

# FirstBrick Java Dersleri

Ders - 01

Programlamaya Giriş

## Programlama Dili Nedir?

Programlama dili, yazılımcının bir **algoritmayı** ifade etmek amacıyla, bir **bilgisayara** ne yapmasını istedğini 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.

# Programlama Dili Nedir?

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 sonuclari anlamlandıramamız gereklidir.

# Programlama Dili Nedir?

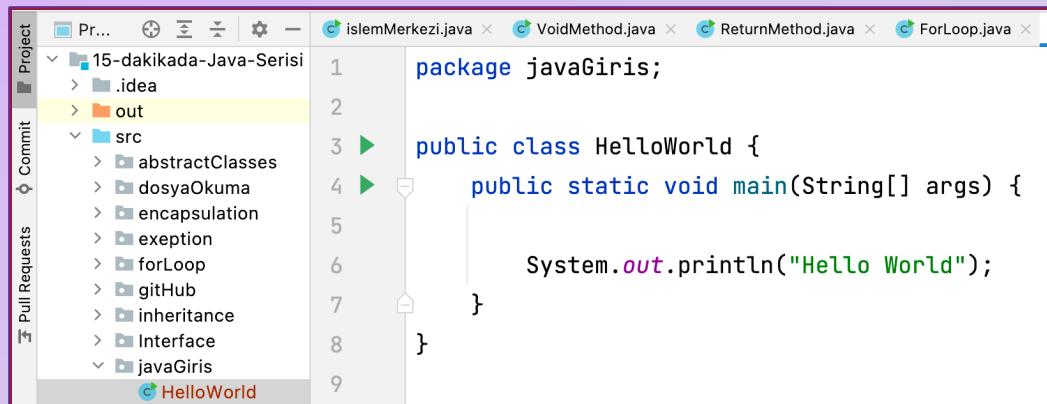
Kisaca ozetlersek, programlama dili bizimle bilgisayar arasindaki iletisimi saglayan dildir.

Peki..Bizimle bilgisayar arasindaki islemlerde patron kim ?

Kimin dedigi olur ?

Tabii ki bilgisayarlar bizim dedigimizi yaparlar

Ama bu istegimizi bilgisayarin anlayacagi ozelliklerde yazmamiz sartiyyla.



The screenshot shows a Java code editor with the following code:

```
1 package javaGiris;
2
3 public class HelloWorld {
4     public static void main(String[] args) {
5         System.out.println("Hello World");
6     }
7 }
```

Ornegin kodlarimizla, Hello World yazdirmak istiyorsak

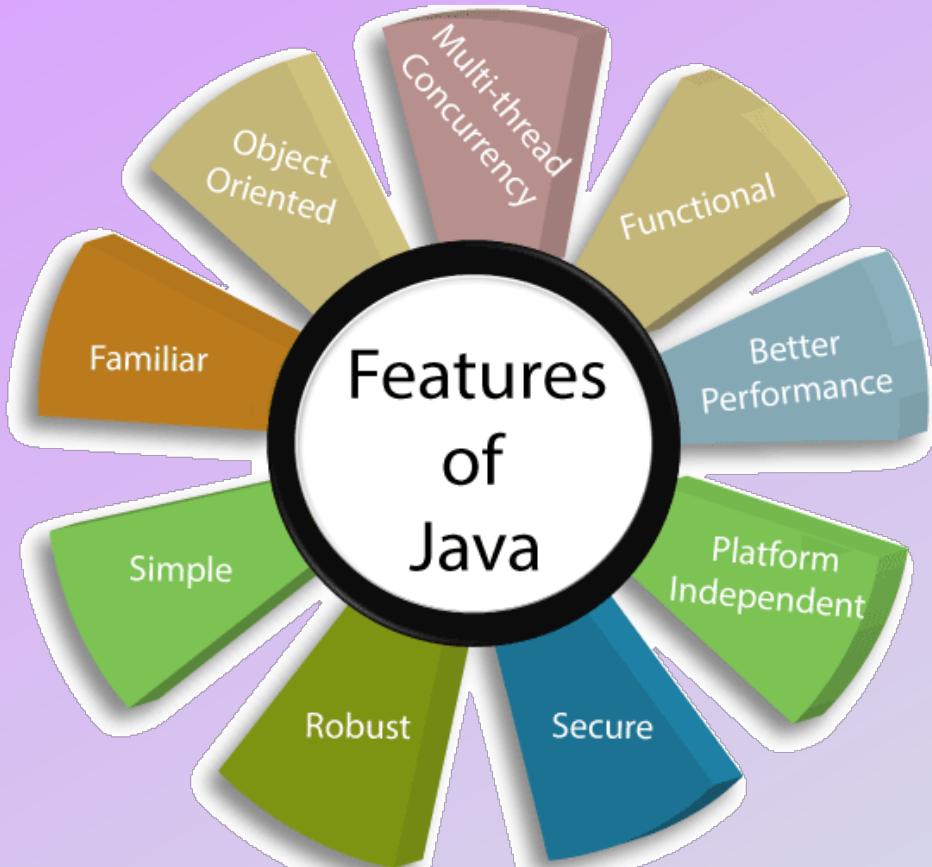
1101010101010010100010101010  
10010111000001000100101101010  
101010010111000110  
01000101001010110001000000101  
0010101110001001010111



JDK Kodlari derler (Compile)

Binary kodlar

Islem



Neden Java

<https://youtu.be/RUYASEnT6pU>

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

Java'yı 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.



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) olusturmaliyiz.

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 Nasil Olusturulur ?



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( Sınıf )'dır.

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

Objeleri olusturabilmek icin oncelikle kalip olusturmaliyiz.



# 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

# 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

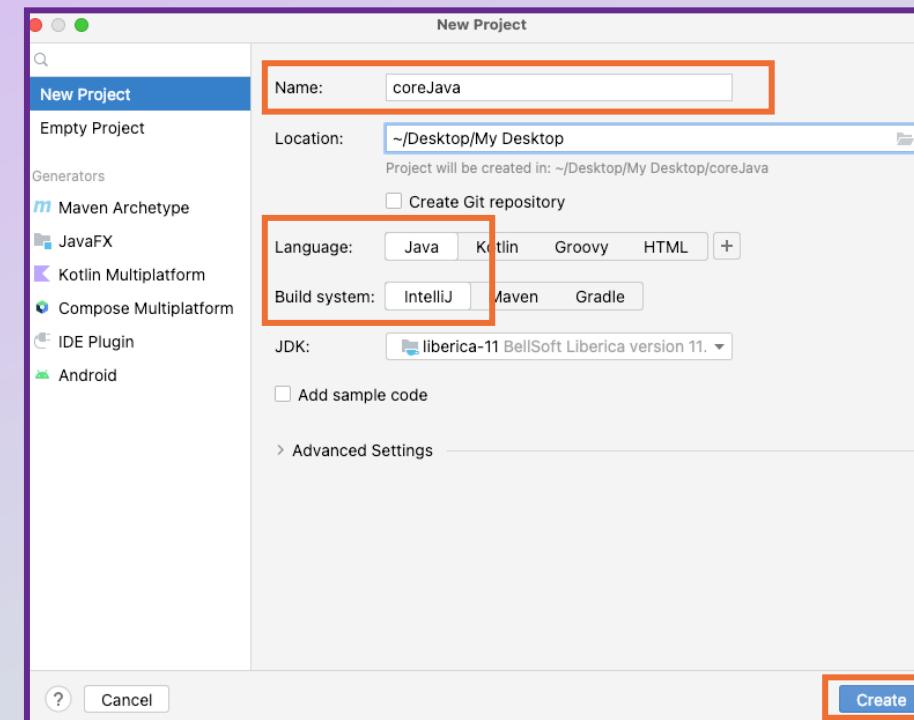
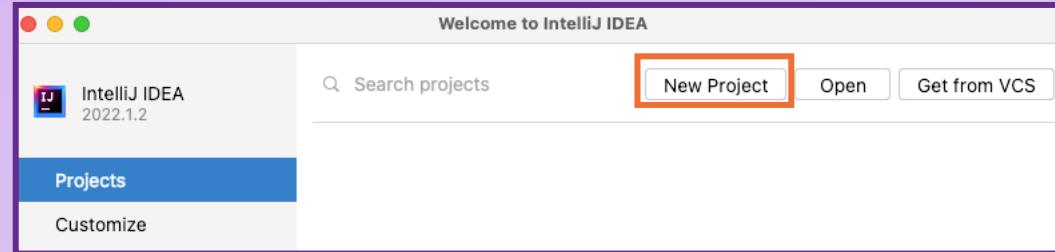
## IntelliJ Nedir?

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

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

## 1- Proje olusturma

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.

## 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

## Class'a Kod Olmayan Yazi (Comment) Ekleme

Kod yazarken ilk hedef calisan bir kod yazmaktir.

Ama asil hedef calisan ve Anlasilabilir kod yazmaktir.

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
```

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

# FirstBrick Java Dersleri

## Ders - 02

Data Nedir?  
Java'da Data Türleri

# Data Nedir ?

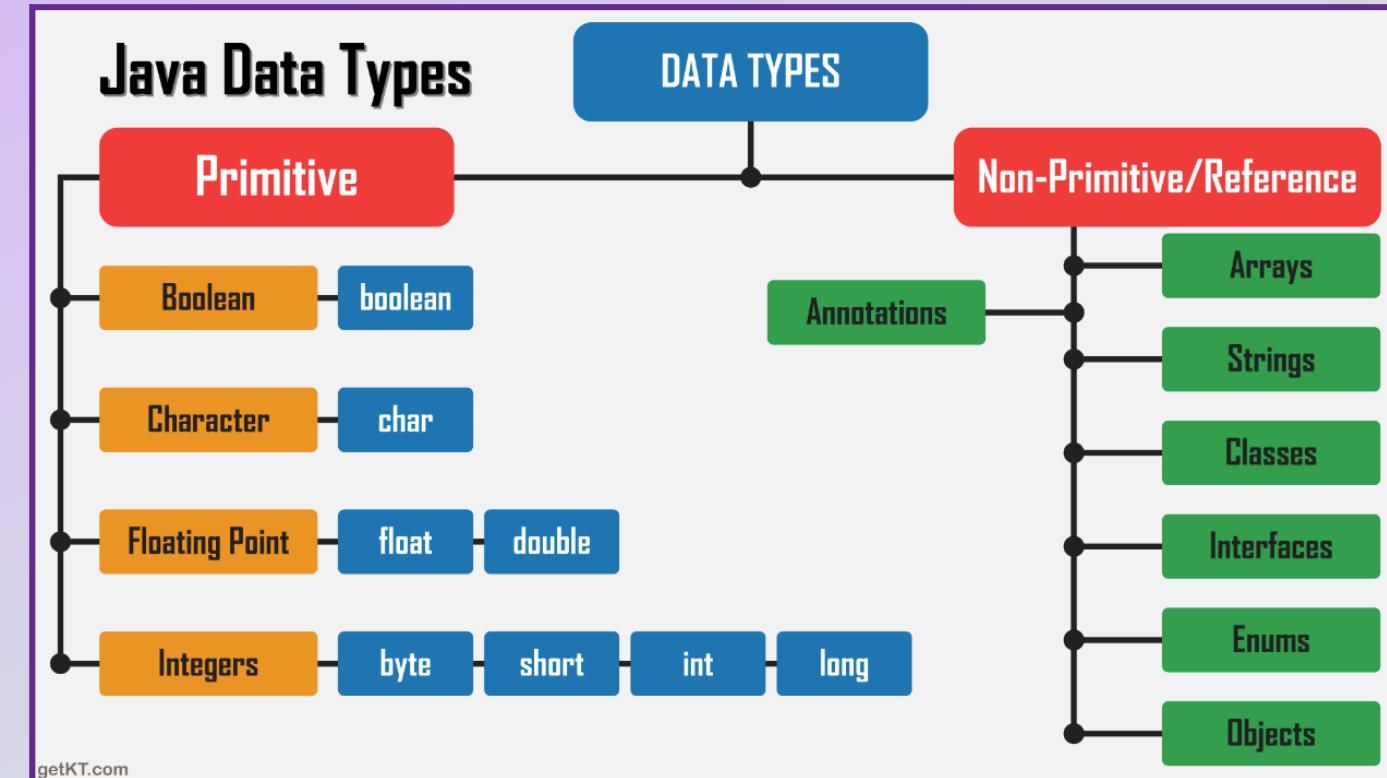
Data is a collection of facts, such as numbers, words, measurements, observations or just descriptions of things.

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 cok farkli olabileceginden, tum programlama dilleri farkli data turlerini kullanabilmek icin **kendi kullanacaklari data turlerini** tanimlamislardir.

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



# Data Hafizada Nasil Saklanir (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.

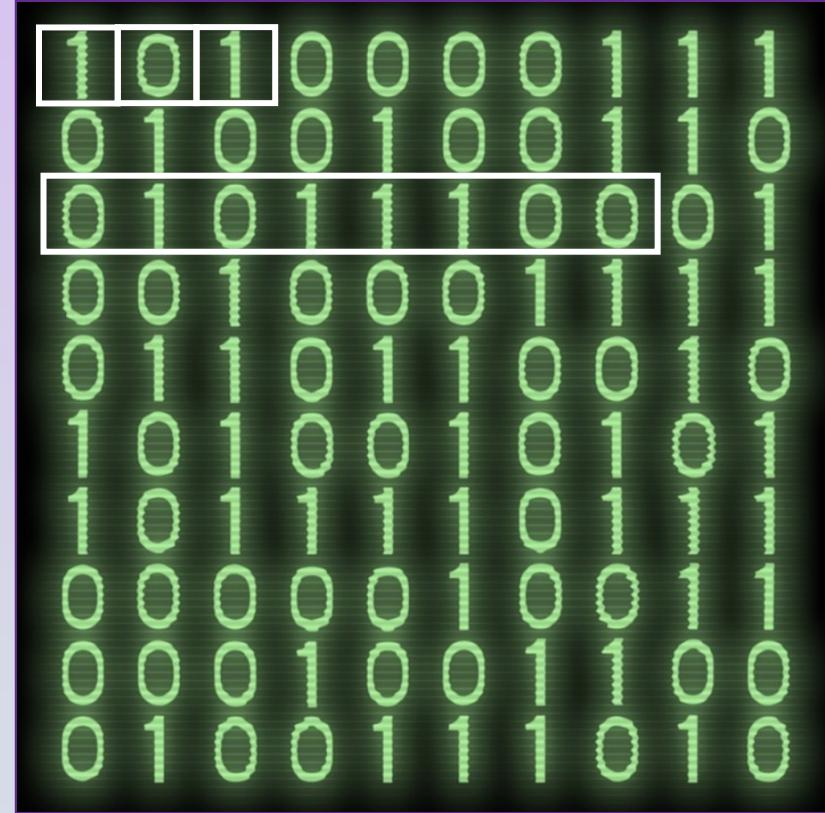
Sayı sistemimiz 10'luk sistem oldugu gibi hafiza da  $2^{10} = 1024$ 'un katlari seklinde yapılandırılmıştır.

1024 byte = 1 KB

1024 KB = 1 MB

1024 MB = 1 GB

1024 GB = 1 TB ....



# Variables (Java Datalari Nasil Kullanir) ?

Java hafizadaki datalari **variable** veya **objeler** yardimiyla kullanir.

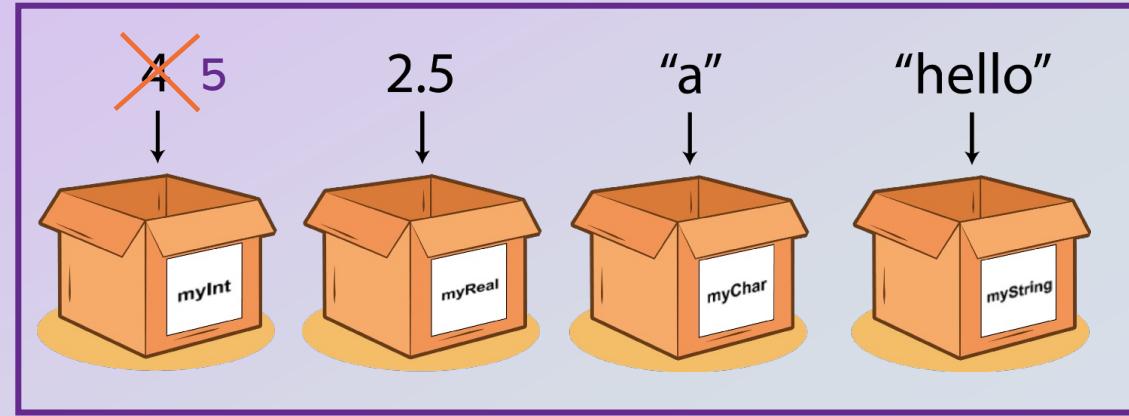
Bir datayi hafizada store etmek istedigimizde, Java hafizada o datayı tutabilecegi bir alan ayirir ve o alani isimlendirir.

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

Ornegin myInt'i bir artırdıgımızda, Java myInt ismindeki variable'i bulur, icindeki deger olan 4'u bir artırip 5 yapar.

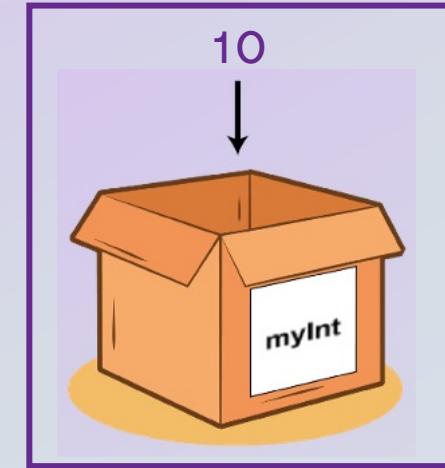
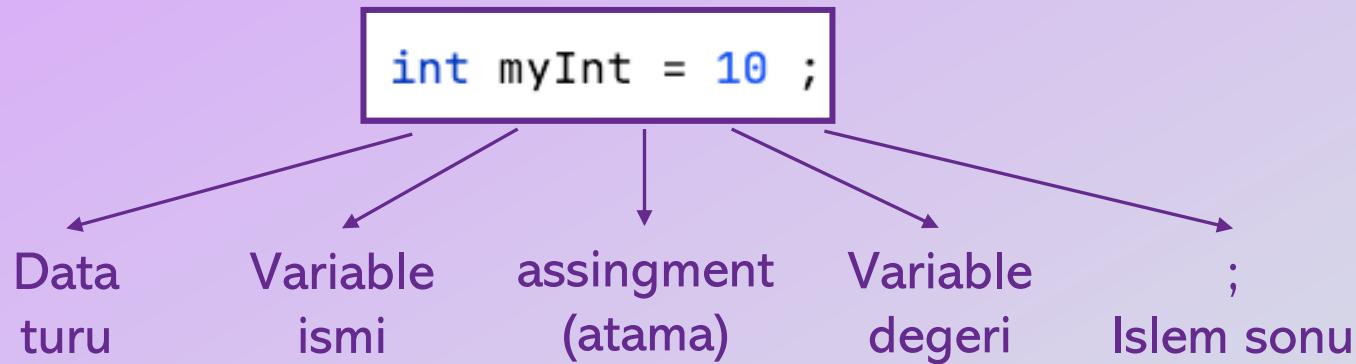
Bu islemden sonra myInt variable'inin degeri 5 olacaktır.

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



# Variable Nasil Olusturulur ?

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



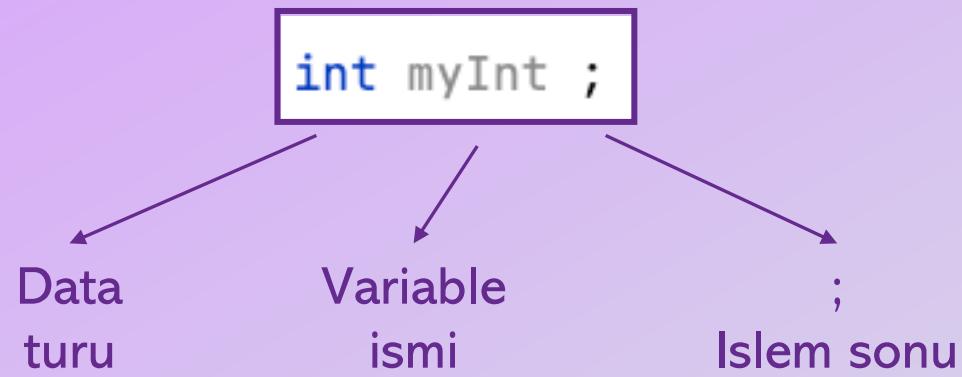
Bir variable olusturmak icin 2 islem vardir;

1- declaration (tanimlama) : esitligin sol tarafı

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

# Variable Declaration

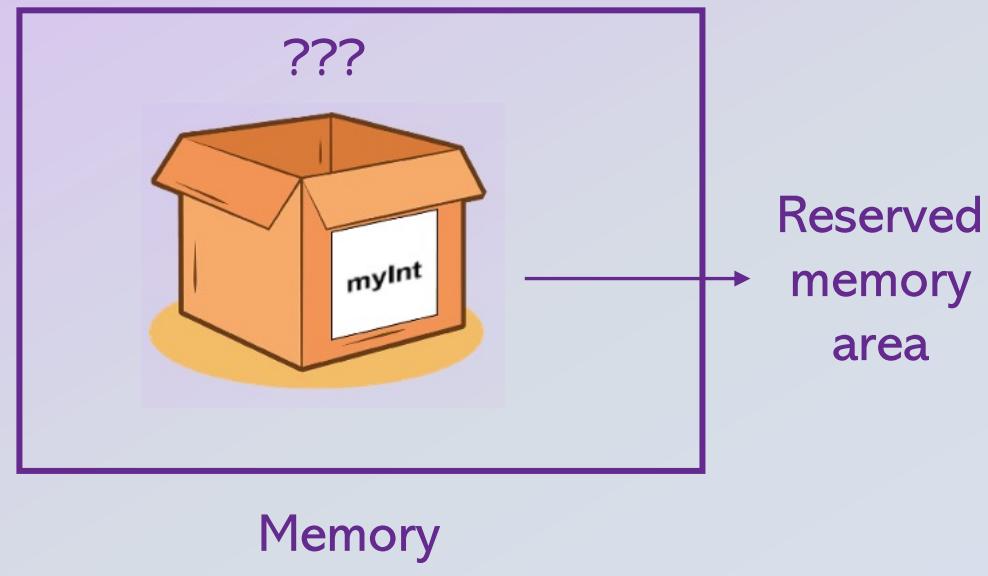
Java'nin datayı store edebilmesi için variable'a ihtiyacı vardır. Bir variable'in oluşturulması için de mutlaka declaration gereklidir.



Declaration için data turu ve variable ismi yeterlidir.

Java'da declaration ve assignment farklı satırlarda yapılabilir.

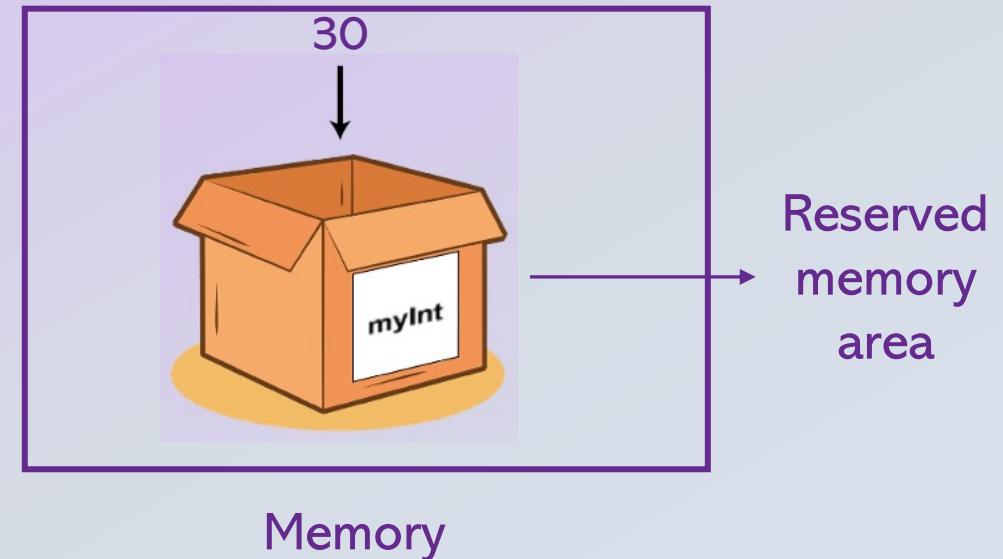
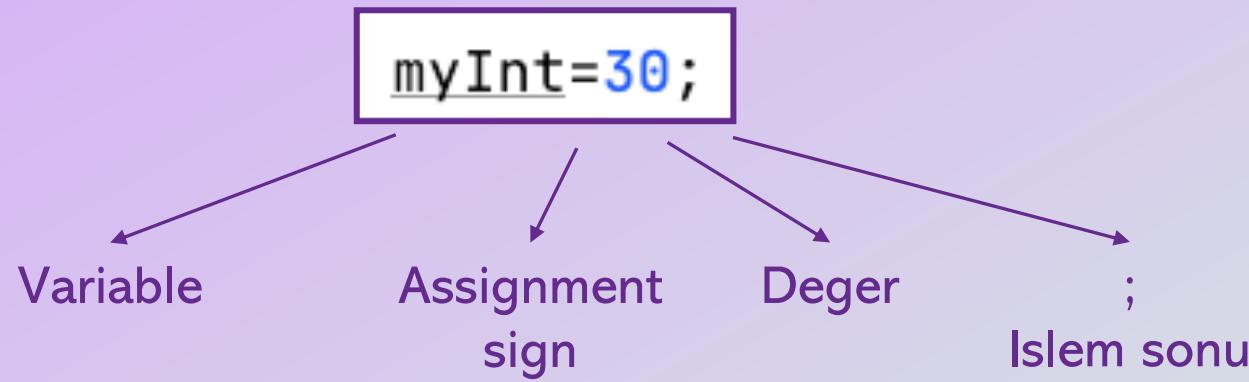
Ancak bir değer ataması olmadan variable'in kullanılması mümkün degildir.



# Variable Deger Atamasi (Assignment)

Deklare edilmiş bir variable'a değer atamaya assignment denir.

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



# Variable Deger Atamasi (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 Deger Atamasi (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;
```

# FirstBrick Java Dersleri

## Ders - 03

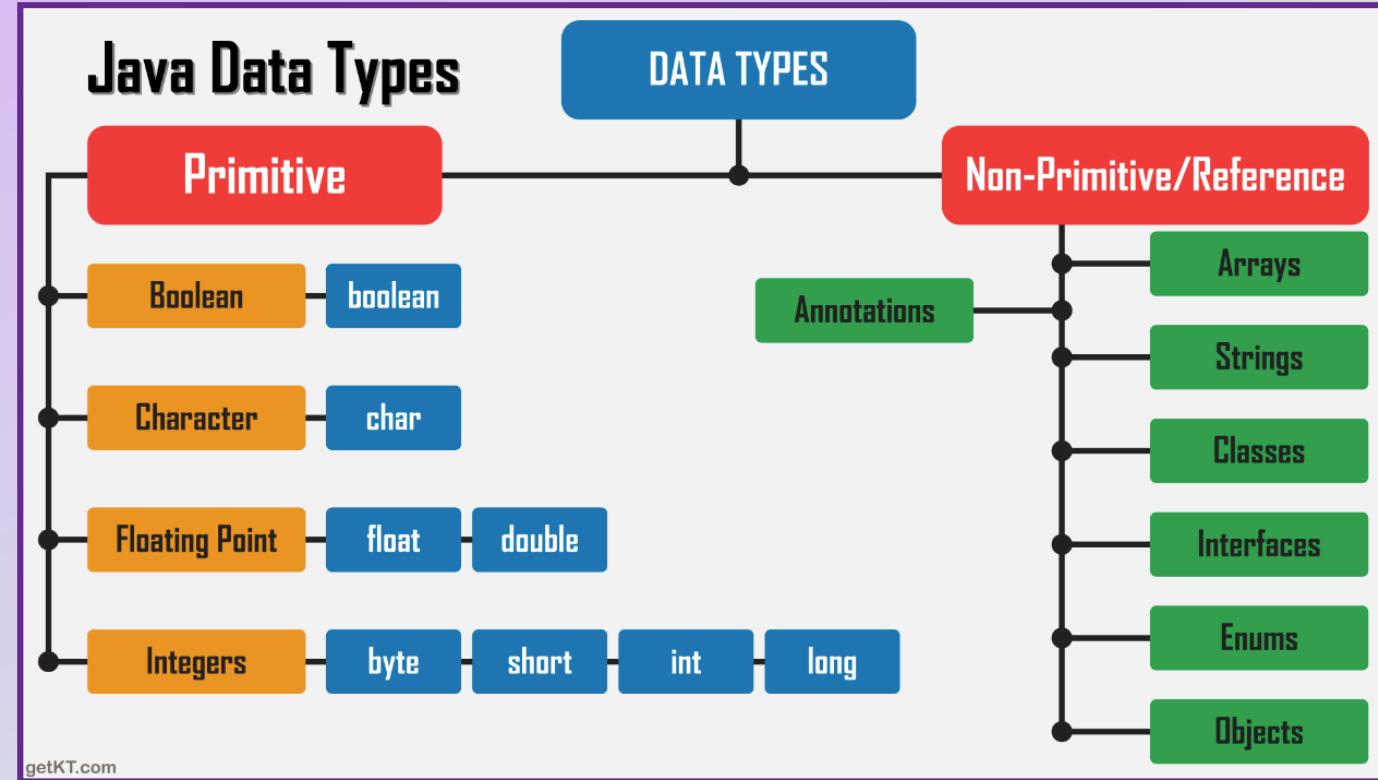
Java'da Data Türleri  
Kullanıcıdan Değer Alma

# Java'da Data Türleri Nelerdir ?

Java'da temelde iki data turu kullanılır.

- 1- Primitive Data Türleri
- 2- Non-Primitive Data Türleri

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



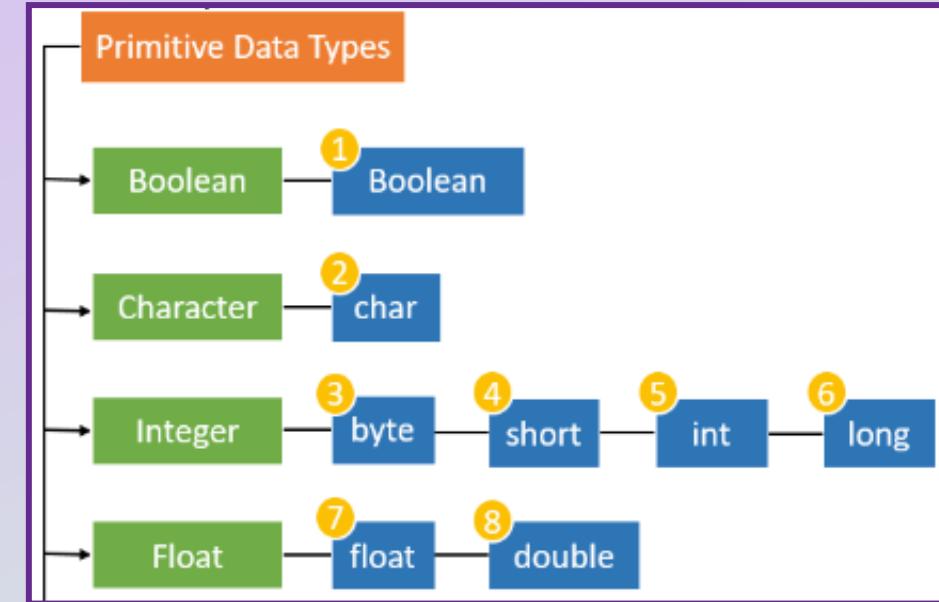
Java'da 8 primitive data turu kullanılır.

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 farkliliği 'c' (tek tırnak) kullanmasıdır. Bir data '' kullanırsa char olmalıdır.

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

hafızada 16 bit yer kaplar

# Primitive Data Turleri

Tam sayi barindiran 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 icin hangi data turunu kullanacagımız, uygulamamızın hafiza kullanımını için önemlidir.

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

Yasları short, int veya long olarak da store edebiliriz ancak bu durumda kullanılacak 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.

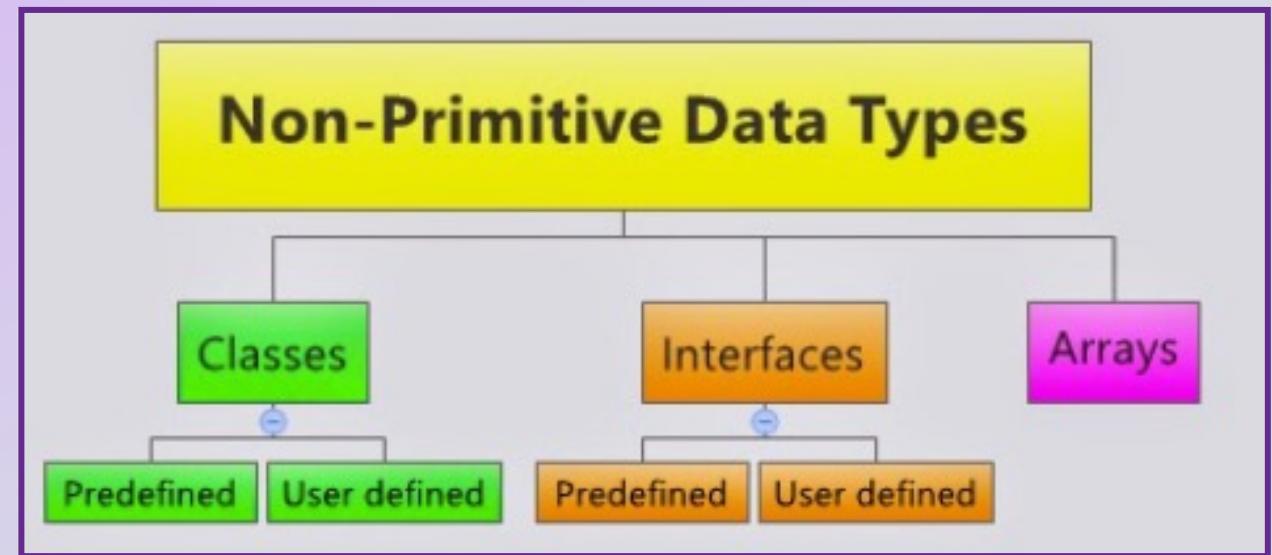
Non-Primitive data turleri ise Java tarafından sınırlanılmamışlardır.

Java da bazi non-primitive data turleri olusturulmustur, biz de yeni non-primitive data turleri olusturabiliriz.

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

Class'lar OOP konsept çerçevesinde bizim obje olusturdugumuz, kaliplar olduğundan, 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.

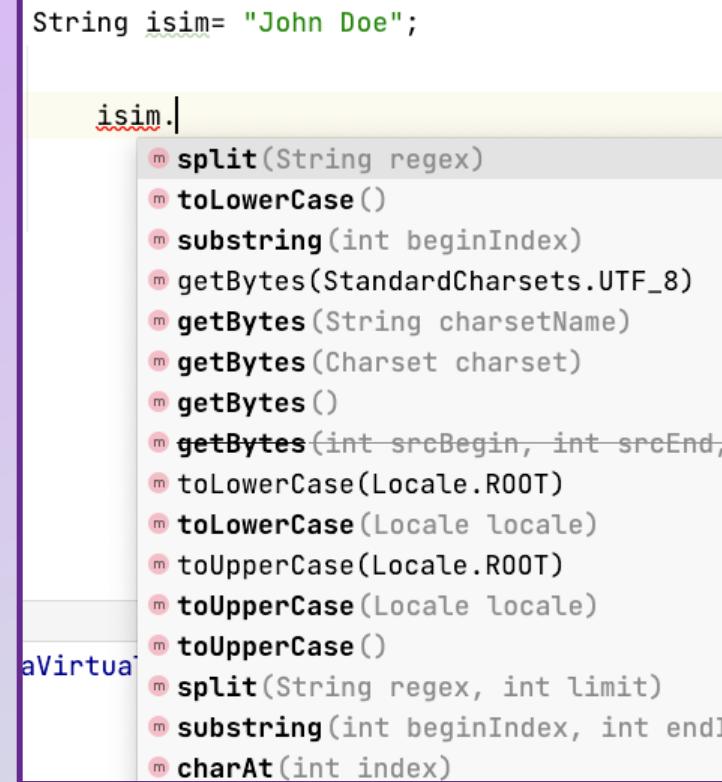


# Non-Primitive Data Turleri

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.



A screenshot of an IDE showing code completion for a String variable. The code `String isim= "John Doe";` is written in the editor. A tooltip or dropdown menu is open over the variable `isim`, displaying a list of available methods:

- `split(String regex)`
- `toLowerCase()`
- `substring(int beginIndex)`
- `getBytes(StandardCharsets.UTF_8)`
- `getBytes(String charsetName)`
- `getBytes(Charset charset)`
- `getBytes()`
- `getBytes(int srcBegin, int srcEnd, Locale locale)`
- `toLowerCase(Locale ROOT)`
- `toLowerCase(Locale locale)`
- `toUpperCase(Locale ROOT)`
- `toUpperCase(Locale locale)`
- `toUpperCase()`
- `split(String regex, int limit)`
- `substring(int beginIndex, int end)`
- `charAt(int index)`

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

# Primitive **vs** Non-Primitive Data Turleri

İki data turu arasında 5 temel farklılık sayabiliriz.

Primitives	Non-Primitives
Tüm Java tarafından oluşturulmuştur.	Java tarafından oluşturulanlar olduğu gibi biz de oluşturabiliriz
Sadece değer içerirler, variable ile kullanılacak hazır method'ları yoktur.	İçerikleri değerin yanında oluşturuldukları class'dan gelen hazır method'lar da 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 başlar (int, char vb)	Data turu isimleri <b>buyuk</b> harfle başlar. (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 büyüklüğine göre hafızada yer kaplarlar. (bir kelime veya binlerce kelime içeren String'lerin boyutları farklı olacaktır)

## Variable'lar Icin Isim Verme Kurallari (Naming Convention)

Variable'lara isim verirken istedigimiz ismi secebiliyoruz 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.

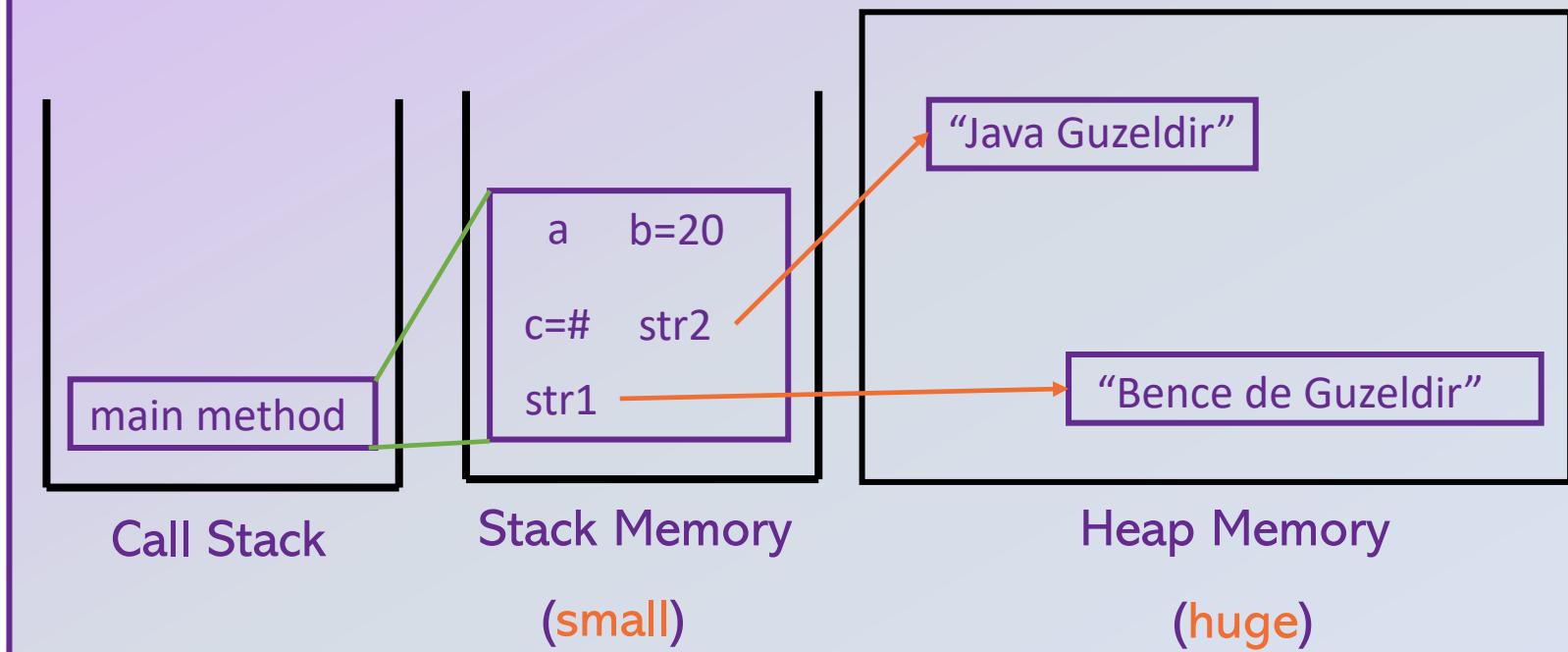
5- Variable isimleri kucuk harfle baslar, birden fazla kelime iceriyorsa **camelCase** kullanilir,  
yani sonraki her kelimenin ilk harfi **buyuk harf**, diger harfleri **kucuk harf** yapilir.

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 istedigimiz icin ornegimizde **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 değerlerini değiştirin(swap).

**Soru 8 (Interview)-** Kullanicidan iki sayı alip, ucuncu bir değişken kullanmadan ikisinin değerlerini değiştirin(swap).

# Free Java Dersleri

## Ders - 04

### Data Casting

# Data Casting ( Datayı Farklı Data Turune Çevirme )

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

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

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

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

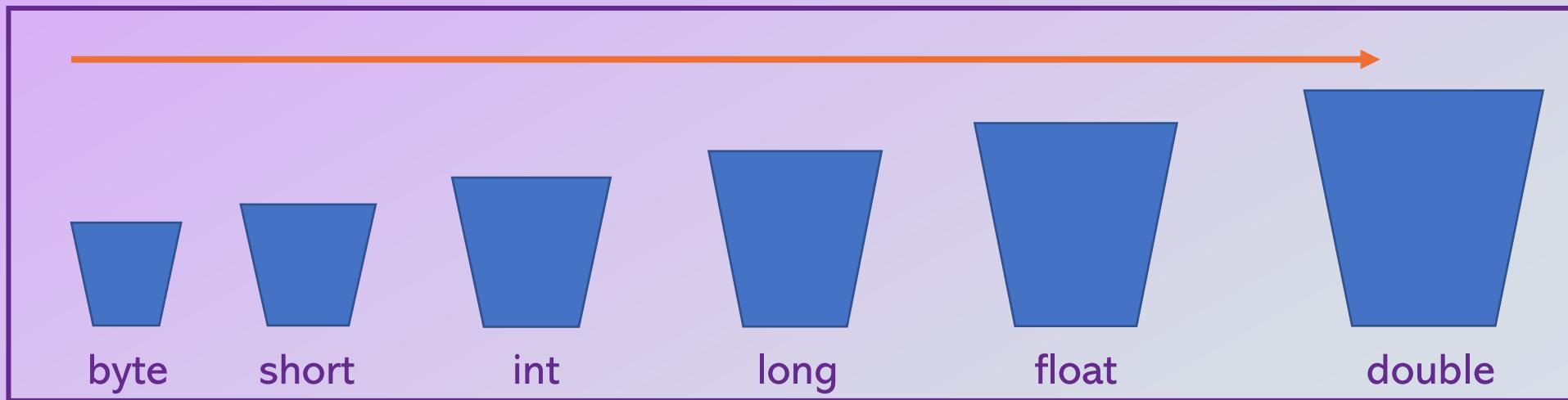
```
double dbl=23.4;  
int sayi= dbl;  
  
int in= 12;  
double db= in;
```

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

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

## Implicit Data Casting ( Auto-widening )

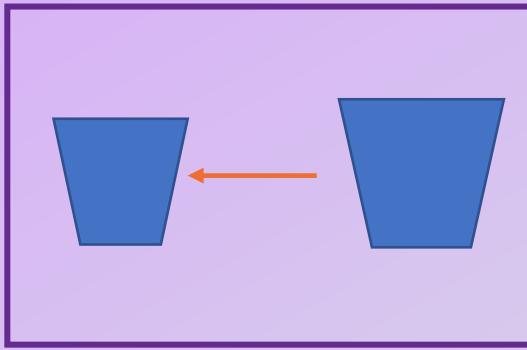
Daha kucuk kapsamli bir data turundeki degeri, daha genis kapsamli data turundeki variable'a atama yapmak istedigimizde, Java bu islemi otomatik olarak yapacaktir



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

## Explicit Data Casting( Daraltma)

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 için 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	'
01	SOH	(Start of Header)	33	!	65	A	97	a
02	STX	(Start of Text)	34	"	66	B	98	b
03	ETX	(End of Text)	35	#	67	C	99	c
04	EOT	(End of Trans.)	36	\$	68	D	100	d
05	ENQ	(Enquiry)	37	%	69	E	101	e
06	ACK	(Acknowledgement)	38	&	70	F	102	f
07	BEL	(Bell)	39	'	71	G	103	g
08	BS	(Backspace)	40	(	72	H	104	h
09	HT	(Horizontal Tab)	41	)	73	I	105	i
10	LF	(Line feed)	42	*	74	J	106	j
11	VT	(Vertical Tab)	43	+	75	K	107	k
12	FF	(Form feed)	44	,	76	L	108	l
13	CR	(Carriage return)	45	-	77	M	109	m
14	SO	(Shift Out)	46	.	78	N	110	n
15	SI	(Shift In)	47	/	79	O	111	o
16	DLE	(Data link escape)	48	0	80	P	112	p
17	DC1	(Device control 1)	49	1	81	Q	113	q
18	DC2	(Device control 2)	50	2	82	R	114	r
19	DC3	(Device control 3)	51	3	83	S	115	s
20	DC4	(Device control 4)	52	4	84	T	116	t
21	NAK	(Negative acknowl.)	53	5	85	U	117	u
22	SYN	(Synchronous idle)	54	6	86	V	118	v
23	ETB	(End of trans. block)	55	7	87	W	119	w
24	CAN	(Cancel)	56	8	88	X	120	x
25	EM	(End of medium)	57	9	89	Y	121	y
26	SUB	(Substitute)	58	:	90	Z	122	z
27	ESC	(Escape)	59	:	91	[	123	{
28	FS	(File separator)	60	<	92	\	124	
29	GS	(Group separator)	61	=	93	]	125	}
30	RS	(Record separator)	62	>	94	^	126	~
31	US	(Unit separator)	63	?	95	-		
127	DEL	(Delete)						

```
char harf= 'a';
```

```
int sayi= 100;
```

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

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

```
char yeniharf=(char)(harf+1);
```

```
System.out.println(yeniharf); // b
```

## 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.

# Free Java Dersleri

## Ders - 05

Wrapper Classes  
Matematiksel İşlemler

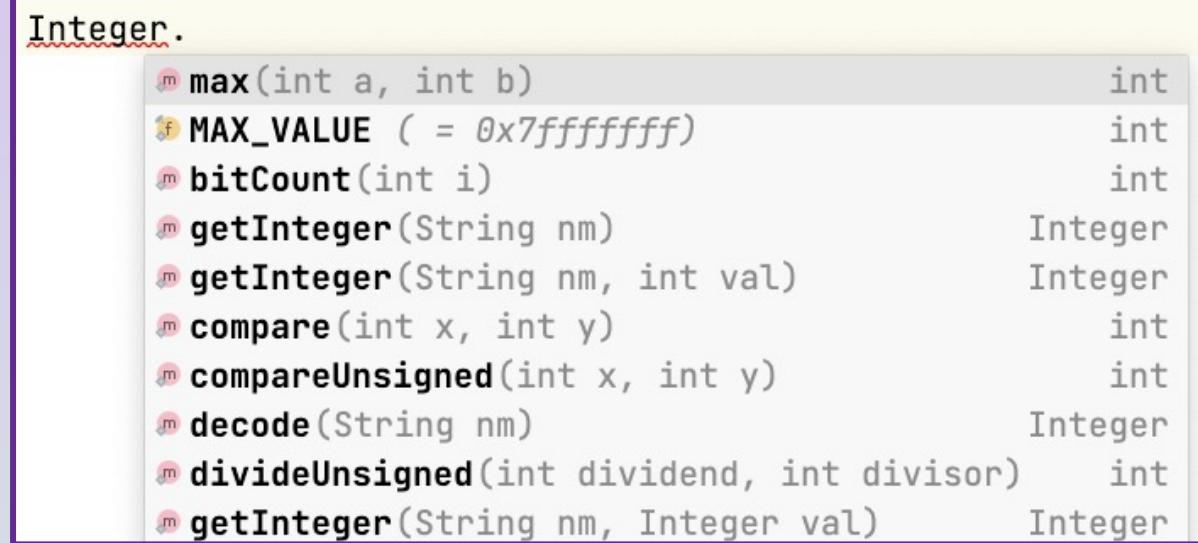
# Wrapper Classes

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 cok faydalı method bulundurlar.



# Wrapper Classes

Wrapper class'lar casting, max-min degerler, karsilastirma gibi bircok hazir 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);

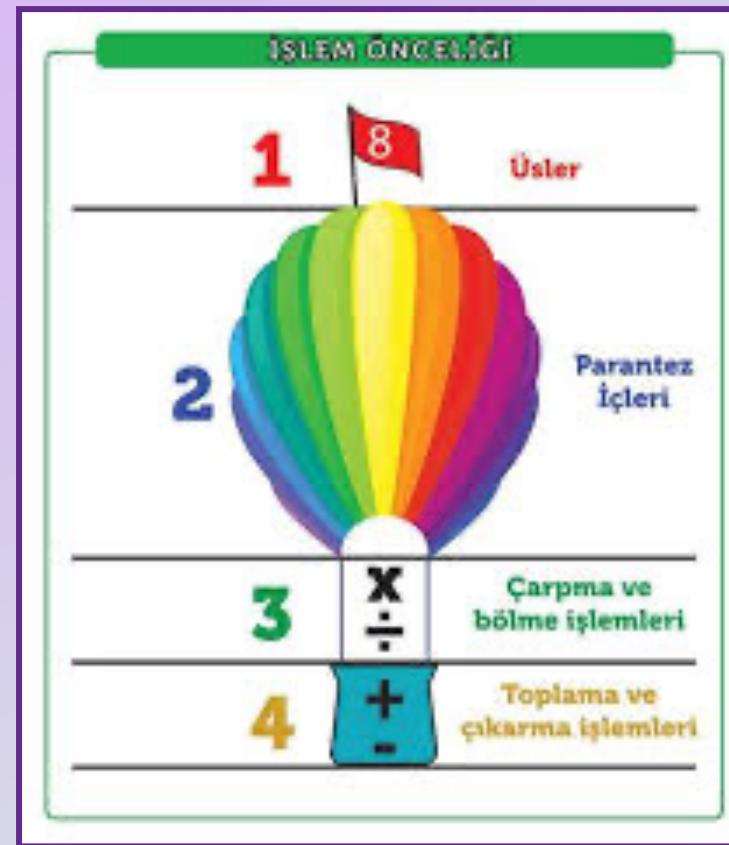
char chr='*';
Character ch='p';
char chr2=101;
System.out.println(Character.valueOf(chr2)); // e
System.out.println(Character.isDigit( ch: '5')); // true
System.out.println(Character.isAlphabetic( codePoint: '9')); // false
System.out.println(Character.isAlphabetic( codePoint: 'a')); //true
```

# 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 ( % )

Java'da Modulus islemi, bir bolme islemindeki kalan sayiyi bize verir.

A handwritten division diagram. On the left, 'Bölen' (Divisor) is written above the vertical bar, and 'Bölünen' (Dividend) is written to its left. On the right, 'Bölüm' (Quotient) is written below the bar, and 'Kalan' (Remainder) is written to its right. The dividend 85 is written above the bar, and the divisor 6 is written to its left. The quotient 14 is written below the bar, and the remainder 1 is written at the bottom. The diagram shows the steps: 6 goes into 8 once, leaving a remainder of 2; then 6 goes into 25 four times, leaving a remainder of 1.

Modulus islemi 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.

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

**Ipucu 1:** Sayi  $\% 10 \Rightarrow$  Bize son basamagi verir

$$1469 \% 10 = 9$$

**Ipucu 2:** Int Sayi /10  $\Rightarrow$  Bize son basamak haric sayiyi verir

```
int sayi=1469;
```

$$\text{sayi} = \text{sayi} / 10 \Rightarrow$$

**sayi'**ya 46 degerini atar

# Free Java Dersleri

## Ders - 06

### Increment Decrement

## 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 ;
```



```
int sayi = 10 ;
```

```
sayi -= 4 ;
```



```
int sayi = 10 ;
```

```
sayi -- ;
```



## 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

```
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.

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

# Pre / Post Increment

```
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**)

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

# Pre / Post Increment

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));
```

# Free Java Dersleri

## Ders - 07

Concatenation  
Relational Operators  
Logical Operators

## 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 yapilir. Sira String ile toplamaya geldiginde toplama yerine Concatenation uygulanir

```
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 Operators ( Karsilastirma Operatorleri )

## 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 Operators ( Karsilastirma Operatorleri )

### 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 Operators ( Mantiksal Operatorler )

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

Mantiktaki AND operatorunun Java'da 2 tane karsiliği 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 carpıma benzetilir.  
Sonucun 1 olması için  
tüm çarpılan sayılar 1 olmalıdır.  
1 tane bile sıfır olsa sonuc 0 olur.  
  
1 \* 1 \* 1 \* 1 = 1  
1 \* 0 \* 1 \* 0 = 0  
1 \* 0 \* 0 \* 0 = 0  
0 \* 1 \* 1 \* 1 = 0