

U N I T Y U E R S E  
A C A D E M Y

# Java

## Ders-01

Genel Bilgilendirme  
Programlamaya Giriş

unityverseacademy.com

Egitmen : Ahmet BULUTLUOZ

# Onemli Hatirlatmalar

Bugun  
IT Dunyasinda  
Ilk Gununuz

- 1- Ders Tam Vaktinde Baslar
- 2- Ders Oncesi Hazirlik Yapin  
(Google en yakin arkadasiniz olmali)
- 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



- 1- Ders tam zamanında baslar
- 2- Ders basında bir önceki günün kısa bir tekrarı yapılır
- 3- Her konu bittiğinde, o konu ile ilgili bir test yapılır.

## 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- Bilgisayarın bizim icin urettigi sonucları anlamlandırmamız gereklidir.

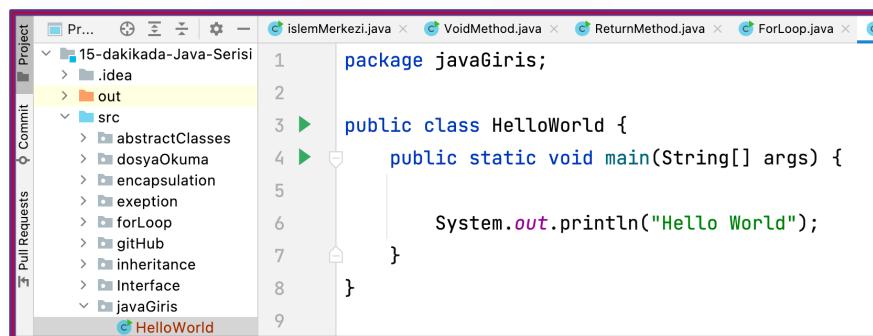
Kisaca ozetlersek, programlama dili bizimle bilgisayar arasindaki iletisimi saglayan dildir.

Peki..Bizimle bilgisayar arasindaki islemlerde patron kim ?

Kimin dedigi olur ?

Tabii ki bilgisayarlar bizim dedigimizi yaparlar

Ama bu istegimizi bilgisayarin anlayacagi ozelliklerde yazmamiz sartiyla.



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

```
package javaGiris;
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Ornegin;  
kodlarimizla, Hello World yazdirmak istiyorsak

```
11010101010100100010101010  
10010111000001000100101101010  
101010010111000110  
010001010010101110001000000101  
0010101110001001010111
```

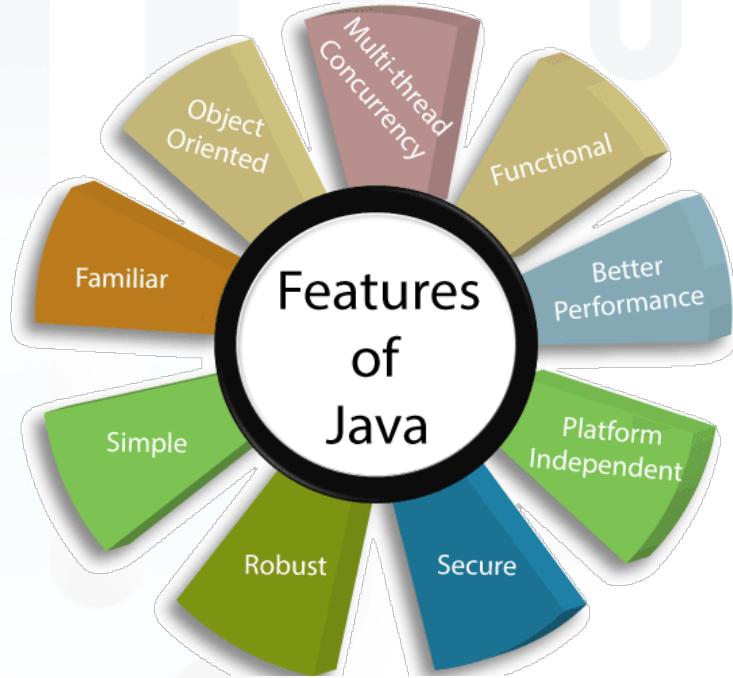
JDK Kodlari derler (Compile)

Binary kodlar



Islem

# Nicin Java ?



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

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

- 1- Ogrenmesi kolay
- 2- Dunyada en çok kullanılan programlama dili  
3 milyar mobil cihazda Java kullanıyor.  
USA'de sirket bilgisayarlarının %97'sinde, kişisel bilgisayarların %89'unda Java kullanılıyor  
Linkedin, Uber, Netflix gibi pek çok popüler uygulama Java tabanlı calisiyor.

- 3- Guvenlidir
- 4- Platform independent'dir
- 5- Ucretsizdir.

Neden Java: <https://youtu.be/RUYASEnT6pU>

# Nicin Java / OOP Konsept



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) 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

Objeleri olusturabilmek icin oncelikle kalip olusturmaliyiz.



Kalip varsa, bu kaliptan 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  
arasinda 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 sözluNotu=92;  
  
    }  
  
    public void başkaMethod(){  
    }  
}
```

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 }
```

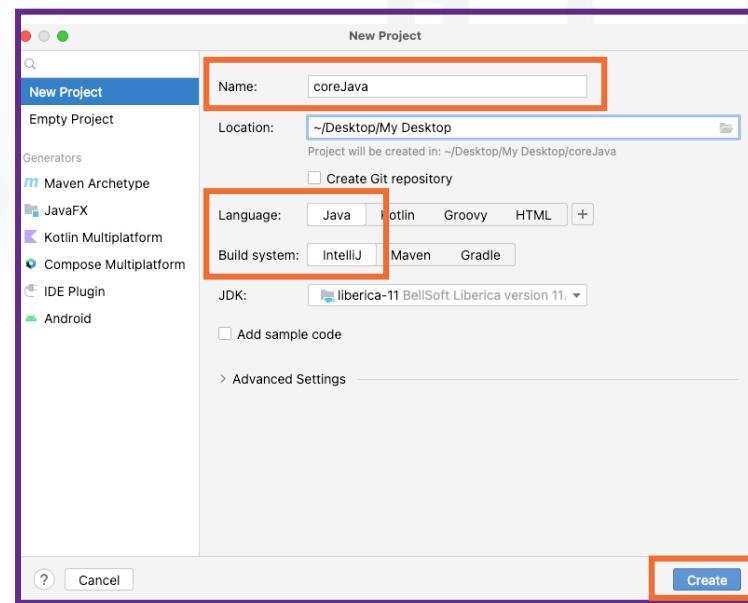
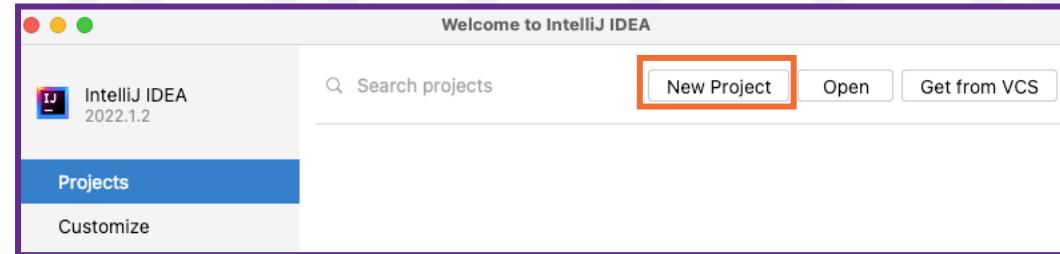
# IntelliJ'de Proje Oluşturma

IntelliJ Nedir?

Java gibi compiler  
programlar çalışmak için  
ide ( Integrated  
Development  
Environment)'ye ihtiyac  
duyarlar.

Birçok ide olmakla  
birlikte, piyasada çok  
kullanıldığı ve kod  
yazımını kolaylastırdığı  
fürsait biz IntelliJ  
kullanacağız.

IntelliJ'de **Projects** menusunde **New Project'i** seçin,



Açılan menü'de

**Name** kısmina projenin ismini yazın,

**Language** kismında Java, **Build system**  
kismında IntelliJ seçili olduğunu 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 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       // Tek satiri comment yapmak icin  
7 }
```



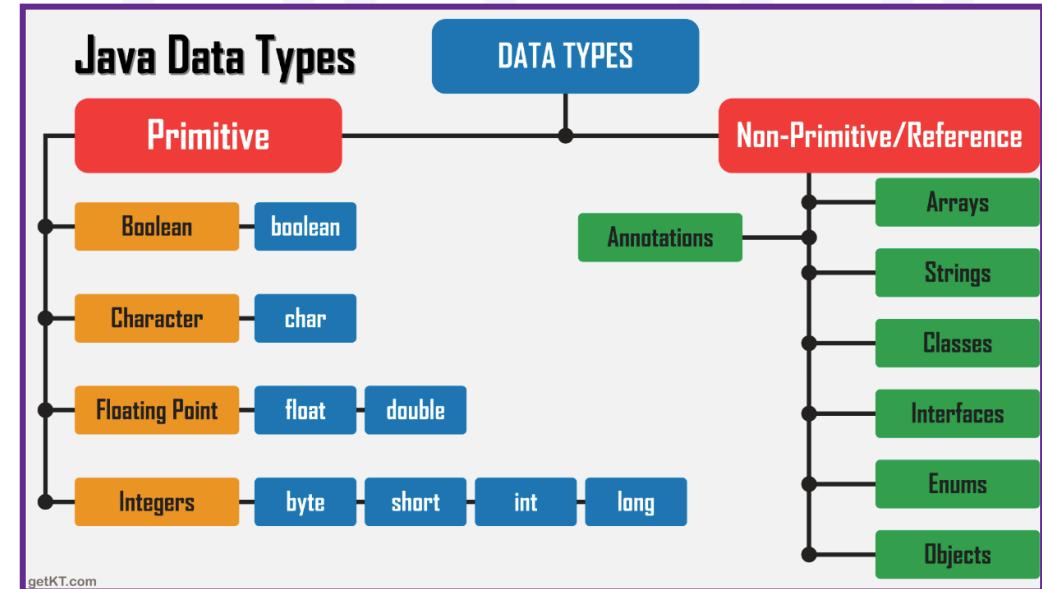
```
10 */  
11      Birden fazla  
12      satiri  
13      comment yapmak icin  
14 */
```

# 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 ogrenmeliyiz.

# Data Saklama(Store)

Her data hafizada(memory) bir yer kaplar.

Bir datanın hafızada saklanacağı en küçük bölüm bit'dir.

Her bir bit **1** veya **0** değerlerini içerir.

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

Her **byte**  $2^8 = 256$  farklı değer alabilir.

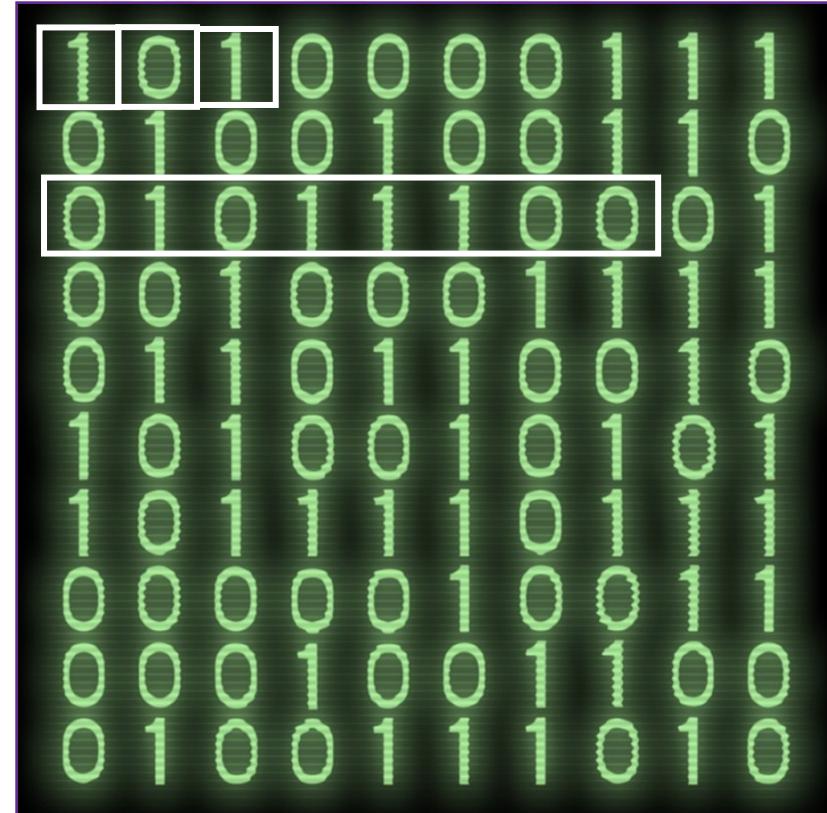
Sayı sistemimiz 10'luk sistem olduğu gibi hafıza da  $2^{10} = 1024$ 'un katları şeklinde yapılandırılmıştır.

1024 byte = 1 KB

1024 KB = 1 MB

1024 MB = 1 GB

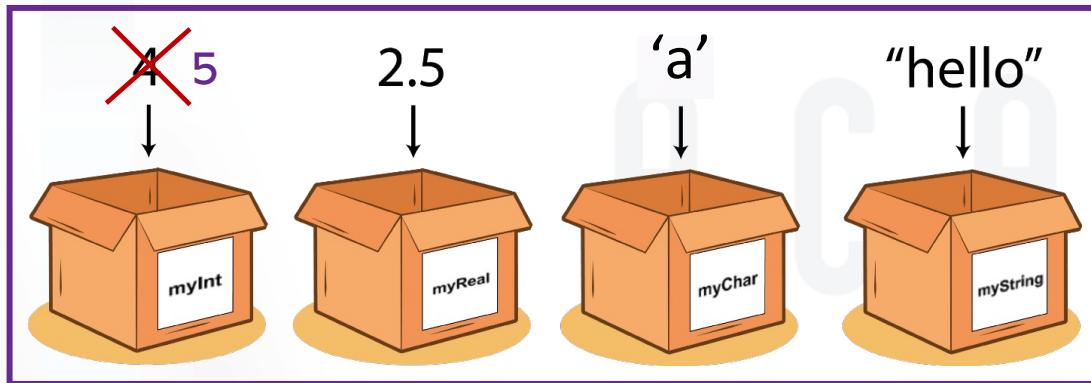
1024 GB = 1 TB ....



# Variables (Data Kullanma )

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.

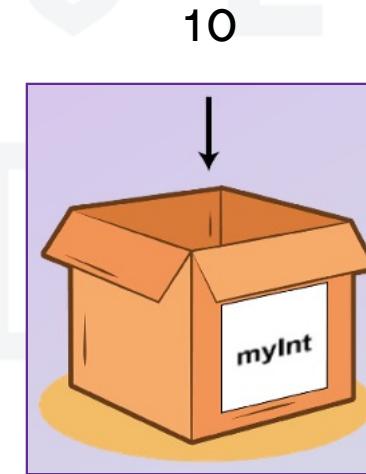
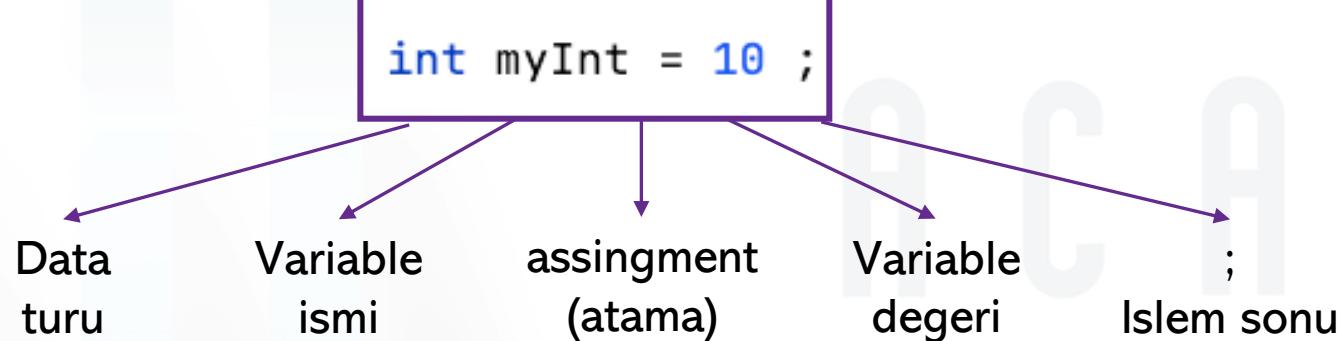
Ornegin myInt'i bir artır dedigimizde, Java myInt ismindeki variable'i bulur, icindeki deger olan 4'u bir artırıp 5 yapar.

Bu islemden sonra myInt variable'inin degeri 5 olacaktir.

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 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 sag 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.

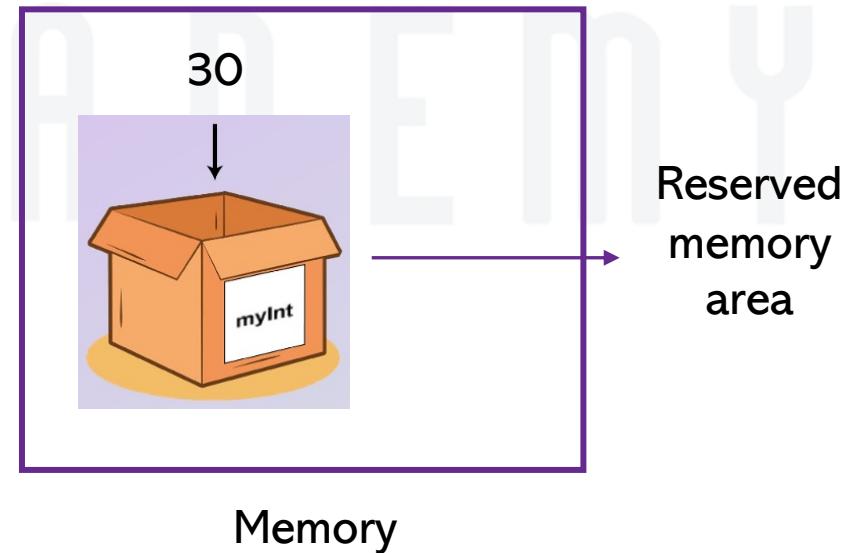
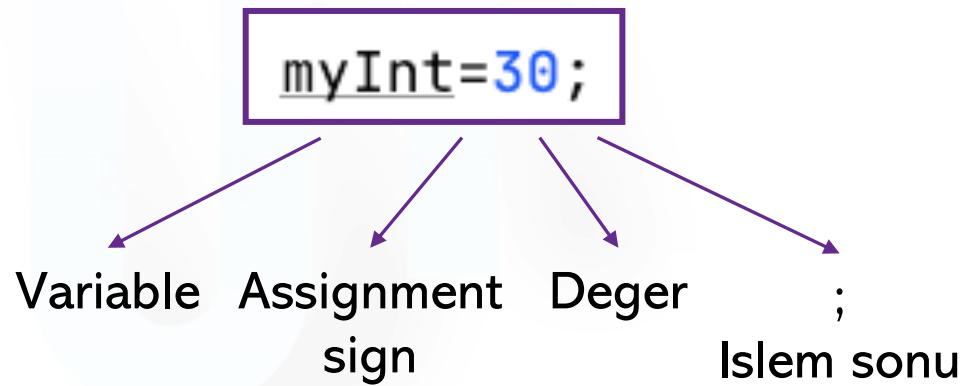
Java'da declaration ve assignment farklı satırlarda yapılabilir.

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 yapılabileceği gibi, farklı satırlarda da yapılabilir.



# 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.

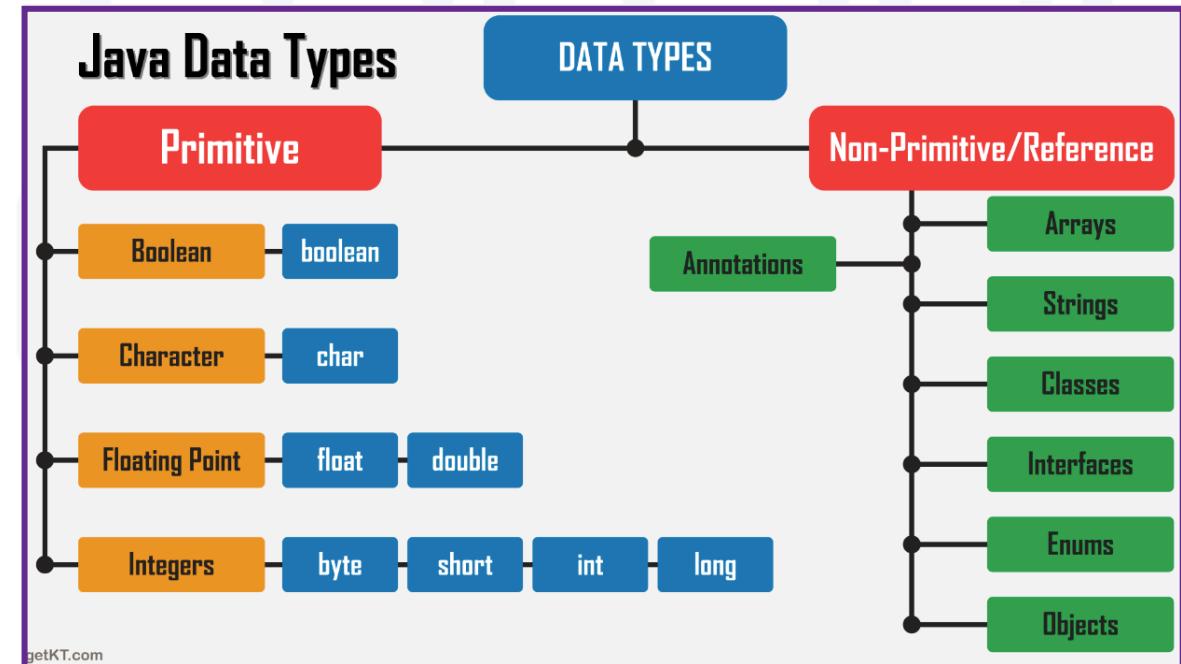
```
int not1=80,not2=90,ortNot= (not1 + not2)/2;
```

# 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.

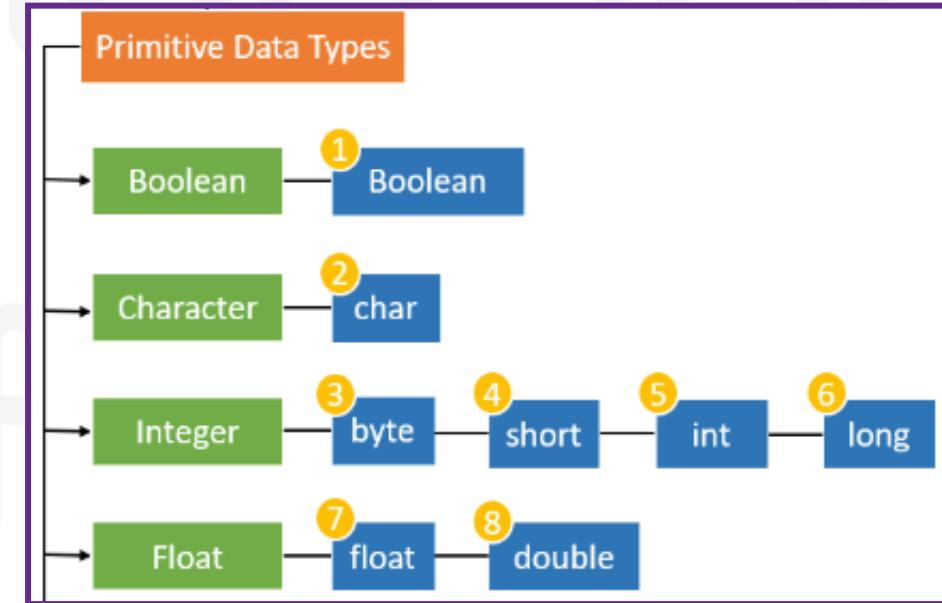


Java'da 8 primitive data turu kullanılır.

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

hafızada **16 bit** yer kaplar

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

# 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 kullanacagimiz, uygulamamizin hafiza kullanimi icin onemlidir.

Ornegin, universitedeki ogrencilerin yaslarini barindiran bir variable icin byte yeterlidir.

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

# Primitive Data Turleri

Ondalikli sayı 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
```

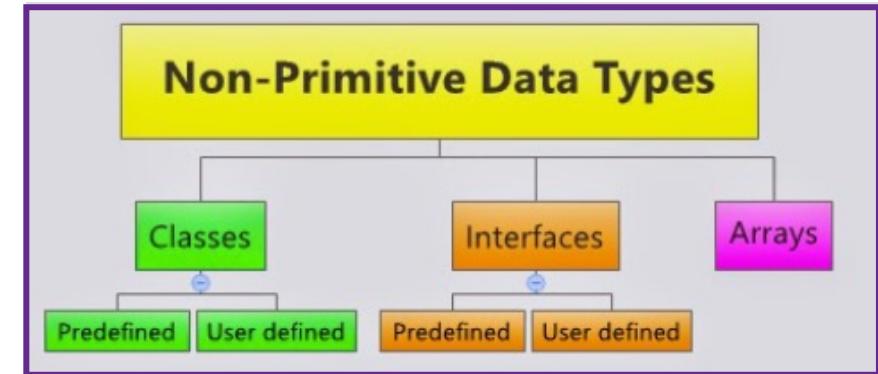
# Non-Primitive Data Turleri

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 olusturulmuştur, biz de yeni non-primitive data turleri olusturabiliriz.

Non-Primitive data turlerinin geneli için **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 içerdiginden, non-primitive data turleri, olusturuldukları class'lardaki method'lari kullanabilirler.

# Non-Primitive Data Turleri

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

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

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

```
String isim= "John Doe";  
  
isim.|  
  └ 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 endIndex)  
  └ 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
Tümü Java tarafından oluşturulmuştur.	Java tarafından oluşturulanlar olduğu gibi biz de oluşturabiliriz
Sadece değer içerirler, variable ile kullanılacak hazır method'ları yoktur.	İçerdikleri değerin yanında oluşturuldukları class'dan gelen hazır method'ları da barındırırlar.
Bir değer atamadan oluşturulabilir ama kullanılmak için mutlaka değer atanmalıdır.	Değer atanmadan <b>null</b> olarak işaretlenebilirler.
Data turu isimleri <b>kucuk harfle</b> başlar (int, char vb.)	Data turu isimleri <b>buyuk harfle</b> 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)

# 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.

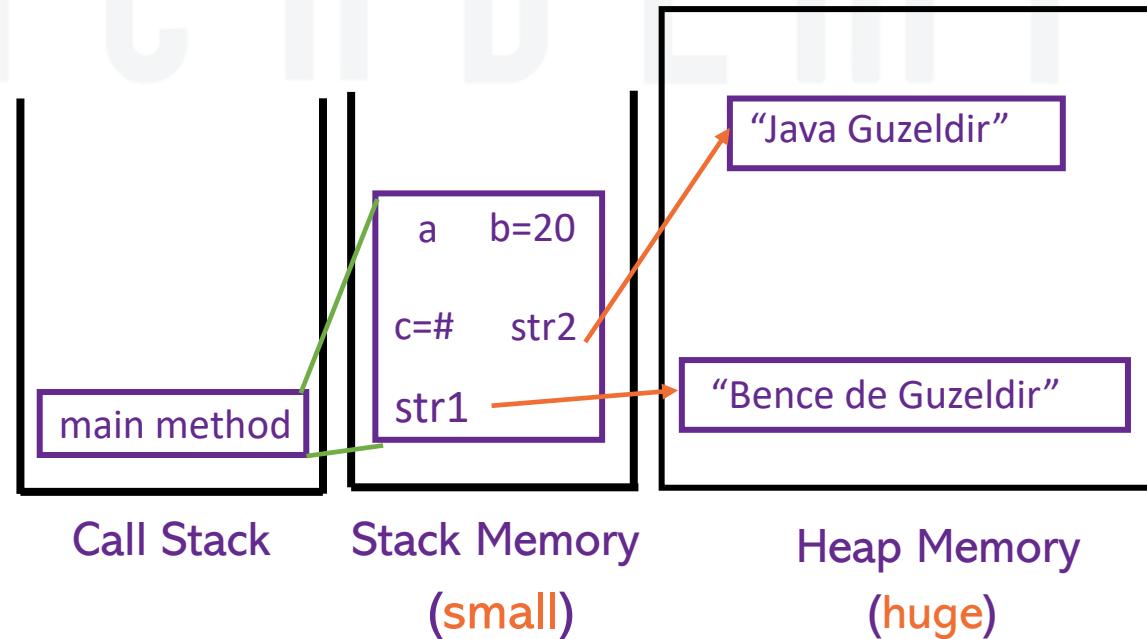
# Hafiza(Memory) Kullanimi

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();
```

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 degisen 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 çevirelemez.

Benzer özelliklerdeki data turundekidataları 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;
```

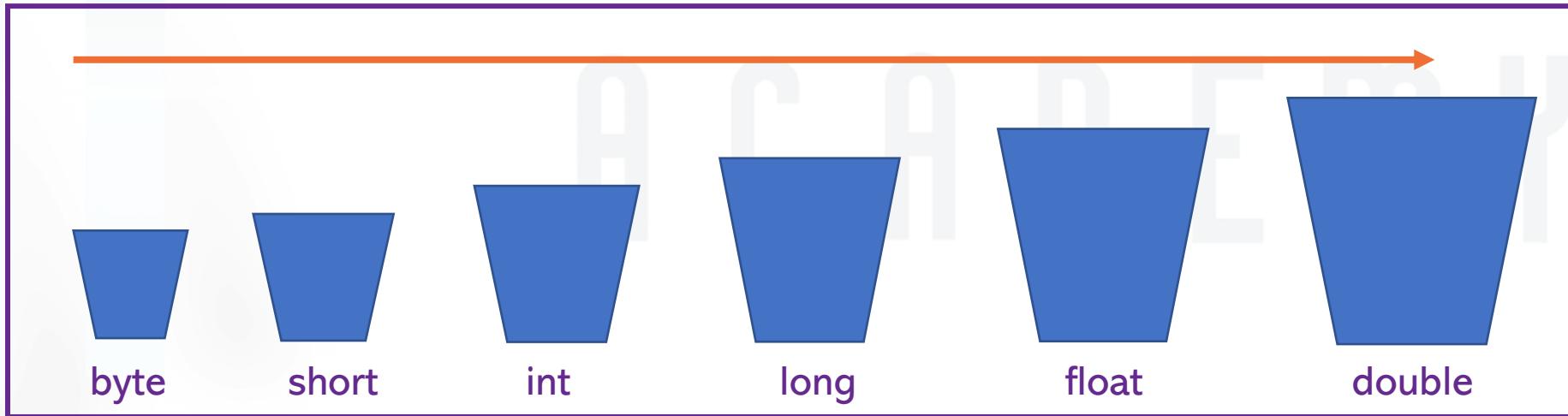
```
double dbl=23.4;  
int sayı= dbl;  
  
int in= 12;  
double db= in;
```

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

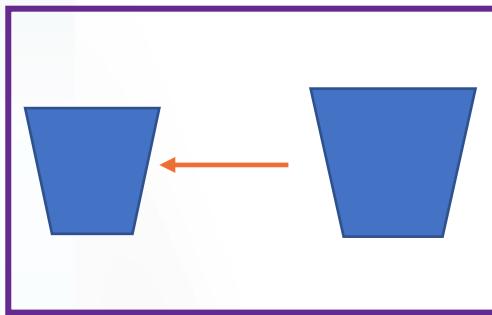
Daha kucuk kapsamlı bir data turundeki degeri, daha genis kapsamlı data turundeki variable'a atama yapmak istedigimizde, Java bu islemi otomatik olarak yapacaktır



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

# Explicit Data Casting

Daha genis kapasiteye sahip bir data turundeki bir degeri, daha dar 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	ß
02	STX	(Start of Text)	34	"	66	B	98	b	130	é	162	ó	194	ł	226	ő
03	ETX	(End of Text)	35	#	67	C	99	c	131	â	163	ú	195	—	227	ò
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	à	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	è	168	¸	200	Ł	232	þ
09	HT	(Horizontal Tab)	41	)	73	I	105	i	137	ë	169	®	201	Ú	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	ú
12	FF	(Form feed)	44	,	76	L	108	l	140	í	172	¼	204	Ł	236	ý
13	CR	(Carriage return)	45	-	77	M	109	m	141	í	173	¡	205	—	237	Ý
14	SO	(Shift Out)	46	.	78	N	110	n	142	À	174	«	206	—	238	—
15	SI	(Shift In)	47	/	79	O	111	o	143	À	175	»	207	—	239	—
16	DLE	(Data link escape)	48	0	80	P	112	p	144	É	176	—	208	ø	240	≡
17	DC1	(Device control 1)	49	1	81	Q	113	q	145	æ	177	—	209	Đ	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	—	245	§
22	SYN	(Synchronous idle)	54	6	86	V	118	v	150	ù	182	À	214	—	246	÷
23	ETB	(End of trans. block)	55	7	87	W	119	w	151	ù	183	À	215	—	247	·
24	CAN	(Cancel)	56	8	88	X	120	x	152	ÿ	184	©	216	—	248	·
25	EM	(End of medium)	57	9	89	Y	121	y	153	ö	185	—	217	—	249	—
26	SUB	(Substitute)	58	:	90	Z	122	z	154	Ù	186	—	218	—	250	·
27	ESC	(Escape)	59	;	91	[	123	{	155	ø	187	—	219	—	251	·
28	FS	(File separator)	60	<	92	\	124		156	£	188	—	220	—	252	·
29	GS	(Group separator)	61	=	93	]	125	}	157	Ø	189	¢	221	—	253	·
30	RS	(Record separator)	62	>	94	^	126	~	158	x	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
```

# 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.

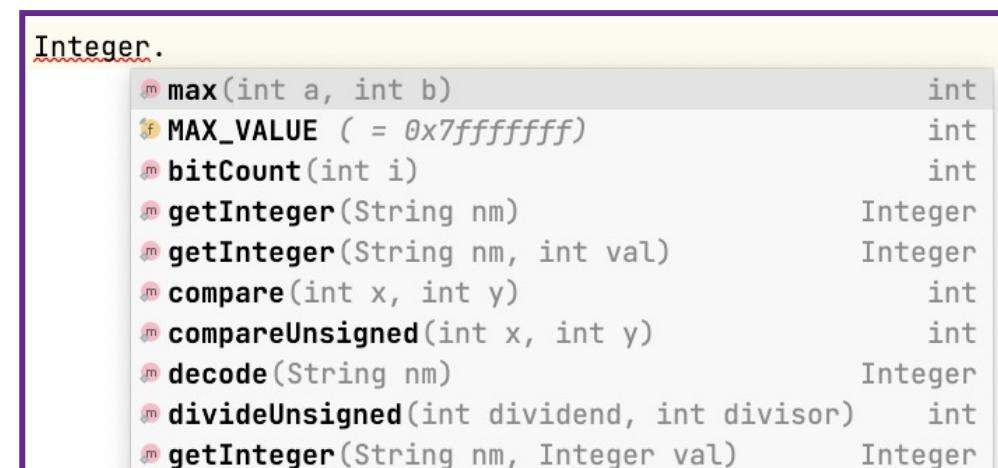
# Wrapper Classes

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.  
Ayrice bu class'lar bir cok faydalı method bulundururlar.



# 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
```



# **Java**

## **Ders-02**

**Matematiksel İşlemler**

**Operatorler**

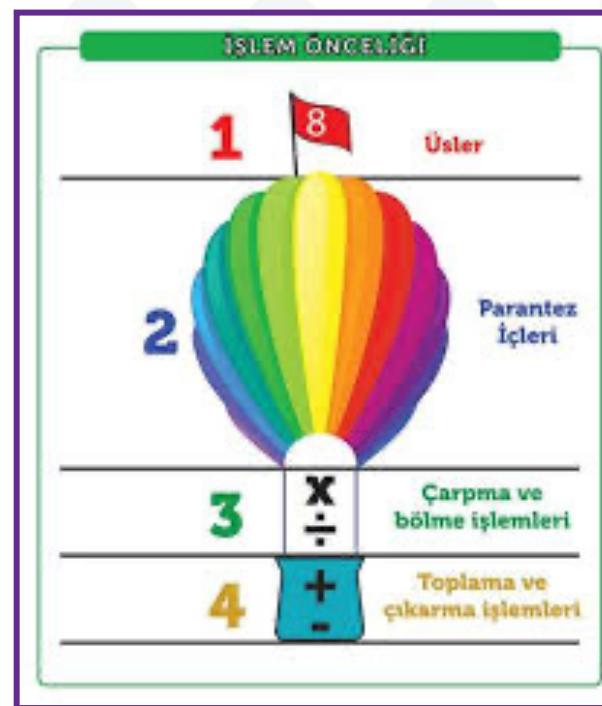
**If Else Statements**

# 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.

The diagram illustrates the long division process for calculating the modulus. A vertical line separates the dividend (Bölünen) from the divisor (Bölen). The dividend is 85, and the divisor is 6. The quotient (Bölüm) is 14, and the remainder (Kalan) is 1. The steps shown are:

$$\begin{array}{r} 8 \ 5 \\ - 6 \ | \ 14 \\ \hline 2 \ 5 \\ - 2 \ 4 \\ \hline 0 \ 1 \end{array}$$

Annotations with red arrows point to each part: 'Bölünen' points to the dividend 85, 'Bölen' points to the divisor 6, 'Bölüm' points to the quotient 14, and 'Kalan' points to the remainder 1.

Modulus işlemi sayesinde

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

mumkun olmaktadır.

# Modulus Soru

**Soru** - 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;
```

```
sayi = sayi / 10  $\Rightarrow$  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++ ;
```

→ 13

→ 30

→ 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 azaltırken 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.
```

# 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)

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 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);
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 (Mantıksal) Operators

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

Mantıktaki AND operatorunun Java'da 2 tane karşılığı 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 çarpmaya benzetilir.  
Sonucun 1 olması için  
tüm çarpılan sayılar 1 olmalıdır.  
1 tane bile sıfır olsa sonuc 0 olur.out

$$\begin{aligned}1 * 1 * 1 * 1 &= 1 \\1 * 0 * 1 * 0 &= 0 \\1 * 0 * 0 * 0 &= 0 \\0 * 1 * 1 * 1 &= 0\end{aligned}$$

# Logical (Mantıksal) Operators

## `&&` ile `&` in farkı

`&&` operatoru birbirine baglı mantıksal ifadeleri incelerken, **ilk false** değeri ile karşılastığında, sonucun **false** olacağını algılar ve geriye kalan mantıksal ifadeleri incelemeden hemen sonucu false olarak atar.

`&` operatoru ise birbirine baglı mantıksal ifadeleri incelerken, herbir mantıksal ifadenin sonucuna göre karar vermez, işlemin sonucuna kadar gider. Tüm işlem bittikten sonra sonuca atama yapar.

`&&` Operatoru tüm mantıksal ifadeleri kontrol etmeden sonuca gidebildiği için daha hızlıdır.

```
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 (Mantiksal) Operators

## 2- OR (veya) Operatoru ||

Mantiktaki OR operatoru ile ayni sekilde kullanilir.

|| operatoru birlestirdigi 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);
```

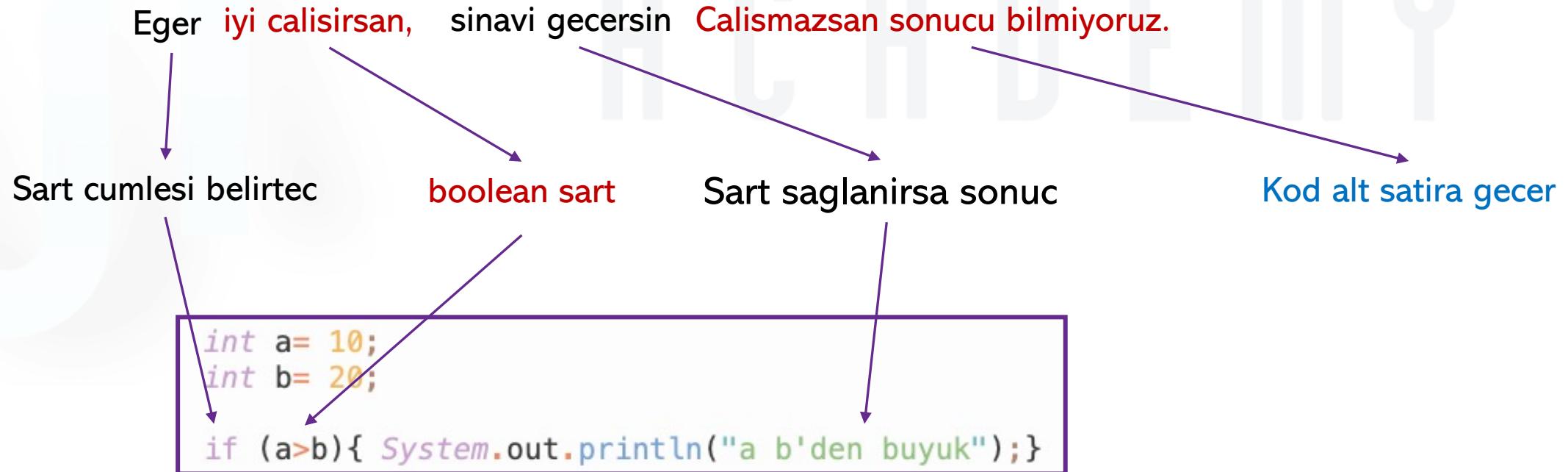
|| operatoru toplamaya benzetilir.  
Sonucun 0 olmasi icin  
tum toplanan sayilar 0 olmalıdır.  
1 tane bile bir olsa sonuc 0 olmaz

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.



# 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");
}
```

# 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 sayı”, 5 ile bolunebiliyorsa “Bes ile bolunebilen sayı” yazdirin.

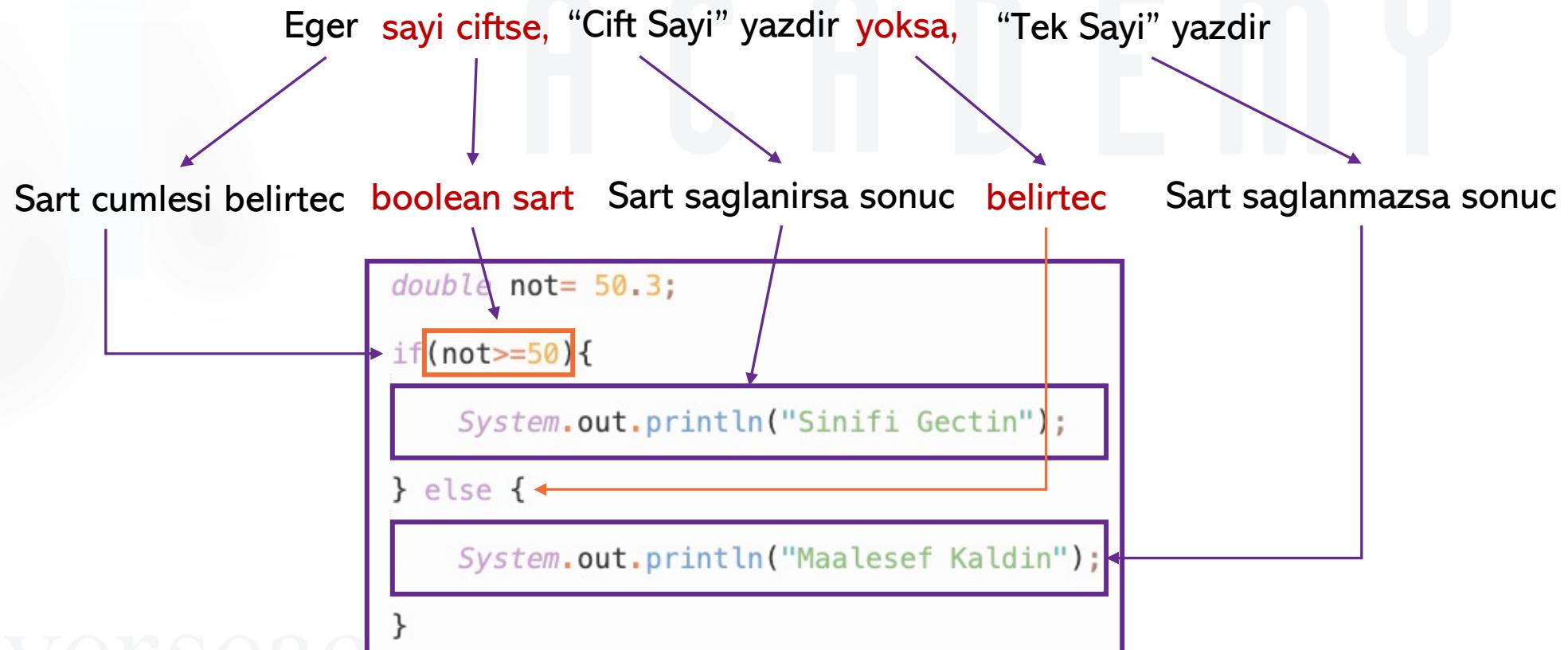
**Soru 4-** Kullanicidan bir ucgenin 3 kenar uzunlugunu alin, ucgen eskenar ise “Eskenar ucgen” yazdirin.

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

# If Else Statements

Basit if cumleleri kodun geri kalani ile ilgilenmiyordu.

Sorularda şartin sağlanması veya sağlanmaması durumunda yapılacaklar belli ise basit if cumlesi yeterli olmayacağıdır.



# 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");  
}
```

# 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 .. Yıl daha calisman gerekir” yazdirin.
- Soru 2-** Kullanicinin kilo(kg) ve boyunu(cm) isteyip vucut kitle endeksini hesaplayin ( $\text{kilo} * 10000 / (\text{boy} * \text{boy})$ ) vucut kitle endeksi 30'dan buyukse obez, 25-30 arasi ise kilolu, 20-25 arasi ise normal, 20'den kucukse zayif 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 çevirmek istediği birimi sorun, istediği birim metre veya santimetre ise çevirip yazdırın, yoksa “istediginiz birim sisteme kayitli degil” yazdırın.

# If Else Statements

## Soru ) Interview Question

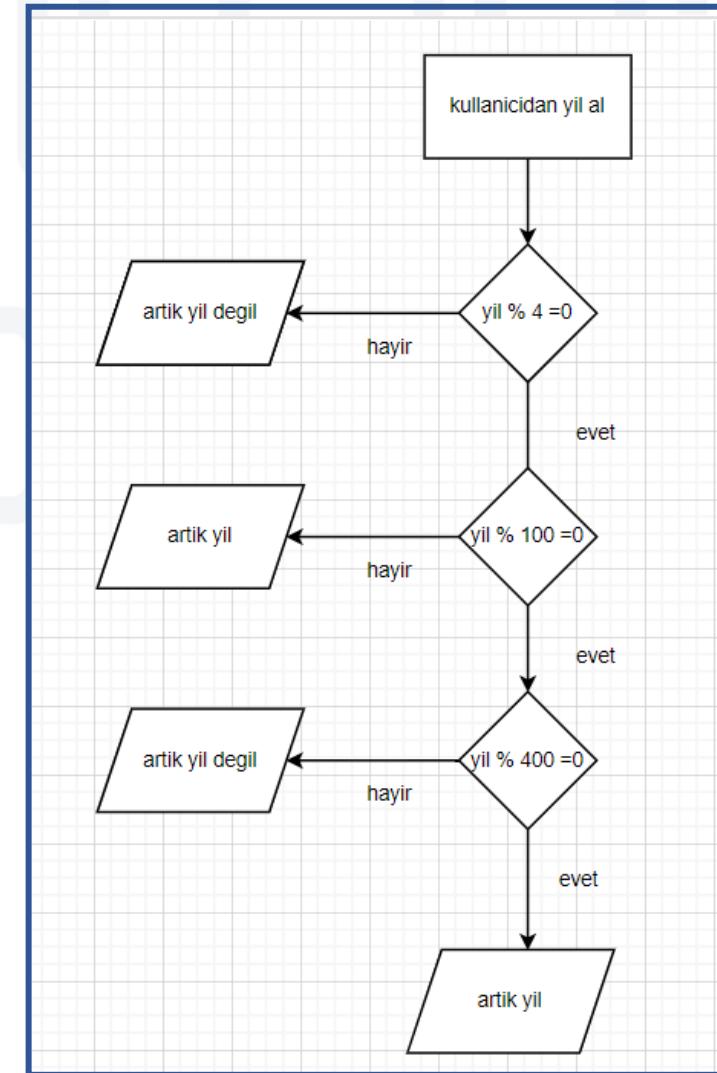
Kullanicidan artik yil olup olmadigini kontrol etmek icin yil girmesini isteyin.

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 .. Yıl daha calisman gerekir” yazdirin.

Buna benzer sorularda, degiskenlerden bir tanesini secip ona gore ana yapıyi kurmalı, 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

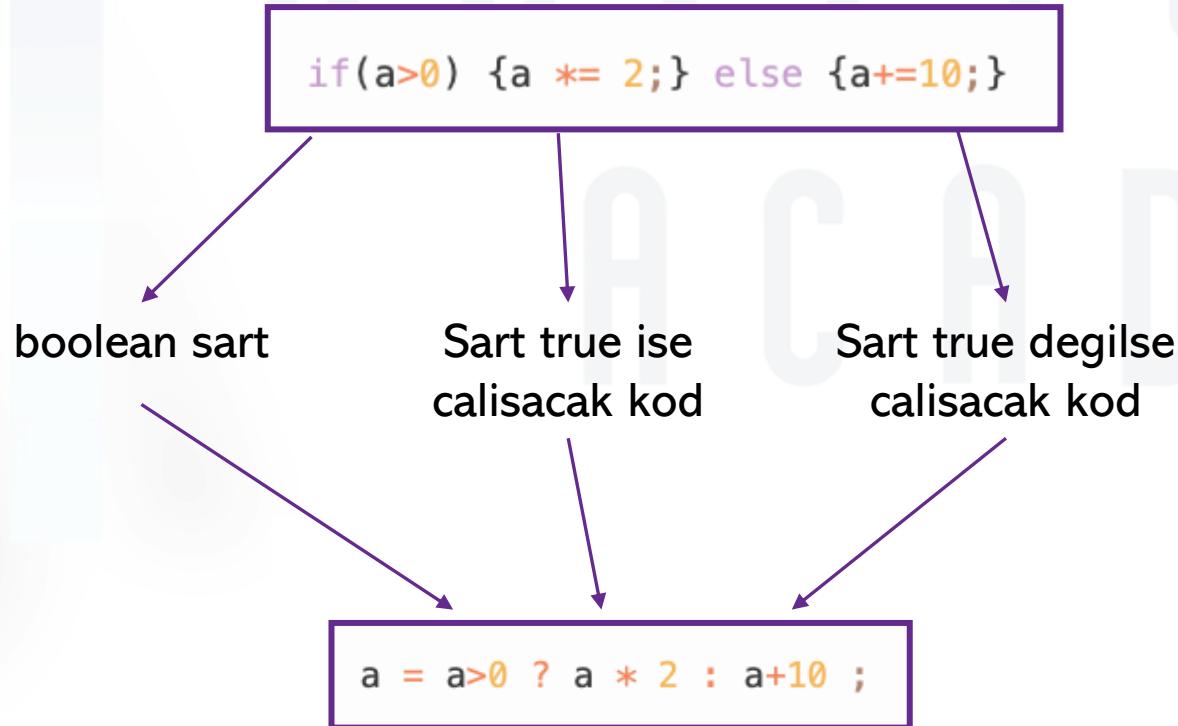
}else {
    // cinsiyet bilgisi hatali
```

# 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 aldigı urun adedi ve ve liste fiyatını alin, kullaniciya 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 katı olup olmadığını yazdırın.
- Soru 4-** Kullanicidan gunu ismini girmesini isteyin, girilen gun hafta ici bir gun ise “Simdi calisma zamani tatile .. gun var” seklinde hafta sonu tatiline kaç gun kaldığını yazdırın, girilen gun hafta sonu ise “Simdi dinlenme zamani” 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 katı” 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 üzere dizayn edilse de bazen kompleks islemleri de ternary ile yapmak isteyebilirsiniz (Tavsiye edilmez).

Ornek : Kullanicidan bir tamsayi alin.

Sayı pozitifse, cift sayı veya cift sayı degil seceneklerinden uygun olani yazdirin

Sayı pozitif degilse, 3 basamaklı 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 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);
```

# **Java**

## **Ders-03**

**Switch Statements**

**String Manipulations**

# 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("Salı");}
else if (rakam==3){System.out.println("Çarşamba");}
else if (rakam==4){System.out.println("Perşembe");}
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 kullaniminda **dikkat edilecek konular**.

- 1- Switch Statement'da switch parantezinde long, double, float ve boolean kullanilamaz.
- 2- Switch Statement'da switch parantezinde yazilan 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 degiliz, ancak bu durumda kodun break gorunceye kadar devam edeceğini unutmamamiz gereklidir.

case'leri gruplandirmak icin bu yontem kullanilabilir.

- 3- switch parantezine yazilan deger hic bir case ile uyusmazsa **default;** satiri devreye girer.  
(if-else if-if... lerin sonundaki else gibi)

# Switch Statements

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

**Soru 2-** Kullanicidan 2 basamaklı bir sayı alip, girilen sayiyi yazı ile yazdirin

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

**Soru 4-** Kullanicidan ISTQB kisaltmasından harfin anlamını öğrenmek istedigini alın ve girilen harfin karşılığını yazdirin.

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

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

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

# String Manipulations

Verilen bir String'i **hazir method'ları** kullanarak **degistirmeye** veya String üzerindeki **bazi bolumleri almaya** denir.

String manipulation yapılarken, degisikligin kalici olmasi isteniyorsa, atama yapılmalıdır.

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

String class'ının ozelliginden dolayı (ileride anlatılacak – immutable class), atama yapılmadan çalıştırılan method'lar variable'da kalıcı değişiklik 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 candir
```

# String Manipulations

3. str.equals( baskaStr ); Verilen iki String'in metinlerini karsilastirir. İki String birbiriyle aynı 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 aynı metinleri içeriyorsa **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 Candir";  
  
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 başladığı 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 parametreli 2 farkli kullanimi vardır

### 7.A- 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.B- 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( )** kullanabilir.

```
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 kullanilabilir

```
// 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 secersek 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
```

# 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 icermediğini kontrol eder, boolean sonuc döndürür.

Aranan String'in kaç tane olduğunu tespit edemez, sadece **var** veya **yok** cevabı 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(""));
```

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

# String Manipulations

## 10. str.startsWith ( baskaString );

Bir String'in baska 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 güzel,cok.";  
  
System.out.println(str.startsWith("J")); // true  
System.out.println(str.startsWith("Java")); // true  
System.out.println(str.startsWith("Java çok güzel,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 :** kullaniciidan bir mail alin

- mail @ icermiyorsa "gecersiz mail"
- mail @gmail.com icermiyorsa, "mail gmail olmali"
- mail @gmail.com ile bitmiyorsa, "mailde yazim hatasi var"

yazdirin.

# 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 guzel,cok.";  
  
System.out.println(str.indexOf('a'));  
// buldugu ilk a'nin index'ini verir : 1  
  
System.out.println(str.indexOf( ch: 'a', fromIndex: 1));  
// 1.index ve sonrasinda 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.

**0** ve pozitif sayilar index olarak kullanilabilecegi icin, Java aradigimiz metin aranan String'de olmadiginda bize **-1** döndürerek, durumu rapor eder.

```
String str="Java çok güzel,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 kullanımını 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 güzel, 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 başa 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 oncesinde arama yapar
System.out.println(str.lastIndexOf( ch: 'x')); // -1
System.out.println(str.lastIndexOf( str: "x", fromIndex: 10)); // -1
```

# **Java**

## **Ders-04**

### **String Manipulations**

### **For Loops**

# String Manipulations

13. str.lastIndexOf ( baskasString/char );

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

**Soru 2 :** Kullanicidan bir String ve aranacak metin alin. Aranan metnin String icerisinde kullanımını 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

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) seçebiliriz.

# 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, rakam veya \_

\W : harf, rakam veya \_ olmayan hersey

# String Manipulations

## 16. str.replaceAll ( Regex , yeniDeger );

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 : 26.70 €

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

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

output : Java ogrenmek ne guzel.

# String Manipulations

**Soru 4 :** Kullanicidan bir sifre isteyip, asagidaki şartları kontrol edin ve kullaniciya düzeltmesi gereken tüm eksikleri söyleyin, eger tum şartları saglarsa, "sifre basariyla kaydedildi" yazdirin

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

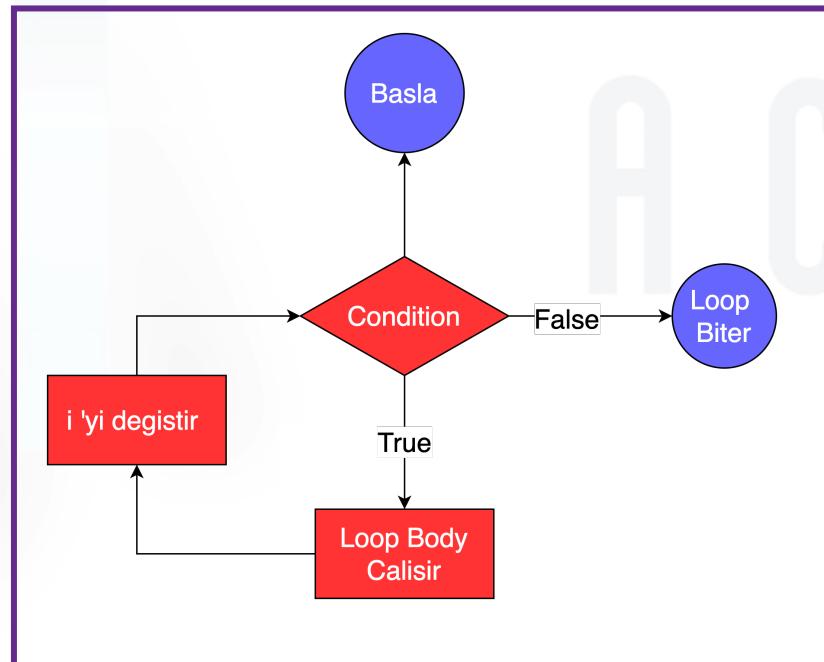
**Soru 5 :** Kullanicidan isim ve soyismini ayrı ayrı alin.

- 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 :( yazdirin.

# 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 sağlanlığında 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 sayiları aynı satırda aralarında bir boşluk bırakarak yazdırın.

**Soru 2-** Kullanıcıdan pozitif bir tamsayı alın, 1'den girilen sayıya kadar(girilen sayı dahil) 7 ile bolunemeyen 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 tamsayıları yazdirin, sira

- 3 ile bolunebilen bir sayıya gelirse, sayı yerine fizz
- 5 ile bolunebilen bir sayıya gelirse sayı yerine buzz
- hem 3 hem 5 ile bolunebilen bir sayıya gelirse sayı 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 sayı 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, Ornegin bir carpim tablosu hazırlamak veya bir futbol liginde oynanacak maclari planlamak için tek bir loop yeterli olmaz.

Nested for-Loop ihtiyacı iki sekilde karsimiza cikabilir.

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 deger aldiginda, 2. sayı bastan sona tum degerleri aliyor

1.Sayı bir deger arttiginda, 2. sayı bastan sona tum degerleri yeniden aliyor

# Nested 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 dongü 1'den dış loop'un i degerine kadar gidiyor.

Soru – Aşağıdaki şekilleri yazdırın

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

**Java**

## **Ders-05**

**Method Olusturma**

**While Loops**

**Do-While Loops**

**Scope**

# Method Olusturma ve Kullanma

Method'lar istedigimiz islemleri bizim adimiza yapan kod bloklaridir.

Belirli bir isi yapmak icin tasarlanmis robotlar gibidirler. Baslangicta tasarlama 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 olusturmayi tercih ederiz ?

- 1- Projemiz icerisinde tekrar tekrar kullanacagimiz 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 kayıt gibi  
islemler de binlerce kez tekrarlanacaktır.

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

```
===== YILDIZ KOLEJI =====
===== ANA MENU =====
1- Okul Bilgileri Goruntule
2- Ogretmen Menu
3- Ogrenci Menu
Q- CIKIS
```

```
===== YILDIZ KOLEJI =====
===== OGRENCI MENU =====
1- Ogrenci Listesi Yazdir
2- Soyisimden Ogrenci Bulma
3- Sınıf ve Sube İle Ogrenci Bulma
4- Bilgilerini Girerek Ogrenci Ekleme
5- Kimlik No İle Kayit Silme
A- ANAMENU
Q- CIKIS
```

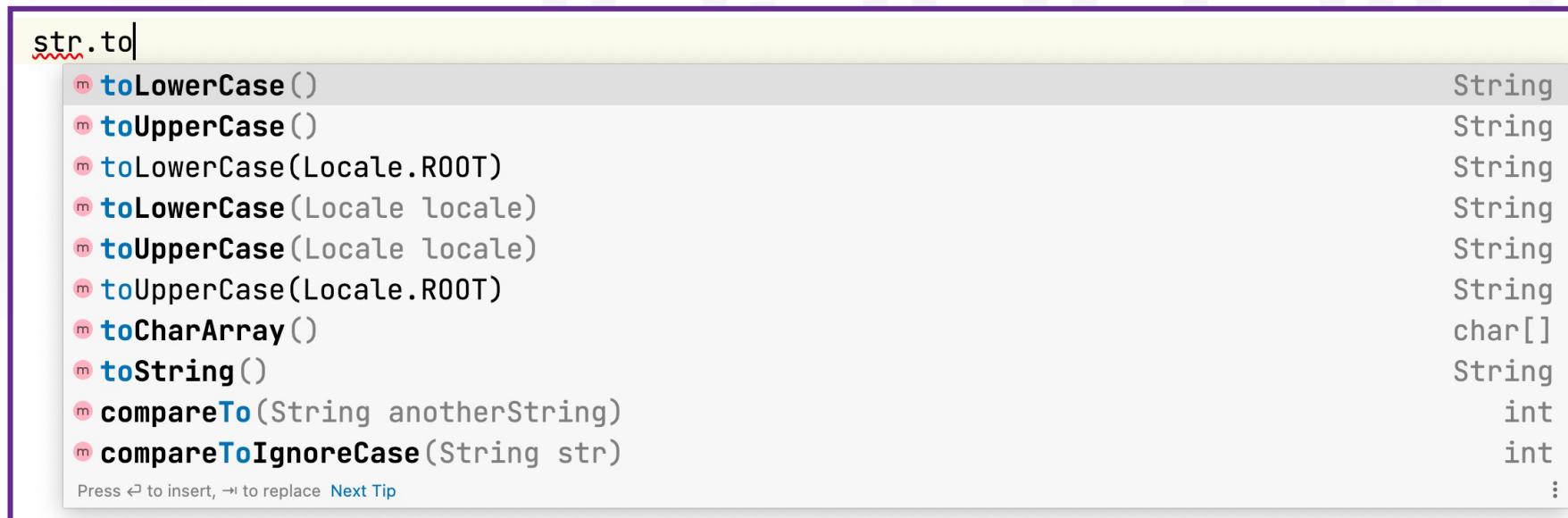
```
===== YILDIZ KOLEJI =====
===== OGRETMETEN MENU =====
1- Ogretmenler Listesi Yazdir
2- Soyisimden Ogretmen Bulma
3- Branstan Ogretmen Bulma
4- Bilgilerini Girerek Ogretmen Ekleme
5- Kimlik No İle Kayit Silme
A- ANAMENU
Q- CIKIS
```

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.



# 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 yatirmaktır, bize bir makbuz getirmesi degildir.

# Method Olusturma ve Kullanma

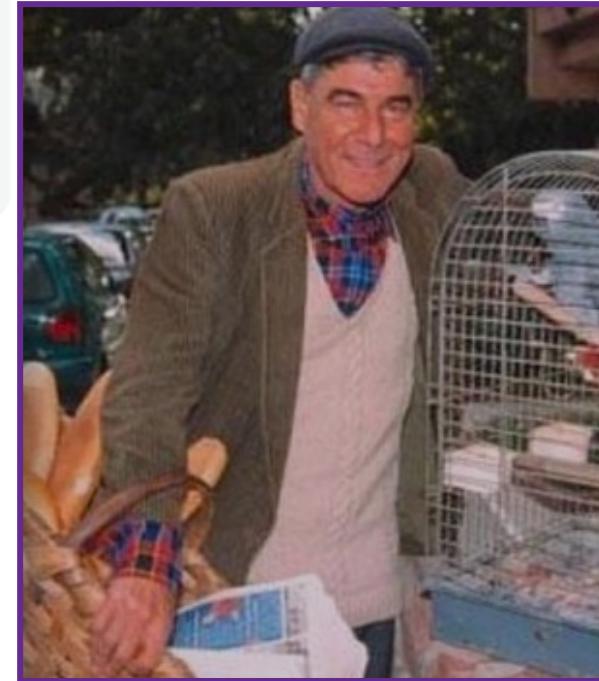
## Method Turleri

### 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 direkt 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 sonucu 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	
no modifier	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 direk ulasabilir.

main method static olarak isaretlendiğinden (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 **yazdirma 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 sayiTopsaDondur(int sayi1, int sayi2){  
    return sayi1+sayi2;  
}  
  
public static void main(String[] args) {  
  
    int sonuc= sayiTopsaDondur(20,30);  
}
```

# Method Olusturma ve Kullanma

6- **method body** : Suslu parantezler arasında kalan ve kodlarımızi yazdigımız bolumdur.

**Soru** : Method nerede olusturulabilir ?

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

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

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

Method'lar cagrildidan calismaz, cagrilinca 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.

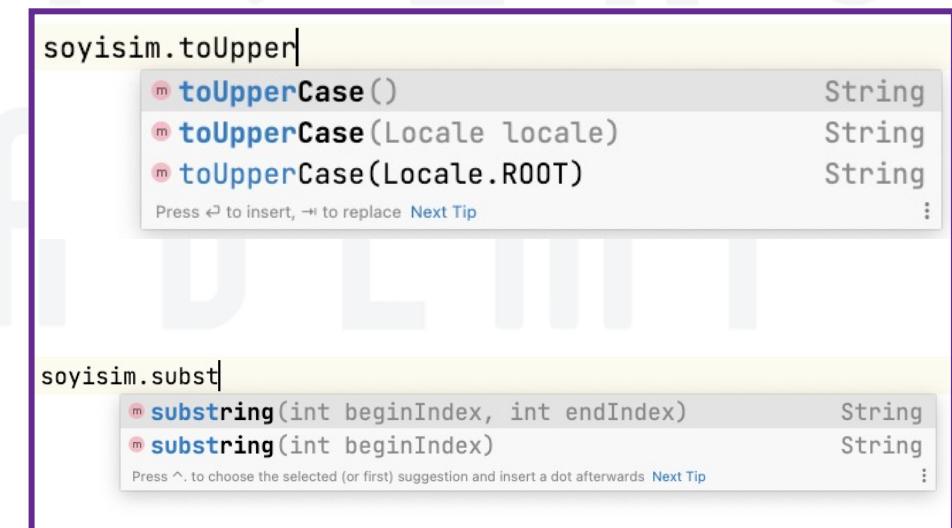
# Method Overloading

**Method overloading** : Bir class'da aynı isimde fakat farklı method signature'ına sahip methodların bulunmasıdır.

**Method overloading**'in amacı aynı işlevi 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 kullaniciya ihtiyacına uygun method'u kullanma imkani vermis oluruz.

# Method Overloading

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

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

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

**1- parametre sayisini degistirmek**

**2- parametrelerin data turunu veya data turu farkli olan parametrelerin yerlerini degistirmek.**  
(ayni 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*sayı2;
}
public static int carpim(int sayı1, int sayı2){

    return sayı1*sayı2;
}

public static double carpim(double sayı1, double sayı2,double sayı3){

    return sayı1*sayı2*sayı3;
}
```

# Method Overloading

Bir class'da aynı isimde birden fazla method olduğunda Java hangisini kullanacağına 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 yapacağını seçer.

```
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 bunları otomatik yazmadığı için manuel olarak yazmamız gereklidir.

```
int s=10;

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

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

Baslangic degerini ve artis degerinin manuel yazılması, while loop'u baslangicta kullanıssız gibi gösterebilir.

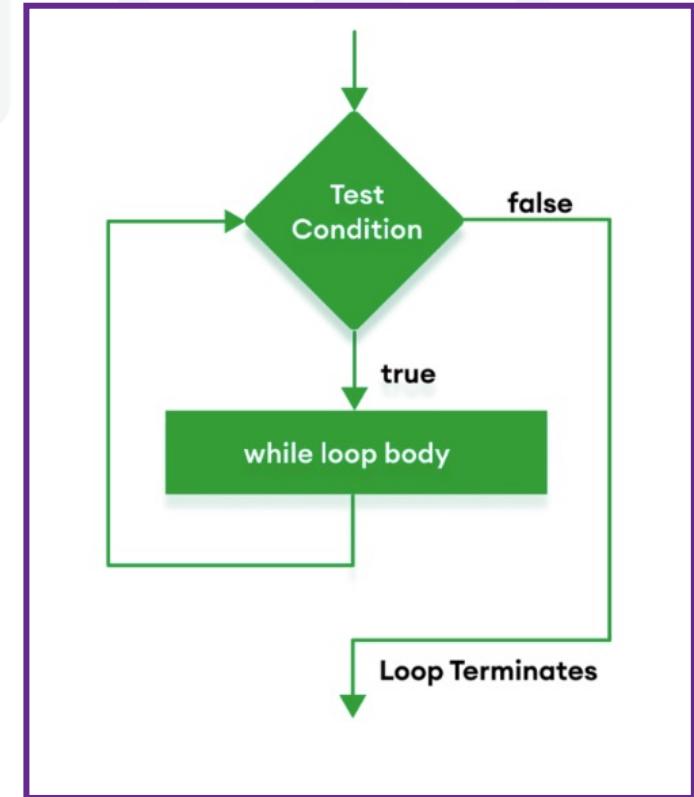
Ozellikle artis/azalis miktarı yazılmazsa kod sonsuz loop'a girecek ve bize sorun çıkaracaktır.

# 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 istiyoruz 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 istiyoruz, kodun durmasini kullanicinin girecegi “**cikis icin O'a basiniz**” gibi bir degere baglayip, kodu tekrar calistirabiliriz.

# While Loop

**Soru :** Kullanicidan toplanmak üzere sayilar isteyin toplam 500 olur veya gecerse 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 sartlari 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 sayi 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 basamaklı 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 sayıyı sayı adedine ve toplama eklemeyin

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

# Do While Loop

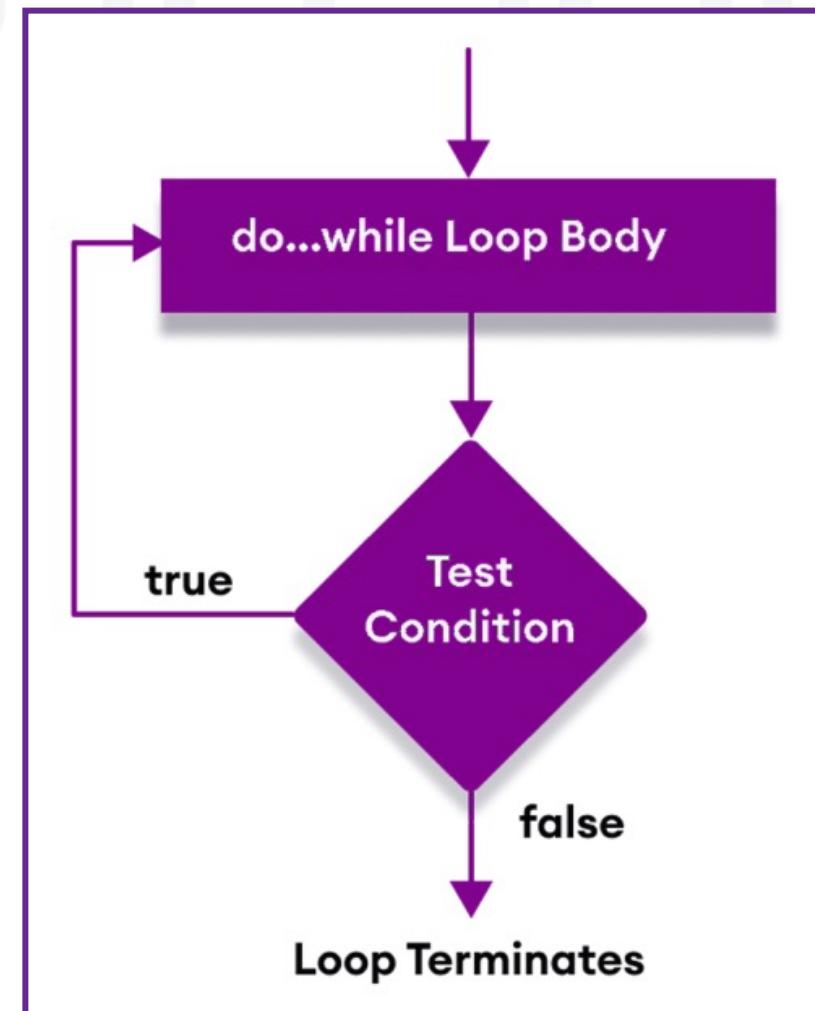
do-while loop, belirtilen koşul doğru olana kadar programın bir bölümünü tekrar tekrar calistirmak için kullanılır.

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 calisır.

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 şartlara 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

# **Java**

## **Ders-06**

**Scope**  
**Arrays**

# Scope

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.

```
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 blogu içerisinde oluşturulan variable'lardır.

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

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

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

```
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 deger atanma mecburiyeti yoktur.

Java variable'i olusturdugumuzu ama deger atamasini ileriki satirlarda yapacagini kabul eder ve CTE vermez.

Ancak local variable'lara deger atamasi yapmadan kullanmaya kalkisirsaniz Java **olmayan degeri** kullanamayacagi icin 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 dışından kullanılamazlar.

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

```
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);
```

```
}
```

# Scope / Class Level Variables



Doktor



Hemsire



Labaratuvar ve personel



Hasta



Tıbbi 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.



Hemsire

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



Tibbi cihazlar

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

# Scope / Class Level Variables

## 2- Class level variables

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

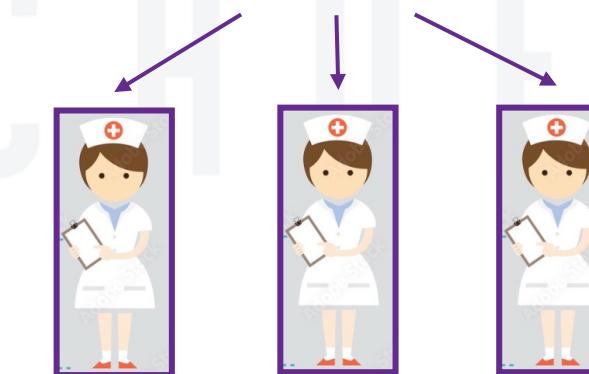
hangi variable'larin ise **tum objeler icin objeye ozel oldugu**, dolayisiyla **static olmamasi** gerektigi asikardir.



H.Ismi : Yildiz

H.adres: Ankara/Cankaya

H.Tel : 2445566



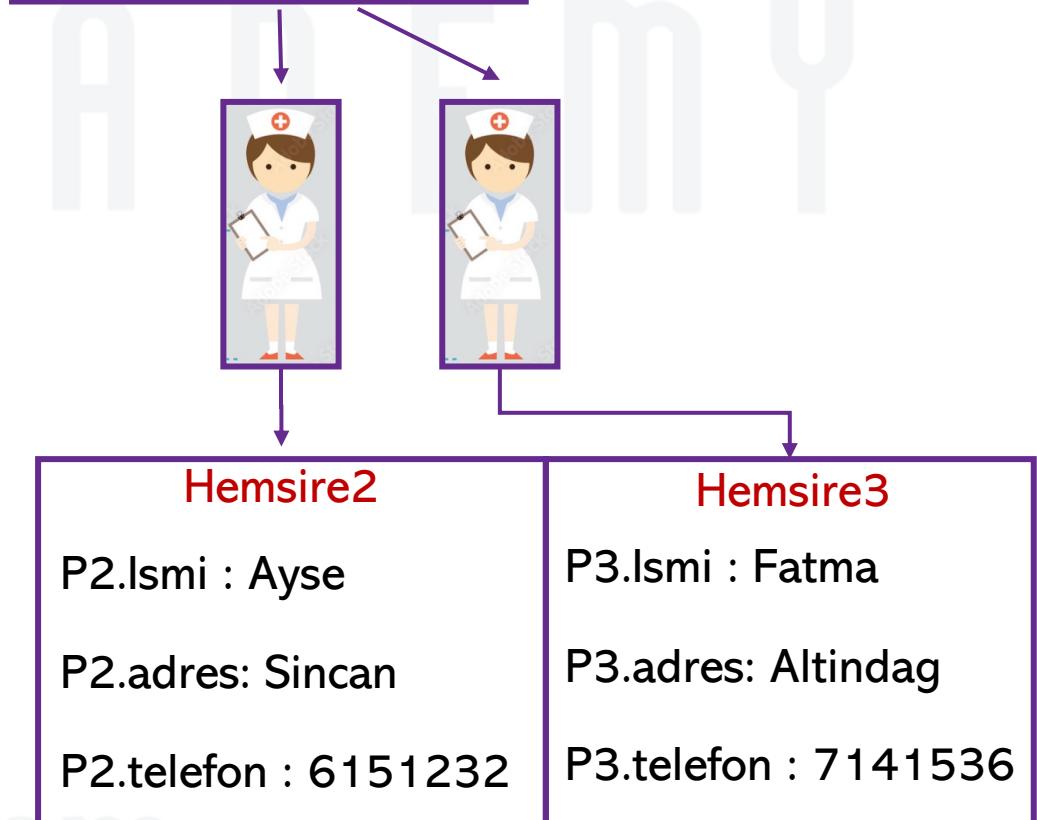
Hemsire1	Hemsire2	Hemsire3
P1.Ismi : Yildiz	P2.Ismi : Ayse	P3.Ismi : Fatma
P1.adres: Cankaya	P2.adres: Sincan	P3.adres: Altindag
P1.Tel : 4164352	P2.Tel : 6151232	P3.Tel : 7141536

# 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.Telpon : 2445566

# 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 tanımlanan** degerleri assign eder ve onlari kullanir.

**Default degerler :**

String : null

Sayısal 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'lar 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 belirtilmemi";
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 belirtilmemi
    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 Level

**Scope** : Class icerisinde olusturulan variable'larin kapsamini (nereden eriselebilecegini) 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 / Class Level Variables

**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 düzeltilmesi 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 }
```

# Arrays

Bugune kadar kullandigimiz data turleri sadece 1 variable'a 1 deger atamasi yapabiliyor.

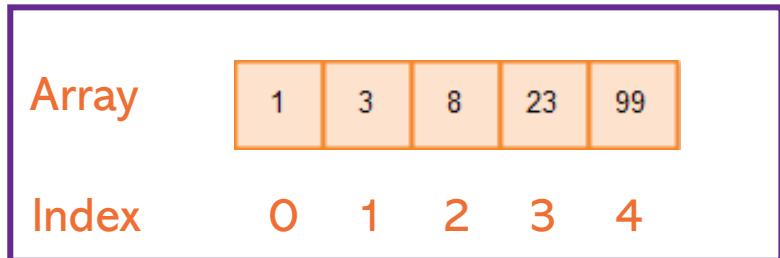
```
int sayi=20;  
String str="Java Candir";  
char ilkHarf='A';  
boolean aktifMi=true;
```

Ancak Java gibi kompleks uygulamalar geliştirmeye uygun bir programlama dilinde birden fazla eleman barindirabilen yapılara da ihtiyac vardır.

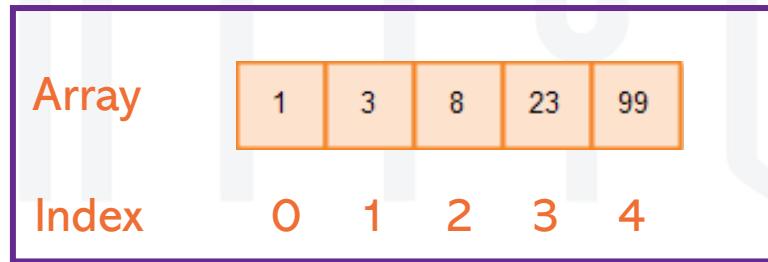
Ornegin; bir sınıfındaki öğrenci listesini oluşturmak, online satış yapan bir uygulamadaki satılan ürünlerin listesini tutmak için birden çok eleman barındıran yapılar gereklidir.

Array(dizi)'ler Java'da **aynı data turundan birden fazla eleman** barindirabilen objelerdir.

Array'ler her bir eleman için ayrı bir alan ayıırlar ve bu elemanlara index bilgisi ile ulaşabiliriz.



# Arrays



1- Bir array olusturulurken 2 sey deklare edilmek zorundadir.

A- Icine konulacak elementlerin data turu

Bir array icinde farkli data turundeki element olamaz.

Array'ler **primitive data** turundeki datalarin degerlerini, **non-primitive data** turundeki datalarin ise referanslarini barindirirlar.

B- Icine kac element konulacagi (length)

Olusturulan bir array'e basta belirtilen index'den daha fazla element atanamaz veya element sayisi azaltilmaz.

```
int [ ] arr = new int [5];
```

# Arrays

2- Bir array 2 sekilde deklare edilir

A-

```
int arr[];
```

IntelliJ sariya boyar

B-

```
int[] arr;
```

Deklare ettik ama deger atamadik, java **referansi** olusturur ama length belli olmadigindan **objeyi** olusturamaz.

3- Bir array'e 2 sekilde deger atanabilir

A- Direk degerler atanabilir

```
int[] arr={1,3,8,23,99};
```

```
[1, 3, 8, 23, 99]
```

B- Uzunlugu belirtilerek olusturulur ama elemanlara deger ataması yapilmaz.

Bu durumda Java belirtilen uzunlukta ve default degerlere sahip bir array olusturur.

```
int[] arr= new int[5];
```

```
[0, 0, 0, 0, 0]
```

# Arrays

4- Bir array'in **length**'i nasıl bulunur ?

```
int[] arr= {1,3,8,23,99};  
  
System.out.println(arr.length); // 5
```

NOT: Array'deki length bir method değil uzunluğu tutan bir variable'dır. Dolayısıyla yanında () parantez yoktur.

5- Bir array'de **istenen index**'deki elemana nasıl ulasılır ?

Array'deki herhangi bir elemana ulaşmak veya değiştirmek için index kullanılır.

Array'de olmayan bir index'i kullanmak isterseniz ArraysIndexOutOfBoundsException exception verir.

```
int[] arr= {1,3,8,23,99};  
  
System.out.println(arr[2]); // 8  
arr[2]=10;  
System.out.println(arr[2]); // 10
```

# Arrays

6- Bir array'in **tum elemanları** nasıl yazdırılır ?

For-Loop kullanalım.

```
int[]arr= {1,3,8,23,99};

for (int i = 0; i <arr.length ; i++) {
    System.out.print(arr[i]+ " ");
}
```

1 3 8 23 99

7- Bir array nasıl yazdırılır ?

Array direkt yazılmaz, direkt yazdırmak isterseniz array'i değil referansını yazdırır.

Array'i yazdırmak için Arrays class'ından `toString()` method'u kullanılmalıdır.

```
int[]arr= {1,3,8,23,99};

System.out.println(arr); // [I@2752f6e2
```

```
int[]arr= {1,3,8,23,99};

System.out.println(Arrays.toString(arr)); // [1, 3, 8, 23, 99]
```

# Arrays

- Soru 1-** Verilen bir int array'in tum elemanlarini 2 artirip bize donduren bir method olusturun. Eski array'i yeni haliyle kaydedin.
- Soru 2-** Verilen bir array'deki pozitif tamsayıları toplayıp sonucu bize donduren bir method yazınız.
- Soru 3-** Verilen bir array'deki tum elementleri bir saga kaydirip, sondaki elementi de basa tasiyacak bir method olusturun, array'i yeni haliyle kaydedin.
- Orn : input : [4,5,6,7] array'in son hali. : [7,4,5,6]
- Soru 4-** Verilen bir array'de istenen bir elemanın var olup olmadigini ve varsa kaç kere kullanildigini yazdırın bir method olusturun.
- Soru 5-** Kullanıcıdan array'in boyutunu ve elementlerini alıp array'i olusturan ve bize donduren bir method olusturun.
- Soru 6-** Verilen String bir array'deki en uzun ve en kısa kelimeleri yazdırın bir method olusturun.
- Soru 7-** Verilen bir array'e istenen bir elemani ekleyip bize donduren bir method yazın, eski array'e yeni degeri atayın.

UNITYVERSE  
ACADEMY

**Java**  
**Ders-07**  
**Arrays**  
**Arrays Lists**

# Arrays

8- Bir array'in elemanları nasıl sıralanır ?

```
int[] arr={1,3,9,5,4,6};  
  
System.out.println(Arrays.toString(arr)); // [1, 3, 9, 5, 4, 6]
```

Array'lerde sıralama yapmak için Arrays class'ından yardım almak gereklidir.

```
Arrays.sort(arr);  
  
System.out.println(Arrays.toString(arr)); // [1, 3, 4, 5, 6, 9]
```

Arrays.sort( ) method'u array'i **Natural Order**'a göre sıralar.

Buyukten kucuge sıralamak isterseniz, sort( ) ile sıralayıp, loop ile tersine çevirmek gereklidir.

# Arrays

9- Bir array'de istenen elemanın varlığı **nasıl kontrol edilir** ?

Array'lerde istenen bir elemanın varlığını kontrol etmek için **binarySearch( )** method'u kullanılır.

Ancak, **binarySearch( )** kullanmadan önce **MUTLAKA** **sort( )** ile sıralama yapılmalıdır.

```
int[] arr={1,3,9,5,4,6};

System.out.println(Arrays.toString(arr)); // [1, 3, 9, 5, 4, 6]

Arrays.sort(arr);

System.out.println(Arrays.toString(arr)); // [1, 3, 4, 5, 6, 9]

System.out.println(Arrays.binarySearch(arr, key: 1)); // 0

System.out.println(Arrays.binarySearch(arr, key: 6)); // 4
```

# Arrays

Arrays.sort( ) kullanilmadan da binarySearch( ) calisir ancak sonuclarinin ne olacagi **BILINEMEZ**.

```
int[] arr={1,3,9,5,4,6};

System.out.println(Arrays.toString(arr)); // [1, 3, 9, 5, 4, 6]

System.out.println(Arrays.binarySearch(arr, key: 1)); // 0

System.out.println(Arrays.binarySearch(arr, key: 6)); // -3

System.out.println(Arrays.binarySearch(arr, key: 8)); // -3
```

binarySearch( ) ile array'de olmayan bir eleman aranırsa, olması gereken sırayı **- ile VERİR**.

```
int[] arr={1,3,9,5,4,6};
Arrays.sort(arr);
System.out.println(Arrays.toString(arr)); // [1, 3, 4, 5, 6, 9]

System.out.println(Arrays.binarySearch(arr, key: -3)); // -1

System.out.println(Arrays.binarySearch(arr, key: 7)); // -6

System.out.println(Arrays.binarySearch(arr, key: 18)); // -7
```

# Arrays

10- İki array'in esitligi **nasıl kontrol edilir** ?

Arrays.equals(arr1,arr2); herbir index icin elemanlari kontrol eder, tum index'lerdeki degerler karsilikli esit ise **true**, farklilik varsa **false** döner.

```
int arr1[] = {2, 1, 7, 6};  
int arr2[] = {7, 1, 6, 2};  
System.out.println(Arrays.equals(arr1, arr2)); → false  
  
int arr3[] = {3, 2, 7, 8, 11};  
int arr4[] = {7, 3, 8, 2, 12};  
Arrays.sort(arr3);  
Arrays.sort(arr4);  
System.out.println(Arrays.equals(arr3, arr4)); → false  
  
int arr5[] = {4,2,6,8,11};  
int arr6[] = {11,4,8,2,6};  
Arrays.sort(arr5);  
Arrays.sort(arr6);  
System.out.println(Arrays.equals(arr5, arr6)); → true
```

# Arrays

## 11- Bir String'i array'e cevirmek

str.split( **StringAyirac** ); bir String'i istedigimiz parcalara ayırarak bir array'e çevirir.

```
String str= "Ali topu at, at Ali at";

String[] arrVirgul=str.split( regex: ", " );
System.out.println(Arrays.toString(arrVirgul));
// [Ali topu at, at Ali at]

String[] arrSpace=str.split( regex: " " );
System.out.println(Arrays.toString(arrSpace));
// [Ali, topu, at,, at, Ali, at]

String[] arrHiclik=str.split( regex: "" );
System.out.println(Arrays.toString(arrHiclik));
// [A, l, i, , t, o, p, u, , a, t, , , a, t, , A, l, i, , a, t]
```

# Arrays

ONEMLI NOT : Varolan bir array'e yeni deger atanabilir mi ?

Varolan bir array'e elementleri yazarak yeni deger atamasi yapamayiz

```
int[] arr= {1,3,8,23,99};  
  
arr= {1,2,3};  
  
int[] arr2=new int[4];  
  
arr2={3,4,5};
```

```
int[] arr= {1,3,8,23,99};  
  
arr= new int[3]; // [0, 0, 0]  
  
int[] arr2=new int[4];  
  
arr2=new int[6]; // [0, 0, 0, 0, 0, 0]
```

Ancak new keyword ile yeni bir deger atayabiliriz.

Bu varolan array'in uzunlugunu degistirmek degil, yepeni bir array olusturmak oldugundan Java CTE vermez.

# Arrays

Soru :

What is the result of the following?

```
int[] random = { 6, -4, 12, 0, -10 };  
int x = 12;  
int y = Arrays.binarySearch(random, x);  
System.out.println(y);
```

- A. 2
- B. 4
- C. 6
- D. The result is undefined.
- E. An exception is thrown.
- F. The code does not compile.

# Multidimensional Arrays

Multidimensional array'ler, içerisinde array bulunduran array'lerdir.

```
int[][] arr= {{3,1,2,4},{1,2},{3,4,5},{10},{2,7}};
```

Bu şekilde deklare edilmiş olan arr array'ini tek katlı olarak yazmak istersek, length'i 5 olan bir array goruruz.

```
int[] arr= {arr[0], arr[1], arr[2], arr[3], arr[4]};
```

outer array

Multidimensional array'lerde en distaki array'e **outer array**, içerisindeki array'lere ise **inner arrays** denir.

```
arr[0] => {3,1,2,4}  
arr[1] => {1,2}  
arr[2] => {3,4,5}  
arr[3] => {10}  
arr[4] => {2,7}
```

inner arrays

# Multidimensional Arrays

Multidimensional array'lerde, bir element'e ulasmak icin once elementin içinde oldugu inner array'e, sonra o inner array'deki index'i kullanarak da elemente ulasmaliyiz.

```
int[][] arr= {{3,1,2,4},{1,2},{3,4,5},{10},{2,7}};
```

0.      1.      2.      3.      4.

0. 1. 2.

Ornegin; 5 elementine ulasmak icin once outer array'den inner array'e ulasalim

Sonra inner array'den 5 elementine ulasalim

```
arr[2][2]
```

# Multidimensional Arrays

```
int[][] arr= {{3,1,2,4}, {1,2}, {3,4,5}, {10}, {2,7}};
```

0.      1.      2.      3.      4.

4 elementi icin → arr [0][3]

2 elementi icin → arr [1][1]

5 elementi icin → arr [2][2]

10 elementi icin → arr [3][0]

7 elementi icin → arr [4][1]

# Multidimensional Arrays

Yazdırma işlemi yapmak istediğimizde, yazdıracağımızın ne olduğunu bilmemiz gereklidir.

```
int[][][] arr= {{3,1,2,4},{1,2},{3,4,5},{10},{2,7}};
```

0.      1.      2.      3.      4.

1- Array'in içerisinde bulunan elementlerden birini yazdırmak istersek direkt yazdırabiliriz.

```
System.out.println(arr[1][1]); // 2  
System.out.println(arr[2][0]); // 3  
System.out.println(arr[4][1]); // 7
```

2- Inner array'lerden birini yazdırmak istersek `Arrays.toString()` kullanırız.

```
System.out.println(Arrays.toString(arr[1])); // [1, 2]  
System.out.println(Arrays.toString(arr[2])); // [3, 4, 5]
```

3- Outer array'i yazdırmak istersek `Arrays.deepToString()` kullanırız.

```
System.out.println(Arrays.deepToString(arr));  
// [[3, 1, 2, 4], [1, 2], [3, 4, 5], [10], [2, 7]]
```

# Multidimensional Arrays

Multi Dimensional Array'in tum elementlerine ulasmamiz gerektiginde, katman sayisi kadar ic ice for loop olusturmaliyiz

Ornegin 2 katli asagidaki array'in tum elementlerini yazdirmak istedigimizde, ic ice 2 loop kullanmaliyiz.

```
int[][] arr= {{3,1,2,4},{1,2},{3,4,5},{10},{2,7}};
```

```
for (int i = 0; i < arr.length ; i++) {  
  
    for (int j = 0; j <arr[i].length ; j++) {  
  
        System.out.print(arr[i][j] + " ");  
    }  
}
```

Output :

```
3 1 2 4 1 2 3 4 5 10 2 7
```

# Multidimensional Arrays

**Soru 1-** Verilen 2 katlı bir array'de bulunan çift sayıları toplayıp, sonucu yazdırın bir method oluşturun.

**Soru 2-** Verilen 2 katlı bir array'de aynı index'e sahip elementleri toplayıp, yeni oluşturacağımız tek katlı bir array'e bu toplamları atayın.

input : int[][] arr = {{3,4,5}, {2,3,6,7}};

output: [5, 7, 11]

**Soru 3-** Verilen 2 katlı bir array'de her bir iç array'deki elementleri toplayıp, yeni oluşturacağımız tek katlı bir array'e bu toplamları atayın.

input : int[][] arr = {{3,1,2,4},{1,2},{3,4,5},{10},{2,7}};

output: [10, 3, 12, 10, 9]

**Soru 4-** Verilen 2 katlı bir array'de bulunan tüm sayıların çarpını bize dönduren bir method oluşturun.

**Soru 5-** Verilen 2 katlı bir array'de her bir inner array'in son elementlerinin toplamını yazdırın bir method oluşturun.

# Array List

ArrayList, dinamik ve yeniden boyutlandirilabilir bir array'dir.

Array'in dezavantajları uzunlugunu en basta belirlemek zorunda olmamız ve uzunluğun sabit olmasıdır. Sabit uzunluk esnek çalışmaya imkan tanımadığından array liste tutmakta çok kullanılmaz.

ArrayList array altyapısını kullanmakla beraber element ekleme ve silme işlemlerine açıktır. Ancak array altyapısı sebebiyle eklemeleri birer birer yapabiliriz. ( veya uzunluğu belirli başka bir ArrayList ekleyebiliriz)

```
List<Integer> sayilar= new ArrayList<>();  
sayilar.add(10);  
sayilar.add(20);  
sayilar.add(30);  
  
System.out.println(sayilar);
```

# Array List

- 1-lleride gorecegimiz üzere List bir class degil interface'dir. Interface'lerden obje olusturulamadigi icin de esitligin saginda List<>() kullanilamaz.

```
List<Integer> sayilar1 = new List<>();  
List<Integer> sayilar2 = new ArrayList<>();  
List<Integer> sayilar3 = new ArrayList<Integer>();
```

- 2- List olusturabilmek icin esitligin sag tarafinda List interface'inin child class'i olan ArrayList<>() veya LinkedList<>() kullanilabilir. Biz simdilik ArrayList'i kullanacagiz.
- 3- Esitligin saginda < > icerisinde data turu yazilmasi mecburi degildir ama istersek yazabiliriz.

# Array List

## 1- ArrayList'e eleman ekleme

```
public static void main(String[] args) {  
  
    List<Integer> sayilar = new ArrayList<>();  
  
    sayilar.add(10); // [10]  
    sayilar.add(20); // [10, 20]  
  
    sayilar.add(index: 1, element: 25); // [10, 25, 20]  
  
    List<Integer> liste = new ArrayList<>();  
    liste.add(101); // [101]  
    liste.add(102); // [101, 102]  
  
    liste.addAll(sayilar); // [101, 102, 10, 256, 20]  
  
    liste.addAll(index: 1,sayilar);  
    // [101, 10, 25, 20, 102, 10, 256, 20]  
}
```

- List'e elemanlar birer birer eklenir ve ekleme sirasina gore listeye yerlestilir.
- List'te istedigimiz bir index'e eleman ekleyebiliriz. Bu durumda ekledigimiz index ve sonrasindaki elementler birer basamak geriye kaydirilir.
- List'tin sonuna istedigimiz bir list'i de ekleyebiliriz
- Eklenecek list, list'tin sonuna degil istedigimiz bir index'e de eklenebilir. Bu durumda yine o index ve sonrasindaki elementler eklenen listenin uzunlugu kadar geriye kaydirilacaktir.

# Array List

## 2- list'in boyutunu belirleme

**list.size( )** Method'u List'in boyutunu bize dondurur.

## 3- list'in bos olup olmadigini kontrol etme

**list.isEmpty ( )**

```
List<Integer> sayilar = new ArrayList<>();
```

```
System.out.println(sayilar.size()); // 0
```

```
sayilar.add(10);
```

```
sayilar.add(20);
```

```
System.out.println(sayilar.size()); // 2
```

```
List<Integer> sayilar = new ArrayList<>();
```

```
System.out.println(sayilar.isEmpty()); // true
```

```
sayilar.add(10);
```

```
sayilar.add(20);
```

```
System.out.println(sayilar.isEmpty()); // false
```

# Array List

- 4- list'de bir elementin olup olmadigini kontrol etme. boyutunu belirleme

list.contains ( arananObje )

NOT : Olusturulacak list uzun ise elementleri tek tek eklemek yerine bir array olusturup, for-loop ile list'e atanabilir.

```
int[] arr={2,3,4,5,3,6,7,3,8,9,1,2,5,3,8,5};

List<Integer> sayilar=new ArrayList<>();

for (int i = 0; i < arr.length ; i++) {
    sayilar.add(arr[i]);
}

System.out.println(sayilar.contains(4)); // true
```

# Array List

- 5- list'den istedigimiz index'deki element'e ulasma

`list.get ( istenenIndex )`

```
List<String> liste=new ArrayList<>();
liste.add("Fatih");
liste.add("Levent");
liste.add("Esra");
liste.add("Seher");

System.out.println(liste.get(2)); // esra
System.out.println(liste.get(1)); // Levent
```

# Array List

## 6- İstenen elementi update etme

`list.set( istenenIndex , yeniDeger )`

```
List<String> liste=new ArrayList<>();
liste.add("Fatih");
liste.add("Levent");
liste.add("Esra");
liste.add("Seher");

liste.set(2, "Yasar");
System.out.println("set'den sonra "+liste);
// [Fatih, Levent, Yasar, Seher]
```

`list.set( istenenIndex , yeniDeger )` ile `list.add( istenenIndex, yeniElement)` birbirinden farklıdır.

`Add( )`'da element silinmez eklenmeden sonra kalanlar geriye kaydırılır, `set( )` de ise o indexdeki eski değer silinir, yeni yazılır, diğer elementlerin yerlerinde bir değişiklik olmaz

# Array List

## 7- İstenen elementi veya elementleri silme

**list.remove ( istenenObje )** : İstenen objeyi siler ve bize boolean sonuç döndürür.

Silmek istedigimiz obje list'de yoksa list'de bir degisiklik olmaz, remove( ) bize **false** döndürür.

**list.remove ( istenenIndex )** : İstenen indexdeki elementi siler ve bize sonuç olarak silinen elementi döndürür.

Silmek istedigimiz index list'de yoksa **exception** olusur.

```
List<String> liste=new ArrayList<>();  
liste.add("Fatih");  
liste.add("Levent");  
liste.add("Esra");  
liste.add("Seher");  
  
System.out.println(liste); // [Fatih, Levent, Esra, Seher]  
  
// remove 1- silme islemi icin obje yazilirsa,  
// objeyi siler ve bize boolean sonuc doner  
System.out.println(liste.remove( o: "Fatih")); // true  
System.out.println(liste); // [Levent, Esra, Seher]  
  
System.out.println(liste.remove( o: "Ahmet")); // false  
  
// remove 2- index ile silersek, bize silinen elementi dondurur  
System.out.println(liste.remove( index: 1)); // Esra  
  
System.out.println(liste); // [Levent, Seher]
```

# Array List

**NOT :** List Integer degerlerden olusuyorsa, remove(**sayı**) yazdigimizda direkt **index** olarak kabul eder ve o index'deki elementi siler.

Istenen Integer objeyi, obje olarak remove etmek isterseniz, oncelikle o objeyi bir variable olarak olusturup, **list.remove(variableismi)** seklinde kullanmalisiniz.

## 8- List'deki tum elementleri silme

**list.clear ( )**

```
List<Integer> sayilar = new ArrayList<>();  
  
sayilar.add(10);  
sayilar.add(15);  
sayilar.add(20);  
sayilar.add(2);  
  
System.out.println(sayilar); // [10, 15, 20, 2]  
  
sayilar.remove(index: 2);  
System.out.println(sayilar); // [10, 15, 2]  
  
Integer silinecek=10;  
sayilar.remove(silinecek);  
System.out.println(sayilar); // [15, 2]  
  
sayilar.clear(); // tum listeyi temizler  
System.out.println(sayilar); // []
```

# Array List

- 9- Bir list'de baska bir list'in tum elementleri silme

**list.removeAll (silinecekList)**

- 10- Iki list'in esit oldugunu kontrol etme

**list.equals (digerList)**

```
List<String> liste=new ArrayList<>();  
liste.add("Fatih");  
liste.add("Levent");  
liste.add("Esra");  
liste.add("Seher");  
  
List<String> silinecekListe=new ArrayList<>();  
silinecekListe.add("Fatih");  
silinecekListe.add("Levent");  
  
liste.removeAll(silinecekListe);  
System.out.println(liste); // [Esra, Seher]  
  
System.out.println(liste.equals(silinecekListe)); // false
```

# Array List

- 11- Bir list'de istenen elementin kullanildigi ilk index'i bulma

**list.indexOf (arananElement)**

- 12- Bir list'de istenen elementin kullanildigi son index'i bulma

**list.lastIndexOf (arananElement)**

```
List<String> liste=new ArrayList<>();  
liste.add("Fatih");  
liste.add("Esra");  
liste.add("Levent");  
liste.add("Esra");  
liste.add("Seher");
```

```
System.out.println(liste.indexOf("Esra")); // 1
```

```
System.out.println(liste.lastIndexOf("Esra")); // 3
```

# Array List

13- Bir list'deki elementleri sıralama

Collections.sort(list)

```
List<String> liste=new ArrayList<>();
liste.add("Eyup");
liste.add("Yahya");
liste.add("Esra");
liste.add("Seher");

System.out.println(liste); // [Eyup, Yahya, Esra, Seher]

Collections.sort(liste);

System.out.println(liste); // [Esra, Eyup, Seher, Yahya]
```

# Array List

- Soru 1-** Verilen bir array'de tekrar eden elementler icin, mukerrer olanlari silip, tum elemanlardan sadece 1 tane yapip bize dondurecek bir method olusturun.
- Soru 2-** Kullanicidan istedigi kadar isim alip, Q'ya bastiginda girdigi isimleri bize liste olarak dondurecek bir method olusturun.
- Soru 3-** Verilen String bir listede istenmeyen harf iceren elementleri silip, kalan kismini list olarak bize donduren bir method olusturun
- Soru 4-** Verilen pozitif bir n tamsayisini alarak, bize ilk n tane tane Fibonacci sayisini bir list olarak donduren bir method olusturun.
- Soru 5-** Kullanicidan pozitif bir tamsayi alip, o tamsayidan kucuk Fibonacci sayilarini bir liste olarak bize donduren bir method olusturun.
- Soru 6-** Verilen pozitif bir tamsayiyi, tam bolebilen tum pozitif tamsayıları bir liste olarak bize donduren bir method olusturun.

# Array List

14- Bir list'in belirli bir bolumunu alma

`list.subList(baslangicIndex,bitisIndexi)`

```
List<String> liste=new ArrayList<>();
liste.add("Eyup");
liste.add("Yahya");
liste.add("Esra");
liste.add("Seher");

System.out.println(liste.subList(1, 3)); // [Yahya, Esra]
```

# Array'i Array List'e Cevirme

Verilen bir array'i Arrays.asList(array) method'u kullanarak list'e cevirebiliriz.

Ancak bu method'u kullanıssız yapan 2 dezavantaj soz konusudur.

- array'den donusturulen list'de add( ) ve remove( ) gibi uzunlugu etkileyen method'lar kullanıldığında Run Time'da exception olusur

```
Integer[] arr= {2,3,4,5,6};

List<Integer> list= Arrays.asList(arr);

System.out.println(list); // [2, 3, 4, 5, 6]

list.add(10);
list.remove(index: 0);

list.set(1,20);
System.out.println("list'i update ettikten sonra list : " + list);
//[2, 20, 4, 5, 6]

System.out.println("list'i update ettikten sonra array : " + Arrays.toString(arr));
//[2, 20, 4, 5, 6]
```

- asList( ) method'u array ve list'i ozdeslestirir, birinde yapılan degisiklik otomatik olarak digerine de islenir.

# ArrayList'i Array'e Cevirme

Verilen bir list'i Array'e 2 turlu cevirebiliriz.

Array'in data turunu girmek istersek sağ tarafta parametre olarak

(**new dataTuru[0]**) yazmaliyiz

**String[ ] arr = liste.toArray(new String[0]);**

Array'in data turunu Object seçersek sağ tarafta parametre yazmaya gerek kalmaz

**Object[ ] arr = liste.toArray( );**

```
List<String> liste=new ArrayList<>();  
liste.add("Eyup");  
liste.add("Yahya");  
liste.add("Esra");  
liste.add("Seher");
```

```
String arr[ ] = liste.toArray(new String[0]);  
System.out.println(Arrays.toString(arr)); // [Eyup, Yahya, Esra, Seher]
```

```
List<Integer> liste2=new ArrayList<>();  
liste2.add(12);  
liste2.add(10);  
liste2.add(5);  
liste2.add(9);
```

```
Integer[] arr2 = liste2.toArray(new Integer[0]);  
System.out.println(Arrays.toString(arr2)); // [12, 10, 5, 9]
```

```
Object[] arr3= liste.toArray();  
Object[] arr4= liste2.toArray();  
System.out.println(Arrays.toString(arr3)); // [Eyup, Yahya, Esra, Seher]  
System.out.println(Arrays.toString(arr4)); // [12, 10, 5, 9]
```

# For Each Loop

U N I T Y U E R S E  
A C A D E M Y

unityverseacademy.com

# For Each Loop

U N I T Y U E R S E  
A C A D E M Y

unityverseacademy.com

# For Each Loop

U N I T Y U E R S E  
A C A D E M Y

unityverseacademy.com

# For Each Loop

U N I T Y U E R S E  
A C A D E M Y

unityverseacademy.com

# For Each Loop

U N I T Y U E R S E  
A C A D E M Y

unityverseacademy.com