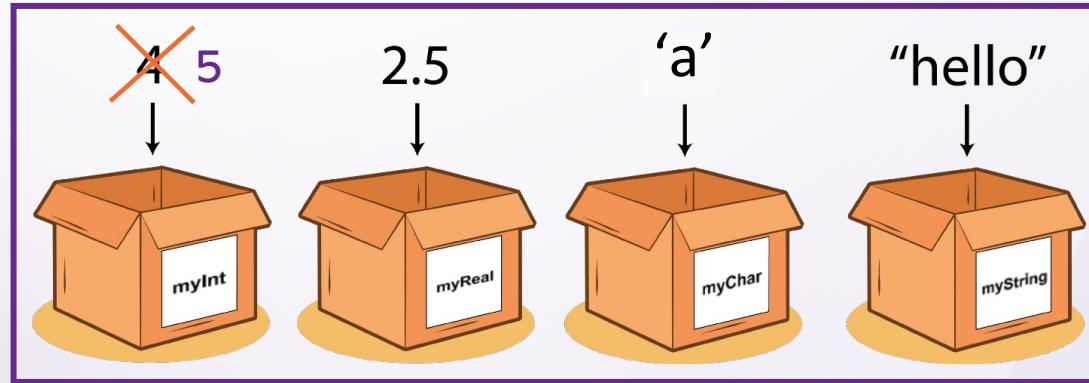
1024 GB = 1 TB ....



# Variables (Data Kullanma )

Java hafizadaki dataları **variable** veya **objeler** yardımıyla kullanır.

Bir datayı hafızada store etmek istedigimizde, Java hafızada o datayı tutabilecegi bir alan ayırır ve o alanı isimlendirir.



Biz ne zaman o data üzerinde degisiklik yapmak istesek, ismini söylememiz yeterli olur.

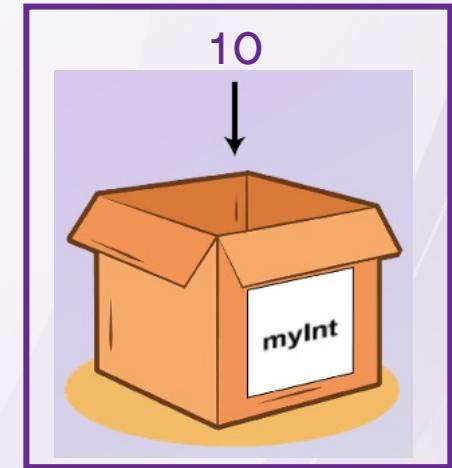
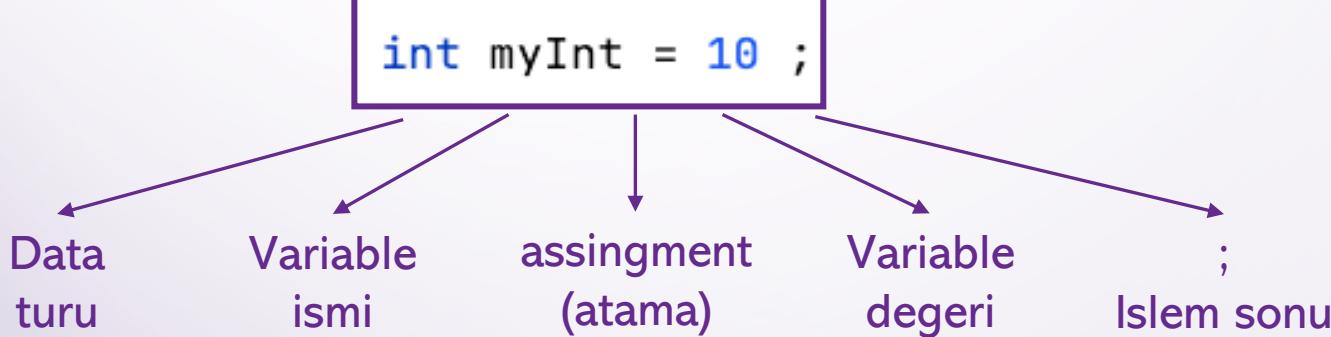
Ornegin myInt'i bir artır dedigimizde, Java myInt ismindeki variable'i bulur, icindeki değer olan 4'u bir artırıp 5 yapar.

Bu islemden sonra myInt variable'ının değeri 5 olacaktır.

Ozetle; biz bir değeri store etmek istedigimizde data turune uygun bir variable oluşturup ona bir isim veririz, ne zaman o datayı kullanmak istesek Java'ya variable ismini söylememiz yeterli olacaktır. myInt'i artır, myInt'i degistir, myInt'i sil vb...

# Variable Nasil Olusturulur ?

Variable olusturmak ve deger atamak icin Java'nin belirledigi syntax asagidaki gibidir.



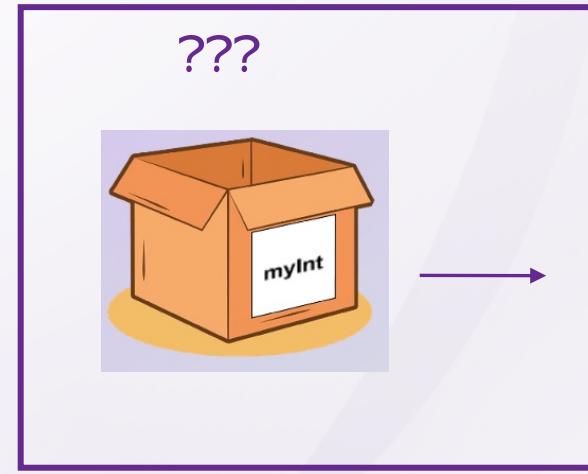
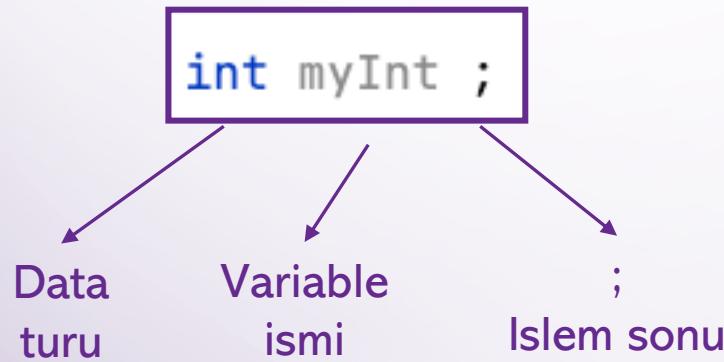
Bir variable olusturmak icin 2 islem vardir;

1- declaration (tanimlama) : esitligin sol tarafı

2- deger atama (assignment) : esitlik ve esitligin sol tarafı

# Variable Declaration

Java'nin datayı store edebilmesi için variable'a ihtiyacı vardır. Bir variable'in oluşturulması için de mutlaka declaration gereklidir.



Declaration için data turu ve variable ismi yeterlidir.

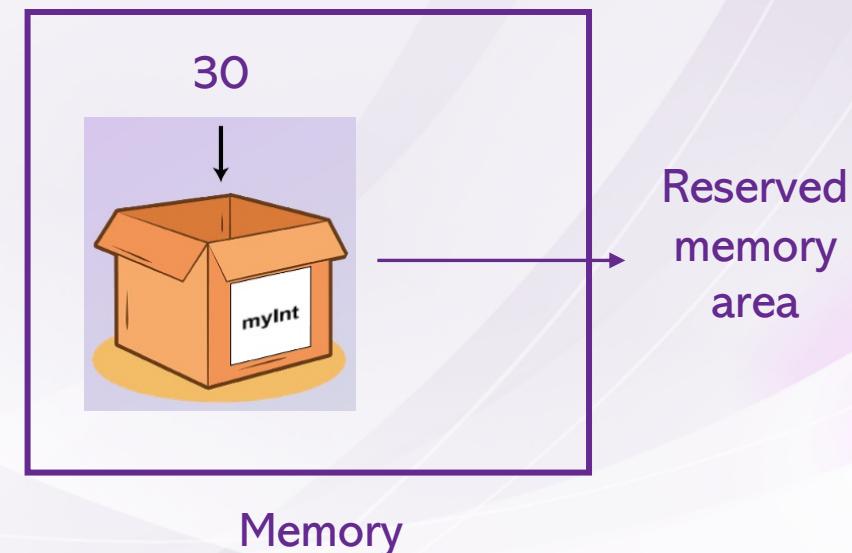
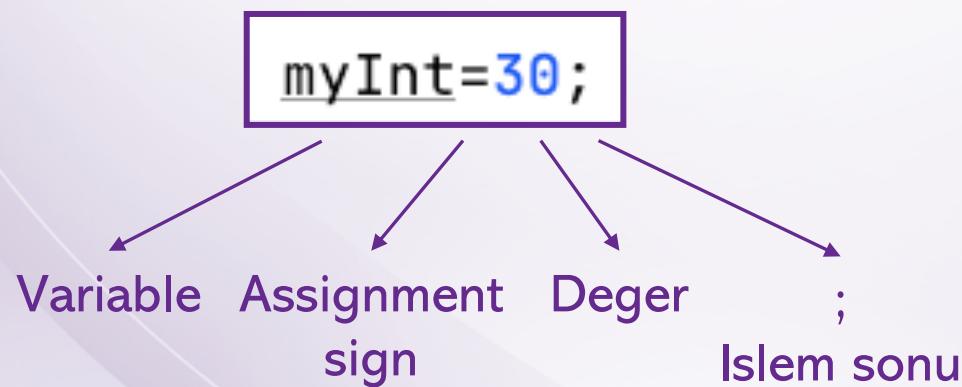
Java'da declaration ve assignment farklı satırlarda yapılabilir.

Ancak bir değer ataması olmadan variable'in kullanılması mümkün degildir.

# Variable Assignment

Deklare edilmiş bir variable'a değer atamaya assignment denir.

Declaration ve assignment farklı iki işlemidir. İkisi aynı satırda yapılabilcegi gibi, farklı satırlarda da yapılabilir.





Wise Quarter  
first class IT courses

# Java

## Team 105 Ders-02

### Variables

### Data Türleri

### Kullanıcıdan Deger Alma

+1 912 888 1630

[www.wisequarter.com](http://www.wisequarter.com)



# Variable Assignment

Declaration ve assignment asagidaki sekillerde yapilabilir.

1- Declaration ve assignment ayni satirda yapilabilir

```
int not=80 ;
String isim="John Doe";
boolean ogrenciMi=true;
double notOrt=89.3;
```

2- Once declaration, sonra assignment yapilabilir

```
int not;
not= 90;
not= (not + 80)/2;
```

NOT : Declaration sadece 1 kere yapilir, assignment ise istendigi kadar yapilabilir.



# Variable Assignment

3- Ayni data turundeki birden fazla variable ayni satirda deklare edilip, sonra tek tek assignment yapilabilir

```
int not1,not2,ortNot;  
  
not1= 80;  
not2= 90;  
ortNot= (not1 + not2)/2;
```

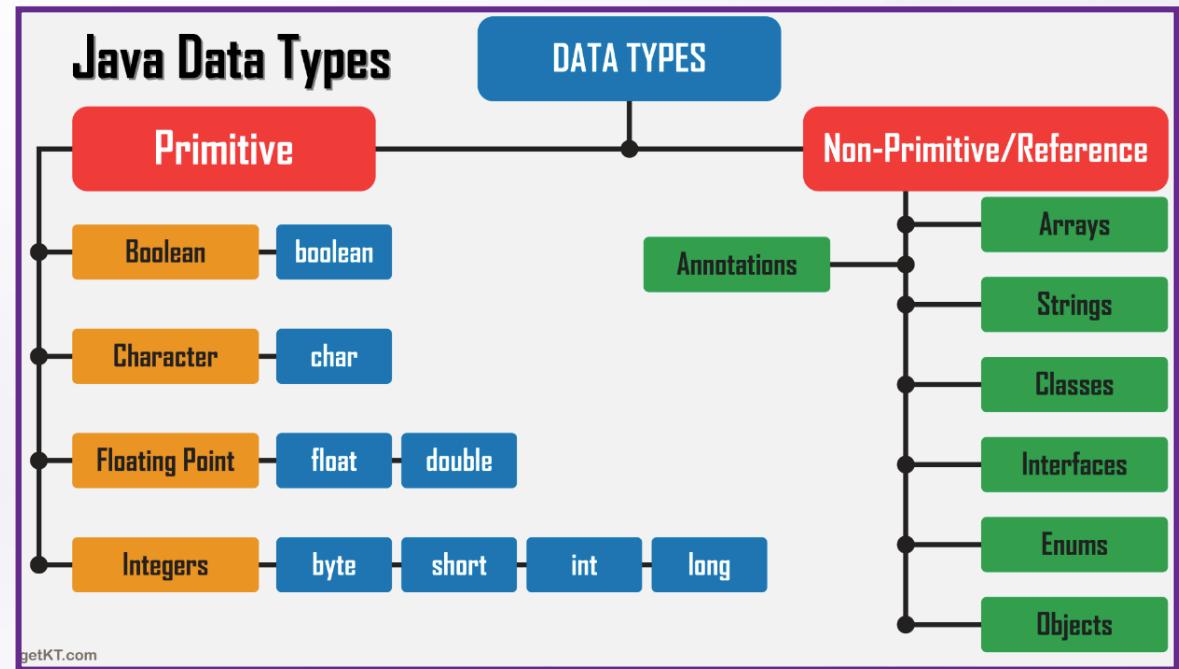
4- Ayni data turunde birden fazla variable tek declaration ile olusturulabilir.

```
int not1=80,not2=90,ortNot= (not1 + not2)/2;
```

# Data Turleri

Java'da temelde iki data turu kullanılır.

- 1- Primitive Data Turleri
- 2- Non-Primitive Data Turleri



Biz baslangicta primitive data turleri ve String kullanarak ilerleyecegiz, daha sonra diger non-primitive data turlerini ogrenecegiz.

# Primitive Data Turleri

Java'da 8 primitive data turu kullanılır.

Primitive data turleri sadece değer store edebilirler ve hafızada kapladıkları alan her data turu için sabittir.

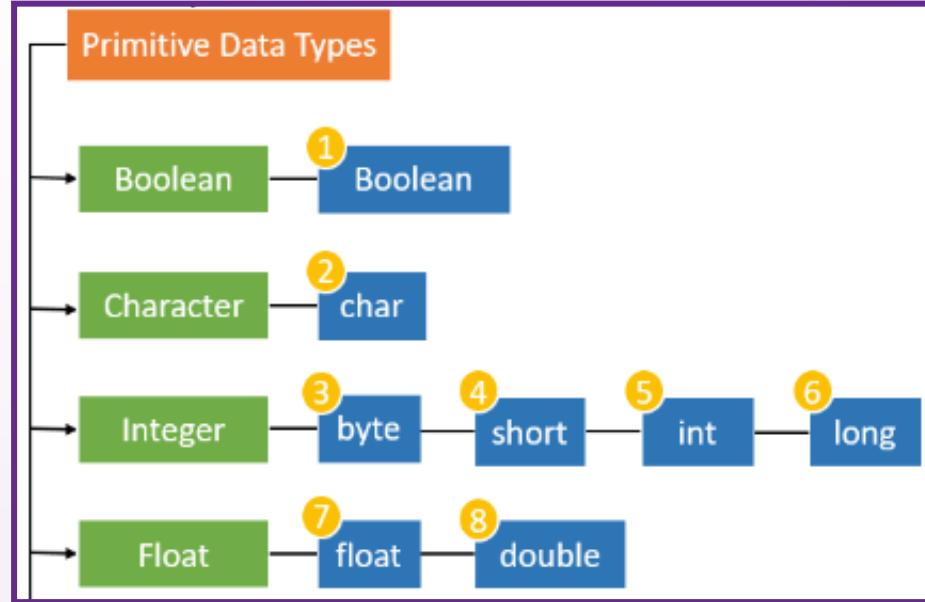
- 1- Boolean : Mantıksal data sonuçlarını store etmek için kullanılır.

boolean data turundeki bir variable sadece 2 değer barındırabilir, true / false

Bilgisayar true için 1, false için 0 değerini tutar, dolayısıyla hafızada sadece 1 bit yer kaplar

- 2- char : Tek bir karakter barındırır. İçerisinde harf, sayı veya özel karakter olabilir. char data turunun en belirgin farklılığı 'c' (tek tırnak) kullanmasıdır. Bir data " kullanırsa char olmalıdır.

hafızada 16 bit yer kaplar



```
char harf='A',sayi= '4',karakter='#';
```

# Primitive Data Turleri

Tam sayı barındıran primitive data turleri

Data Turu	Hafiza Boyut	minimum deger	maximim deger
3- byte	8 bit	$-2^7 = -128$	$2^7 - 1 = 127$
4- short	16 bit	$-2^{15} = -32.768$	$2^{15} - 1 = 32.767$
5- int	32 bit	$-2^{31} = -2.147.483.648$	$2^{31} - 1 = 2.147.483.647$
6- long	64 bit	$-2^{63} = -9.223.372.036.854.755.808$	$2^{63} - 1 = 9.223.372.036.854.755.807$

Bir variable için hangi data turunu kullanacağımız, uygulamamızın hafiza kullanımı için önemlidir.

Örneğin, üniversitedeki öğrencilerin yaslarını barındıran bir variable için byte yeterlidir.

Yasları short, int veya long olarak da store edebiliriz ancak bu durumda kullanılabilecek hafıza miktarı katlanarak artacaktır.

# Primitive Data Turleri

Ondalikli sayi barindiran primitive data turleri

Data Turu	Hafiza Boyut	min-max deger	Ondalikli basamak sayisi
7- float	32 bit	$\pm 3.40282347E+38F$	6-7 basamak
8- double	64 bit	$\pm 1.79769313486231570E+308$	15-16 basamak

Ondalikli sayilar icin hafiza durumu ve ondalik kismin uzunluguna gore data turu secilebilir.

Deger atamasi yaptigimizda Java'nin float ile double'i ayirt edebilmesi icin, float sayilarin yaninda **f** veya **F** kullanmamiz gerekir.

```
float a=20f;
float b=6f;
System.out.println(a/b); // 3.3333333
```

```
double c=20;
double d=6;
System.out.println(c/d); // 3.333333333333335
```

# Non-Primitive Data Türleri

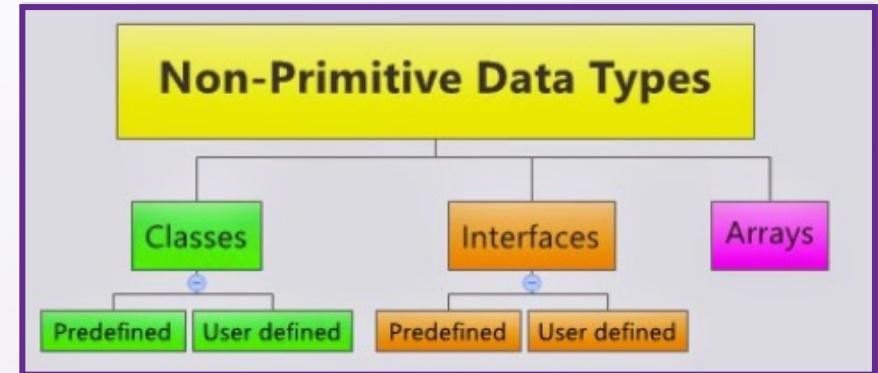
Primitive data türleri Java tarafından oluşturulmuştur ve biz yeni bir PRIMITIVE DATA TURU oluşturamayız.

Non-Primitive data türleri ise Java tarafından sınırlanılmamışlardır. Java da bazi non-primitive data türleri oluşturmuştur, biz de yeni non-primitive data türleri oluşturabiliriz.

Non-Primitive data türlerinin geneli için **object** tabiri kullanılabilir, cunku tüm non-primitive data türlerinin kendilerine ait bir class tarafından oluşturulan objelerdir.

Class'lar OOP konsept çerçevesinde bizim obje oluşturduğumuz, kalıplar olduğundan, non-primitive data türleri bu class'lardan oluşturulan objelerdir.

Class'lar method'lar da içerdiginden, non-primitive data türleri, oluşturuldukları class'lardaki method'lari kullanabilirler.





# Non-Primitive Data Turleri

Non-Primitive data turlerinden, simdilik String'i kullanacagiz. Ilterleyen derslerde Java'nin olusturdugu tum non-primitive data turlerini gorecegiz.

```
String isim= "John Doe";
```

String'i variable'lara deger atamak icin **""** kullaniriz.

```
String isim= "John Doe";
```

isim.|

- m **split**(String regex)
- m **toLowerCase**()
- m **substring**(int beginIndex)
- m **getBytes**(StandardCharsets.UTF\_8)
- m **getBytes**(String charsetName)
- m **getBytes**(Charset charset)
- m **getBytes**()
- m **getBytes**(int srcBegin, int srcEnd,
- m **toLowerCase**(Locale.ROOT)
- m **toLowerCase**(Locale locale)
- m **toUpperCase**(Locale.ROOT)
- m **toUpperCase**(Locale locale)
- m **toUpperCase**()
- m **split**(String regex, int limit)
- m **substring**(int beginIndex, int endIndex)
- m **charAt**(int index)

Non-primitive data turunde olusturulmus herhangi bir variable'in **ismini** yazip **.'**ya basarsaniz, o variable ile kullanabilecegimiz method'lari gorebilirsiniz.

# Primitives & Non-Primitives

İki data turu arasında 5 temel farklilik sayabiliriz.

Primitives	Non-Primitives
Tüm Java tarafından oluşturulmuştur.	Java tarafından oluşturulanlar olduğu gibi biz de oluşturabiliriz
Sadece değer icerirler, variable ile kullanılabilecek hazır method'ları yoktur.	İçerikleri değerin yanında oluşturdukları class'dan gelen hazır method'ları barındırırlar.
Bir değer atamadan oluşturulabilir ama kullanılmak için mutlaka değer atanmalıdır.	Değer atanmadan <b>null</b> olarak işaretlenebilirler.
Data turu isimleri <b>kucuk</b> harfle baslar (int, char vb.)	Data turu isimleri <b>buyuk</b> harfle baslar. (String vb..)
Primitive data turundeki variable'ların hafızada kapladıkları alan sabittir. Değeri küçük de olsa, büyük de olsa hafızada belirlenen miktarda alan ayrıılır.	Hafızada kapladıkları alan sabit degildir. Data turu ve içerdigi datanın büyüklüğüne göre hafızada yer kaplarlar. (bir kelime veya binlerce kelime içeren String'lerin boyutları farklı olacaktır)

# Naming Convention (Isim Verme Kurallari)

Variable'lara isim verirken istedigimiz ismi secebiliriz ancak asagidaki kurallara uyulmasi gereklidir.

1- Variable isimleri buyuk-kucuk harf duyarlidir (**case sensitive**).

Not, NOT, not, nOT ... birbirinden farklidir.

2- Variable isimlerinde **harf**, **rakam**, **\_** ve **\$** kullanilabilir,  
bosluk veya \* gibi ozel karakterler kullanilamaz.

3- Variable isimleri harf ile baslamlidir, rakam ile baslayamaz.  
**\_** ve **\$** ile baslayabilir ama kullanilmasi tavsiye edilmez.

4- Variable isimleri olarak keyword (Java'da tanimli anahtar kelimeler) kullanilamaz.  
for, int, short, class vb. variable ismi olamaz.

5- Variable isimleri kucuk harfle baslar, birden fazla kelime iceriyorsa **camelCase** kullanilir, yani sonraki her kelimenin ilk harfi **buyuk harf**, diger harfleri **kucuk harf** yapilir.



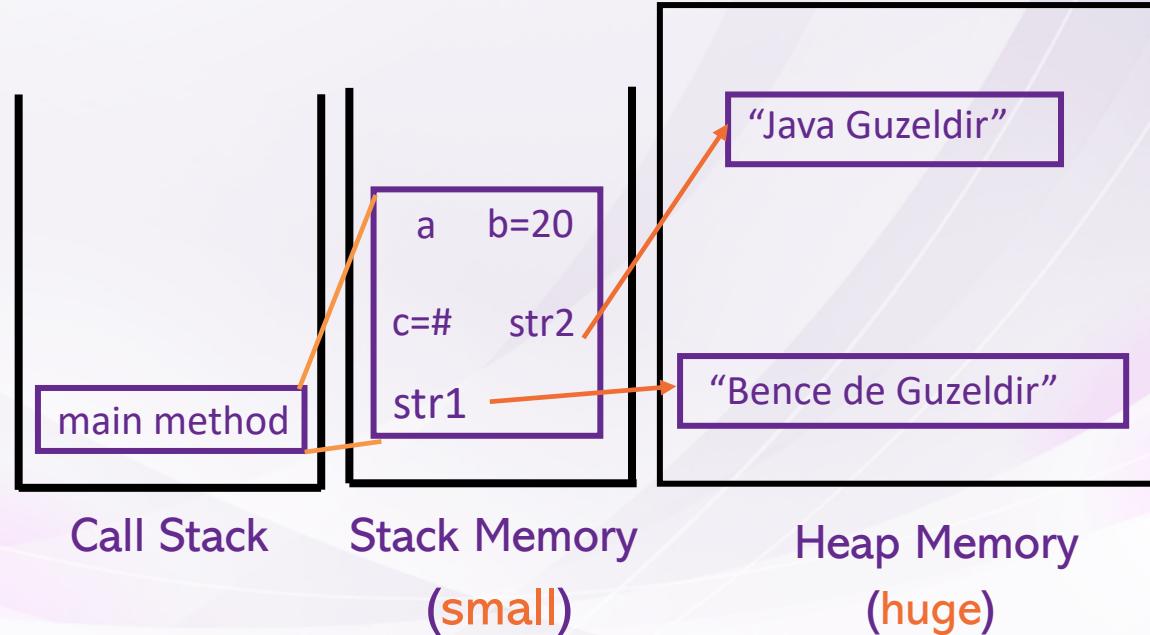
# Hafiza(Memory) Kullanimi

Java'da her method calistiginda, o methoda ait bir stack ve ona ait bir stack memory alani olusturulur.

Ayrica her method'un kullandigi heap memory vardir.

Stack memory'de primitive data turundeki variable'lar ve degerleri ile non-primitive data turundeki variable'larin referanslari olurken, heap memory'de non-primitive variable'larin degerleri store edilir.

```
public static void main(String[] args) {  
  
    int a;  
  
    int b=20;  
  
    char c= '#';  
  
    String str1;  
  
    String str2="Java Guzeldir";  
  
    str1="Bence de Guzeldir";  
  
}
```





# Kullanicidan Deger Alma (Scanner)

Kodlarimizi yazdigimiz IntelliJ veya benzeri ide'lere disardan bilgi almak icin Java Util kutuphanesinden Scanner Class'ina ihtiyacimiz vardir.

## 1. Adim

Scanner Class'inda var olan hazir method'lari kullanabilmek icin Scanner class'indan bir obje olusturmaliyiz.

```
Scanner scan= new Scanner(System.in);
```

## 2. Adim

Scanner calisinda kullanicidan bir bilgi bekleyecektir, kullanicinin kendisinden ne istendigini bilmesi icin bir aciklama yazdiralim.

```
System.out.println("Lutfen bir tamsayi giriniz");
```





# Kullanicidan Deger Alma (Scanner)

## 3. Adim

Kullanicinin girdigi degeri alabilmesi icin Scanner class'indan uygun method'u kullanalim.

Kullanicidan tamsayi istersek **nextInt()** kullanmamiz uygun olacaktir.

**nextInt()** bize kullanicinin girdigi tamsayiyi getirecektir, bunu programimizda kullanabilmek icin uygun data turundeki bir variable'a atayalim.

```
int girilensayi=scan.nextInt();
```

scan.nextInt()	
m next()	String
m next(String pattern)	String
m next(Pattern pattern)	String
m nextBigDecimal()	BigDecimal
m nextBoolean()	boolean
m nextBigInteger()	BigInteger
m nextBigInteger(int radix)	BigInteger
m nextByte()	byte
m nextByte(int radix)	byte
m nextDouble()	double
m nextFloat()	float
m nextInt()	int
m nextInt(int radix)	int
m nextLine()	String
m nextLong()	long
m nextLong(int radix)	long

Bu adimlar sonucunda kullanicinin girdigi deger girilenSayi variable'i olarak kodumuza eklenmis oldu.

# Sorular (Variables ve Scanner)

**Soru 1-** Kullanicidan uc farkli data turunde deger alip, girilen degerleri aciklamalariyla yazdirin.

**Soru 2-** Kullanicidan bir double, bir de int sayi alip bunların toplamini ve carpimini yazdirin.

**Soru 3-** Kullanicidan ismini, soyismini ve yasini alip, asagidaki formmatta yazdirin.

Isminiz : John

Soyisminiz : Doe

Yasiniz : 44

Kaydiniz basariyla tamamlanmistir.

**Soru 4-** Kullanicidan bir dikdortgenin 2 kenar uzunlugunu alip, dikdortgenin alanini yazdirin.

**Soru 5-** Kullanicidan ismini, soyismini ve yasini alip asagidaki formmatta yazdirin.

girilen bilgiler : J Doe, 44

**Soru 6-** Kullanicidan bir cemberin yaricapini alip, cevresini ve alanini yazdirin.

**Soru 7 (Interview)-** Kullanicidan iki sayı alip ikisinin degerlerini degistirin(swap).

**Soru 8 (Interview)-** Kullanicidan iki sayı alip, ucuncu bir degisken kullanmadan ikisinin degerlerini degistirin(swap).



Wise Quarter  
first class IT courses

# Java

## Team 105 Ders-03

### Data Casting

### Wrapper Classes

### Matematiksel İşlemler

+1 912 888 1630

[www.wisequarter.com](http://www.wisequarter.com)



The future at your fingertips

# Data Casting (Data Turunu Degistirme)



Java'da bir data turundeki datayı başka bir data turune cevirmeye **data casting** denir. Ancak her data turu birbirine cevrilemez.

Benzer özelliklerdeki data turundekidataları birbirine kolayca cevirebilirken, bazi casting işlemleri için ekstra kod yazmamız gereklidir, bazi casting işlemleri ise imkansızdır.

```
int sayı= "John Doe";  
String str= false;
```

```
String isim="John Doe";  
int sayı= isim;  
  
boolean dogruMu=false;  
String str= dogruMu;
```

```
double dbl=23.4;  
int sayı= dbl;  
  
int in= 12;  
double db= in;
```

Java'da bir kodun altı kırmızı çizili oluyorsa, orada java'nın çözemediği bir sorun vardır ve siz o sorunu çözmedikçe Java çalışmazacaktır. (Sadece o class değil diğer class'lar da çalışmaz)

Java'da hem primitive, hem de non-primitive data turleri için data casting yapmak mümkün ancak biz simdilik primitive data turleri için data casting konusunu irdeleyelim.

# Implicit Data Casting (Auto-Widening)

Daha kucuk kapsamlı bir data turundeki degeri, daha genis kapsamlı data turundeki variable'a atama yapmak istedigimizde, Java bu islemi otomatik olarak yapacaktir

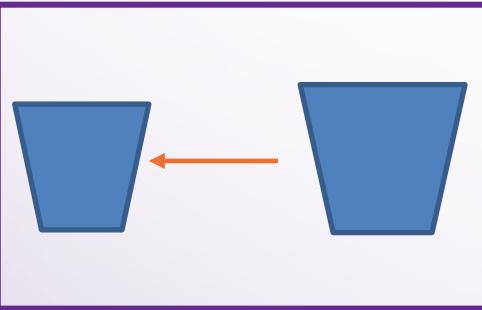


```
byte a = 12;  
  
int b = a;  
  
double c = b;
```



# Explicit Data Casting

Daha genis kapasiteye sahip bir data turundeki bir degeri, daha dar kapsamli bir variable'a atamak istedigimizde, Java bunu otomatik olarak yapmayacaktir.



```
int a = 12;  
int c = 567;  
  
byte b = a;  
byte d = c;
```

```
int a = 12;  
int c = 567;  
  
byte b = (byte) a; // 12  
byte d = (byte) c; // 55
```

Atanan deger'in data turu genis kapsamli oldugundan deger dar kapsamli variable'in sinirlari icinde olabilecegi gibi, sinirlarindan buyuk de olabilir.

Bu durumda Java, data kaybi veya degisikligi ihtimalinin farkinda oldugumuzun bilmek ister.

Sorumluluğu almak icin cast etmek istedigimiz degerin onune (cast etmek istedigimiz data turunu) yazarsak, java bu casting'i data turu sinirlarina gore yapar.



# Char Data Turu ve ASCII Table

Char data turu sadece 1 karakter icerir, ancak degerlerin ascii degerlerini tuttugundan, matematiksel islemlerde ascii kodlarina gore islemlere dahil olur.

ASCII control characters		ASCII printable characters								Extended ASCII characters							
00	NULL (Null character)	32	space	64	@	96	`	128	Ç	160	á	192	ł	224	ó		
01	SOH (Start of Header)	33	!	65	A	97	a	129	Ü	161	í	193	ł	225	ó		
02	STX (Start of Text)	34	"	66	B	98	b	130	é	162	ó	194	ł	226	ó		
03	ETX (End of Text)	35	#	67	C	99	c	131	â	163	ú	195	ł	227	ó		
04	EOT (End of Trans.)	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	ö		
05	ENQ (Enquiry)	37	%	69	E	101	e	133	à	165	Ñ	197	+	229	ö		
06	ACK (Acknowledgement)	38	&	70	F	102	f	134	â	166	º	198	å	230	µ		
07	BEL (Bell)	39	'	71	G	103	g	135	ç	167	º	199	À	231	þ		
08	BS (Backspace)	40	(	72	H	104	h	136	ê	168	¸	200	Ł	232	Þ		
09	HT (Horizontal Tab)	41	)	73	I	105	i	137	ë	169	®	201	Ł	233	Ú		
10	LF (Line feed)	42	*	74	J	106	j	138	è	170	¬	202	Ł	234	Ó		
11	VT (Vertical Tab)	43	+	75	K	107	k	139	í	171	½	203	Ł	235	Ù		
12	FF (Form feed)	44	,	76	L	108	l	140	î	172	¼	204	Ł	236	Ý		
13	CR (Carriage return)	45	-	77	M	109	m	141	í	173	ı	205	=	237	Ý		
14	SO (Shift Out)	46	.	78	N	110	n	142	Ã	174	«	206	‡	238	—		
15	SI (Shift In)	47	/	79	O	111	o	143	Ã	175	»	207	¤	239	·		
16	DLE (Data link escape)	48	0	80	P	112	p	144	É	176	„	208	ð	240	≡		
17	DC1 (Device control 1)	49	1	81	Q	113	q	145	æ	177	„	209	ð	241	±		
18	DC2 (Device control 2)	50	2	82	R	114	r	146	Æ	178	„	210	È	242	–		
19	DC3 (Device control 3)	51	3	83	S	115	s	147	ô	179	—	211	È	243	¾		
20	DC4 (Device control 4)	52	4	84	T	116	t	148	ö	180	—	212	È	244	¶		
21	NAK (Negative acknowl.)	53	5	85	U	117	u	149	ò	181	À	213	ı	245	§		
22	SYN (Synchronous idle)	54	6	86	V	118	v	150	û	182	Ã	214	ı	246	+		
23	ETB (End of trans. block)	55	7	87	W	119	w	151	ú	183	Ã	215	ı	247	·		
24	CAN (Cancel)	56	8	88	X	120	x	152	ÿ	184	©	216	ı	248	°		
25	EM (End of medium)	57	9	89	Y	121	y	153	Ó	185	—	217	ı	249	..		
26	SUB (Substitute)	58	:	90	Z	122	z	154	Ü	186	—	218	—	250	·		
27	ESC (Escape)	59	;	91	[	123	{	155	ø	187	—	219	—	251	ı		
28	FS (File separator)	60	<	92	\	124		156	£	188	—	220	—	252	³		
29	GS (Group separator)	61	=	93	]	125	}	157	Ø	189	¢	221	—	253	²		
30	RS (Record separator)	62	>	94	^	126	~	158	×	190	¥	222	—	254	■		
31	US (Unit separator)	63	?	95	—			159	f	191	—	223	—	255	nbsp		
127	DEL (Delete)																

```
char harf= 'a';
int sayi= 100;

System.out.println( harf + sayi); // 197
System.out.println( harf+1); // 98

char yениharf=(char)(harf+1);
System.out.println(yениharf); // b
```

# Wrapper Classes

Java primitive data turleri, kod yazarken mutlaka kullanacagimiz data turleridir. Ancak primitive data turleri sadece deger tasiyabilirler, **class olmadiklari icin** hazir method'lara sahip degillerdir.

Wrapper class'lar primitive data turlerini iceren **class'lardir**. Bu class'lardan olusturulan objeler primitive data turleri ile kullanabilirler.

```
int sayi=10;  
Integer sayiW= 20;  
  
sayiW=sayi;  
sayi= sayiW+5;
```

Wrapper class'lardan objelere primitive data turundeki degerler atanabilir. Ayrca bu class'lar bir çok faydalı method bulundururlar.

Integer.

m <b>max</b> (int a, int b)	int
f <b>MAX_VALUE</b> ( = 0x7fffffff)	int
m <b>bitCount</b> (int i)	int
m <b>getInteger</b> (String nm)	Integer
m <b>getInteger</b> (String nm, int val)	Integer
m <b>compare</b> (int x, int y)	int
m <b>compareUnsigned</b> (int x, int y)	int
m <b>decode</b> (String nm)	Integer
m <b>divideUnsigned</b> (int dividend, int divisor)	int
m <b>getInteger</b> (String nm, Integer val)	Integer



# Wrapper Classes

Wrapper class'lar casting, max-min degerler, karsilastirma gibi bircok hazır method'lara sahiptirler.

```
int sayi=10;
Integer sayiW= 20;
System.out.println(Integer.MAX_VALUE); // 2147483647
System.out.println(Integer.max( a: 34, b: 465)); // 465

boolean kontrol=true;
Boolean kont=false;
String knt="false";
boolean sonuc = Boolean.valueOf(knt);

char chr='*';
Character ch='p';
char chr2=101;
System.out.println(Character.valueOf(chr2)); // e
System.out.println(Character.isDigit( ch: '5')); // true
System.out.println(Character.isAlphabetic( codePoint: '9')); // false
System.out.println(Character.isAlphabetic( codePoint: 'a')); //true
```



Wise Quarter  
first class IT courses

# Java

## Team 105 Ders-04

Matematiksel İşlemler  
Increment - Decrement

+1 912 888 1630  
[www.wisequarter.com](http://www.wisequarter.com)

  
[/wisequarter](#)

 /wisequarter



[www.wisequarter.com](http://www.wisequarter.com)



# Onceki Dersten Hatirladiklarimiz

- 1- Data Casting : Java'da uygun data turleri arasında gecis mumkundur. Hem primitive data turleri icin hem de non primitive data turleri icin data casting mumkundur.
- 2- Her data turu birbirine cast edilemez. Ornegin boolean'ı String'e veya char'ı boolean'a cevirememeyiz. Ancak sayı degeri iceren byte, short, int, long, float ve double gibi benzer data turleri birbirine cevrilebilir.
- 3- Eger daha dar kapsamlı bir data turundeki deger, geniş kapsamlı data turundeki variable'a atanirsa java bunu otomatik olarak yapar. (Auto widening)
- 4- Eger geniş data turundeki bir deger, dar data turundeki bir variable'a atanirsa, java bu casting'ı otomatik yapmaz, sorumluluğunu bizim almamizi ister. Sorumluluğu almak icin degerin onune ( donusturulecek data turu) yazilir. Ancak explicit narrowing yapildiginda deger kayiplari veya deger baskalasimi olabilir.
- 5- char data turu matematiksel islemlerde ASCII tablosundaki degeri ile isleme girer. Eger sayisal bir ifadenin char olarak yazilmasini istersek onune (char) yazarak explicit data casting yapariz
- 5- Wrapper Class : Non-primitive data turlerinde oldugu gibi primitive data turlerinin de hazir method'lari olmasi icin Java Wrapper class'lari olusturmustur. Wrapper class'lar primitive'ler ile ayni degerleri kullanir ama hazir method'lari da vardir.

# Sorular (Data Casting)

**Soru 1-** Int olarak verilen 3 degerin ortalamasini double olarak yazdiran bir kod yazin

**Soru 2-** Kullanicidan bir harf alin ve alfabebe o harften sonraki 3 harfi yazdirin.

**Soru 3-** Kullanicidan bir sayi alin, kullanici kac degerini girerse girsin, o sayiyi -128 ile 127 arasindaki bir sayiya donusturup yazdirin.

**Soru 4-** Kullanicidan iki double sayi alin, ilk sayiyi ikinci sayiya bolun ve bolum isleminin sonucununun tamsayi kismini yazdirin.

**Soru 5-** Kullanicidan bir double, bir tamsayi alin, double sayiyi ikinci sayiya bolun ve bolum isleminin sonucununun tamsayi kismini yazdirin.

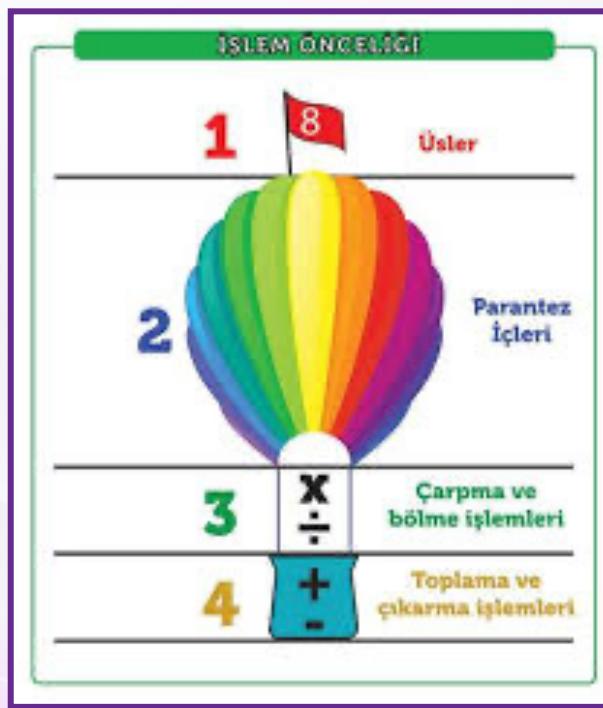


# Java'da Matematiksel İşlemler

Java Matematik işlemlerini sorunsuz yapar, Ancak biz işlemleri yazarken matematik kurallarına uygun olarak yazmazsa ummadığımız sonuçlarla karşılaşabiliriz.

$$24 + ( 5 * 2^3 - 3^3 )^2 - 13$$

$$14 - 5 * 2 + 3 * 4 - 8$$



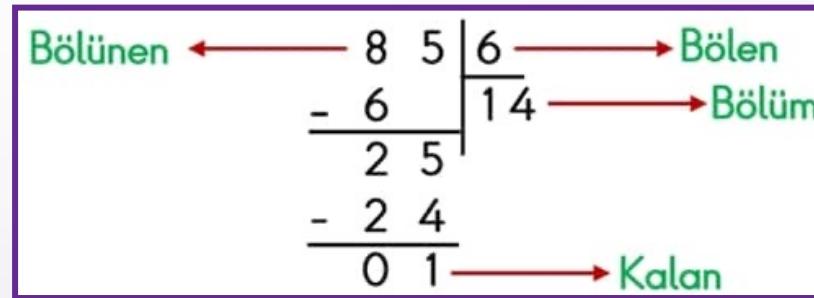
$$24 / 6 * 2 - 7 * 4 + 9$$

$$8 * 5 + 2 * ( 12 / 4 ) - 19$$



# Modulus (% Kalan Bulma)

Java'da Modulus işlemi, bir bolme işlemindeki kalan sayiyi bize verir.



Modulus işlemi sayesinde

- Cift sayilar ( sayı %2 )
- Bir sayinin birler basamagini bulma ( sayı %10 )
- Bir sayı (ornegin 5) ile tam bolunebilen sayıları bulma ( sayı % 5 )

mumkun olmaktadır.

# Modulus Soru

Soru 1- Kullanicidan 4 basamakli pozitif bir tamsayi alip rakamlar toplamini bulalim

**Ipucu 1:** Sayi % 10 => Bize son basamagi verir

$$1469 \% 10 = 9$$

**Ipucu 2:** Int Sayi /10 => Bize son basamak haric sayiyi verir

**int** sayi=1469;

**sayi = sayi / 10 =>**

**sayi'ya 46 degerini atar**



# Increment (Deger Artirma)

Toplama veya carpma yaparak bir variable'in degerini artirabiliriz.

Increment isleminin kalici olmasi icin 3 farkli sekilde assignment yapilabilir.

```
int sayi = 10 ;
```

```
sayi= sayi +3 ;
```



```
int sayi = 10 ;
```

```
sayi *= 3 ;
```



```
int sayi = 10 ;
```

```
sayi++ ;
```





# Decrement (Deger Azaltma)

Cikarma veya bolme yaparak bir variable'in degerini azaltabiliriz.

Decrement isleminin kalici olmasi icin 3 farkli sekilde assignment yapilabilir.

```
int sayi = 10 ;  
  
sayi = sayi - 2 ; → 8
```

```
int sayi = 10 ;  
  
sayi -= 4 ; → 6
```

```
int sayi = 10 ;  
  
sayi -- ; → 9
```



# Pre – Post Increment

```
int sayi = 10 ;  
  
sayi++ ;
```

```
int sayi = 10 ;  
  
sayi -- ;
```

Sayı değerini kalıcı olarak 1 artırırken **sayi++**, 1 azaltırırken **sayi--** kullanabileceğimizi gormustuk.

**++** ve **--** sayidan önce de kullanılabilir, sonuc yine aynı olacaktır.

```
int sayi = 10 ;  
  
++sayi ;
```

```
int sayi = 10 ;  
  
--sayi ;
```

Pre-Increment veya Pre-Decrement ile Post-Increment ve Post-Decrement arasındaki fark, bu işlem farklı bir işlem ile birlikte kullanılırsa ortaya çıkar.

# Pre – Post Increment

Post-Increment'te, increment diger islemden **sonra** yapilir .

```
int sayi = 10 ;  
  
System.out.println(sayi++); ; → Once sayiyi yazdirir (10) , sonra degeri artirir (11)  
  
System.out.println(sayi); → Ust satirda deger (11) oldu, (11) yazdirir.
```

Pre-Increment'te, increment diger islemden **once** yapilir .

```
int sayi = 10 ;  
  
System.out.println(++sayi); ; → Once sayiyi artirir (11) , sonra yazdirir (11)  
  
System.out.println(sayi); → Ust satirda deger (11) oldu, (11) yazdirir.
```



Wise Quarter  
first class IT courses

# Java Team 105 Ders-05

## Concatenation Operatorler

+1 912 888 1630  
[www.wisequarter.com](http://www.wisequarter.com)

/wisequarter

/wisequarter



[www.wisequarter.com](http://www.wisequarter.com)



# Onceki Dersten Hatirladiklarimiz

- 1- Java çok iyi derecede matematik bilir, işlemleri hatalı yapar. Ancak biz matematiksel işlemleri oncelik hatası yaparsak, Java olması gereken gibi işlem yapacağından bizim umdugumuzdan farklı bir sonuc çıkabilir.
- 2- Matematiksel işlemleri oncelik sırası üs-parantez, çarpma-bolme, toplama-cıkarma. Esit oncelikli işlemlerden önce soldaki yapılır
- 3- modulus % operatoru : bir bolme işleminde kalanı bize verir. Bu operatoru birler basamagini bulmada sayı%10 , bir sayının çift veya tek olmasını bulmakta sayı%2 veya herhangi bir sayıya tam bolunebildigini kontrol etmede sayı % istenenSayı kullanırız.
- 4- increment ve decrement : bir sayının değerini toplama veya çarpma yaparak artırabilir, çıkarma veya bolme yaparak azaltabiliriz. Bunları 3 farklı yazımla yapabiliriz. 1) sayı = sayı +10; , 2) sayı \*=3; , 3) sayı++, --sayı
- 5- Sadece 3.yazım şekli için pre veya post increment/decrement ayırımı söz konusudur.
  - bir işlemde (yazdırma veya atama) ++sayı varsa önce sayıyı artırıp sonra yazdırma veya atama işlemini yaparız
  - sayı++ varsa önce yazdırma veya atama işlemini yapıp sonra artırırız\*\*\* pre ve post increment farkı sadece yazıldığı satır için geçerlidir, bir alt satırda geçildiğinde ikisinde de sayının değeri 1 artmış veya azalmış olur.

# Pre – Post Increment

Post-Increment'te, increment diger islemden **sonra** yapilir .

```
int a = 10 ;  
int b= a++;  
  
System.out.println(a);  
System.out.println(b);
```

Once b'ye atama yapar ( $b=10$ ) ,  
sonra a degerini artirir ( $a=11$ )  
(11)  
(10)

Pre-Increment'te, increment diger islemden **once** yapilir .

```
int a = 10 ;  
int b= ++a;  
  
System.out.println(a);  
System.out.println(b);
```

Once artirma yapar ( $a=11$ ) ,  
sonra b'ye atama yapar( $b=11$ )  
(11)  
(11)



# Pre – Post Increment Soru

Soru : Asagidaki kod calistirilirsa konsolda gorunecek sonuclar neler olur?

```
int a=10;  
  
System.out.println("a'nin degeri : " + ++a);  
  
int b= a++;  
  
System.out.println("b'nin degeri : " + b);  
  
int c= b++ + a ;  
  
System.out.println("c'nin degeri : " + c);  
  
System.out.println("Son toplam : " +(a+b+c));
```



# Concatenation (String Birlestirme )

Bir String'i, baska bir String veya primitive deger ile + isareti kullanarak isleme sokarsak Java bu degiskenleri birlestirerek yeni bir String olusturur

```
String a = "Hello";
String b = "World";
System.out.println(a+b);
System.out.println(a+" "+b);
```

HelloWorld  
Hello World

**Not :** Eger matematiksel bir islemin icinde String kullanilirsa, matematikteki oncelikler dikkate alınarak islem yapılır. Sira String ile toplamaya geldiginde toplama yerine Concatenation uygulanır

```
String a = "Hello";
int b = 2;
int c = 3;
```

```
System.out.println(a+b+c);
```

Hello23

```
System.out.println(c+b+a);
```

5Hello

```
System.out.println(a+(b+c));
```

Hello5

```
System.out.println(a+b*c);
```

Hello6



# Concatenation (String Birlestirme )

Soru : Sadece verilen variable'lari kullanarak istenen String'leri elde edelim.

```
String s1= "Java";
String s2= " ";
String s3= "kolay";
String s4= "";

int a= 3;
int b= 4;
```

12 Java kolay  
7 Java kolay  
34Java kolay  
Java12kolay  
Java34kolay  
Java7kolay



# Relational (Karsilastirma) Operators

## 1- Esitlik (Cift esitlik isareti) : ==

Java'da, matematikten farkli olarak = isareti assignment(atama) islemi yapar, esitligi kontrol etmez

Java'da, iki degerin esit olup olmadigini kontrol etmek icin == kullanilir ve sonuc olarak bize true veya false doner.

```
int a=10;
int b=15;

System.out.println(a==b);

System.out.println(a==b-5);

boolean c;

System.out.println(c=15==b);

c= 15*a==10*b;

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



# Relational (Karsilastirma) Operators

## 2- Esit Degildir : !=

Java'da, herhangi bir mantiksal degerin basina konulan !=, o mantiksal ifadenin degerini tersine cevirir.

!true → false

!(5==5) → false

5 !=5 → false

```
int a=10;
int b=15;

System.out.println(a!=b);

System.out.println(a!=b-5);

boolean c;

System.out.println(c=15!=b);

c= 15*a !=10*b;

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



# Logical (Mantıksal) Operators

## 1- And (ve) Operatoru `&&` , `&`

Mantıktaki AND operatorunun Java'da 2 tane karsılığı vardır. İşlevleri aynı olmakla birlikte iç işleyisi ve hız açısından aralarında küçük bir fark vardır.

`&&` operatoru birlestirdiği 2 boolean ifadenin ikisi de true ise sonucu true yapar, bunun dışındaki tüm durumlarda sonucu false yapar. (`&&` operatoru mükemmeliyetcidir.)

```
int a=10;
int b=15;

System.out.println(a>b  && b>0);

System.out.println(a<=b-5 && a>b-8);

boolean c;

System.out.println(c=15>=b  && a<0);

c= a>=b && 3*a<4*b;

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

`&&` operatoru carpmaya benzetilir.  
`Sonucun 1 olması için`  
tüm çarpılan sayılar `1` olmalıdır.  
`1` tane bile sıfır olsa sonuc `0` olur.`out`

```
1 * 1 * 1 * 1 = 1
1 * 0 * 1 * 0 = 0
1 * 0 * 0 * 0 = 0
0 * 1 * 1 * 1 = 0
```

# Logical (Mantıksal) Operators

## && ile &' in farki

**&&** operatoru birbirine bagli mantıksal ifadeleri incelerken, **ilk false** degeri ile karsilastiginda, sonucun **false** olacagini algilar ve geriye kalan mantıksal ifadeleri incelemeden hemen sonucu **false** olarak atar.

**&** operatoru ise birbirine bagli mantıksal ifadeleri incelerken, herbir mantıksal ifadenin sonucuna gore karar vermez, islemin sonucuna kadar gider. Tum islem bittikten sonra sonuca atama yapar.

**&&** Operatoru tum mantıksal ifadeleri kontrol etmeden sonuca gidebildigi icin daha hizlidir.

```
int a=10;
int b=15;

System.out.println(a<b && b<10 && b>=a && a<0);

System.out.println(a<b & b<10 & b>=a & a<0);
```



# Logical (Mantıksal) Operators

## 2- OR (veya) Operatoru ||

Mantıktaki OR operatoru ile aynı şekilde kullanılır.

|| operatoru birlestirdiği 2 boolean ifadenin ikisi de false ise sonucu false yapar, bunun disindaki tum durumlarda sonucu true yapar. (OR operatoru iyimserdir.)

```
int a=10;
int b=15;

System.out.println(a>b || b>0);

System.out.println(a<=b-5 || a>b-8);

boolean c;

System.out.println(c=15>=b || a<0);

c= a>=b || 3*a<4*b;

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

|| operatoru toplamaya benzetilir.  
*Sonucun 0 olması için*  
tüm toplanan sayılar 0 olmalıdır.  
1 tane bile bir olsa sonuc 0 olmaz

```
1 + 1 + 1 + 1 != 0
1 + 0 + 0 + 0 != 0
0 + 0 + 0 + 0 == 0
0 + 1 + 1 + 1 != 0
```

# If Statements

If Statements Java'da kod yazarken mutlaka ihtiyac duyacagimiz temel kod bloklarindan biridir.

Gunluk dilimizde oldugu gibi, bir şart ve ona bagli bir sonucu ifade eder.

Eger **iyi calisirsan**, **sinavi gecersin** **Calismazsan sonucu bilmiyoruz.**

Sart cumlesi belirtec

**boolean şart**

Sart saglanirsa sonuc

```
int a= 10;  
int b= 20;  
  
if (a>b){ System.out.println("a b'den buyuk");}
```

Kod alt satira gecer



# If Statements

Basit if cumleleri, kod'un geriye kalani ile baglantili degildir.

Belirlenen boolean sarti kontrol eder, o sart saglanirsa **if body** calisir, sart saglanmazsa **if body** calismaz

```
int a= 10;
int b= 20;

if (a>b){
    System.out.println("a b'den buyuk");
}

if (a<100){
    System.out.println("a 100'den kucuk");
}

if (b>0){
    System.out.println("b 0'dan buyuk");
}
```



# If Statements

If cumlesindeki boolean şart daha onceden de tanımlanabilir.

```
int a= 10;
int b= 20;

boolean sonuc=a>b;
if (sonuc){
    System.out.println("a b'den buyuk");
}

sonuc= a<100;
if (sonuc){
    System.out.println("a 100'den kucuk");
}

sonuc= b>0;
if (sonuc){
    System.out.println("b 0'dan buyuk");
}
```



Wise Quarter  
first class IT courses

# Java

## Team 105 Ders-06

### If Statements

### If Else Statements

+1 912 888 1630

[www.wisequarter.com](http://www.wisequarter.com)



The future at your fingertips



# Socrative Quiz

1) <https://b.socrative.com/login/student/>  
adresine gidin

(google'da Socrative student aranınca cıktı)

2) Room Name **BULUTLUOZ** yazın

3) Isminizi yazın

4) Done butonuna basin

Sure : 15 Dakika



# If Statements Sorular

**Soru 1-** Kullanicidan bir sayi isteyin, sayiyi kontrol edip 5 ile bolunebiliyorsa  
“Sayi 5'in tam kati” yazdirin.

**Soru 2-** Kullanicidan bir harf alin, harf ile baslayan bir ay varsa yazdirin.  
NOT: Buyuk harf, kucuk harf hassasiyeti olmasin.  
Kullanici o veya O yazdiginda output Ocak olsun.

**Soru 3-** Kullanicidan bir sayi alin, sayi 3 ile bolunuyorsa ”Uc ile bolunebilen  
sayi”, 5 ile bolunebiliyorsa “Bes ile bolunebilen sayi” yazdirin.

**Soru 4-** Kullanicidan bir ucgenin 3 kenar uzunlugunu alin, ucgen eskenar ise  
“Eskenar ucgen” yazdirin.

**Soru 5-** Kullanicidan notunu alin 50 veya daha buyukse ”Sinifi Gectin”, 50'den  
kucukse “Maalesef kaldin” yazdirin.



# If Else Statements

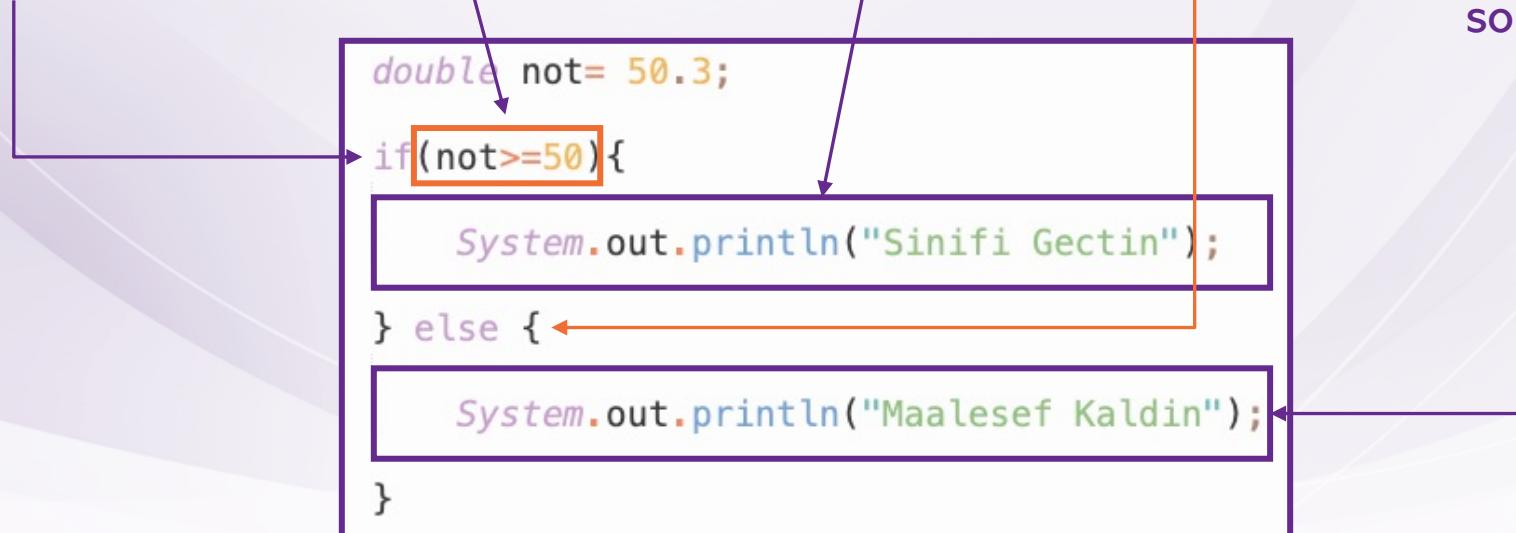
Basit if cümleleri kodun geri kalani ile ilgilenmiyor.

Sorularda şartın sağlanması veya sağlanmaması durumunda yapılacaklar belli ise basit if cümlesi yeterli olmayacağından.

Eğer sayı çiftse, "Çift Sayı" yazdır yoksa, "Tek Sayı" yazdır

Şart cümlesi belirteç boolean şart Sart sağlanırsa sonuc belirteç

Sart sağlanmazsa sonuc



# If Else Statements Sorular

**Soru 1-** Kullanicidan bir ucgenin 3 kenar uzunlugunu alin, ucgen eskenar ise “Eskenar ucgen” yazdirin, degilse “Eskenar degil” yazdirin.

**Soru 2-** Kullanicidan notunu alin 50 veya daha buyukse ”Sinifi Gectin”, 50’den kucukse ”Maalesef kaldin” yazdirin.

**Soru 3-** Kullanicidan yasini isteyin, 65 yas ve uzeri ise ”Emekli olabilirsin” yazdirin, yoksa emekli olmasi icin calismasi gereken yil sayisini yazdirin.

**Soru 4-** Kullanicidan bir karakter girmesini isteyin, girilen karakterin buyuk harf olup olmadigini yazdirin.

**Soru 5-** Kullanicidan bir harf isteyin, girilen karakter kucuk harf ise onu buyuk harf olarak yazdirin, yoksa girilen harfi yazdirin



# If Else If ... Statements

Bazen karsilastigimiz bir durumda secenek sayisi 2'den fazla olur. Bu durumda bir tane if else cumlesi sorunu cozmez.

Ornek :

Ogrencinin notu 85 ve ustu ise AA,

(85 ve ustu degilse) 65 ve ustu ise BB,

(65 ve ustu de degilse) 50 ve ustu ise CC,

(geriye kalanlar) DD

```
double not= 50.3;  
  
if(not>=85){  
    System.out.println("Notunuz AA");  
}  
else if(not>=65){  
    System.out.println("Notunuz BB");  
}  
else if(not>=50) {  
    System.out.println("Notunuz CC");  
}  
else {  
    System.out.println("Notunuz DD");  
}
```



Wise Quarter  
first class IT courses

# Java

## Team 105 Ders-07

If Else If Statements  
Nested If Else Statements

+1 912 888 1630  
[www.wisequarter.com](http://www.wisequarter.com)

/wisequarter

/wisequarter



[www.wisequarter.com](http://www.wisequarter.com)



# If Else Statements Sorular

- Soru 1-** Kullanicidan cinsiyetini ve yasini alin, Kadın, 60 yas ve uzeri , Erkek 65 yas ve uzeri emekli olabilir. Cinsiyet ve yasini dikkate alarak “Emekli olabilirsın” veya “Emekli olmak icin .. Yil daha calisman gerekir” yazdirin.
- Soru 2-** Kullanicinin kilo(kg) ve boyunu(cm) isteyip vucut kitle endeksini hesaplayin ( $kilo * 10000 / (boy * boy)$ ) vucut kitle endeksi 30'dan buyukse obez, 25-30 arasi ise kilolu, 20-25 arasi ise normal, 20'den kucukse zayıf yazdirin.
- Soru 3-** Kullanicidan aldiği ürün adedi ve ve liste fiyatını alın, kullanıcıya musteri kartı olup olmadığını sorun. Musteri kartı varsa 10 urunden fazla alırsa %20, yoksa %15 indirim yapın, Musteri kartı yoksa 10 urunden fazla alırsa %15, yoksa %10 indirim yapın
- Soru 4-** Kullanicidan mesafeyi kilometre olarak alın ve cevirmek istedigi birimi sorun, istedigi birim metre veya santimetre ise cevirip yazdirin, yoksa “istediginiz birim sisteme kayitli degil” yazdirin.

# If Else Statements

## Soru ) Interview Question

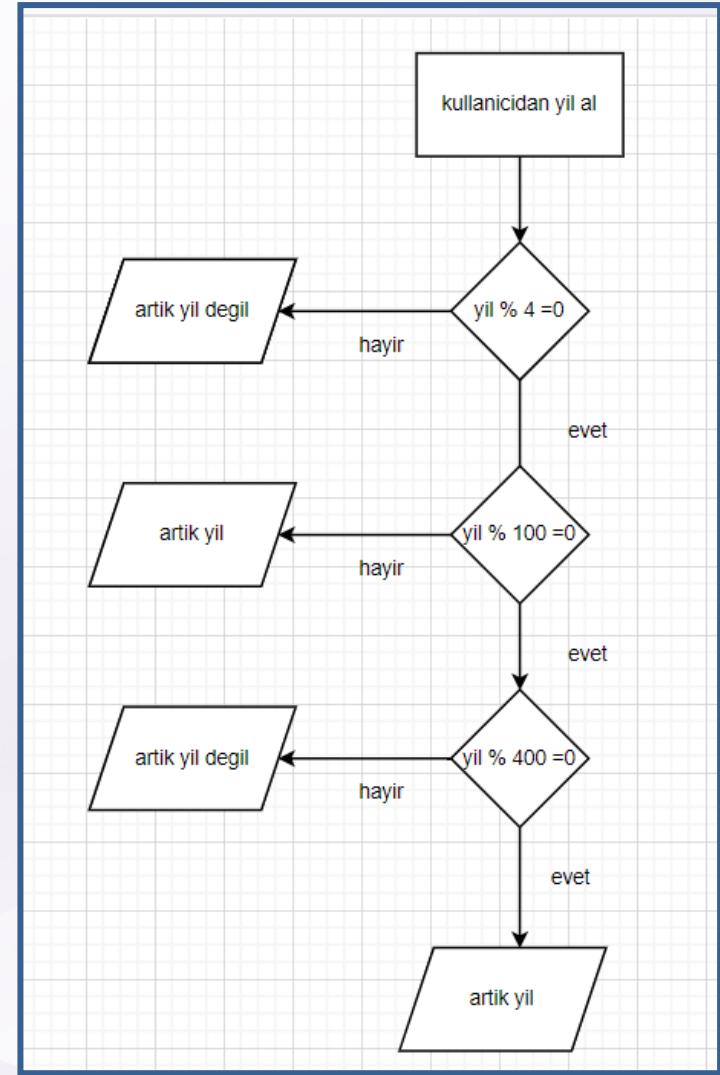
Kullanicidan artik yil olup olmadigini kontrol etmek icin yil girmesini isteyin.

Kural 1: 4 ile bolunemeyen yillar artik yil degildir

Kural 2: 4 ile bolunup 100 ile bolunemeyen yillar artik yildir

Kural 3: 4'un kati olmasina ragmen 100 ile bolunebilen yillardan sadece 400'un kati olan yillar artik yildir

<https://app.diagrams.net/>



# Nested If Statements

Eger kontrol edilecek degisken birden fazla ise ic ice loop'lar olusturmamiz gerekir.

**Ornegin :** Kullanicidan cinsiyetini ve yasini alin, Kadın, 60 yas ve uzeri , Erkek 65 yas ve uzeri emekli olabilir. Cinsiyet ve yasini dikkate alarak “Emekli olabilirsin” veya “Emekli olmak icin .. Yil daha calisman gerekir” yazdirin.

Buna benzer sorularda, degiskenlerden bir tanesini secip ona gore ana yapiyi kurmali, ondan sonra diger degiskene gore detay olusturulmalidir.

```
String cinsiyet= "Kadin";
int yas= 61;

if(cinsiyet.equals("Kadin")){
    // 60'dan buyukse emekli yoksa degil

} else if(cinsiyet.equals("Erkek")){
    // 65'den buyukse emekli yoksa degil

}else {
    // cinsiyet bilgisi hatali
```



# Nested If Statements Sorular

- Soru 1-** Kullanicidan cinsiyetini ve yasini alin, Kadın, 60 yas ve uzeri , Erkek 65 yas ve uzeri emekli olabilir. Cinsiyet ve yasini dikkate alarak “Emekli olabilirsın” veya “Emekli olmak icin .. Yıl daha calisman gerekir” yazdirin.
- Soru 2-** Kullanicidan aldiği ürün adedi ve ve liste fiyatını alın, kullanıcıya musteri kartı olup olmadığını sorun. Musteri kartı varsa 10 urundan fazla alırsa %20, yoksa %15 indirim yapın, Musteri kartı yoksa 10 urundan fazla alırsa %15, yoksa %10 indirim yapın
- Soru 3-** Kullanicidan bir sayı alın sayı tek ise negatif veya pozitif tek sayı olduğunu yazdırın, sayı çift sayı ise 10'un tam kati olup olmadığını yazdırın.
- Soru 4-** Kullanicidan günü ismini girmesini isteyin, girilen gün hafta içi bir gün ise “Simdi çalışma zamanı tatil.. gün var” şeklinde hafta sonu tatiline kaç gün kaldığını yazdırın, girilen gün hafta sonu ise “Simdi dinlenme zamanı” yazdırın.



Wise Quarter  
first class IT courses

# Java

## Team 105 Ders-08

Ternary Operators  
Switch Statements

+1 912 888 1630  
[www.wisequarter.com](http://www.wisequarter.com)

/wisequarter

/wisequarter

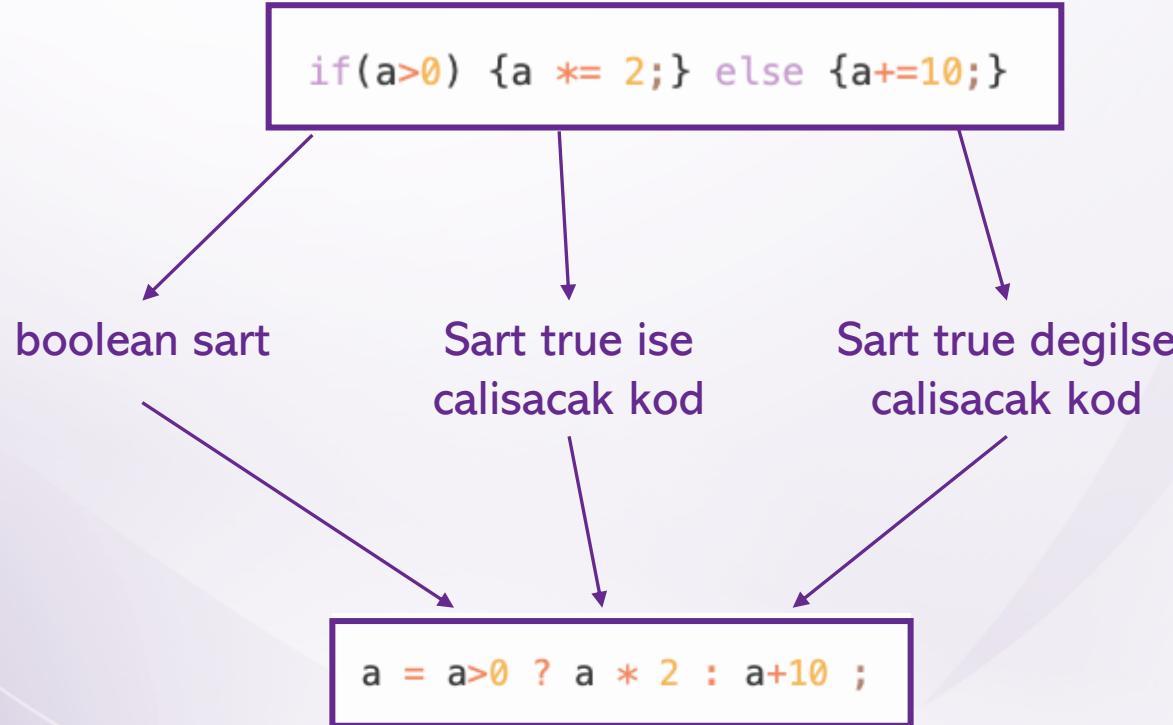


[www.wisequarter.com](http://www.wisequarter.com)



# Ternary Operator

Ternary, if-else statements ile yapabilecegimiz basit islemleri, daha basit bir formda kodlama imkani verir.



If-else statements'da if ve else body'lerinde kompleks kodlar yazabiliriz,  
Ancak ternary'de sadece deger veya deger hesaplayacagimiz basit kodlar yazabiliriz.



# Ternary Operator

Ternary,sadece deger dondurdugu icin, ya yazdirmali veya bir variable'a atamalisiniz.

```
a>0 ? a * 2 : a+10 ;
```

```
System.out.println( a > 0 ? a * 2 : a + 10);
```

```
a= a > 0 ? a * 2 : a + 10;
```



# Ternary Operator

Bir Ternary Ifadenin sonucunu yazdirdigimizda, şart sağlanırsa veya sağlanmazsa yazdırılacak datanın turu onemli olmaz

```
int a = 10;  
  
System.out.println(a > 0 ? "girilen sayı pozitif" : a + 10);
```

Ancak, ternary ifade'nin sonucunu bir variable'a atama yapacaksak, şart sağlanırsa veya sağlanmazsa elde edilecek sonucun aynı data turunde olması gereklidir.

```
int a = 10;  
  
a= a > 0 ? "girilen sayı pozitif" : a + 10;
```

```
a= a > 0 ? a * 2 : a + 10;
```

# Ternary Operator Sorular

**Soru 1-** Kullanicidan bir sayi isteyin, sayiyi kontrol edip 5 ile bolunebiliyorsa  
“Sayi 5'in tam kati” yazdirin.

**Soru 2-** Kullanicidan bir ucgenin 3 kenar uzunlugunu alin, ucgen eskenar ise  
“Eskenar ucgen” yazdirin, degilse “Eskenar degil” yazdirin.

**Soru 3-** Kullanicidan bir harf isteyin, girilen karakter kucuk harf ise onu buyuk  
harf olarak yazdirin, yoksa girilen harfi yazdirin

**Soru 4-** Kullanicidan notunu alin 50 veya daha buyukse ”Sinifi Gectin”, 50'den  
kucukse ”Maalesef kaldin” yazdirin.

**Soru 5-** Kullanicidan iki sayi alin ve buyuk olmayan sayiyi yazdirin

**Soru 6-** Kullanicidan bir sayi alin ve mutlak degerini yazdirin



# Ternary Operator

Asagidaki kod'lar calistiginda konsolda ne yazdiracagini bulun

```
int a = 10;

System.out.println(a>0 ? "Sayi Pozitif" : "Sayi Pozitif degil");

System.out.println(a>20 ? a*a : a++);

System.out.println(a<100 || a<0 ? 3*a+1 : 2 + a /5 );

int x=10;
int y=15;

int z = a>0 ? y++ : --x;

System.out.println(x +" , "+y+" , "+ z);
```

# Nested Ternary Operator

Ternary operatoru basit islemlerde kullanilmak uzere dizayn edilse de bazen kompleks islemleri de ternary ile yapmak isteyebilirsiniz (Tavsiye edilmez).

Ornek : Kullanicidan bir tamsayi alin.

Sayi pozitifse, cift sayi veya cift sayi degil seceneklerinden uygun olani yazdirin

Sayi pozitif degilse, 3 basamaklı veya 3 basamaklı degil seceneklerinden uygun olani yazdirin

```
System.out.println(a>0 ? Sayi pozitifse calisacak kod : Sayi pozitif degilse calisacak kod);
```

```
a%2==0 ; "sayi cift sayi" : "sayi cift sayi degil"
```

```
a<=-100 && a>-1000 ? "3 basamakli" : "3 basamakli degil"
```



# Nested Ternary Operator

Asagidaki kod'lar calistiginda konsolda ne yazdiracagini bulun

```
int a = 10;  
int b = 20;
```

```
System.out.println( a > 5 ? a > 0 ? 100 : 50 : a < 20 ? a + 5 : a - 5);
```

```
System.out.println( b < a ? b > 0 ? b+a : b-a : a < 10 ? a * 5 : b/a);
```

```
System.out.println( a == b ? a > b ? a : b : a < b ? a + b : a - b);
```



# Switch Statements

If else statement ile cozdugumuz sorularda olasi durumların sayisi arttıkça, if else if... yapısı kurgulaması ve anlasılması zor hale gelebilir.

Ornegin kullanıcının rakam olarak girdiği gün numarının ismini yazmamız için 8 if else gereklidir, bu durum sonrasında kodu inceleyenler için uzun ve zor olabilir.

```
Scanner scan = new Scanner(System.in);
System.out.println("Lütfen bir rakam giriniz");
int rakam= scan.nextInt();

if (rakam==1){System.out.println("Pazartesi");}
else if (rakam==2){System.out.println("Sali");}
else if (rakam==3){System.out.println("Carsamba");}
else if (rakam==4){System.out.println("Persembe");}
else if (rakam==5){System.out.println("Cuma");}
else if (rakam==6){System.out.println("Cumartesi");}
else if (rakam==7){System.out.println("Pazar");}
else {System.out.println("Gun sayisi gecersiz");}
```



# Switch Statements

Bu tur sorulari daha anlasilir bir kod ile cozmek icin switch – case yapisi kullanabiliriz

```
switch (rakam){  
    case 1: System.out.println("Pazartesi");  
    break;  
    case 2: System.out.println("Sali");  
    break;  
    case 3: System.out.println("Carsamba");  
    break;  
    case 4: System.out.println("Persembe");  
    break;  
    case 5: System.out.println("Cuma");  
    break;  
    case 6: System.out.println("Cumartesi");  
    break;  
    case 7: System.out.println("Pazar");  
    break;  
    default: System.out.println("Gun sayisi gecersiz");  
}
```

# Switch Statements

Switch Statement kullanımında dikkat edilecek konular.

- 1- Switch Statement'da switch parantezinde long, double, float ve boolean kullanılamaz.
- 2- Switch Statement'da switch parantezinde yazılan degere uygun case calisir ve **break** gorunceye veya switch case bitinceye kadar calismaya devam eder.

```
switch (rakam){  
    case 1: System.out.println("Pazartesi");  
    break;  
    case 2: System.out.println("Sali");  
    break;  
    case 3: System.out.println("Carsamba");  
    break;  
    case 4: System.out.println("Persembe");  
    break;  
    case 5: System.out.println("Cuma");  
    break;  
    case 6: System.out.println("Cumartesi");  
    break;  
    case 7: System.out.println("Pazar");  
    break;  
    default: System.out.println("Gun sayisi gecersiz");  
}
```

break; komutunu her case'den sonra kullanmak zorunda değiliz, ancak bu durumda kodun break gorunceye kadar devam edeceğini unutmamamız gereklidir.

case'leri grüplendirmek için bu yöntem kullanılabilir.

- 3- switch parantezine yazılan değer hiç bir case ile uyusmazsa **default;** satırı devreye girer. (if-else if-if... lerin sonundaki else gibi)



# Switch Statements

**Soru 1-** Kullanicidan bir rakam alip, rakami yaziyla yazdirin.

**Soru 2-** Kullanicidan 2 basamakli bir sayi alip, girilen sayiyi yazi ile yazdirin

**Soru 3-** Kullanicidan ay numarasini alip ay ismini yazdirin

**Soru 4-** Kullanicidan ISTQB kisaltmasindan harfin anlamini ogrenmek istedigini alin ve girilen harfin karsiligini yazdirin.

I : International S : Software T : Testing Q : Qualifications B: Board

**Soru 5-** Kullanicidan gun numarasini alip hafta ici veya hafta sonu yazdirin

**Soru 6-** Kullanicidan ay numarasini alip mevsimi yazdirin.



Wise Quarter  
first class IT courses

# Java

## Team 105 Ders-09

### String Manipulations

+1 912 888 1630  
[www.wisequarter.com](http://www.wisequarter.com)

/wisequarter

/wisequarter



The future at your fingertips

[www.wisequarter.com](http://www.wisequarter.com)

# String Manipulations

Verilen bir String'i **hazir method'ları** kullanarak **degistirmeye** denir.

String manipulation yapilirken, degisikligin kalici olmasi isteniyorsa, atama yapilmalidir.

```
String str= "Java candir";  
  
System.out.println(str.toUpperCase()); // JAVA CANDIR  
  
System.out.println(str); // Java candir
```

String class'inin ozelliginden dolayi (ileride anlatilacak – immutable class), atama yapilmadan calistirilan method'lar variable'da kalici degisiklik yapmazlar.

```
String str= "Java candir";  
  
str=str.toUpperCase();  
  
System.out.println(str); // JAVA CANDIR
```

# String Manipulations

1. str.toUpperCase( );

Verilen String'i buyuk / kucuk harfe cevirir

2. str.toLowerCase( );

```
String str= "Java candir";  
  
str=str.toUpperCase();  
  
System.out.println(str); // JAVA CANDIR  
  
System.out.println(str.toLowerCase()); // Java candir
```

Eger bu degisimi yaparken ingilizce disinda bir dili esas almak isterseniz Locale secennegi kullanilir.

```
str="JAVA CANDIR";  
System.out.println(str.toLowerCase(Locale.GERMAN)); // java candir  
System.out.println(str.toLowerCase(Locale.forLanguageTag("Tr"))); // java candır
```

# String Manipulations

3. str.equals( baskaStr ); Verilen iki String'in metinlerini karsilastirir. İki String birbiriyle ayni metinleri iceriyorsa **true**, herhangi bir farklilik varsa **false** dondurur.

```
String str1= "Fatih";
String str2= "fatih";
String str3= new String( original: "Fatih" ); // Fatih

System.out.println(str1.equals(str3)); // true
System.out.println(str1.equals(str2)); // false
```

**NOT :** Diger primitive data turlerinde kullandigimiz == (double equal sign)'nin iki String'i karsilastirmak icin kullanilmasi tavsiye edilmez.

lleride detayini gorecegiz( **String Pool** ).

== karsilastirirken hem metne hem de stack memory'deki referansa baktigi icin tamamen ayni metne sahip iki String'i karsilastirirken **bazen true, bazen false** donecektir.

```
String str1= "Fatih";
String str2= "Fatih";
String str3= new String( original: "Fatih" ); // Fatih

System.out.println(str1==str2); // true
System.out.println(str1==str3); // false
```

# String Manipulations

## 4. str.equalsIgnoreCase( baskaStr );

Verilen iki String'in metinlerini karsilastirir. Case-sensitive olmadan birbiriyle ayni metinleri iceriyorsa **true**, herhangi bir farklilik varsa **false** dondurur.

```
String isim1 = "Kadir";
String isim2 = "kadir";
String isim3 = "Kadir ";

System.out.println(isim1.equals(isim2)); // false
System.out.println(isim1.equalsIgnoreCase(isim2)); // true

System.out.println(isim1.equals(isim3)); // false
System.out.println(isim1.equalsIgnoreCase(isim3)); // false
```

**NOT :** equalsIgnoreCase( ); sadece buyuk / kucuk harf farkliliklarini ignore eder, farkli bir karakter bulunmasi durumunda **herzaman false** donecektir (bosluk da bir karakterdir)

# String Manipulations

5. str.charAt( istenenIndex );

Verilen bir String'in istenen index'indeki char karakteri bize döndürür.

```
String str= "Java Candır";  
  
System.out.println(str.charAt(0)); // J  
System.out.println(str.charAt(3)); // a  
System.out.println(str.charAt(10)); // r
```

**NOT 1:** Java'da index 0'dan baslar. İlk harfe ulaşmak için str.charAt(0); yazmalısınız

**NOT 2:** Index 0'dan basladığı için son index toplamKaraktersayısı -1 olacaktır.

Yukarıdaki örnekte karakter sayısı 11 iken, son harfe ulaşmak için charAt(10); kullanılmalıdır.

**NOT 3:** Son index'den daha büyük bir index yazdığında java hata verir

```
System.out.println(str.charAt(20));
```

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException Create breakpoint : String index out of range: 20  
at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:47)  
at java.base/java.lang.String.charAt(String.java:693)  
at day08_switchStatements_StringManipulations.C10_charAt.main(C10_charAt.java:17)
```

# String Manipulations

## 6. str.length();

Verilen bir String'deki karakter sayisini bize döndürür.

```
String str= " Uzunkavaklaraltindayataruyumazoglu";  
  
System.out.println(str.length()); // 35
```

**NOT 1:** Index 0'dan basladigi icin son index str.length() -1 olacaktir. Yukarıdaki ornekte length(uzunluk) 11 oldugundan, son index  $35-1 = 34$  olacaktir.

**NOT 2:** Her hangi bir karakter istenirken index sayisi degil, sondan kacinci harf oldugu veriliyorsa (`length - sondanKacinciKarakter`) index olarak kullanilabilir. Ornegin sondan 3.karakter isteniyorsa,

```
System.out.println(str.charAt(str.length()-3)); // g
```

# String Manipulations

## 7. str.substring( istenenParametre );

Verilen bir String'in istedigimiz bir bolumunu bize döndürür. İstedigimiz bolume uygun olarak 1 parametre veya 2 parametrelili 2 farkli kullanimi vardir

str.substring( tekParametre );

Parametre olarak girilen index'den String'in sonuna kadar olan bolumunu bize döndürür.

```
String str= "Java Guzeldir";  
  
System.out.println(str.substring( beginIndex: 2)); // va Guzeldir  
  
System.out.println(str.substring( beginIndex: 10)); // dir
```

Eger sondan istenen kadar karakteri elde etmek istersek length( )-istenenkarakterSayisi kullanilir.

```
System.out.println(str.substring( beginIndex: str.length()-3)); // dir  
  
System.out.println(str.substring( beginIndex: str.length()-1)); // r
```

# String Manipulations

7. str.substring( istenenParametre );

str.substring( baslangicIndex, bitisIndex );

Parametre olarak girilen iki index'den baslangic index'i dahil, bitis index'i haric bolumunu bize döndürür.

```
String str= "Java Guzeldir.";  
  
System.out.println(str.substring(1,3)); // av  
  
System.out.println(str.substring(5,10)); // Guzel  
  
System.out.println(str.substring(0,12)); // Java Guzeldi
```

Eger parametre olarak sondan belirlenen index'i istersek length( ) kullanilabilir.

```
System.out.println(str.substring(0, str.length()-3)); // Java Guzeld
```

# String Manipulations

## 7. str.substring( istenenParametre );

charAt(istenenIndex); method'u bize istenen indexdeki karakteri dondurur fakat char oldugu icin sonrasinda String method'ini kullanamayiz, bunun yerine substring kullanabilir

```
// sadece 5.index'deki harfi yazdiralim
System.out.println(str.substring(5,6));

// sadece 2.indexteki harfi buyuk harf olarak yazdiralim
System.out.println(str.substring(2,3).toUpperCase());
```

Eger parametre olarak ayni index'i baslangic ve bitis olarak secerek bize hiclik döndürür.

```
System.out.println(str.substring(3,3));
// hiclik yazdirir, konsolda birsey gorunmez
```

Eger parametre olarak girilen bitis index'i, baslangictan kucuk olursa hata olusur.

```
System.out.println(str.substring(5,2));
// RTE 5.index'den sonra 2.index'i bulamaz
```

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException Create breakpoint : begin 5, end 2, length 14
```



Wise Quarter  
first class IT courses

# Java

## Team 105 Ders-10

### String Manipulations

+1 912 888 1630  
[www.wisequarter.com](http://www.wisequarter.com)

/wisequarter

/wisequarter



The future at your fingertips

[www.wisequarter.com](http://www.wisequarter.com)

# String Manipulations

## 8. str.concat( baskaString );

Istedigimiz String'in sonuna baska bir String ekler.

Daha once String variable'lar ile + (toplama) islemi yaptigimizda, Java'nin bunu matematiksel toplama olarak degil birlestirme (concat) olarak algiladigini soylemistik.

+ islemini method ile yapmak istersek concat( ) kullanabiliriz.

```
System.out.println(a+d+b+d+s+t); // Java Guzeldir 54
System.out.println(a.concat(d).concat(b).concat(d).concat(str: s+t+c)); // Java Guzeldir 9
```



# String Manipulations

## 9. str.contains ( baskaString );

Bir String'in başka bir String'i icerip icermedigini kontrol eder, boolean sonuc döndürür.

Aranan String'in kaç tane olduğunu tespit edemez, sadece **var** veya **yok** cevabi verir.

```
String str= "Java cok guzel, cok.";  
  
System.out.println(str.contains("Java")); // true  
  
System.out.println(str.contains("java")); // false  
  
System.out.println(str.contains("cok")); // true  
  
System.out.println(str.contains("a")); // true  
  
System.out.println(str.contains(" ")); // true  
  
System.out.println(str.contains(""))); // true
```

**NOT :** contains( ) parametre olarak **char** kabul etmez, ananan charSequence yani String olmalıdır

# String Manipulations

## 10. str.startsWith ( baskaString );

Bir String'in başka bir String ile baslayip, baslamadigini kontrol eder, boolean sonuc döndürür.

2 kullanım sekli vardir. **Tek parametreli** olursa str'in baş kismini kontrol eder

```
String str="Java çok guzel,cok.";  
  
System.out.println(str.startsWith("J")); // true  
System.out.println(str.startsWith("Java")); // true  
System.out.println(str.startsWith("Java çok guzel,cok.")); // true  
System.out.println(str.startsWith(""))); // true
```

**2 parametre** olursa, Java'ya baslangic olarak hangi index'i kullanmasini istedigimizi de verebiliriz.

Ornegin, 5.index ve sonrasi "cok" ile mi basliyor diye kontrol edebiliriz.

```
System.out.println(str.startsWith( prefix: "cok", toffset: 5)); // true  
System.out.println(str.startsWith( prefix: "guzel", toffset: 10)); // false
```

# String Manipulations

## 11. str.endsWith ( baskaString );

Bir String'in başka bir String ile bitip, bitmediğini kontrol eder, boolean sonuc döndürür.

```
String str="Java çok güzel,cok.";  
  
System.out.println(str.endsWith("cok")); // false  
System.out.println(str.endsWith("cok.")); // true  
System.out.println(str.endsWith(""))); // true
```

SORU : kullanıcidan bir mail alın

- mail @ icermiyorsa "gecersiz mail"
- mail @gmail.com icermiyorsa, "mail gmail olmali"
- mail @gmail.com ile bitmiyorsa, "mailde yazım hatası var"

yazdırın.



# String Manipulations

## 12. str.indexOf ( baskaString/char );

Bir String icerisinde aradigimiz bir String veya char degerin ilk kullanım index'ini döndürür.

2 parametre kullanırsak aramaya hangi index'den baslayacagini da söyleyebiliriz.

```
String str="Java çok güzel,cok.";  
  
System.out.println(str.indexOf('a'));  
// bulduğu ilk a'nın index'ini verir : 1  
  
System.out.println(str.indexOf( ch: 'a', fromIndex: 1));  
// 1.index ve sonrasında a arar : 1  
  
System.out.println(str.indexOf( ch: 'a', fromIndex: 2)); // 3  
  
System.out.println( str.indexOf("cok")); // 5  
  
System.out.println(str.indexOf( str: "cok", fromIndex: 6)); // 15
```

# String Manipulations

## 12. str.indexOf ( baskaString/char );

indexOf( ) method'u bize int index döndürür.

String icerisinde olmayan bir metin aradigimizda Java'nin bunu bize bir integer ile anlatmasi gereklidir. O ve pozitif sayilar index olarak kullanilabilecegi icin, Java aradigimiz metin aranan String'de olmadiginda bize **-1** döndürerek, durumu rapor eder.

```
String str="Java cok guzel,cok.";  
  
System.out.println(str.indexOf("Soner")); // -1  
System.out.println(str.indexOf('t'));
```

# String Manipulations

12. str.indexOf ( baskaString/char );

**Soru 1 :** Kullanicidan bir String ve aranacak metin alin. String'in aranan metni icerip icermedigini indexOf( ) method'u kullanarak yazdirin.

**Soru 2 :** Kullanicidan bir String ve aranacak metin alin. Aranan metnin String icerisinde kullanimini kontrol ederek asagidaki cumlelerden uygun olanini yazdirin.

- String aranan metni icermiyor
- Aranan metin String'de sadece 1 kere kullanilmis
- Aranan metin String'de sadece 1'den fazla kullanilmis

# String Manipulations

## 13. str.lastIndexOf(arananString );

Aranan String veya char'in verilen metindeki en son kullanımını bulur ve index'ini döndürür.

```
String str= "Java çok guzel, çok";  
  
System.out.println(str.indexOf("cok")); // 5  
System.out.println(str.lastIndexOf( str: "cok")); // 16  
  
System.out.println(str.indexOf('o')); // 6  
System.out.println(str.lastIndexOf( ch: 'o'))); // 17
```

str.lastIndexOf(arananString, sonIndex ); şeklinde kullanırsak aramaya sonIndex olarak girilen index'den başlar ve basa doğru devam eder.

Her iki kullanımda da arananString/char'i bulamazsa -1 döndürür.

```
System.out.println(str.lastIndexOf( str: "cok" , fromIndex: 10)); // 5  
// 10.index ve öncesinde arama yapar  
System.out.println(str.lastIndexOf( ch: 'x'))); // -1  
System.out.println(str.lastIndexOf( str: "x" , fromIndex: 10)); // -1
```

# String Manipulations

13. str.lastIndexOf( baskaString/char );

**Soru 1 :** Kullanicidan bir String ve aranacak metin alin. String'in aranan metni icerip icermeyagini lastIndexOf( ) method'u kullanarak yazdirin.

**Soru 2 :** Kullanicidan bir String ve aranacak metin alin. Aranan metnin String icerisinde kullanimini kontrol ederek asagidaki cumlelerden uygun olanini yazdirin

- String aranan metni icermiyor
- Aranan metin String'de sadece 1 kere kullanilmis
- Aranan metin String'de sadece 1'den fazla kullanilmis

# String Manipulations

## 14. str.isEmpty( );

Verilen bir String'in bos olup olmadigini boolean olarak döndürür.

```
String str= "Java ogren, 70000 euro offer al";  
  
System.out.println(str.isEmpty()); // false  
  
String str2="";  
  
System.out.println(str2.isEmpty()); // true
```

String'in bos olmasi(length=0) ile sadece space'lerden olusmasi farklidir.  
Space'lerden olusan bir String'in uzunlugu 0 olmayacagi icin isEmpty( ) bize false döndürür.

Bir String'in sadece space'lerden olusmus oldugunu kontrol icin str.isBlank( ) kullanilabilir.

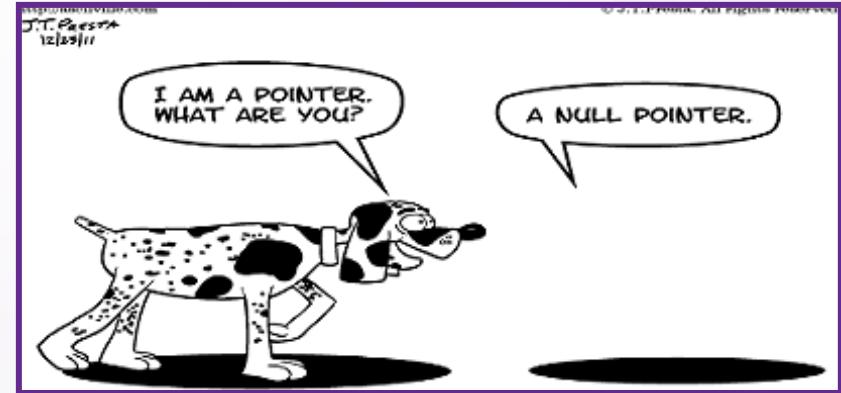
```
String str3= " ";  
System.out.println(str3.isEmpty()); // false  
System.out.println(str3.isBlank()); // true
```



# String Manipulations

## Null Pointer

Null pointer bir deger degil isaretcidir.



```
String isim1=null;  
  
String isim2;  
  
String isim3="";
```

Yandaki 3 isim variable'nin durumlari birbirinden tamamen farklidir.

Isim3'e bir deger atanmistir. Bu degeri yazdirabilir veya method'larda kullanilabilir.

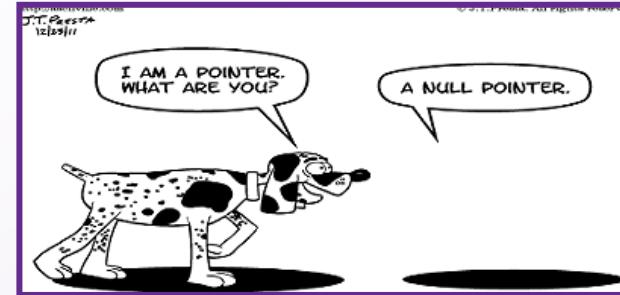
```
System.out.println(isim3);  
//hiclik yazdirir, konsolda birsey gorunmez  
  
System.out.println(isim3.length()); // 0
```



# String Manipulations

## Null Pointer

```
String isim1=null;  
  
String isim2;  
  
String isim3="";
```



İsim1 ve isim2'nin durumları biraz daha benzerdir.

İkisi de oluşturulmuş ve ikisine de **deger atanmamıştır**.

İsim1 null pointer ile işaretlendiği için Java isim1'in kullanım sorumluluğunu bize bırakır.

İsim2'yi kullanmanıza ise (değer atanmadığı surece) izin vermez.

```
System.out.println( isim2 ); // CTE  
System.out.println( isim2.length() ); // CTE  
  
System.out.println(isim1); // null  
System.out.println(isim1.length()); // NullPointerException
```

# String Manipulations

## 15. str.replace ( degisecekString , yeniDeger );

Bir String'in icinde bulunan **degisecekString'lerin** tumunu **yeniDeger** yapar.

Parametre olarak char da kullanilabilir, bu durumda **degisecekChar'larin** tumunu **yeniChar** yapar.

```
String str= "Java ogren, isi kap";  
  
System.out.println(str.replace( oldChar: 'J', newChar: 'j'));  
// java ogren, isi kap  
  
System.out.println(str); // Java ogren, isi kap  
  
str=str.replace( target: "isi", replacement: "offer'i");  
// String'deki degisikligin kalici olmasi icin atama yapmaliyiz  
  
System.out.println(str); // Java ogren, offer'i kap
```

**degisecekString** ile **yeniDeger'in** ayni uzunlukta olması şart değildir. Daha uzun veya daha kısa olabilir.

**degisecekString'i** tamamen silmek istiyorsak **yeniDeger'i** “” (hiclik) secebiliyoruz.

# String Manipulations

16. str.replaceFirst ( degisecekString , yeniDeger );

Bir String'in icinde bulunan degisecekString'lerden ilkini yeniDeger yapar.

degisecekString String olabilecegi gibi regex de olabilir.

```
String str = "Herkesin github'i olmali";  
  
System.out.println(str.replaceFirst( regex: "e" , replacement: "a"));  
// Harkesin github'i olmali  
  
System.out.println(str.replaceFirst( regex: "\\\w" , replacement: "1"));  
// 1erkesin github'i olmali
```

## Regex (Regular Expressions)

\s : space

\S : space olmayan hersey

\s+ : yanyana birden fazla space

\d : digits

\D : digit olmayan hersey

\w : harf veya rakam

\W : harf veya rakam olmayan hersey

# String Manipulations

Bir String'in icinde bulunan degisecekRegex'e uyan tum karakterleri yeniDeger yapar.

str.replace'den farki tek tek harfleri veya metinleri degistirmek yerine, parametre olarak girilen regex'in kapsadigi tum karakterleri degistirmesidir.

Tum rakamlar, tum space'ler, rakam olmayan tum karakterler vb...

```
str="J1a2va3 G4u5z6e7l8d9i0r.";

// ornegin yukarida metin'de tum rakamlardan tek seferde kurtulalim
str=str.replaceAll( regex: "\\\d", replacement: "");

System.out.println(str); // Java Guzeldir.

// eger birden fazla bosluk olan yerleri tek space yapmak istersek
str="Java      Guzel bir programlama      dili";

str=str.replaceAll( regex: "\\\s+", replacement: " ");

System.out.println(str); // Java Guzel bir programlama dili
```

# String Manipulations

## 18. str.repeat ( tekrarSayisi);

Bir String'i tekrarSayisi kadar tekrar ettirir.

```
String str= "Java Candir.";  
  
System.out.println(str.repeat( count: 4));  
// Java Candir.Java Candir.Java Candir.Java Candir.
```

## 19. str.trim ( );

Bir String'in basında ve sonunda (**varsa**) bulunan space'leri siler.

```
str= "    Ali kos    ";  
  
str=str.trim();  
  
System.out.println(str); // Ali kos
```

# String Manipulations

**Soru 1 :** Kullanicidan bir cumle alin

- cumlede ev geciyorsa, "home home sweet home" yazdirin
- cumlede is geciyorsa, "calismak guzeldir"
- ikisini de iceriyorsa "Hem ev lazim hem is"
- hicbirini icermiyorsa "cok calisman lazim" yazdirin

**Soru 2 :** Kullanicinin belirli bir formatta verdigi String fiyatları toplayip yazdirin.

input1 : “15.30 €” , input2 : “11.40 €”

output : 36.70 €

**Soru 3 :** Kullanicidan alınan metindeki istenmeyen rakam ve ozel karakterleri silip, sadece ilk harfi buyuk diger harfler kucuk harf yapan bir program yazın.

input : java1 ogRe2@nMek3 #ne +Gu=zel

output : Java ogrenmek ne guzel.

# String Manipulations

**Soru 4 :** Kullanicidan bir sifre isteyip, asagidaki şartlari kontrol edin ve kullaniciya duzeltmesi gereken tum eksikleri soyleyin, eger tum şartlari saglarsa, "sifre basariyla kaydedildi" yazdirin

- ilk harf kucuk harf olmali
- son karakter rakam olmali
- sifre bosluk icermemeli
- uzunlugu en az 10 karakter olmali

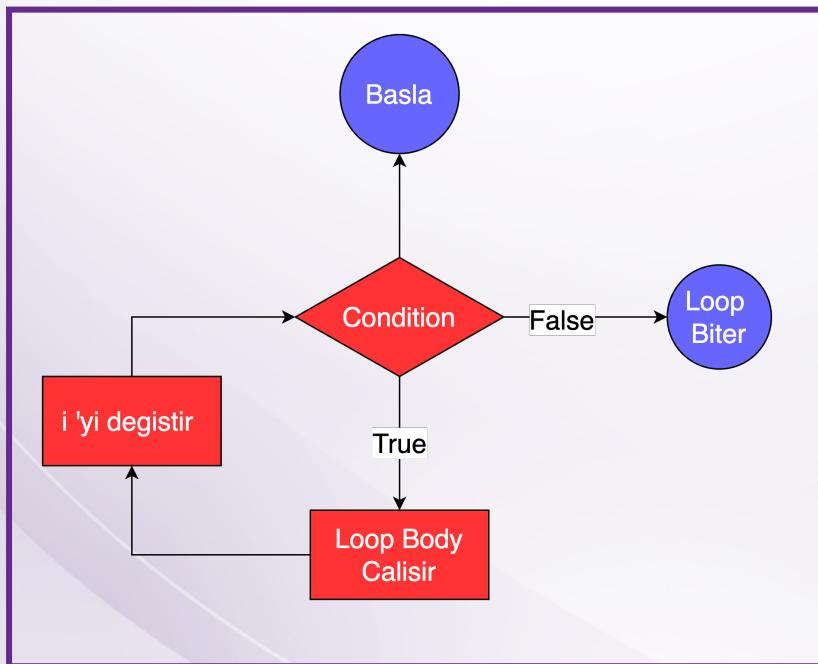
**Soru 5 :** Kullanicidan isim ve soyismini ayri ayri alin.

- ismi daha uzun ise, isim ve soyismi ilk harf buyuk kalanlar kucuk seklinde yazdirin
- soyisim daha uzun ise ismi ilk harf buyuk digerleri kucuk, soyismi buyuk harflerle yazdirin.

**Soru 6 :** Kullanicidan alınan bir String alın, String'in uzunluğu çift sayı ise tam ortasına :) ekleyin, String'in uzunluğu tek sayı ise ortadaki harfi silin ve yerine :( yazdırın.

# For Loops

For Loop, belirli sayıda çalıştırılması gereken bir döngüyü verimli bir şekilde yazmanıza olanak tanıyan bir tekrar kontrol yapısıdır.



For döngüsü, bir görevin kaç kez tekrarlanacağını bildiğinizde kullanışlıdır.

```
for (int i = 0; i < 10 ; i++) {  
    /* şart sağlandığında çalışacak kod */  
    System.out.print(i + " ");  
}
```



# For Loops

NOT 1 : Condition i'nin tum degerleri icin hep true oluyorsa

```
for (int i = 0; i >-10 ; i++) {  
    System.out.print(i + " ");  
}
```

Sonsuz loop olusur

NOT 2 : Condition i'nin ilk degeri icin bile false oluyorsa

```
for (int i = 0; i >10 ; i++) {  
    System.out.print(i + " ");  
}
```

For loop calisir ancak loop body calismaz

# For Loops

**Soru 1-** 1'den 100'e kadar sayilari aynı satırda aralarında bir bosluk bırakarak yazdırın.

**Soru 2-** Kullanıcıdan pozitif bir tamsayı alın, 1'den girilen sayıya kadar(girilen sayı dahil) 7 ile bolunebilen sayıları yazdırın.

**Soru 3-** Kullanıcıdan başlangıç ve bitiş değeri olarak pozitif sayılar alın, sınırlar dahil olarak aralarındaki tüm sayıların toplamını yazdırın. Bitiş değeri başlangıç değerinden küçükse, uyarı yazdırıp işlemi sonlandırın

**Soru 4-** Kullanıcıdan başlangıç ve bitiş değeri olarak pozitif sayılar alın, sınırlar dahil olarak aralarındaki tüm sayıların toplamını yazdırın. Bitiş değeri başlangıç değerinden küçük olsa da program calıssın

**Soru 5-** Kullanıcıdan 20'den küçük bir sayı alıp, bu sayının faktöryel değerini hesaplayın.

**Soru 6-** Kullanıcıdan 20'den küçük bir sayı alıp, bu sayının faktöryel değerini hesaplayın. Konsolda faktöryel hesabının yapılısını da yazdırın.

$$\text{Or : } 6! = 6 * 5 * 4 * 3 * 2 * 1 = 720$$

# For Loops

**Soru 7-** Kullanicidan pozitif bir tamsayi alip, rakamlar toplamini yazdirin.

**Soru 8 (interview)-** Kullanicidan pozitif bir sayi alin, 1'den baslayarak tum tamsayilari yazdirin, sira

- 3 ile bolunebilen bir sayiya gelirse, sayi yerine **fizz**
- 5 ile bolunebilen bir sayiya gelirse sayi yerine **buzz**
- hem 3 hem 5 ile bolunebilen bir sayiya gelirse sayi yerine **fizzBuzz** yazdirin

**Soru 9 (interview)-** Kullanicidan bir String isteyin ve String'i tersten yazdirin.

**Soru 10 (interview)-** Kullanicidan bir String isteyin ve String'i tersine cevirip kaydedin.

**Soru 11-** Kullanicidan pozitif bir tamsayi isteyip, sayinin asal sayi olup olmadigini kontrol edin ve sonucu yazdirin.

# Nested For Loops

For Loop, belirli sayıda çalıştırılması gereken bir döngüyü verimli bir şekilde yazmanıza olanak tanıyan bir tekrar kontrol yapısıdır.

Bazen verilen görevi yapmak için tek bir loop yeterli olmaz, Örneğin bir carpım tablosu hazırlamak veya bir futbol liginde oynanacak maçları planlamak için tek bir loop yeterli olmaz.

Nested for-Loop ihtiyacı iki şekilde karşımıza çıkabilir.

1 -	1	2	3	4	1*1	1*2	1*3	1*4
	2	4	6	8	2*1	2*2	2*3	2*4
	3	6	9	12	3*1	3*2	3*3	3*4
	4	8	12	16	4*1	4*2	4*3	4*4

1.Sayı bir değer alındığında, 2. sayı baştan sona tüm değerleri alıyor

1.Sayı bir değer arttığında, 2. sayı baştan sona tüm değerleri yeniden alıyor

# For Loops

- 2 - 1            1. satir 1'den 1'e kadar yazdir
- 1 2            2. satir 1'den 2'e kadar yazdir
- 1 2 3            3. satir 1'den 3'e kadar yazdir
- 1 2 3 4.        4. satir 1'den 4'e kadar yazdir

Bu tarz sorularda iç dongu 1'den dış loop'un iç degerine kadar gidiyor.

**Soru –** Aşağıdaki şekilleri yazdırın

*	* * * * *	* * * * *
**	* * * * *	* * * *
***	* * * * *	* * *
****	* * * * *	*



# Method Olusturma ve Kullanma

Method'lar istedigimiz islemleri bizim adimiza yapan kod bloklaridir.



Belirli bir isi yapmak icin tasarlanmis robotlar gibidirler. Baslangicta tasarlamaasi ve sorunsuz calismasi emek ve zaman ister ancak calismaya baslayinca, yaptigi islemi sorunsuz olarak tekrar tekrar yaptirabiliriz.

Method'lar da robotlar gibi calismasini istedigimizde calisir. Java calismaya main method'dan baslar ve bir Method Call (Method Cagirma) gorurse o method'u bulur ve calistirir.

# Method Olusturma ve Kullanma

Nicin bir islemi main method icerisinde yapmak yerine method olusturmamizi tercih ederiz ?

## 1- Projemiz icerisinde tekrar tekrar

kullanicigimiz bir islem icin her seferinde  
yeniden kod yazmak yerine bir kere yazip  
ihtiyacimiz oldukca kullanmak (OOP  
Concept)

Ornegin faktoryel hesaplamak zor bir islem  
degildir, ama her ihtiyacimiz oldugunda  
yeniden faktoryel hesaplamak icin kod  
yasmak yerine bir kere method olarak yazip  
ne zaman lazim olsa kullanmak daha pratik  
olacaktir.



## 2- Calistigimiz class'i ve main method'u basit bir yapida tutup, sectigimiz uygun isme sahip method'larla kodumuzu daha anlasilabilir hale getirmek.



# Method Olusturma ve Kullanma

Bir okul projesi yaptigimizi dusunelim.

Ogretmen ekleme, ogrenci  
kayit gibi islemler de  
binlerce kez  
tekrarlanacaktır.

```
===== YILDIZ KOLEJI =====
===== ANA MENU =====
```

- 1- Okul Bilgileri Goruntule
- 2- Ogretmen Menu
- 3- Ogrenci Menu
- Q- ÇIKIŞ

```
===== YILDIZ KOLEJI =====
===== OGRENCI MENU =====
```

- 1- Ogrenci Listesi Yazdir
- 2- Soyisimden Ogrenci Bulma
- 3- Sinif ve Sube Ile Ogrenci Bulma
- 4- Bilgilerini Girerek Ogrenci Ekleme
- 5- Kimlik No Ile Kayit Silme
- A- ANAMENU
- Q- ÇIKIŞ

```
===== YILDIZ KOLEJI =====
===== OGRETMETEN MENU =====
```

- 1- Ogretmenler Listesi Yazdir
- 2- Soyisimden Ogretmen Bulma
- 3- Branstan Ogretmen Bulma
- 4- Bilgilerini Girerek Ogretmen Ekleme
- 5- Kimlik No Ile Kayit Silme
- A- ANAMENU
- Q- ÇIKIŞ

Tum bu kodlari tek bir class'da ve main method'da yapmak uygulamamizi kontrol edilemez ve anlasilamaz bir hale getirecektir.

Gunumuzde Instagram, facebook gibi sosyal media uygulamalarin, bankacilik, alisveris siteleri gibi ticari programlarin veya e-devlet gibi bir ulkenin tum insanlarini ve yapılan tum resmi islemleri kapsayan uygulamalarin olusturulmasi ve kullaniminin pratik olarak yapisilmesi icin MUTLAKA method'lara ihtiyacimiz olacaktir.



# Method Olusturma ve Kullanma

Bir method'un sonuc olarak bize bir deger dondurmeyi saglar. Matemetik islemlerindeki sonuc gibidir.

Ornegin String method'larini incelerken, hem ne is yaptigina, hem de bize sonuc olarak hangi data turunden bir sonuc dondurdugune bakiyorduk.

str.to

• <b>toLowerCase()</b>	String
• <b>toUpperCase()</b>	String
• <b>toLowerCase(Locale.ROOT)</b>	String
• <b>toLowerCase(Locale locale)</b>	String
• <b>toUpperCase(Locale locale)</b>	String
• <b>toUpperCase(Locale.ROOT)</b>	String
• <b>toCharArray()</b>	char[]
• <b>toString()</b>	String
• <b>compareTo(String anotherString)</b>	int
• <b>compareToIgnoreCase(String str)</b>	int

Press ↵ to insert, → to replace [Next Tip](#)

# Method Olusturma ve Kullanma

Method'lar bize bir sonuc döndürüp döndürmedigine gore 2'ye ayrılır.

1- Bazi method'lar gorevlerini yapar ama bize herhangi bir data turunde sonuc dondurmezler. Bu tur method'larin return type'i void olur.

Ornegin ogrenci kaydi yapan bir method dusunelim, amac , kayit yapan kisiye bir sonuc dondurmekten ziyade ogrenciyi kayit yapmaktir. Belki kayit islemi tamamlandi diye bir sonuc yazdirilabilir ama bu yazdirma islemi asil amac degildir.



“Kayit basariyla yapildi” yazan ama kayit yapmayan bir method calisti Kabul edilemez.

Bu tur method'lari fatura yatirmaya yolladigimiz cocugumuz gibi dusunebiliriz. Amac faturayı yatırmaktır, bize bir makbuz getirmesi degildir.



# Method Olusturma ve Kullanma

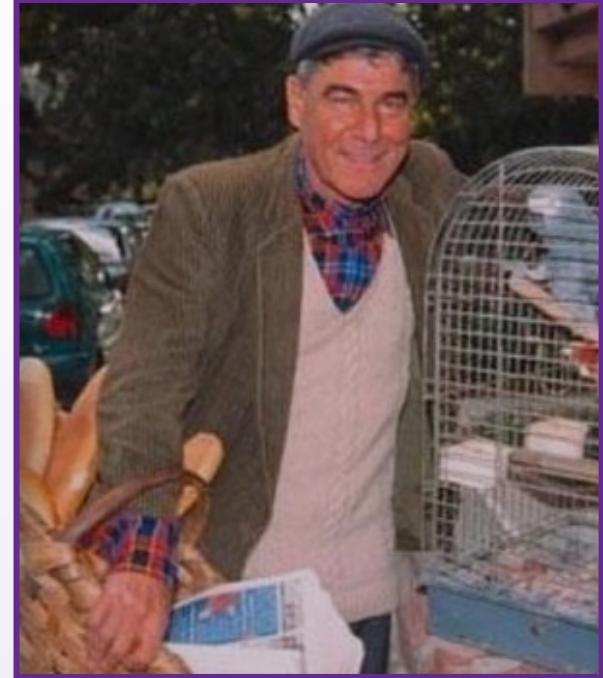
## Method Turleri

### 2- Bize sonuc döndüren method'lar.

Cogu zaman method'lar bize bir sonuc döndürmesi icin olusturulur.

Markete alisverise giden kapici gibidir, bizim istedigimiz urunu getirir. Onun getirmesi yetmez biz de kapiçinin getirdigi urunu ondan almaliyiz.

```
String str= "Java Guzeldir.";  
  
str.toUpperCase(); // bize String dondurur
```



Bu method calistiginda konsolda bir sey de goremeyez, str da degismez.

Bize sonuc döndüren method'lar ya direk yazdirilmali veya data turune uygun bir variable'a atanmalidir.

```
System.out.println(str.toUpperCase());  
  
str= str.toUpperCase();
```



# Method Olusturma ve Kullanma

void mi yoksa return type'li method mu tercih edilmelidir ?

Bu tercih bize verilen gereksinim (requirements)'a gore bizim belirleyecegimiz bir durumdur.

Ancak return type'i olan method'lar daha avantajlidir. Void bir method'a sonuc dondurtemeyiz ama sonuc donen bir method'u System.out.println( ) icinde kullanip sonunu yazdirabiliriz.

```
String str= "Java Guzeldir.";  
  
str.toUpperCase(); // bize String dondurur  
  
System.out.println(str.toUpperCase());  
  
str= str.toUpperCase();
```

# Method Olusturma ve Kullanma

Bir method olusturmak istedigimizde kullanilacak syntax soyledir.

```
public static void toplama(int sayi1, int sayi2){  
    System.out.println( sayi1 + sayi2 );  
}
```

1- access modifier : method'a proje icerisinden nerelerden ulasabilecegini belirler.

**public** : Proje icerisinde tum class'lardan kullanilabilir.

**protected** : Sadece icinde bulundugu package ve child class'lardan kullanilir

**default** : Sadece icinde bulundugu paket(package)'den kullanilir

**private**: Sadece bulundugu class'da kullanilabilir

Access Levels					
Modifier	Class	Package	Subclass	World	
public	Y	Y	Y	Y	
protected	Y	Y	Y	N	
<i>no modifier</i>	Y	Y	N	N	
private	Y	N	N	N	



# Method Olusturma ve Kullanma

Bir method olusturmak istedigimizde kullanilacak syntax soyledir.

```
public static void toplama(int sayi1, int sayi2){  
    System.out.println( sayi1 + sayi2 );  
}
```

**2- static :** Access modifier olmadigi halde method ve variable'lar icin erisimi duzenler.

static olarak isaretlenmis method'lar, method disinda bulunan variable ve method'lardan sadece static olarak isaretlenmis olanlara direkt ulasabilir.

main method static olarak isaretlendiginden (**simdilik**) main method'dan cagiracagimiz method'lari da static yapacagiz.

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



# Method Olusturma ve Kullanma

3- return type : Method'un hangi data turunden bir sonuc urettigini belirtir.

Gorevi sadece birsey yazdirmak olan method'larin return type'i void olarak belirlenir.

Method'un gorevi bize bir sonuc dondurmek ise, dondurecegi dataya uygun bir return type secilmeli,

method'un sonunda ise return keyword'u ile beklenen data turunden bir deger dondurulmelidir.

```
public static void sayiTposta(int sayi1, int sayi2){  
    System.out.println(sayi1+sayi2);  
}  
  
public static int sayiTpostaDondur(int sayi1, int sayi2){  
    return sayi1+sayi2;  
}  
  
public static void main(String[] args) {  
  
    sayiTposta(5,10); // 15 yazdirir  
  
    int sonuc= sayiTpostaDondur(20,30);  
    // 50 dondurup sonuc'a assign eder  
  
    // istersek sonucu yazdirabiliriz  
    System.out.println(sonuc); // 50
```

Return type'i void olan method'lar cagrildiginda, sadece **yazirma islemi** yapabilir, Void olmayan method'lar ise bize bir deger dondurur ve **biz de o degeri kaydederiz**,

Sonunu bir variable'a atadiktan sonra istedigimiz zaman yazdirmak mumkun olacaktir.



# Method Olusturma ve Kullanma

```
public static void toplama(int sayi1, int sayi2){  
    System.out.println( sayi1 + sayi2 );  
}
```

**4- method ismi :** Method ismi olarak istedigimiz ismi secebiliriz, ancak method'un islevi ile isminin uyumlu olması tercih edilir.

Method isimleri kucuk harfle baslar ve camelCase kuralina uygun olur.

**5- parametre :** ( ) icerisine yazilan variable'lardir. Bir method cagrildigi zaman (method call) parametrelerine uygun argument'ler ile cagrilmalidir.

Java, herhangi bir method call ile karsilastiginda once method call'daki argument'ler ile method'daki parametre'leri karsilastirir uyumlu degilse CTE verir.

```
public static int sayiToplama(int sayi1, int sayi2){  
    return sayi1+sayi2;  
}  
  
public static void main(String[] args) {  
  
    int sonuc= sayiToplama(20,30);  
}
```



# Method Olusturma ve Kullanma

6- method body : Suslu parantezler arasında kalan ve kodlarimizi yazdigimiz bolumdur.

Soru : Method nerede olusturulabilir ?

Cevap : Class icerisinde, main method veya diger var olan method'larin disinda olmalidir.

```
public class asd {  
  
    public static int sayiToplama(int sayi1, int sayi2) {  
        return sayi1+sayi2;  
    }  
  
    public static void main(String[] args) {  
  
        int sonuc= sayiToplama(20,30);  
  
    }  
}
```

Method'larin main method'dan once veya sonra olmasinin farki yoktur.

Method'lar cagrilmadan calismaz, cagriliyor da nerede olursa olsun Java bulup calistirir.

# Method Olusturma ve Kullanma

**Soru 1-** Kullanicidan input olarak verilen bir String, baslangic ve bitis indexlerine gore baslangic index'i dahil, bitis index'i haric olacak sekilde aradaki harfleri yazdiran bir method olusturun.

- kullanici baslangic degeri olarak, bitis degerinden buyuk bir sayi girerse, hata mesaji verin
- kullanici str'da olan index'lerden daha buyuk bir index girerse hata mesaji yazdirin.

**Soru 2-** Kullanicidan main method icinde ayri ayri isim ve soyismini alin Isim ve soyismi ilk harfleri buyuk diger harfler kucuk olacak sekilde duzenleyip, isim bosluk soyisim seklinde bize donduren bir method olusturun  
**input :** isim : Ali soyisim :YILMAZ.    **output :** Ali Yilmaz

**Soru 3-** Kullanicidan main method icinde pozitif bir tamsayi alin. Girilen sayinin asal sayi olup olmadigini kontrol edip, sonuc olarak “asal sayi” veya “asal sayi degil” sonuclarini donduren bir method olusturun.

**Soru 4-** Kullanicidan main method icinde bir tamsayi alin. Girilen sayinin pozitif tam bolenleri sayisini bulup bize donduren bir method olusturun.

# Method Overriding

**Method overloading** : Bir class'da ayni isimde fakat farkli method signature'ina sahip methodların bulunmasıdır.

**Method overloading'in amacı** ayni islevi farklı parametrelerle, farklı şekilde gerçekleştirebilmektir.

Ornegin, String'deki substring method'unda

- 1 parametre girersek o index'den sona kadar olan metni verir
- 2 parametre girersek, 1.index dahil, 2.index haric olmak üzere aradaki metni verir.



Boylece kullanıcıya ihtiyacına uygun method'u kullanma imkani vermiş oluruz.

# Method Overriding

Bir class'da aynı isimde birden fazla method olusturabilmek için signature'larini degistirmek gereklidir..

**Method signature :** Method ismi, parametre sayisi ve parametrelerin dizilisi demektir.

Method overloading için ismin  
aynı olması gereklidir,  
signature degistirmek için iki  
yontem kalır.

- 1- parametre sayisini degistirmek
- 2- parametrelerin data turunu veya  
data turu farklı olan parametrelerin  
yerlerini degistirmek.  
(aynı data turundeki parametrelerin  
yerini degistirmek signature'i  
degistirmez)

```
System.out.println(carpim(2,3)); // int int 6
System.out.println(carpim(2,3.4)); // double double 6.8
System.out.println(carpim(3,4,5)); // double double double 60.0
}

public static double carpim(double sayi1, double sayi2){
    return sayi1*sayi2;
}
public static int carpim(int sayi1, int sayi2){
    return sayi1*sayi2;
}

public static double carpim(double sayi1, double sayi2,double sayi3){
    return sayi1*sayi2*sayi3;
}
```

# Method Overriding

Bir class'da aynı isimde birden fazla method olduğunda Java hangisini kullanacağı karar vermek için

1- Oncelikle method ismi ve parametre sayısına bakar.

2- Aynı isim ve parametre sayısında birden fazla method varsa, argument ve parametrelerin uyumuna bakar.

- Argumentlerle parametrelerin %100 uyustugu method varsa onu kullanır.

- %100 uyumlu parametre bulamazsa casting ile çalışacak method'lara bakar  
- casting ile çalışacak method birden fazla ise en az casting yapacagini secer.

```
System.out.println(carpim(2,3)); // int int 6
System.out.println(carpim(2,3.4)); // double double 6.8
System.out.println(carpim(3,4,5)); // double double double 60.0
}

public static double carpim(double sayi1, double sayi2){

    return sayi1*sayi2;
}
public static int carpim(int sayi1, int sayi2){

    return sayi1*sayi2;
}

public static double carpim(double sayi1, double sayi2,double sayi3){

    return sayi1*sayi2*sayi3;
}
```

# While Loop

For Loop, belirli sayıda çalıştırılması gereken bir döngüyü verimli bir şekilde yazmanıza olanak tanıyan bir tekrar kontrol yapısıdır.

For loop kullanırken ihtiyacımız olan

- baslangic degeri,
- bitis sarti (condition)
- artis/azalis yontemi

bilgilerine while loop icin de ihtiyac duyuyoruz,  
ama java bunlari otomatik yazmadigi icin  
manuel olarak yazmamiz gerekir.

```
int s=10;  
  
while(s<100){  
    // calisacak kodlar  
    s++;  
}
```

```
for (int i = 0; i <100 ; i++) {  
    // calisacak kodlar  
}
```

Baslangic degerini ve artis degerinin manuel  
yazilmasi, while loop'u baslangicta kullanissiz gibi  
gosterebilir.

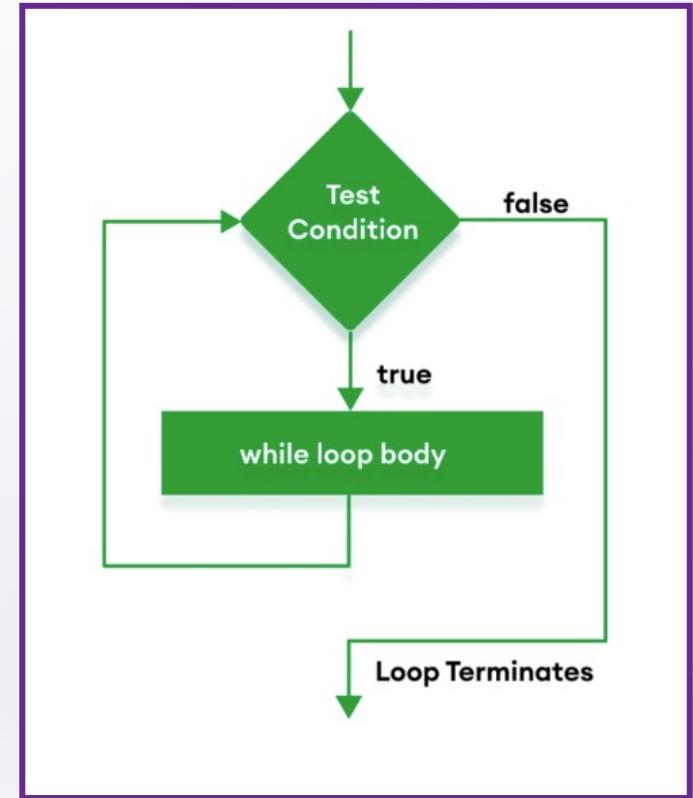
Ozellikle artis/azalis miktarı yazilmazsa kod sonsuz  
loop'a girecek ve bize sorun cikaracaktir.

# While Loop

Ancak while loop bazi durumlarda for loop'dan avantajli olacaktir.

Bir loop'un kac kere calisacagi belli degilse, veya bitis sarti loop degiskene degil, baska bir degiskene bagli ise while loop daha kullanisli olacaktir.

Ornegin kullanicidan sifre istiyorsak ve yanlis giris oldugu muddetce tekrar istememiz gerekiyorsa adim sayisini bilmemiz mumkun olmadigindan while loop tercih edilebilir.



Veya kullanici istedigi muddetce kodumuzun ayni islemi yapmasini istiyorsak, kodun durmasini kullanicinin gireceği “**cikis icin O'a basiniz**” gibi bir degere baglayip, kodu tekrar tekrar calistirabiliriz.



# While Loop

Soru : Kullanicidan toplanmak üzere sayilar isteyin toplam 500 olur veya gecerce toplami yazdirin.

```
Scanner scan= new Scanner(System.in);
int sayi=0;
int toplam=0;

while (toplam<=500){
    System.out.println("Lutfen toplamak üzere sayı girin");
    sayi= scan.nextInt();
    toplam +=sayi;
}

System.out.println("girilen sayıların toplam : "+ toplam);
```

# While Loop

**Soru :** Kullanicidan Kullanicidan sifre isteyin asagidaki şartlari saglamayan sifrelerde hatalari yazdirip, kullanicinin yeni sifre girmesi isteyin Gecerli bir sifre yazilincaya kadar bu islemi tekrar edin gecerli sifre yazilinca “sifreniz basari ile kaydedildi” yazdirin

sartlar :

- sifrenin ilk karakteri kucuk harf olmali
- sifrenin son karakteri sayı olmali

```
Scanner scan = new Scanner(System.in);
boolean sifreDogrumu=false;
String sifre="";
char ilkHarf;
char sonHarf;

while(!sifreDogrumu){ // sifreDogrumu==false

    System.out.println("Lutfen sifre giriniz");
    sifre= scan.nextLine();
    ilkHarf=sifre.charAt(0);
    sonHarf=sifre.charAt(sifre.length()-1);

    if (ilkHarf<'a' || ilkHarf>'z'){
        System.out.println("sifrenin ilk harfi kucuk harf olmali");

    }else if(sonHarf<'0' || sonHarf>'9'){
        System.out.println("sifrenin son karakteri rakam olmali");

    }else{
        System.out.println("Sifre basari ile kaydedildi");
        sifreDogrumu=true;
    }
}
```

# While Loop

**Soru 1-** While loop kullanarak 2 basamakli 7 ile bolunebilen pozitif tamsayıları yazdırın.

**Soru 2-** While loop kullanarak kullanıcıdan alınan sayının rakamlar toplamını bulun.

**Soru 3-** While loop kullanarak verilen bir String'i terse çevirip, bu halini bize donduren bir method oluşturun.

**Soru 4-** Kullanıcıdan toplanmak üzere pozitif tamsayılar isteyin Kullanıcıya bitirmek istediginde 0'a basmasını söyleyin

Kullanıcı bitirmek istediginde toplam kaç adet pozitif tam sayı girdiginive bunların toplamının kaç olduğunu yazdırın

Kullanıcı negatif sayı girerse "negatif sayı kullanamazsınız" yazdırın bu negatif sayiyi sayı adedine ve toplama eklemeyin

**Soru 5-** Kullanıcıdan bir sayı ve hesaplamak istediği ussunu isteyin. While loop kullanarak verilen sayının istenilen ussunu hesaplayıp yazdırın bir method oluşturun.



# Do While Loop

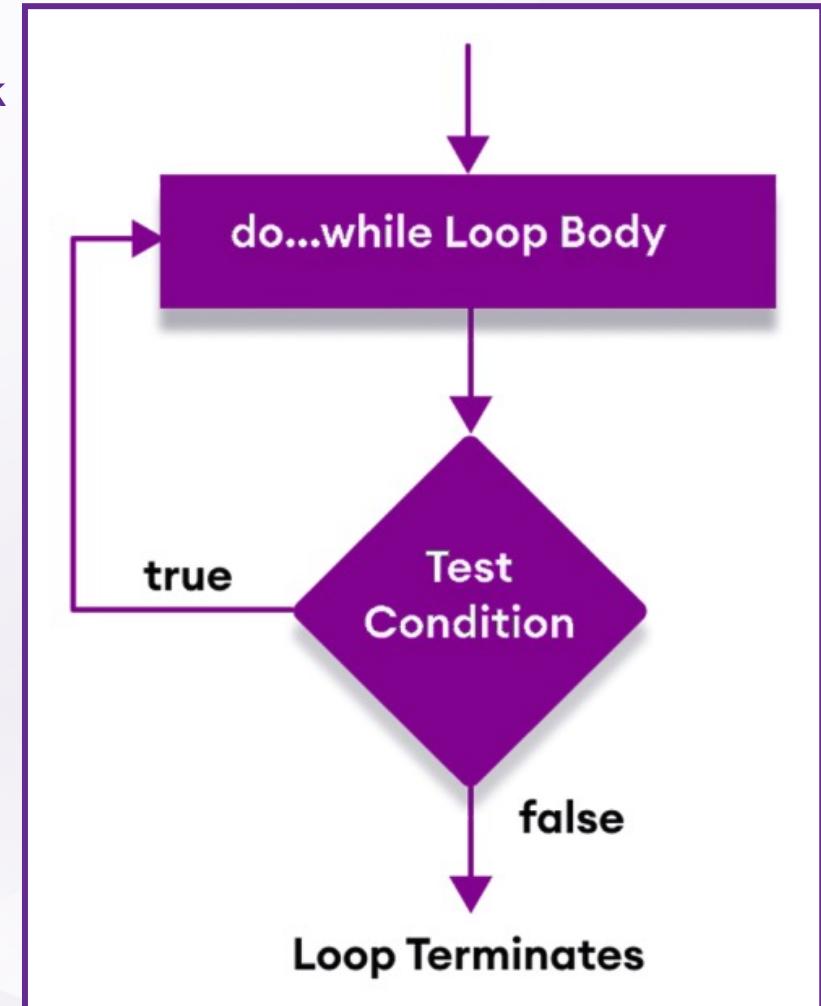
do-while loop, belirtilen koşul doğru olana kadar programın bir bölümünü tekrar tekrar calistirmak için kullanılır.

Java do-while döngüsüne **çıkış kontrol** döngüsü denir.

While döngüsü ve for döngüsünden farklı olarak do-while, döngü gövdesinin sonundaki koşulu kontrol eder.

Java do-while döngüsü, döngü gövdesinden sonra koşul kontrol edildiğinden en az bir kez calisir.

Tekrar sayısı belirli değilse ve döngüyü **en az bir kez çalıştırmanız** gerekiyorsa, bir do-while döngüsü kullanmanız önerilir.



# Do While Loop

**Soru 1-** 'k' harfinden 't' harfine kadar harfleri yazdirin.

**Soru 2-** Kullanicidan bir sifre girmesini isteyin. Girilen sifreyi asagidaki sartlara gore kontrol edin ve sifredeki hatalari yazdirin.

Kullanici gecerli bir sifre girinceye kadar bu islemi tekrar edin ve gecerli sifre girdiginde “Sifreniz Kabul edilmistir” yazdirin.

- Sifre kucuk harf icermelidir
- Sifre buyuk harf icermelidir
- Sifre ozel karakter icermelidir
- Sifre en az 8 karakter olmalidir.

**Soru 3-** Kullanicidan bir pozitif sayi isteyin, sayinin tam kare olup olmadığını bulunuz, tamkare ise true değilse false yazdiriniz.

Ornek : input : 16, output: 4

Scope bir variable'in erisilebildigi alandir.

Scope, variable'in olusturuldugu yer goz onunde bulundurularak 2 ana gruba ayrilir.

### 1- Local variable'lar

Local variable'lar bir method veya kod blogu icerisinde olusturulan variable'lardir.

Local variable'larin scope'u icerisinde olusturulduklari kod blogudur ve o blogun disinda kullanilamazlar.

### 2- Class level variable'lar

Class level variable'lar method ve kod bloklarinin disinda olusturulurlar ve scope'lari tum class'dir.

Ancak static keyword kullanilip kullanilmamasina gore erisimleri ve kullanimlari farkli olur.

# Scope

```
public class Scope {  
  
    static String hastaneIsim;  
    String persIsim;  
    String persSoyisim;  
    String persNo;  
    int cihazNo;  
    String servis;  
  
    public static void main(String[] args) {  
  
        int sayi=10;  
  
        for (int i = 0; i <20 ; i++) {  
            System.out.println(i);  
        }  
    }  
  
    public void method1(){  
        String str="Java";  
    }  
}
```

# Scope

## 1- Local variables :

Local variable'lar bir method veya kod bloğu içerisinde oluşturulan variable'lardır.

Local variable'ların scope'u içerisinde oluşturdukları kod blogudur ve o blok içerisinde kullanılabılırler.

Ancak scope'lari disinda kullanılamazlar. Kullanmak isterseniz CTE olusur.

Tum method'larda kullanmak istediginiz variable'lari class level'da oluşturmalisiniz.

```
public static void main(String[] args) {  
  
    int sayi=10;  
  
    System.out.println(str);  
  
    sayi++;  
    System.out.println(sayi);  
}
```

```
public void method1(){  
    String str="Java";  
    str=str.toUpperCase();  
    System.out.println(str);  
    System.out.println(sayi);  
}
```

# Scope

## 1- Local variables :

Local variable'lar deklare edilirken değer atanma mecburiyeti yoktur.

Java variable'i olusturdugumuzu ama değer atamasını ileriki satırlarda yapacağımızı kabul eder ve CTE vermez.

Ancak local variable'lara değer ataması yapmadan kullanmaya kalkışırsanız Java **olmayan değeri** kullanamayacağı için CTE verir.

```
public static void main(String[] args) {  
  
    int sayi;  
  
    sayi++;  
  
}
```

```
public void method1(){  
    String str;  
  
    System.out.println(str);  
}
```



# Scope

## 1- Local variables :

### Loop variables

Bir loop içerisinde olusturulan variable'larin scope'u olusturuldukları loop'tur, yani o loop disindan kullanilamazlar.

```
public static void main(String[] args) {  
  
    for (int i = 0; i < 10 ; i++) {  
        int sayi= 20;  
        System.out.println(i+ sayi);  
    }  
  
    sayi++;  
  
    System.out.println(i);  
}
```

**NOT :** Her ne kadar bir method içerisinde olsa da loop variable'larinin scope'u içinde bulundukları **method degil**, içerisinde olusturuldukları loop'tur.



# Scope / Class Level Variables



Doktor



Hemsire



Labaratuvar ve personel



Hasta



Tibbi cihazlar

# Scope / Class Level Variables

## 2- Class level variables

Class level variables variable'lar instance ve static olmak üzere ikiye ayrılırlar.

Class level'da oluşturulacak bir variable'in static veya instance yapılmasına o variable'in class'dan oluşturulacak objeler ile ilişkisine bakılarak karar verilir.



Hastane adı, hastane adresi, telefonu gibi bilgiler tüm objeler için ortaktır ve her bir obje için ayrı ayrı atama yapılmasına gerek yoktur.

Ancak, personel adı, personel adresi, telefonu veya cihaz no vs.. gibi bilgiler objelere özeldir ve her obje için birbirinden farklı olabilir.



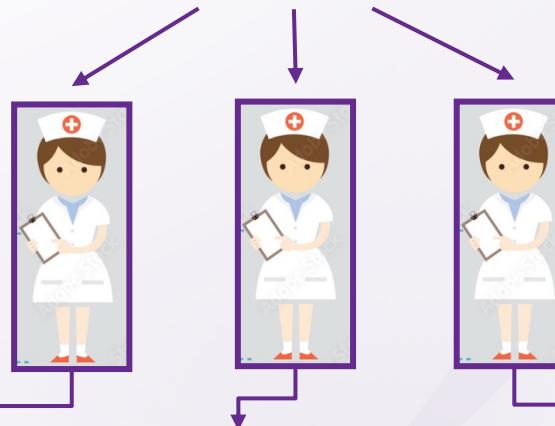
Tibbi cihazlar

# Scope / Class Level Variables

## 2- Class level variables

Yandaki sema incelenirse hangi variable'larin **tum objeler icin ortak oldugu** dolayisiyla **static** olmasi gerektigi

hangi variable'larin ise **tum objeler icin objeye ozel** oldugu, dolayisiyla **static olmamasi** gerektigi asikardir.



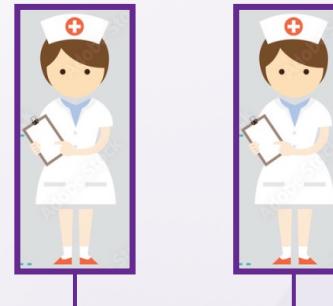
Hemsire1	Hemsire2	Hemsire3
P1.Ismi : Yildiz	P2.Ismi : Ayse	P2.Ismi : Fatma
P1.adres: Cankaya	P2.adres: Sincan	P2.adres: Altindag
P1.Telefon : 4164352	P2.telefon : 6151232	P2.telefon : 7141536

# Scope / Class Level Variables

## 2- Class level variables

Static (**class**) variables : Class'a ait **tek bir variable** olusturulur, tum objeler icin bu variable'in degeri ortaktir.

Instance (**object**) variables : Class level'da olusturulur, ancak her bir object olusturuldugunda Java ilk atanan degerlere sahip yeni bir variable olusturup objeye baglar. Dolayisiyla Java instance bir variable icin **obje sayisinda kopya variable'lar** olusturur.



H.Ismi : Yildiz	H.adres: Ankara/Cankaya	H.Telefon : 2445566
Hemsire2 P2.Ismi : Ayse P2.adres: Sincan P2.telefon : 6151232	Hemsire3 P2.Ismi : Fatma P2.adres: Altindag P2.telefon : 7141536	

# Scope / Class Level Variables

## 2- Class level variables

Kural 1 : Static veya instance variable'lara deger atama mecburiyeti yoktur.

Biz deger atamasi yaparsak, o degeri kullanir. Deger atamazsak Java variable'lar icin default olarak tanimlanan degerleri assign eder ve onlari kullanir.

Default degerler :

String : null

Sayisal primitive'ler : 0

Char : hiclik

Boolean : false

```
static String hastaneIsim;
static String hastaneTel="03123454354";
String persIsim;
String persSoyisim="Soyisim belirtildi";
boolean izindeMi;
int yas;

public static void main(String[] args) {

    System.out.println(hastaneIsim); // null
    System.out.println(hastaneTel); // 03123454354

    Scope per1=new Scope();
    System.out.println(per1.persIsim); // null
    System.out.println(per1.persSoyisim); // Soyisim belirtildi
    System.out.println(per1.izindeMi); // false
    System.out.println(per1.yas); // 0
}
```

# Scope / Class Level Variables

**Kural 2 :** Static variable'lar, static olduklari icin tum class'dan direk kullanilabilirler,  
(tum static method'lар ve static olmayan method'lardan)

Instance variable'lar static olmadiklari icin static method'lardan direk kullanilamazlar.

Instance variable'lara static method'lardan ulasmak ve/veya kullanmak icin obje olusturmamiz gerekir.

Instance variable'lari static olmayan method'lardan direk kullanabiliriz.

```
static String hastaneIsim;
static String hastaneTel="03123454354";
String persIsim;
String persSoyisim="Soyisim belirtildi";
boolean izindeMi;
int yas;

public static void main(String[] args) {

    System.out.println(hastaneIsim); // null
    System.out.println(hastaneTel); // 03123454354

    Scope per1=new Scope();
    System.out.println(per1.persIsim); // null
    System.out.println(per1.persSoyisim); // Soyisim belirtildi
    System.out.println(per1.izindeMi); // false
    System.out.println(per1.yas); // 0
}
```

# Scope / Class Level Variables

## 2- Class level variables

**Kural 3 :** Static variable'lara, baska class'lardan erismek icin **classAdi.staticVariableAdi** yazmamiz yeterlidir.

Instance variable'lara baska class'dan ulasmak ve/veya kullanmak icin obje olusturmamiz gereklidir.

```
public class Scope {  
  
    static String hastaneIsim;  
    static String hastaneTel="03123454354";  
    String persIsim;  
    String persSoyisim="Soyisim belirtildi";  
    boolean izindeMi;  
    int yas;
```

```
public class Runner {  
    public static void main(String[] args) {  
  
        System.out.println(Scope.hastaneIsim); // null  
        System.out.println(Scope.hastaneTel); // 03123454354  
  
        Scope per1=new Scope();  
        System.out.println(per1.persIsim); // null  
        System.out.println(per1.persSoyisim); // Soyisim belirtildi  
        System.out.println(per1.izindeMi); // false  
        System.out.println(per1.yas); // 0  
    }  
}
```



Scope : Class icerisinde olusturulan variable'larin kapsamini (nereden erisebilecegini) belirler

Temel olarak 4 Scope'dan bahsedebiliriz

Class Level'da olusturulan variable'lar class'in tamaminda gecerlidir, ancak direk erisim icin static keyword belirleyicidir

1- static olarak tanimlanan variable'lara tum method'lardan ulasilabilir

2- static olarak tanimlanmayan (instance) variable'lara sadece static olmayan method'lardan ulasilabilir

Local olarak olusturulan variable'lar sadece tanimlandiklari scope'da gecerlidirler.  
(Herkes oturdugu mahallede taninir)

3- bir method'da olusturulan variable'lara sadece o method'dan ulasilabilir

4- Loop icerisinde olusturulan variable'a loop disindan erisilemez

```
2 ► public class ScopeNedir {  
3 →     static int sayi=5;  
4 →     String ders="Java";  
5 ►     public static void main(String[] args) {  
6         sayi=100;  
7         ders="Java Course";  
8         int mainsayi=20;  
9         ders2="API";  
10        for (int i = 0; i < 10; i++) {  
11            System.out.println(i);  
12            String ders3="SQL";  
13        }  
14        System.out.println(i);  
15        ders3="API";  
16    }  
17    public static void staticMethod(){  
18        sayi=110;  
19        System.out.println(ders);  
20        mainSayi=10;  
21        System.out.println(ders2);  
22    }  
23    public void staticOlmayanMethod(){  
24        System.out.println(sayi);  
25        ders="Java Course";  
26        System.out.println(mainSayi);  
27        String ders2="Selenium";  
28    }
```



# Scope / Ozet

Soru : Yandaki class için aşağıdaki soruları yanıtlayın.

1- hangi satırlarda local variable'lar vardır ?

2- class level'da oluşturulan variable'ların scope'ları ve değerleri nelerdir ?

3- Hangi satırlarda CTE vardır ve düzeltmesi gereklidir ?

```
3 public class Scope {  
4  
5     static String str;  
6     String tel="03123454354";  
7  
8     public static void main(String[] args) {  
9  
10        Scope obj=new Scope();  
11        System.out.println(tel);  
12        obj.str="Java ne guzel";  
13        int sayi=15;  
14        method2(sayi);  
15        method1();  
16    }  
17  
18    public void method1(){  
19        tel="03124324343";  
20        String isim= "John Doe";  
21        boolean dogruMu;  
22        int sayi;  
23    }  
24  
25    public static void method2(int sayi){  
26        str=str+".";  
27        tel=tel.substring(1);  
28        int sayi=10;  
29    }  
30}
```



Wise Quarter  
first class IT courses

# Arrays