



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)

# Bugun IT Dunyasinda Ilk Gununuz

## Onemli Hatirlatmalar

- 1- Ders Tam Vaktinde Baslar
- 2- Ders Oncesi Hazirlik Yapin  
(Google en yakin arkadasiniz olmalı)
- 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 gosterin
- 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 istediğini 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.

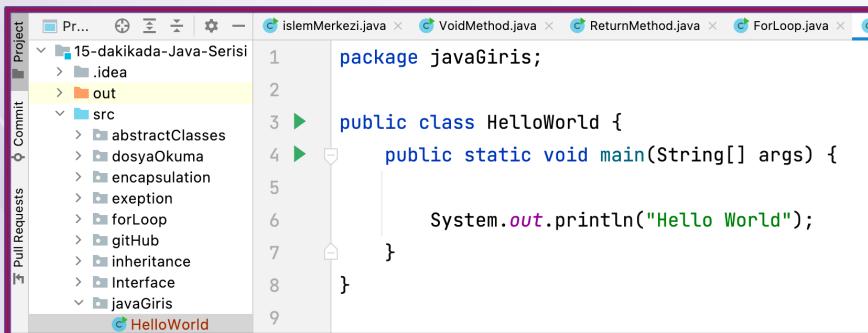
Kisaca ozetlersek, programlama dili bizimle bilgisayar arasindaki iletisimi saglayan dildir.

Peki..Bizimle bilgisayar arasindaki islemlerde patron kim ?

Kimin dedigi olur ?

Tabii ki bilgisayarlar bizim dedigimizi yaparlar

Ama bu istegimizi bilgisayarin anlayacagi ozelliklerde yazmamiz sartiyla.



```
Pr... Project 15-dakikada-Java-Serisi islemMerkezi.java VoidMethod.java ReturnMethod.java ForLoop.java
  > .idea
  > out
  > src
    > abstractClasses
    > dosyaOkuma
    > encapsulation
    > exception
    > forLoop
    > gitHub
    > inheritance
    > Interface
    > javaGiris
      > HelloWorld.java
      > HelloWorld.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 }
```

Ornegin;  
kodlarimizla, Hello World yazdirmak  
istiyororsak

```
110101010101010010100010101010
100101111000001000100101101010
101010010111000110
010001010010101110001000000101
0010101110001001010111
```

JDK Kodlari derler (Compile)



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.

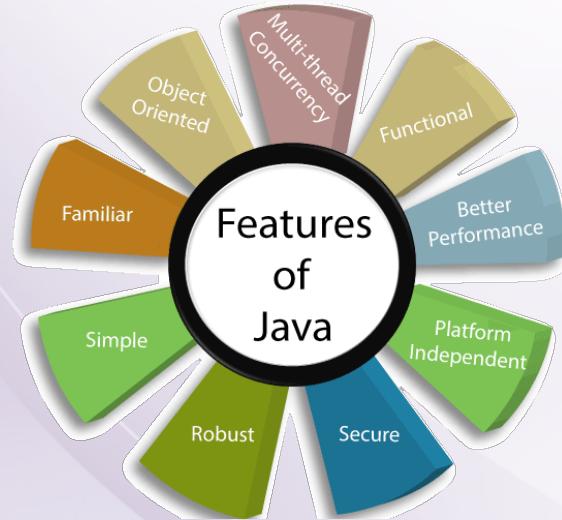
USA'de şirket bilgisayarlarının %97'sinde, kişisel bilgisayarların %89'unda Java kullanılıyor

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



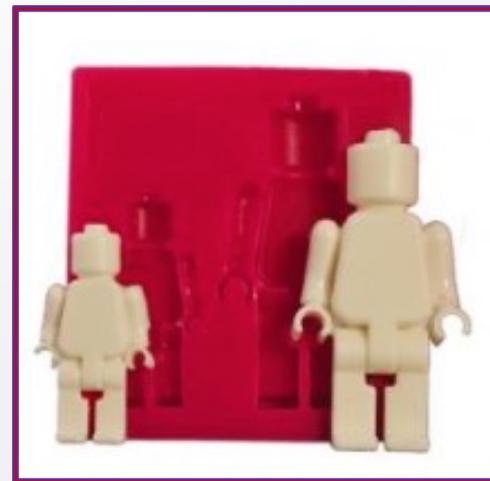
Java'da, once Object (Nesne) olusturmalıyız.

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.

# Class Body'sinde Neler Bulunur?

```
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

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

# Main Method Nedir ?

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 }
```

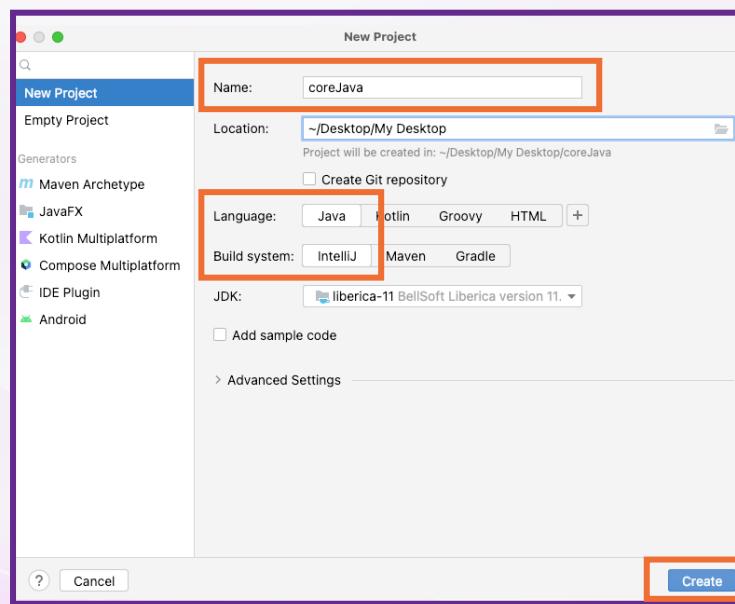
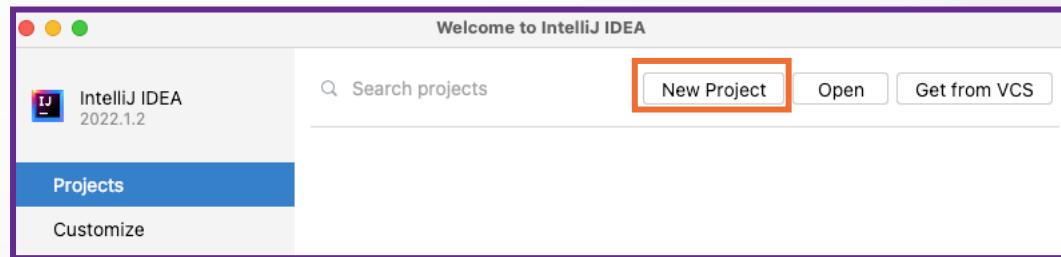
# IntelliJ'de Proje Olusturma

## 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

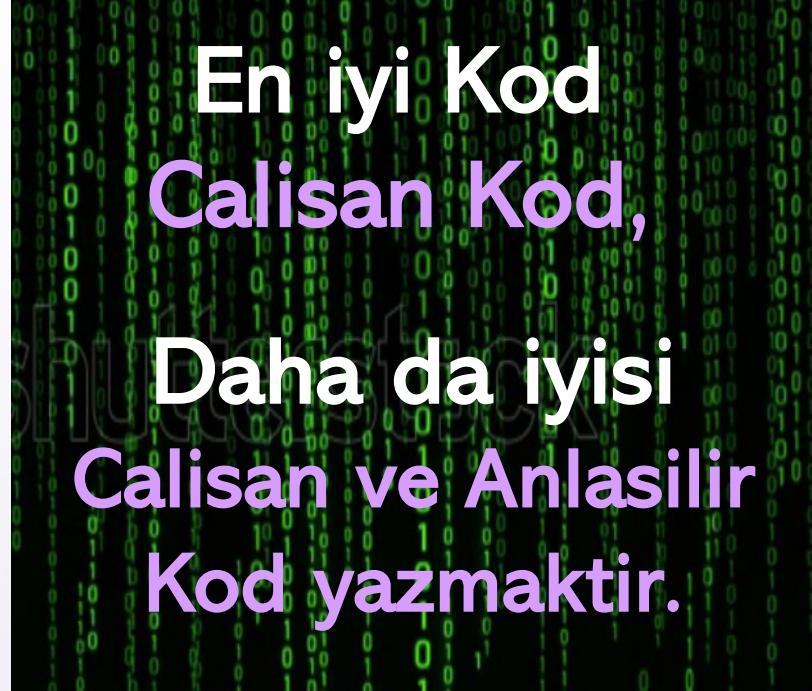
# Kod'a Comment Ekleme

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.

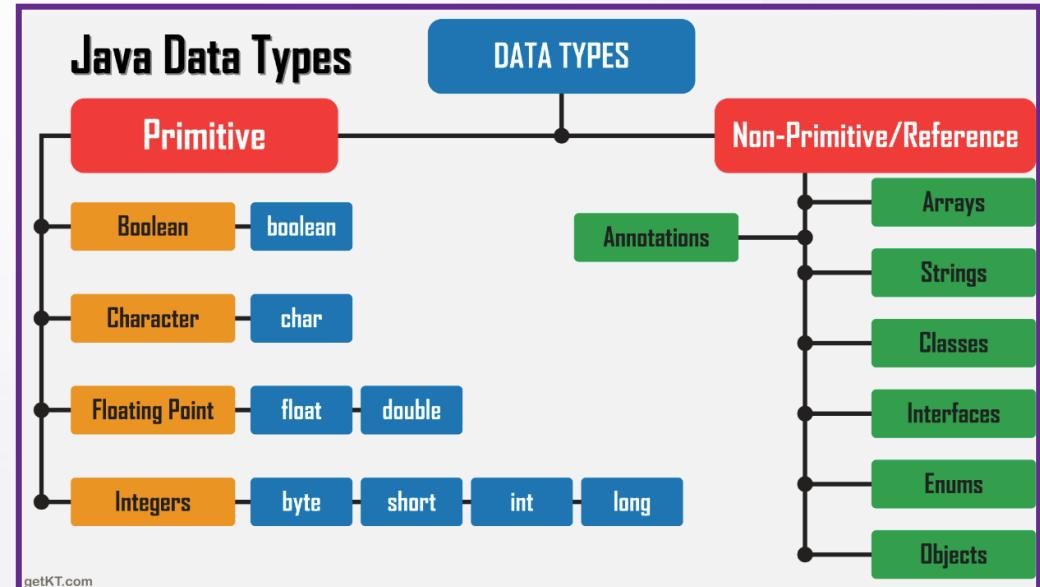
```
10 */  
11 Birden fazla  
12 satiri  
13 comment yapmak icin  
14 */
```

# 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

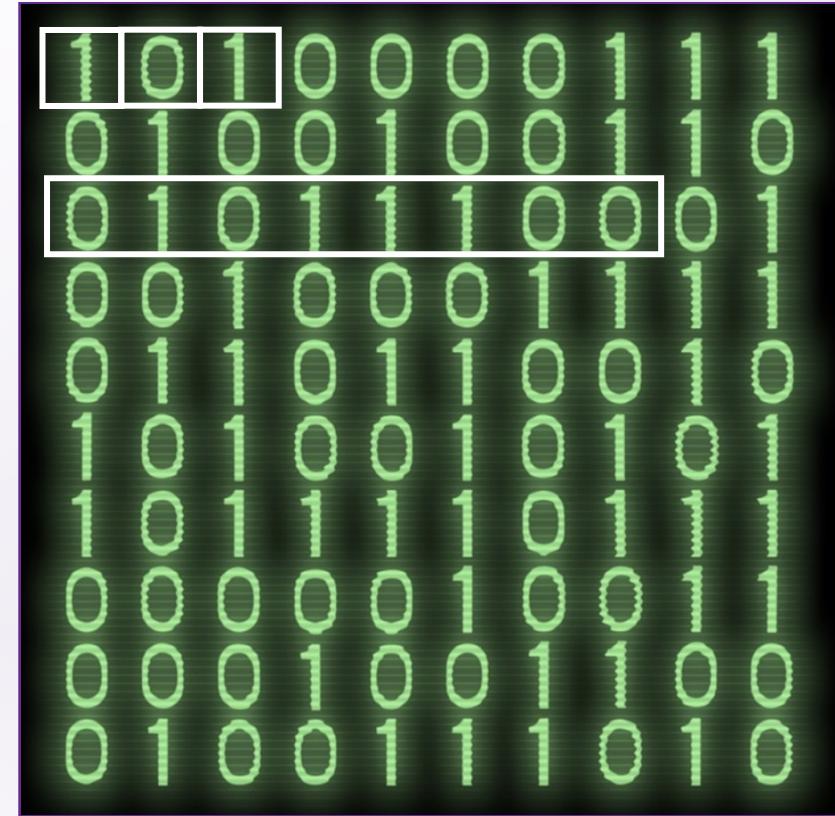
$2^{10} = 1024$ 'un katlari seklinde yapılandirilmistir.

1024 byte = 1 KB

1024 KB = 1 MB

1024 MB = 1 GB

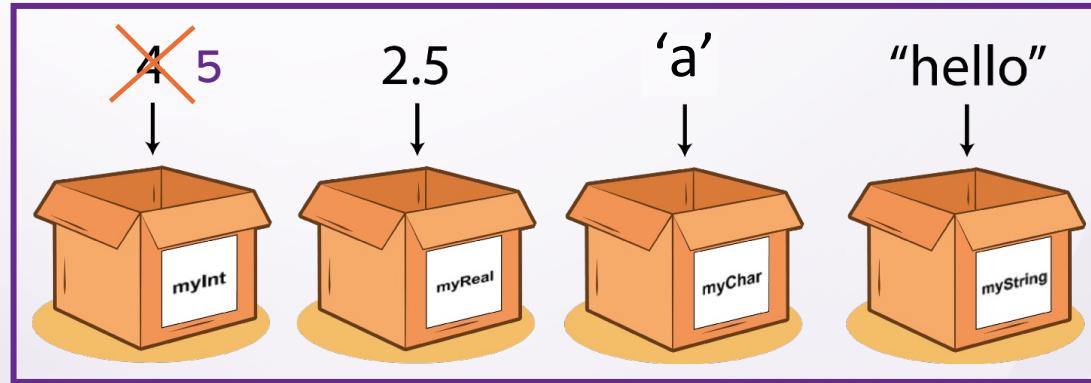
1024 GB = 1 TB ....



# 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.

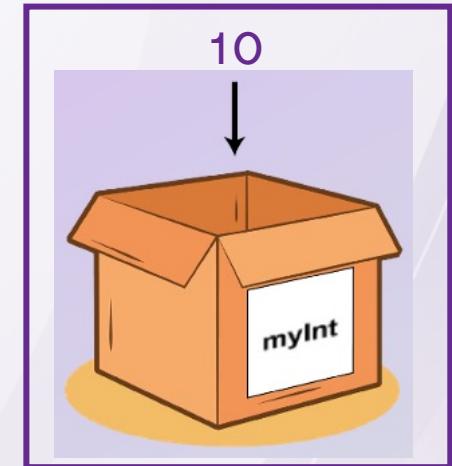
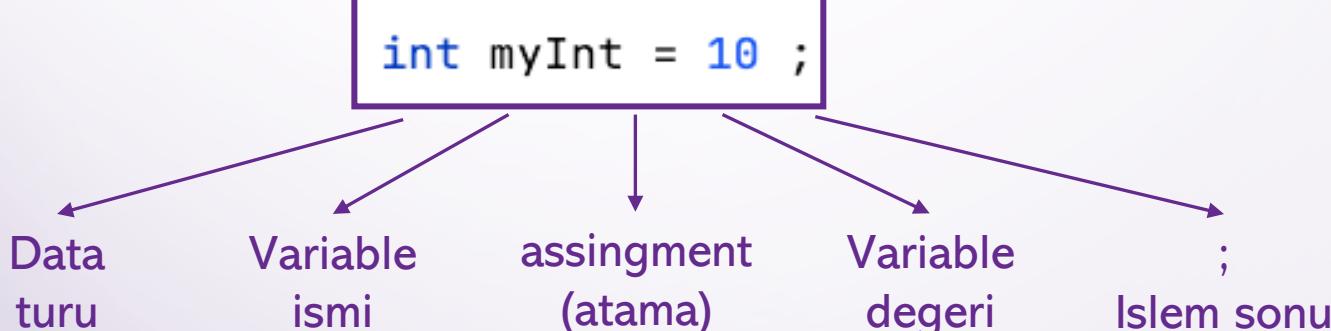
Ornegin myInt'i bir artır dedigimizde, Java myInt ismindeki variable'i bulur, icindeki değer olan 4'u bir artırıp 5 yapar.

Bu islemden sonra myInt variable'inin 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.



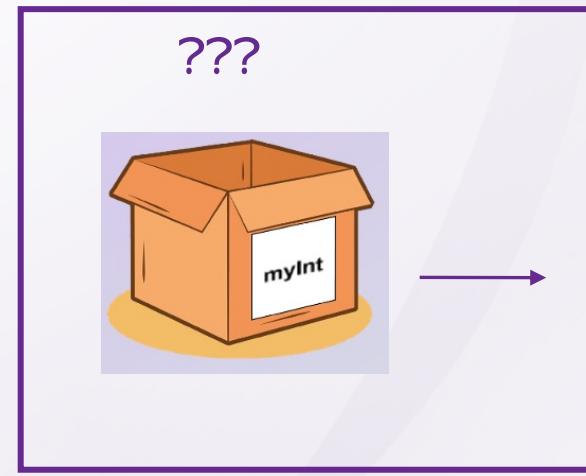
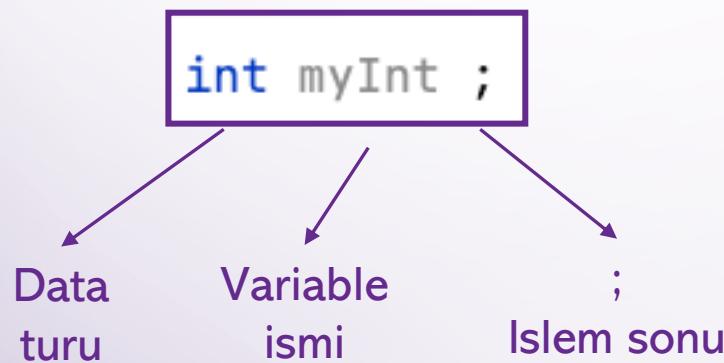
Bir variable olusturmak icin 2 islem vardir;

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.

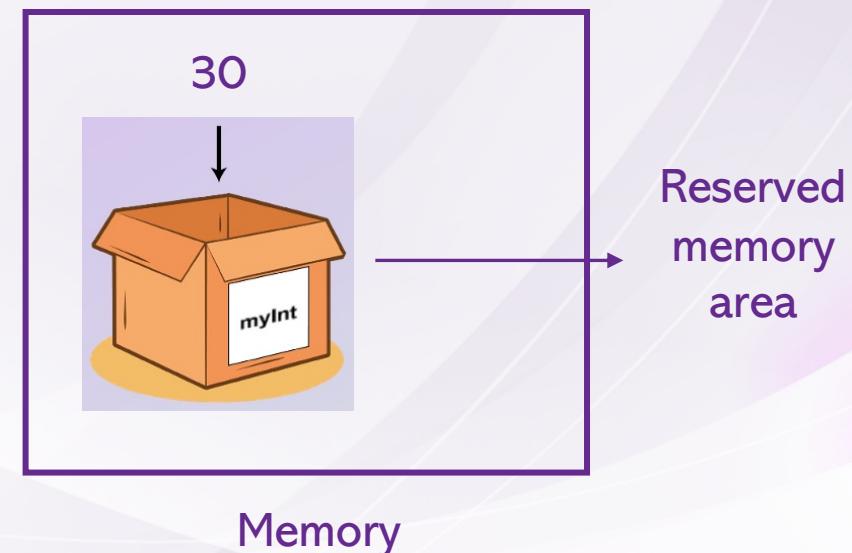
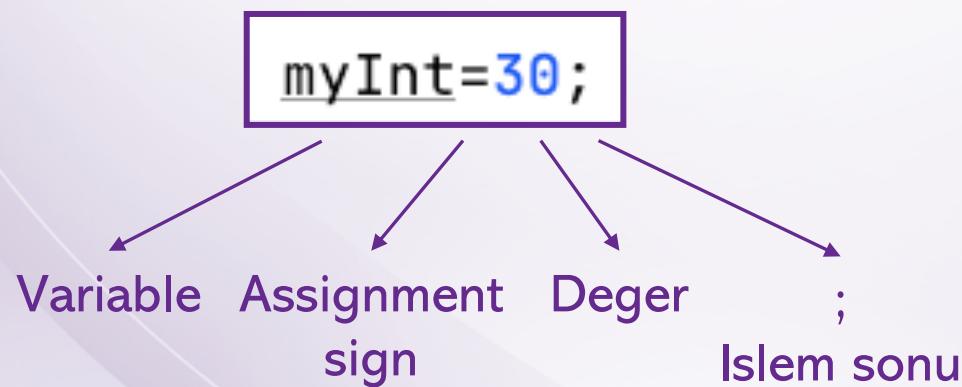
Java'da declaration ve assignment farklı satırlarda yapılabilir.

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)



The future at your fingertips

# 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;
```

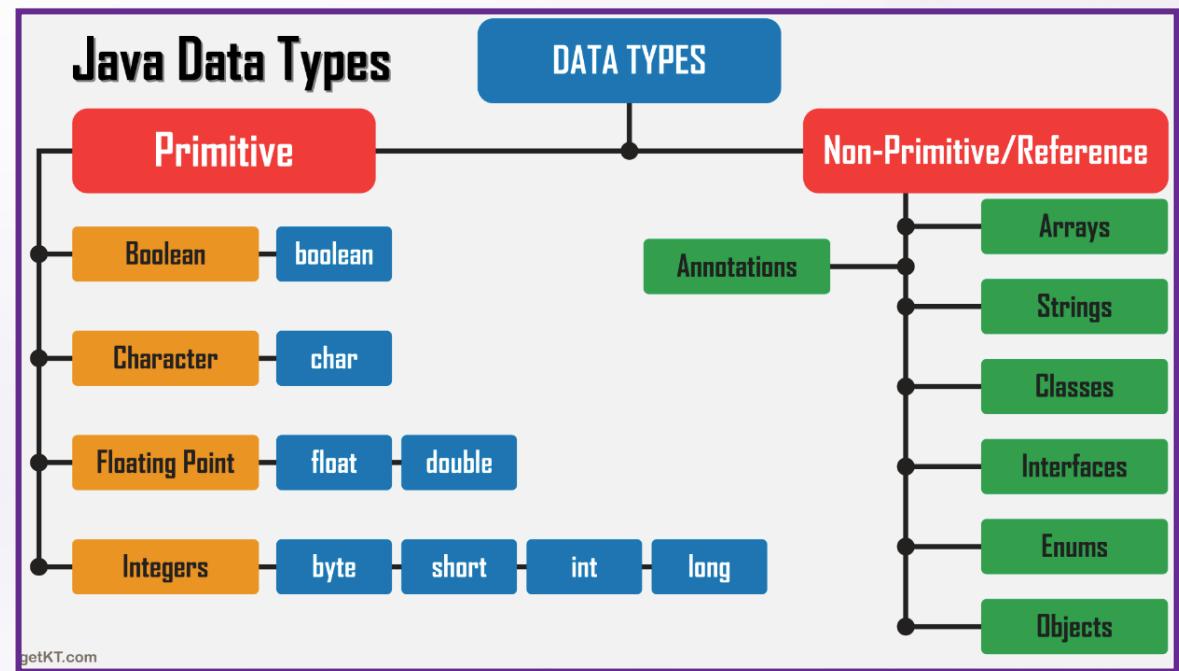
4- Ayni data turunde birden fazla variable tek declaration ile olusturulabilir.

```
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.

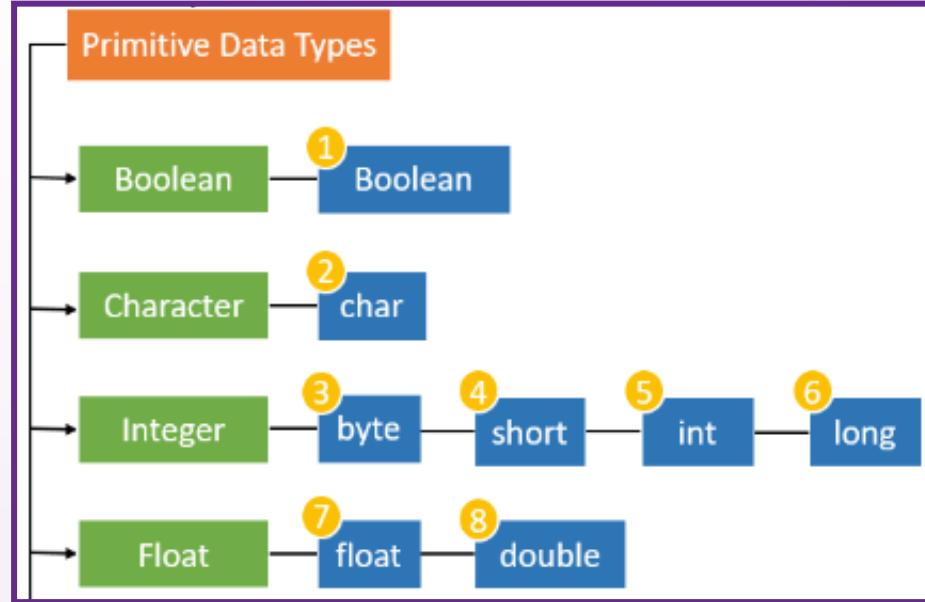
- 1- Boolean : Mantıksal data sonuçlarını store etmek için kullanılır.

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 Turleri

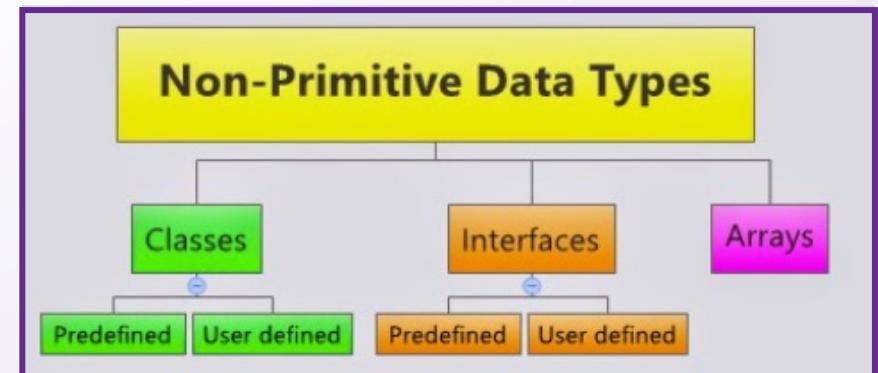
Primitive data turleri Java tarafından olusturulmustur ve biz yeni bir PRIMITIVE DATA TURU olusturamayiz.

Non-Primitive data turleri ise Java tarafından sinirlandirilmamislardir. Java da bazi non-primitive data turleri olusturulmustur, biz de yeni non-primitive data turleri olusturabiliriz.

Non-Primitive data turlerinin geneli icin **object** tabiri kullanilabilir, cunku tum non-primitive data turlerinin kendilerine ait bir class tarafından olusturulan objelerdir.

Class'lar OOP konsept cercevesinde bizim obje olusturdugumuz, kaliplar oldugundan, non-primitive data turleri bu class'lardan olusturulan objelerdir.

Class'lar method'lar da icerdiginden, non-primitive data turleri, olusturulduklari 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
Tumu Java tarafindan olusturulmustur.	Java tarafindan olusturulanlar oldugu gibi biz de olusturabiliriz
Sadece deger icerirler, variable ile kullanilacak hazir method'lari yoktur.	Icerdikleri degerin yaninda olusturulduklari class'dan gelen hazir method'lar da barindirirlar.
Bir deger atamadan olusturulabilir ama kullanabilmek icin mutlaka deger atanmalidir.	Deger atanmadan <b>null</b> olarak isaretlenebilirler.
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'larin hafizada kapladiklari alan sabittir. Degeri kucuk de olsa, buyuk de olsa hafizada belirlenen miktarda alan ayrilir.	Hafizada kapladiklari alan sabit degildir. Data turu ve icerdigi datanin buyuklugune gore hafizada yer kaplarlar. (bir kelime veya binlerce kelime iceren String'lerin boyutlari farkli olacaktir)

# Naming Convention (Isim Verme Kurallari)

Variable'lara isim verirken istedigimiz ismi secebilmiziz 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.

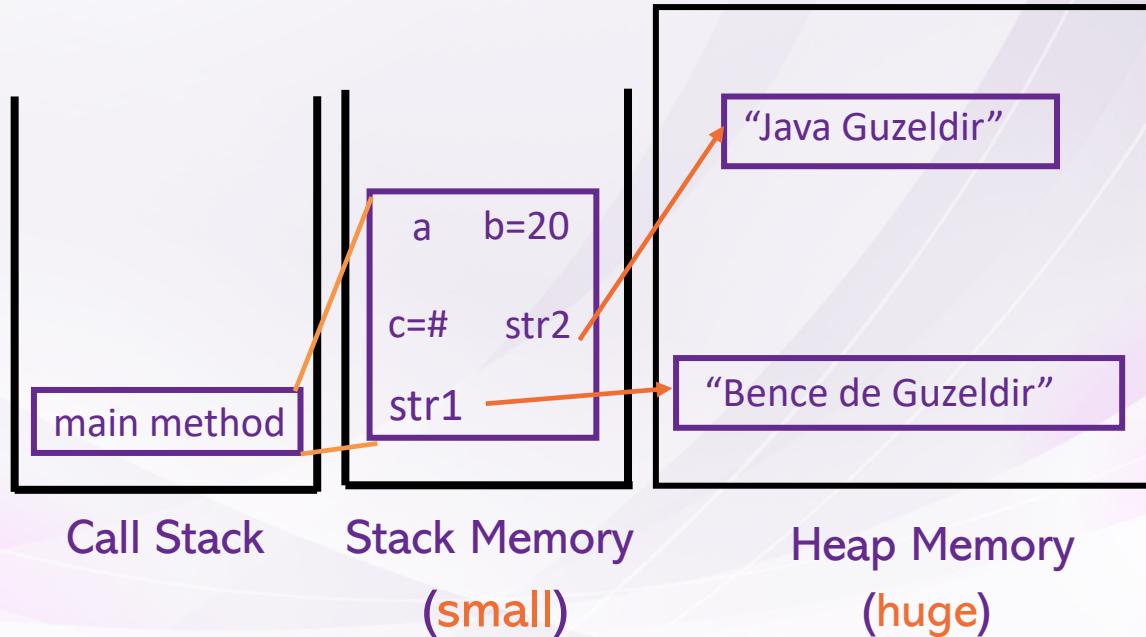
# Hafiza(Memory) Kullanimi

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).

# 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 turundeki dataları birbirine kolayca çevirebilirken, bazı casting işlemleri için ekstra kod yazmamız gereklidir, bazı 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 türleri için data casting yapmak mümkün ancak biz simdilik primitive data türleri için data casting konusunu irdeleyelim.

# Implicit Data Casting (Auto-Widening)

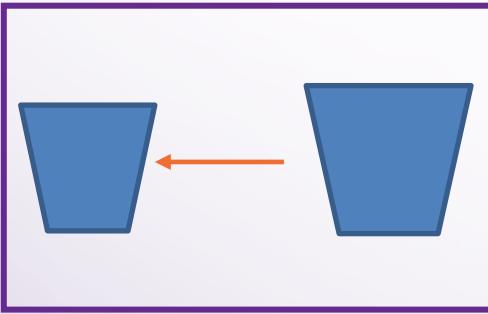
Daha kucuk kapsamlı bir data turundeki degeri, daha genis kapsamlı data turundeki variable'a atama yapmak istedigimizde, Java bu islemi otomatik olarak yapacaktır



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

# Explicit Data Casting

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



```
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 kapsamlı olduğundan deger dar kapsamlı variable'in sınırları içinde olabileceği gibi, sınırlarından büyük de olabilir.

Bu durumda Java, data kaybi veya degisikligi ihtimalinin farkında olduğumuzun bilmek ister.

Sorumluluğu almak için cast etmek istedigimiz degerin onune (cast etmek istedigimiz data turunu) yazarsak, java bu casting'i data turu sınırlarına göre 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
01	SOH (Start of Header)	33	!	65	A	97	a	129	Ü	161	í
02	STX (Start of Text)	34	"	66	B	98	b	130	é	162	ó
03	ETX (End of Text)	35	#	67	C	99	c	131	â	163	ú
04	EOT (End of Trans.)	36	\$	68	D	100	d	132	ä	164	ñ
05	ENQ (Enquiry)	37	%	69	E	101	e	133	à	165	Ñ
06	ACK (Acknowledgement)	38	&	70	F	102	f	134	â	166	º
07	BEL (Bell)	39	'	71	G	103	g	135	ç	167	º
08	BS (Backspace)	40	(	72	H	104	h	136	ê	168	¸
09	HT (Horizontal Tab)	41	)	73	I	105	i	137	ë	169	®
10	LF (Line feed)	42	*	74	J	106	j	138	è	170	™
11	VT (Vertical Tab)	43	+	75	K	107	k	139	í	171	½
12	FF (Form feed)	44	,	76	L	108	l	140	î	172	¼
13	CR (Carriage return)	45	-	77	M	109	m	141	í	173	i
14	SO (Shift Out)	46	.	78	N	110	n	142	Ã	174	«
15	SI (Shift In)	47	/	79	O	111	o	143	À	175	»
16	DLE (Data link escape)	48	0	80	P	112	p	144	É	176	„
17	DC1 (Device control 1)	49	1	81	Q	113	q	145	æ	177	„
18	DC2 (Device control 2)	50	2	82	R	114	r	146	Œ	178	„
19	DC3 (Device control 3)	51	3	83	S	115	s	147	ô	179	„
20	DC4 (Device control 4)	52	4	84	T	116	t	148	ö	180	„
21	NAK (Negative acknowl.)	53	5	85	U	117	u	149	ò	181	À
22	SYN (Synchronous idle)	54	6	86	V	118	v	150	û	182	Ã
23	ETB (End of trans. block)	55	7	87	W	119	w	151	ú	183	À
24	CAN (Cancel)	56	8	88	X	120	x	152	ÿ	184	©
25	EM (End of medium)	57	9	89	Y	121	y	153	Ó	185	„
26	SUB (Substitute)	58	:	90	Z	122	z	154	Ü	186	„
27	ESC (Escape)	59	;	91	[	123	{	155	ø	187	„
28	FS (File separator)	60	<	92	\	124		156	£	188	„
29	GS (Group separator)	61	=	93	]	125	}	157	Ø	189	¢
30	RS (Record separator)	62	>	94	^	126	~	158	×	190	¥
31	US (Unit separator)	63	?	95	_			159	f	191	„
127	DEL (Delete)							192	„	193	„

```
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
```

# Sorular (Data Casting)

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

**Soru 2-** Kullanicidan bir harf alin ve alfabede 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.

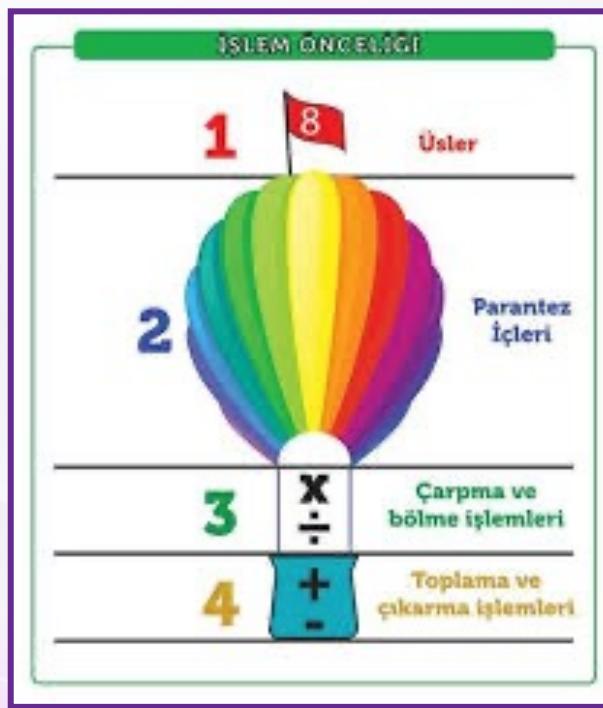
**Soru 5-** Kullanicidan bir double, bir tamsayi alin, double sayiyi ikinci sayiya bolun ve bolum isleminin sonucununun tamsayi kismini yazdirin.

# Java'da Matematiksel İşlemler

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

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

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

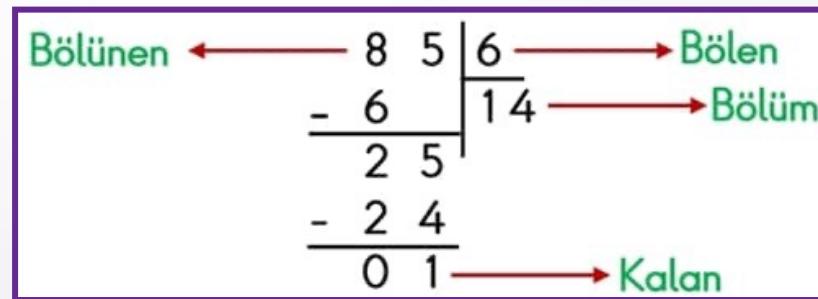


$$24 / 6 * 2 - 7 * 4 + 9$$

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

# 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 sayilari 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 ; → 13
```

```
int sayi = 10 ;  
  
sayi *= 3 ; → 30
```

```
int sayi = 10 ;  
  
sayi++ ; → 11
```

# 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++); ;  
  
System.out.println(sayi);
```

Once sayiyi yazdirir (10) , sonra degeri artirir (11)  
Ust satirda deger (11) oldu, (11) yazdirir.

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

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

Once sayiyi artirir (11) , sonra yazdirir (11)  
Ust satirda deger (11) oldu, (11) yazdirir.

# 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)  
(10)

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

Hello23  
5Hello  
Hello5  
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 (Mantiksal) Operators

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

Mantiktaki AND operatorunun Java'da 2 tane karsiligi vardir. Islevleri ayni olmakla birlikte ic isleyisi ve hiz acisindan aralarinda kucuk bir fark vardir.

`&&` operatoru birlestirdigi 2 boolean ifadenin ikisi de true ise sonucu true yapar, bunun disindaki tum durumlarda sonucu false yapar. (`&&` operatoru mukemmeliyetcidir.)

```
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` olmasi icin  
tum carpilan sayilar `1` olmalidir.  
`1` tane bile sifir 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");
}
```

# 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

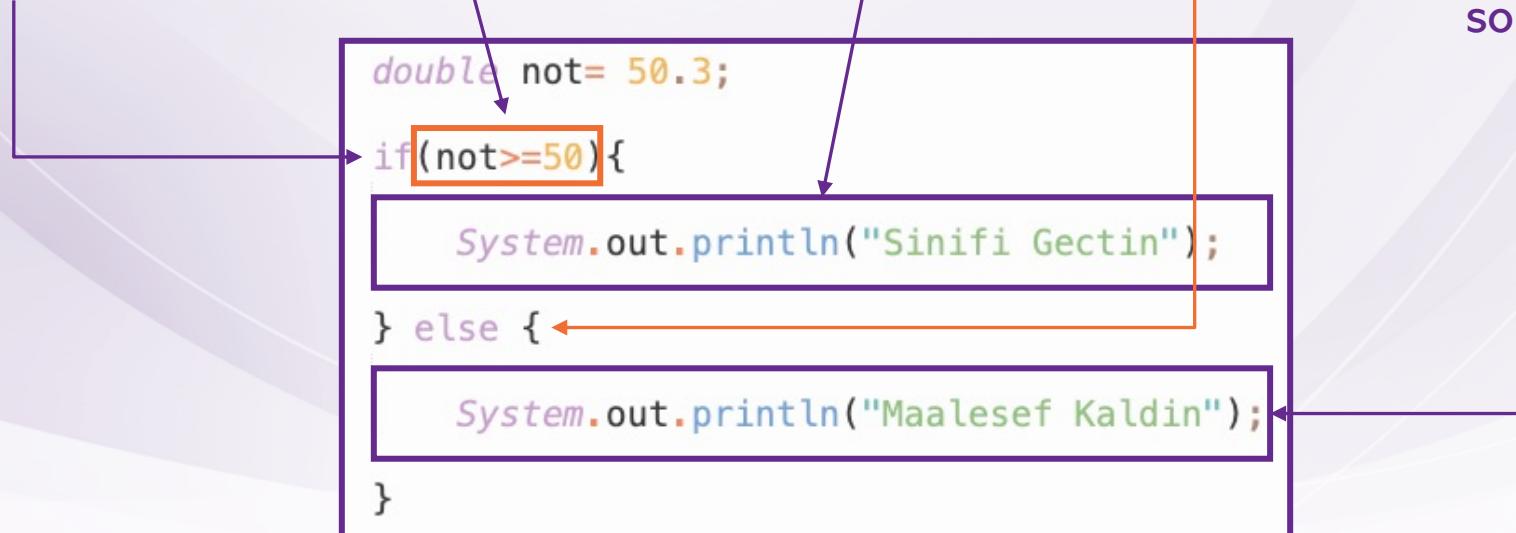
Basit if cümleleri kodun geri kalani ile ilgilenmiyor.

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 olabilirsın” 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");  
}
```

# 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

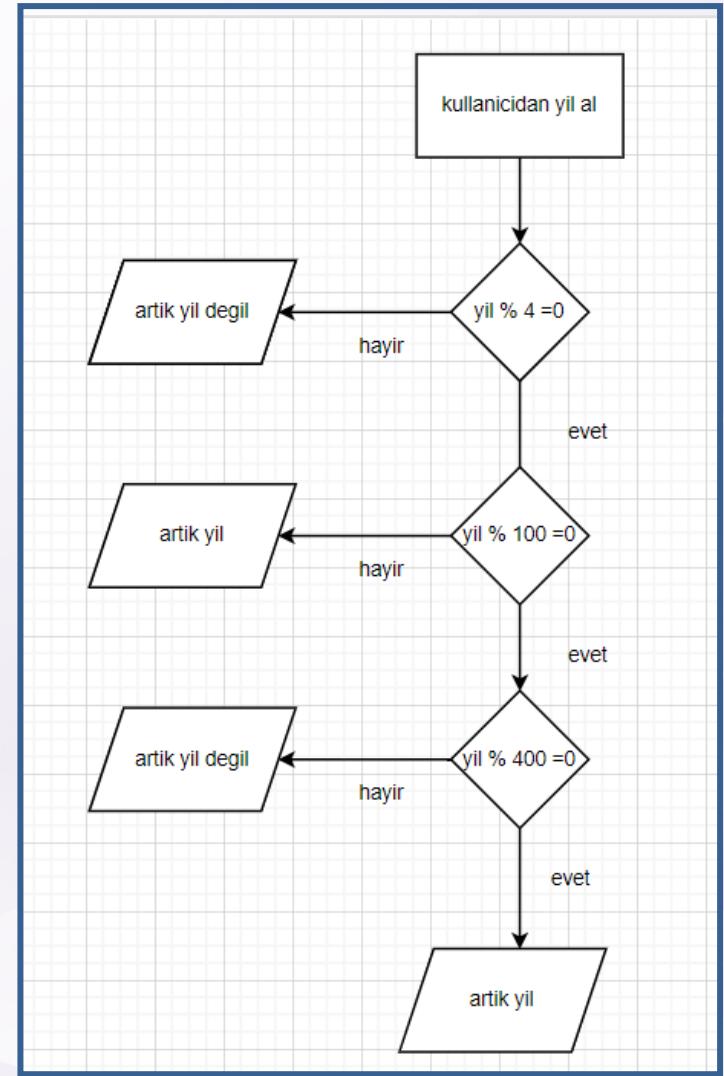
Kullanicidan artik yil olup olmadigini kontrol etmek icin yil girmesini isteyin.

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

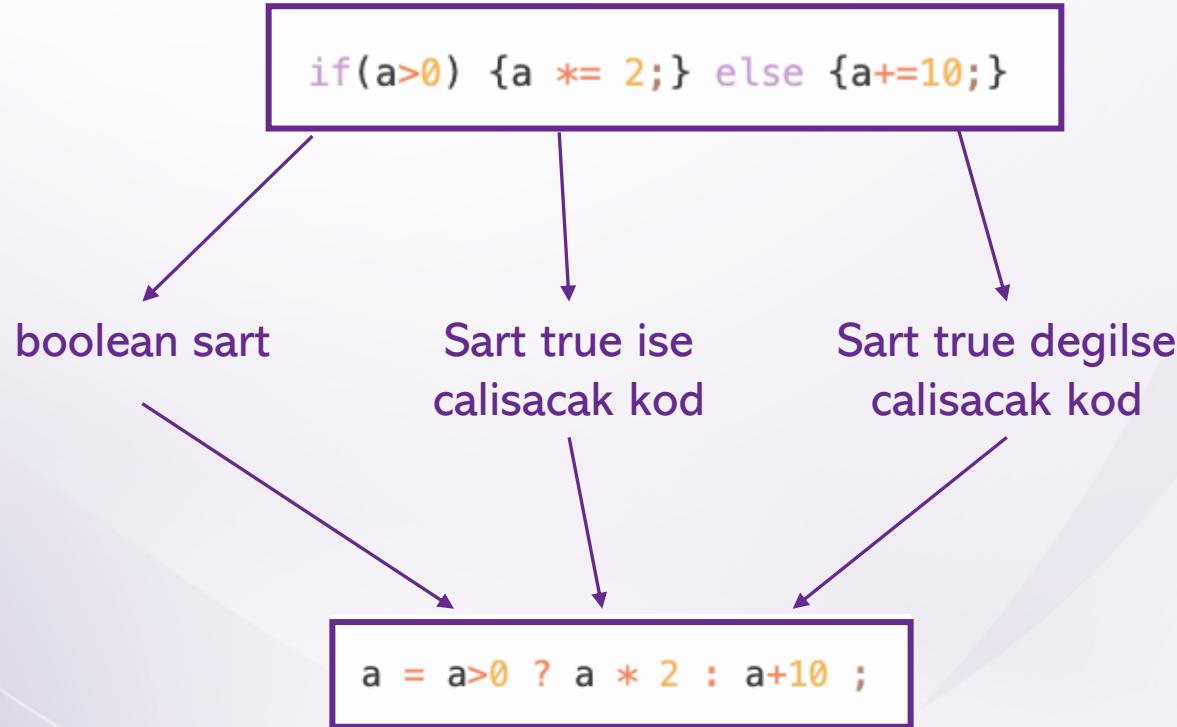
}else {
    // cinsiyet bilgisi hatali
```

# 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, kullaniciya 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 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.

# 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 basamakli veya 3 basamakli 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 basamaklı" : "3 basamaklı değil"
```

# 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);
```



Wise Quarter  
first class IT courses

# Switch Statements