



**BATCH** : Batch 59

**LESSON** : Java 01

**DATE** : 19.02.2022

**SUBJECT** : Genel Hatırlatmalar  
Java Giriş

**GOOGLE CLASSROOM da bugünkü ATTENDANCE bölümünü doldurmayı unutmayalım !!**



techproeducation



techproeducation



techproeducation



techproeducation



techproedu



## Genel Hatırlatmalar



1. Derslere Hazirlanın ve Zamanında Katılın
2. Dersi Dikkatli Dinleyin
3. Derste Aktif Olun
4. Anlamadıklarınızı Sorun
5. Ödevlerinizi Yapın (Kod yazma araba kullanma gibidir)
6. Her Dersten Sonra Tekrar Yapın



## Genel Hatirlatmalar

7. Basari = Egitim + Calismak
8. Grup calismalari yapin, En iyi ogrenme yontemi ogretmektir
9. Mentoring toplantılarini kacirmayin
10. Maillerinizi gunluk kontrol edin
11. Yoklama yapiliyor zooma isminizle girin
12. Teknik destek slack @technical support
13. Ders esnasinda canli destek  
Free : Nur, Zafer , Y.Selim  
Batch 59 : Elif, Merve, Feyza, Yusuf, Kenan
14. Customer service +1 917 768 74 66

**"TEACHERS  
CAN OPEN  
THE DOOR,  
BUT YOU  
MUST ENTER  
IT YOURSELF."**

~ CHINESE PROVERB



# Gorulecek Dersler

## Automation Engineer:

Java	Selenium Grid
Selenium	Git, GitHub
SDLC	HTML & CSS
API	Bootstrap
SQL	Java Script
Jenkins	Lambda
JDBC	Project

## Java Developer

Core Java	UML Diagram
Advance Java	Multi Thread
Oracle SQL	Hibernate
JDBC	MongoDB
HTML5 & CSS	SpringMVC
Bootstrap	Restful API
JavaScript	Micro Services with Spring Boot
React.js	Git-GitHub
SDLC	
Market Session	

## Mobile Developer

Core Java	Git-GitHub
Oracle SQL	Bootstrap
SDLC	React.js
HTML5 & CSS	JavaScript
Advance Java	React Native
	Project



## Mentoring

Mentoring toplantıları her hafta team tarafından ortak belirlenen gün ve saatte düzenli şekilde yapılmaktadır.

- ✓ Mentoring faaliyetleri STUDENT COACHING (öğrenci danışmanlığı) olarak yapılmaktadır.
- ✓ Mentoring faaliyetlerinde...
  - Haftanın görülen derslerin değerlendirmesi...
  - Derslerle ilgili döküman desteğinin sağlanması....
  - Ödev proje vs çalışmaların takip edilmesi...
  - Team work'lerin takip edilmesi...
  - FlipGrid çalışmalarının takip edilmesi...
  - Java verbal çalışmalarının takip edilmesi...
  - Java coding çalışmalarının takip edilmesi...
  - Interview çalışmalarının takip edilmesi...

DÜZENLİ OLARAK YAPILMAKTADIR....



## Ders İşleyisi - Bilmeniz Gerekenler

1. Maillerinizi günlük kontrol edin
2. Dersleri zoom'dan izliyoruz ama mesajlaşma için slack kullanıyoruz



- İki slack kanalımız var
- Direk mesaj
- Kod paylaşma (**snippet**)
- Mesaj silme ve edit
- Pin yapma



3. Google Classroom
- Tüm ders notları, zoom linki ve videolar Google Classroom'dan paylaşılacak
  - Maillerinize davetiye gönderildi
  - Youtube videoları



## Ders Isleyisi - Bilmeniz Gerekenler



- 1-Ders esnasında öğrencilerin dikkatini dagitacak paylasimlar yapmayin
- 2-Ders esnasında ders ve konu disinda paylasim yapmayiniz.
- 3-Diyaloglarınızda asgari nezaket ve saygı kurallarına azami dikkat ediniz.
- 4-Ders esnasında ders hocasına direct mesaj yazmayiniz
- 5-Derste code paylaşırken SNIPPET ve screenshot kullanmaya dikkat ediniz
- 6-Dersi iyi takip ediniz, öncesinden sorulmuş ve cevaplanmış soruyu tekrar sormamaya azami gayret gösteriniz.
- 7-CODE ve SYNTAX hatalarınız için MENTOR'lerimiz, KURULUM hatalarınız için TECHNICAL SUPPORT yardımcı olacaktır.
- 8-CODE ve SYNTAX hatalarınızı DM olarak değil benzer hataları alanların da yararlanması için öğrenci yardimlasi channel'den paylaşınız.



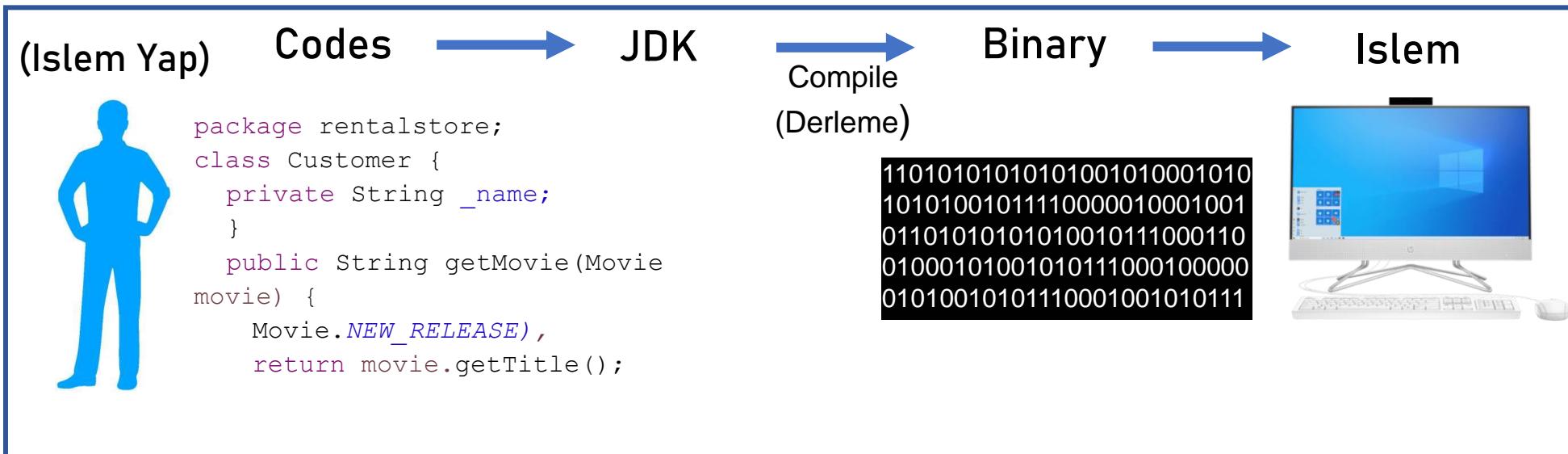
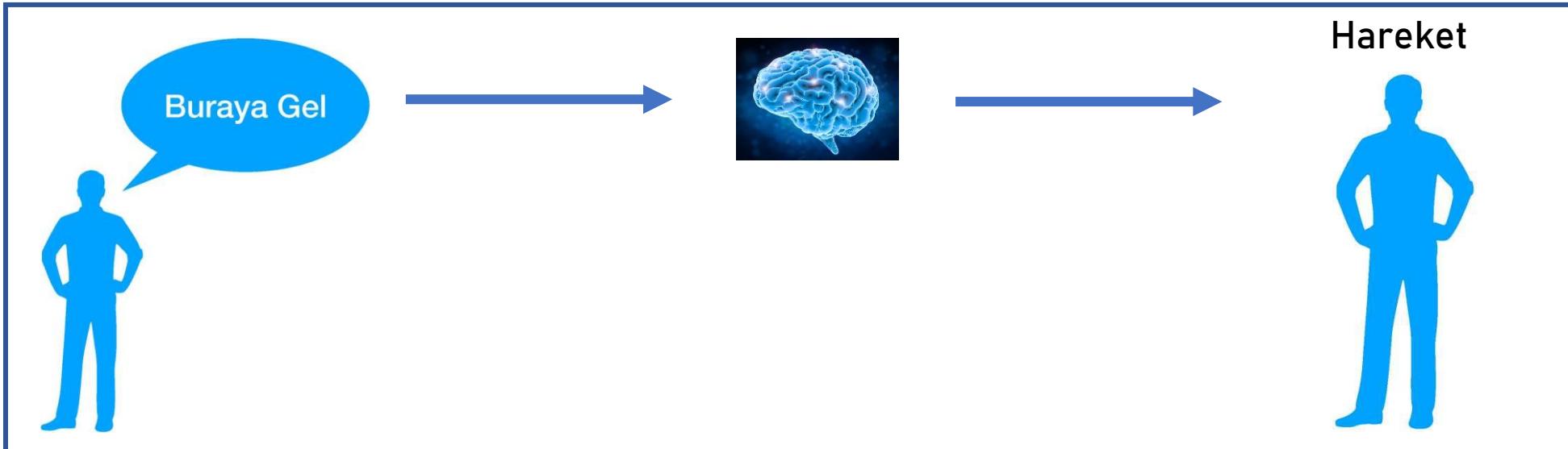
## Ders Isleyisi - Bilmeniz Gerekenler

1. Ders tam zamanında baslar.
2. Dersin basında 10 dakika bir önceki günün kısa tekrarı yapılır
3. Her konu bittiğinde ertesi gün kısa tekrardan sonra Socrative testi yapılır ( 10 -15 dk) sonra o sorular çözülerek konu tekrarı yapılır





# Programlama Dili Nedir ?





## Nicin Java ?

1- Öğrenmesi kolay

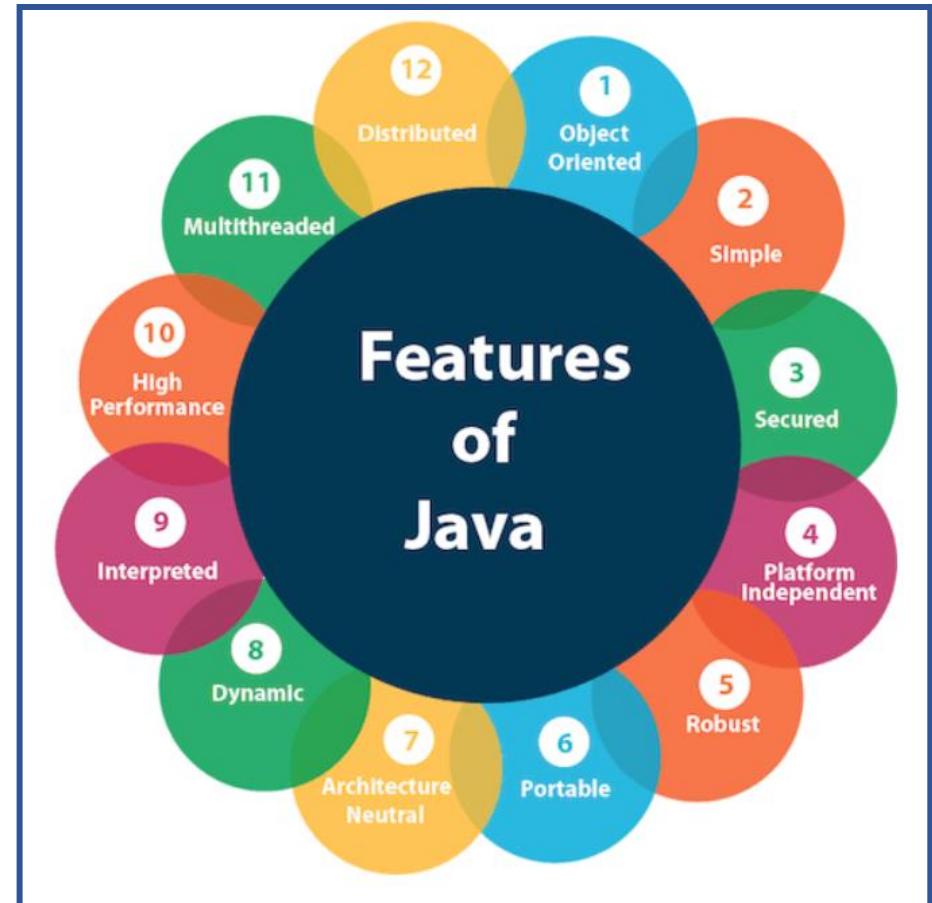
2- Dünyada en çok kullanılan programlama dili

Sun'a göre 3 milyar cihaz Java kullanıyor. Şu anda Java'nın kullanıldığı birçok cihaz var.

Bunlardan bazıları şu şekildedir:

- Acrobat reader, medya oynatıcı, antivirüs vb.
- Masaüstü Uygulamaları
- Bankacılık uygulamaları gibi Kurumsal Uygulamalar
- Cep Telefonu
- Akıllı kart uygulamaları
- Robotik uygulamaları
- Oyunlar

3- Java "Object Oriented Programming (OOP)" Language' dir.





# Object Oriented Programming Nedir?



Objects (Nesne)



Application (Urun)

1- Feature (Fields veya Variables)

Pasif ozellik (renk,sekil,isim)

2- Functionality (Method)

Aktif ozellik (tasima,degistirme)



# Bir Object Nasıl Olusturulur?



Class(Object Kalibi)

Field      Method  
(Variables)    (Functions)



Object

Birden fazla Obje birlestirilir



Application



# Object Nasıl Kullanılır ?



Ogretnen



Personel



Ogrenci



Dersler

09:00	TÜRKÇE-1
09:30	MATEMATİK-1
10:00	TÜRKÇE-2
10:30	MATEMATİK-2
11:00	TÜRKÇE-3
11:30	MATEMATİK-3
12:00	TÜRKÇE-4
12:30	MATEMATİK-4
13:00	İYEP TÜRKÇE

Notlar



# Bir Class Hangi Bolumlerden Olusur?

```
public class C2_MethodCreation2 {
```

1

2

3

2

```
}
```

// Class sonu

1 – Class Declaration

2 – Curly braces : Suslu parantez

3 – Class Body : Suslu parantezler  
arasinda kalan ve kodlarimizi  
yazdigimiz bolum



# Bir Class'in Icinde Neler Bulunur?

```
public class C2_MethodCreation2 {  
  
    private double ortalama; 1  
    public int sonuc;  
  
    public static void main(String[] args) {  
        ortalama(85.2 ,90.3); // method call 2  
    }  
  
    public static void ortalama(double sayi1, double sayi2) {  
        System.out.println("girdiginiz iki sayinin ortalamasi : " + (sayi1+sayi2)/2);  
    }  
} // Class sonu
```

1 – Field / Variables

2 – Main Method

3 – Method



# Class Olustururken (Declaration) Kullanican Keyword'ler

```
public class MyFirstClass {}  
1     2      3      4
```

- 1 **public** : Access Modifier (Erisim duzenleyici) : class'a kimlerin erisebilecegini belirler. Public olursa her yerden erisilebilir  
**default** : Sadece bulundugu Package'den kullanilabilir
- 2 **class** : Yazdigimiz kodun class oldugunu belirtir
- 3 **MyFirstClass** : Olusturdugumuz class'in ismidir. Class'a istedigimiz ismi verebiliriz ancak isim verilirken genelde class'da yapılan isleme uygun bir isim secilmesine dikkat edilir.  
Isim mutlaka buyuk harfle baslar, birden fazla kelimededen olusursa sonraki kelimelerin ilk harfleri de buyuk harf yazilir (Camel Case)
- 4 Body (Class Body) : { } arasında kalan kodlarimizi yazdigimiz bolumdur



# Method Oluştururken Kullanılan Keyword'ler

```
public int myFirstMethod () {}  
1   2       3       4 5
```

- 1 **public** : Access Modifier (Erisim duzenleyici):method'a kimlerin erisebilecegini belirler  
**private**: Sadece bulunduğu class'da kullanilabilir  
**protected** : Sadece icinde bulunduğu class ve child class'lardan kullanilir
- 2 **Int** : Return Type, methodun ne urettigini ve bize dondurdugunu belirtir
- 3 **myFirstMethod** :Olusturdugumuz method'un ismidir. Isim mutlaka kucuk harfle baslar, birden fazla kelimededen olusursa sonraki kelimelerin ilk harfleri buyuk harf yazilir (Camel Case)
- 4 **() parantez**: Methodlarda isimden sonra parantez kullanilir ve gerektiginde parantez icinde parametre yazilir.
- 5 Body (Method Body) : { } arasında kalan kodlarimizi yazdigimiz bolumdur



## Main Method

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



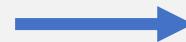
- main method, java'nin calismaya basladigi giristir. (**Entry Point**)
- main method olusturulurken yazilmasi gereken syntax (kod dizimi) degistirilemez
- Parantez icinde yazilan (String[] args) java'nin calismasi icin gerekli olan parametreleri barindirir ve olmasi sarttir.

Araba



Motor

Java Project



Main Method



# Yorum Cumlesi (Comment) Nasıl Eklenir ?

```
public class Example {  
    // Bir satiri comment haline getirmek icin // kullanilir  
    String isim = "Mehmet";  
  
    /*  
     * Eger birden fazla  
     * satiri yorum haline  
     * getirmek istiyorsak  
     * kullanilir  
     */  
    int sayi=10;  
    double not=75.70;  
}  
  
boolean ogrenciMi =false;
```

➤ **Comments** : Java tarafindan calistirilmayan, amaci kodların aciklanması veya bir konuda bilgi vermek olan cümlelerdir

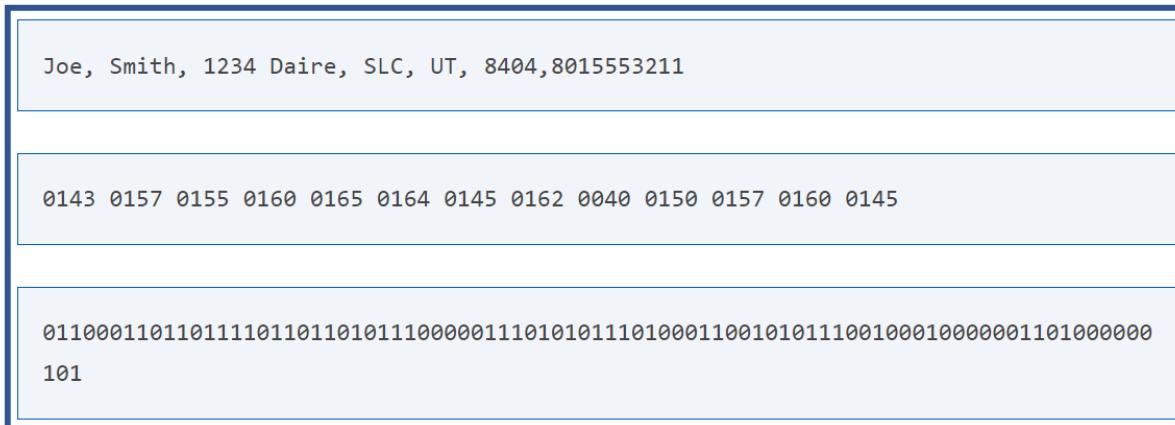
➤ Genelde iki kullanim vardir

- 1) **Tek satirlik comment**
- 2) **Cok satirlik comment**

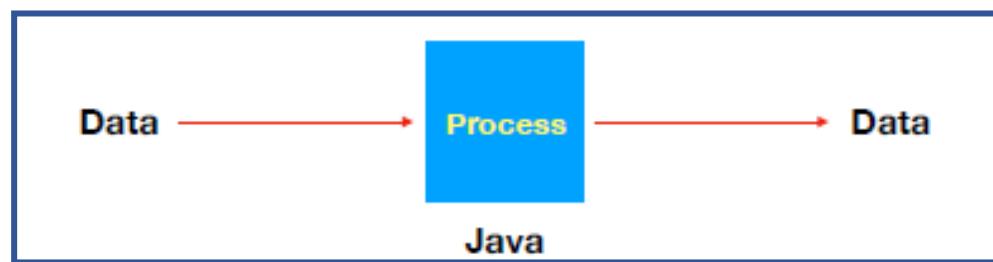


# Data Nedir?

Data bilgisayar tarafından işlenen (**processed**) veya depolanan (**stored**) bilgidir.



Java'nın kullandığı (**use**) veya ürettiği (**produce**) her şey data'dır.

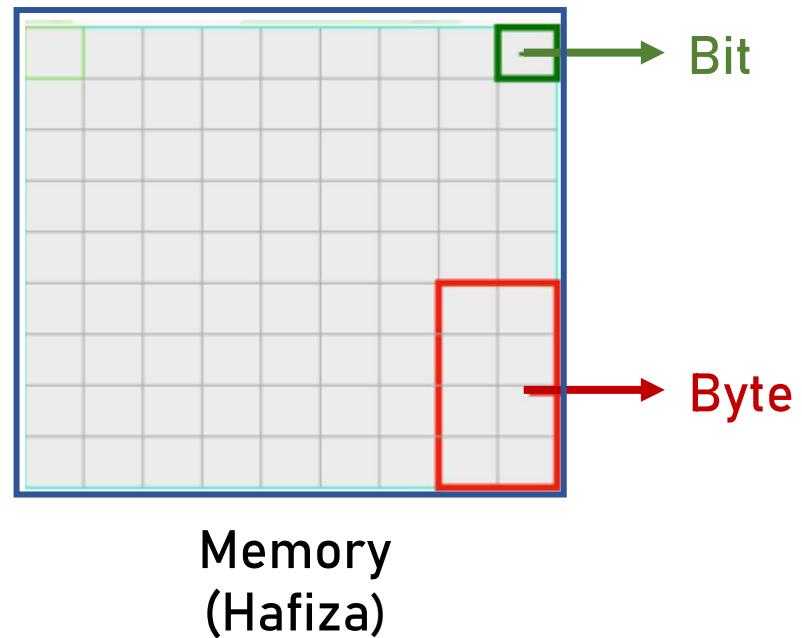
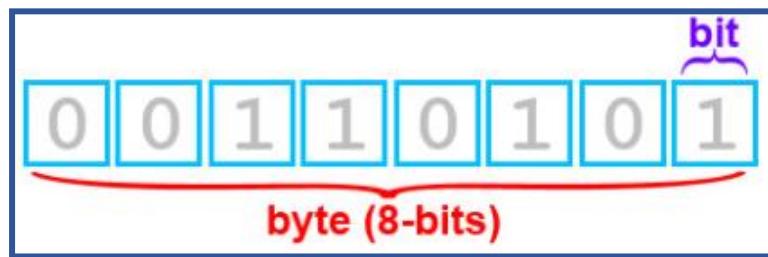




# Bit

**bit** hafızadaki en küçük data parçasıdır. Her “bit” bir binary value içerir, 0 veya 1.

Note: 8 bit =1 byte





**BATCH** : Batch 60

**LESSON** : Java 02

**DATE** : 19.02.2022

**SUBJECT** : **Java Giris**  
**Variables**

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



# IntelliJ Kullanım

## 1- Proje oluşturma

File -- New -- Project -- (Java Project) Next -- java2022WinterTr -- finish

## 2- Package (paket) oluşturma

src dosyasına sağ click -- New -- Package -- day01variables -- finish

## 3- Class oluşturma

day01variables dosyasına sağ click -- New -- Class -- C01\_Variables01 -- finish

## 4- Main method oluşturma

**public static void main(String[] args)** yazarak main methodu oluşturalim



# Variables (Degisken) Olusturma

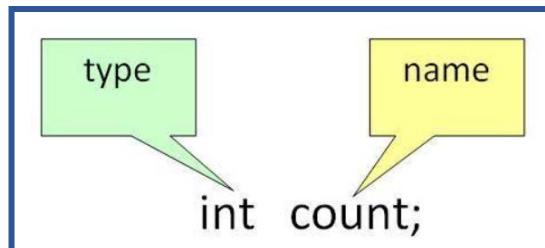
## Declaration

Variable bellekte (memory) ayrılmış olan alanın (reserved area) adıdır.

Variable içinde değer saklayan bir konteynirdir (container).

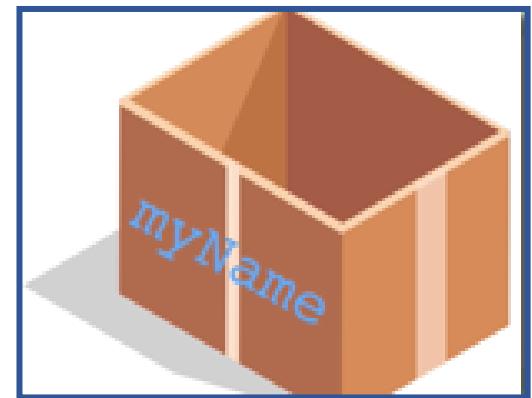
Bir değişkende saklanan değer, program yürütülürken değiştirilebilir.

Java'da, tüm değişkenler kullanılmadan önce deklare edilmelidir (variable declaration)



Variable declaration için iki seyi belirtmemiz gerekiyor

- 1- Data type (data turu)
- 2- Variable Name (degisken ismi)





# Variables Deger Atama (Assignment)

Varolan bir variable'a deger atamaya assignment (atama) denir.

1- Deger atamasi yapilirken data turune uygun deger atanmalidir. Diger turlu Java hata verir.

```
5 public class Example {  
6  
7     String isim ="Mehmet";  
8     boolean ogrenciMi =false;  
9     int not=85;  
10    double ortalama= 78.3;  
11  
12    String ad =75;  
13    boolean emekliMi ="true";  
14    int maas=true;  
15    double yas= "kuru";
```



# Variables Deger Atama (Assignment)

2- İlk önce declaration, daha sonra atama yapılabilir.

```
String isim ;
boolean ogrenciMi;
int not;
double ortalama;

isim ="Mehmet";
ogrenciMi =false;
not=85;
ortalama= 78.3;
```

3- Bir defa declaration yapıldıktan sonra, birden fazla atama yapılabilir. Java son değeri tutar, öncekini siler.

```
5 public class Example {
6 public static void main(String[] args) {
7
8     int level=1;
9
10
11
12
13     level=2;
14
15
16
17     level=3;
18
19 }
20 }
21 }
```



## Variables Deger Atama (Assignment)

4- Ayni data turunde birden fazla variable tek komutla deklare edilebilir.

```
9  int level,yas,maas;  
10  
11  level=5;  
12  yas=20;  
13  maas=10000;
```

5- Ayni data turunde birden fazla variable tek komutla deklare edilip deger atanabilir.

```
9  int level=5, yas=20, maas=10000;
```



# Data Types

Java'da iki data tipi kullanilmaktadir

- **Primitive Data Types** : boolean, char, byte, short, int, long, float ve double
- **Non- Primitive Data Types** : String,

ilerleyen derslerde gorecegimiz primitive olmayan Array, List, Object gibi her data non-primitive'dir.



# Primitive Data Types

- 1) **boolean** Data Type: true veya false barindirir. Hafizada **1 bit** kullanir  
Sadece dogru veya yanlis seklinde cevap verilebilecek variable'larda kullanilir

```
boolean isExpensive = true;  
boolean isCold = false;
```

- 2) **char** Data Type : Tek karakter barindirir. Hafizada **16 bit** kullanir  
Harf, sayi veya simbol bakilmaksizin sadece 1 karakter kullanacak variable'larda kullanilir

```
char letter = 'a';  
char digit = '3';  
char cymbol = '#';
```

**Note:** char degerlerini single quote arasina yazilir.



## Primitive Data Types

3) **byte** Data Type: -128 den 127'e (dahil) tamsayilar icin kullanilabilir. Hafizada **8 bit** kullanir

byte age = **73**;

4) **short** Data Type: -32.768 den 32.767'e (dahil) tamsayilar icin kullanilabilir. Hafizada **16 bit** kullanir

short koyNufusu = **27,324**;

5) **int** Data Type: -2.147.483.648 den 2.147.483.647'e (dahil) tamsayilar icin kullanilabilir. Hafizada **32 bit** kullanir

int turkiyeNufusu = **67,324.564**;

6) **long** Data Type: -9.223.372.036.854.755.808 den ,223.372.036.854.755.807'e (dahil) tamsayilar icin kullanilabilir. Hafizada **64 bit** kullanir



## Primitive Data Types

7) **float** Data Type: Kucuk ondalik sayilar icin kullanilabilir. Hafizada **64 bit** kullanir

```
float floatVar2 = -2.123456f;
```

**Not:** float sayilarin sonunda “ **f** ” yazilmalidir, yazilmazsa java sayiyi double kabul eder

8) **double** Data Type: Buyuk ondalik sayilar icin kullanilabilir. Hafizada **64 bit** kullanir

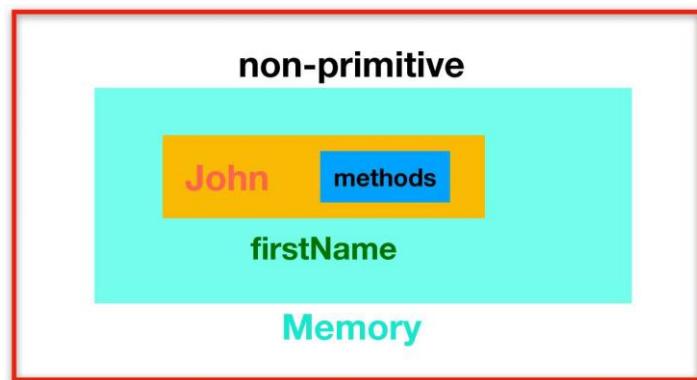
```
double doubleVar2 = -2.1234567907800000000123
```



# Non-Primitive Data Type

## String Data Type:

String pes pese dizilmiş char'lardan oluşur. Kelimeler, cümleler, matematiksel işlem yapılmayacak sayısal değerler de String olarak tanımlanabilir



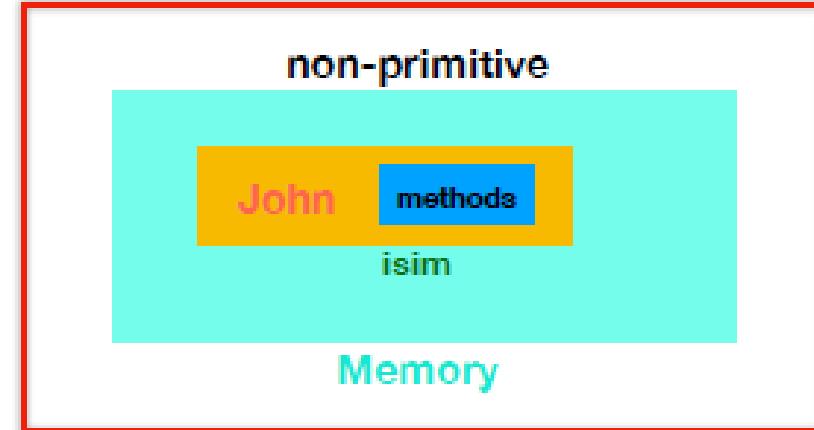
String okulAdi = "Yildiz Koleji, Cankaya Ankara #";  
String telNo = "5321234567";  
String ilkHarf = "A";

**Note:** String'ler çift tırnak (double quotes) arasına yazılır.

**Note:** Baska non-primitive data type'lar da var, daha sonra öğrenecegiz.



# Primitive VS Non-Primitive Data Types



- 1) Primitive'ler sadece value icerir, non-primitive'ler value ve methodlar icerir.
- 2) Primitive'ler kucuk harf ile, non-primitive'ler buyuk harf ile baslar.
- 3) Primitive'leri Java olusturur biz primitive data turu olusturamayiz.  
Non-primitive'leri biz de olusturabiliriz, Java da olusturabilir. Or: String'i Java olusturmustur.
- 4) Primitive'lerin buyuklukleri data type'ing gore sabittir. non-primitive'ler icin sabit buyukluk soz konusu degildir.



**BATCH** : Batch 60

**LESSON** : Java 03

**DATE** : 19.02.2022

**SUBJECT** : Kullanıcıdan Deger Alma

Data Casting

Increment/ Decrement

- techproeducation
- techproeducation
- techproeducation
- techproeducation
- techproedu



## Önceki Dersten Aklımızda Kalanlar

1. OOP concept : nesne tabanlı programlama demektir, biz oluşturduğumuz class'lar sayesinde objeler üretebiliriz ve bu objeleri birleştirerek kompleks uygulamalar geliştirebiliriz(lego gibi)
2. Class hangi bölümlerden oluşur ?
  - class declaration : keyword'ler sayesinde class'ı kimlerin kullanabileceğini görebiliriz, class ismi büyük harfle başlar ve CamelCase şeklinde yazılır
  - { } curly braces / suslu parantez : Classın nerede başlayıp bittiğini gösterir
  - Class Body : curly braces arasında kalan ve kodlarımızı yazdığımız bolumdur
- 3- Class içerisinde neler olur ?
  - Main method : arabanın motoru gibidir, Java kodlarımız çalıştırılmaya main method'dan başlar
  - normal method'lar : method'lar bizim adımıza belirlediğimiz işlemleri yapıp, işlem sonunda da istediğimiz sonucu bize dondururlar
  - variable : bizim için değerleri saklayan konteynır'lardır



## Önceki Dersten Aklımızda Kalanlar

4- variable : bizim istedigimiz deger koyabilmemiz icin hafiza da ayrılan bolumun adidir.  
Ornegin bir oyunda level bilgisi icin bir variable tanimlarsak, oyunun hangi asamasında  
olursa olsun level variable'ina baktigimizda icinde level degerimizi gorebiliriz  
Biz programimiz icerisinde ne zaman variable ismini yazsak, java o variable yerine en son  
atanan degeri kullanır.

### 5- Data turleri

- primitive data turleri : boolean, char, byte, short, int, long, float, double
- non-primitive data turleri : String (ilerde pek cok cesidini gorecegiz, biz de istersek  
non-primitive data turu olusturabiliriz)

### 6- iki data turu arasindaki farklar

- p'ler sadece depolama yapar, np'ler ise hem depolama yapar hem de kendilerine  
ozel methodlar sayesinde istedigimiz degisimleri yaparlar
- p data turlerinin isimleri kucuk harfle baslar, np data turlerinin isimleri buyuk  
harfle baslar
- p'ler 8 tanedir ve biz yeni p data turu uretemeyiz, ancak np'leri java da uretebilir biz  
de uretebiliriz, dolayisiyla np data turu sayisi sinirlandirilamaz



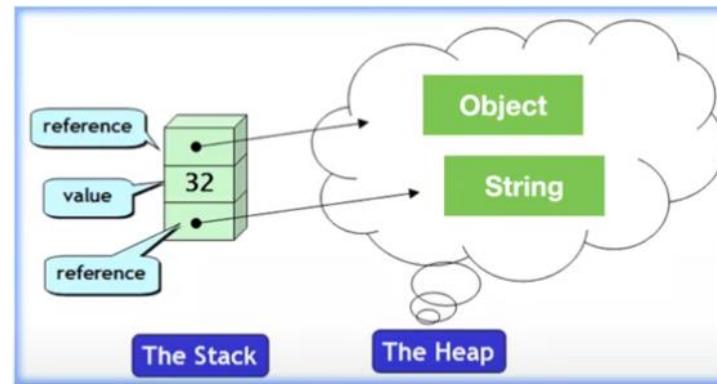
## Variable ve Method'lar Nasıl Adlandırılır

1. Java variable isimleri **case sensitive** (Buyuk kucuk harfe duyarlidir)dir.  
“money”, “Money” veya “MONEY” birbirinden farklidir
2. Java variable isimleri “harf”, “\$” veya “\_” ile baslamlmalıdır.  
Fakat “\$” ve “\_” ile baslamak tavsiye edilmez.
3. Java variable isimlerinde, ilk harften sonra sayi, “\$” ve “\_” kullanılabilir.
4. Variable isimleri icin Java'ya ozel terimler (key word) kullanılamaz. (int, for, if, import vb).
5. Variable isimleri kucuk harflerle baslar, camel case kullanılır
6. Variable isimleri 1'den fazla kelime içeriyorsa, ilk kelimededen sonraki her kelimenin ilk hafi buyuk harf ile baslamlıdır. firstName, bigApple, ageJohnWalker gibi. Buna camelCase denir.



# Memory (Hafiza) Kullanimi

Javada kullanılan iki hafıza vardır



**Stack => small**

**Heap => huge**

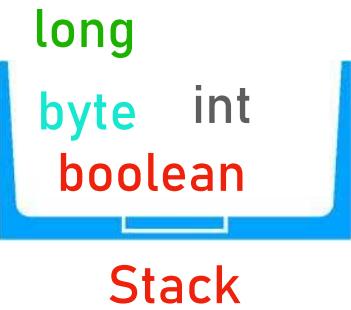
**1- Stack Memory** : primitive data tiplerine ait değerleri ve Non-primitive datalara (Object) ait referansları(adres) barındırır

**2- Heap Memory** : Non-primitive data'lari depolamak(**store**) icin kullanılır

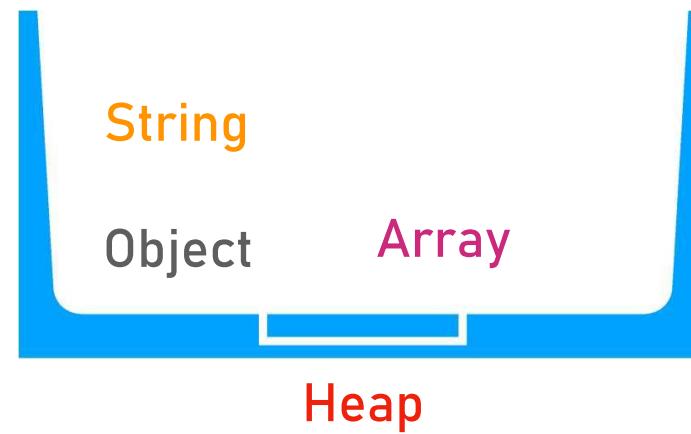


# Memory (Hafiza) Kullanımı

reference of an Object



String  
Object  
Array





## Variables Class Work

- 1- Farkli 3 data turunde variable olusturun ve bunlari yazdirin
- 2- isim ve soyisim icin iki variable olusturun ve bunlari  
isminiz : Mehmet  
soyisminiz : Bulutluoz  
seklinde yazdirin
- 3- Iki farkli tamsayi data turunde 2 variable olusturun bunların toplamini yazdirin
- 4- Bir tamsayi ve bir ondalikli variable olusturun ve bunların toplamini yazdirin
- 5 - char data turunde bir variable olusturun ve yazdirin
- 6- Bir tamsayi, bir de char degiskeni olusturun ve bunların toplamini yazdirin.



# ASCII Table

ASCII control characters			ASCII printable characters			Extended ASCII characters		
00	NULL	(Null character)	32	space	64	@	96	'
01	SOH	(Start of Header)	33	!	65	A	97	a
02	STX	(Start of Text)	34	"	66	B	98	b
03	ETX	(End of Text)	35	#	67	C	99	c
04	EOT	(End of Trans.)	36	\$	68	D	100	d
05	ENQ	(Enquiry)	37	%	69	E	101	e
06	ACK	(Acknowledgement)	38	&	70	F	102	f
07	BEL	(Bell)	39	'	71	G	103	g
08	BS	(Backspace)	40	(	72	H	104	h
09	HT	(Horizontal Tab)	41	)	73	I	105	i
10	LF	(Line feed)	42	*	74	J	106	j
11	VT	(Vertical Tab)	43	+	75	K	107	k
12	FF	(Form feed)	44	,	76	L	108	l
13	CR	(Carriage return)	45	-	77	M	109	m
14	SO	(Shift Out)	46	.	78	N	110	n
15	SI	(Shift In)	47	/	79	O	111	o
16	DLE	(Data link escape)	48	0	80	P	112	p
17	DC1	(Device control 1)	49	1	81	Q	113	q
18	DC2	(Device control 2)	50	2	82	R	114	r
19	DC3	(Device control 3)	51	3	83	S	115	s
20	DC4	(Device control 4)	52	4	84	T	116	t
21	NAK	(Negative acknowl.)	53	5	85	U	117	u
22	SYN	(Synchronous idle)	54	6	86	V	118	v
23	ETB	(End of trans. block)	55	7	87	W	119	w
24	CAN	(Cancel)	56	8	88	X	120	x
25	EM	(End of medium)	57	9	89	Y	121	y
26	SUB	(Substitute)	58	:	90	Z	122	z
27	ESC	(Escape)	59	;	91	[	123	{
28	FS	(File separator)	60	<	92	\	124	
29	GS	(Group separator)	61	=	93	]	125	}
30	RS	(Record separator)	62	>	94	^	126	~
31	US	(Unit separator)	63	?	95	-		
127	DEL	(Delete)						



## Variables Class Work

### Interview Question

1- Verilen sayi1 ve sayi2 variable'larinin degerlerini degistiren (SWAP) bir program yaziniz

Orn : sayi1=10 ve sayi2=20;

kod calistiktan sonra

sayi1=20 ve sayi2=10

2- Verilen sayi1 ve sayi2 variable'larinin degerlerini 3.bir variable olmadan degistiren (SWAP) bir program yapiniz



## Kullanicidan Deger Alma

1) **Scanner scan = new Scanner( System.in );**

**scan** : olusturdugumuz scanner'in ismidir ve istedigimiz ismi vermemiz mumkundur. Ancak genelde scan ismi kullanilir.

Bu tur isimlendirmelerde genel kurallara uymamiz kodumuzun anlasilabilir olmasi acisindan faydalı olacaktir.

2) **System.out.println( "Lutfen 100'den kucuk pozitif iki tamsayı giriniz" );**

Kullanicuya girmesini istedigimiz degerler icin aciklayici bilgi vermeliyiz.

Burada aciklama olarak ne yazdirlsa kodumuz calisir, hatta birsey yazdirmasak da calisir ancak kullanici kendisinden ne istedigimizi bilmezse deger girmesi gerektigini veya ne tur bilgi girmesi gerektigini bilemez



## Kullanicidan Deger Alma

3) `scan.nextInt()` ile girilen degerleri alabiliriz. Istedigimiz data tipine gore next'ten sonra yazilacak kisim degisir.

```
int num1 = scan.nextInt()  
int num2 = scan.nextInt()
```

<code>nextBoolean()</code>	Reads a boolean value from the user
<code>nextByte()</code>	Reads a byte value from the user
<code>nextDouble()</code>	Reads a double value from the user
<code>nextFloat()</code>	Reads a float value from the user
<code>nextInt()</code>	Reads a int value from the user
<code>nextLine()</code>	Reads a String value from the user
<code>nextLong()</code>	Reads a long value from the user
<code>nextShort()</code>	Reads a short value from the user



# Kullanicidan Deger Alma

## Sorular

**Soru 1)** Kullanicidan iki tamsayi alip bu sayilarin toplam,fark ve carpimlarini yazdirin

**Soru 2)** Kullanicidan karenin bir kenar uzunlugunu alin ve karenin cevresini ve alanini hesaplayip yazdirin

**Soru 3)** Kullanicidan yaricap isteyip cemberin cevresini ve dairenin alanini hesaplayip yazdirin

**Soru 4)** Kullanicidan dikdortgenler prizmasinin uzun, kisa kenarlarini ve yuksekligini isteyip prizmanın hacmini hesaplayip yazdirin

**Soru 5)** Kullanicidan ismini ve soyismini isteyip asagidaki sekilde yazdirin

Isminiz : Mehmet

Soyisminiz : Bulut

Kursumuza katiliminiz alınmıştır,tesekkur ederiz

**Soru 6)** Kullanicidan ismini ve soyismini alip aralarinda bir bosluk olusturarak asagidaki sekilde yazdirin

Isim – soyisim : Mehmet Bulutluoz

**Soru 7)** Kullanicidan ismini alip isminin bas harfini yazdirin.



BATCH : Batch 59-60  
LESSON : Java 04  
DATE : 23.02.2022  
SUBJECT : Data Casting

Increment / Decrement  
Matematiksel Operatorler

- techproeducation
- techproeducation
- techproeducation
- techproeducation
- techproedu



## Önceki Dersten Aklımızda Kalanlar

1. Scanner class'l kullanicidan bilgi almak icin kullanilir.
2. Uc adimda kullanicidan degeri aliriz
  - Scanner olusturma, parameter olarak System.in yazmaliyiz
  - Kullaniciya ne girecegini soyleyen bir mesaj yazdirma
  - Olusturdugumuz scan objesi ve girilecek dataya uygun next method'u ile kullanicinin girdigi degeri alip, gelen dataya uygun data turunde olusturdugumuz variable'a atama yapariz
- 3- Kullanicidan aldigimiz deger metin ise next( ) veya nextLine( ) method'u kullanilir. Next method'u sadece ilk bosluga kadar olan metni alirken, nextLine tum satiri alir
- 4- Java da kullanilan 2 tur hafiza vardir.  
stack- primitive dataturundeki variable'larin aldiği degerler ve non-primitive'lerin referansları bulunur  
Heap : Non primitive data turundeki datalar



## Data Casting / Veri Sınıfı Değiştirme

- Java'da kod yazarken bir veri tipinden diğer bir veri tipine aktarım yapmamız gerekebilir.
- Veri tiplerinde bir variable'a , olusturuldugu data tipinden farkli bir data turunden deger atanmasina Data Casting denir.
- Data casting yaparken aklımızdan cikarmamamız gereken konu data tiplerinin sınırlarıdır. Data tipinin sınırlarını aşan data casting islemlerinde hata almamamız için dikkat etmemiz gereken bazi durumlar olacaktır.
- Hatırlayacagımız sekilde Java'da sayılarla ilgili data tiplerinin sıralaması su sekildeydi

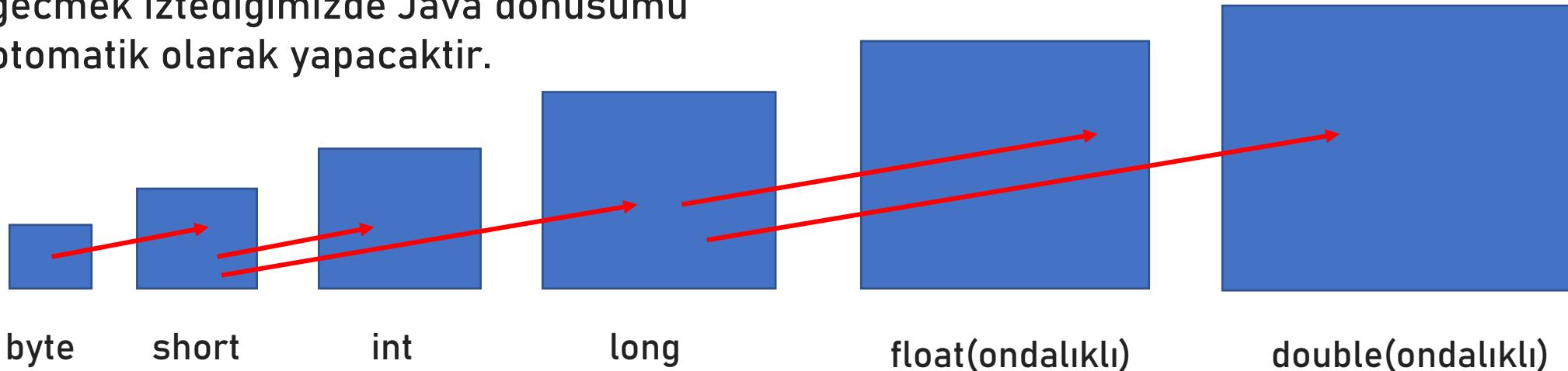
`byte < short < int < long < float(ondalıklı) < double(ondalıklı)`



# Data Casting / Veri Sınıfı Değiştirme

## 1) Auto Widening (Otomatik Genişletme)

Dar veri tipinden daha geniş bir very tipine gecmek istediğimizde Java dönüşümü otomatik olarak yapacaktır.



Orn :    `byte num1 = 12;`

`short num2 = num1;      // yazdırırsak 12 olarak yazdırır`

`int num3 = num2;        // yazdırırsak 12 olarak yazdırır`

`float num4=num3;      // yazdırırsak 12.0 olarak yazdırır`

`double num5=num4;     // yazdırırsak 12.0 olarak yazdırır`



# Data Casting

## 2) Explicit Narrowing (Manuel Daraltma)

```
public class Main {  
    public static void main(String[] args) {  
        double myDouble = 9.78;  
        int myInt = (int) myDouble; // Manual casting: double to int  
  
        System.out.println(myDouble); // Outputs 9.78  
        System.out.println(myInt); // Outputs 9  
    }  
}
```

- Genis veri tipinden daha dar bir veri tipine gecmek istedigimizde Java donusumu otomatik olarak YAPMAYACAKTIR.
- Bu durumda Java Casting'in bir problem olusturabilecegini varsayarak sizden MANUEL ONAY isteyecektir.
- Narrowing Casting bazi datalari kaybetmemize yol acabilir, bazen de sayiyi kendi sinirlari icinde kalan baska bir sayiya donusturebilir



## Data Casting

**Soru 1 )** byte veri tipinde bir degisken olusturun, short,int,float ve double data tiplerinde birer degisken olusturup adim adim widening yapin ve yazdirin

**Soru 2 )** int veri turunde bir degisken olusturun ve adim adim narrowing yapin ve yazdirin

**Soru 3 )** Float data turunde bir variable olusturun ve yazdirin

**Soru 4 )** double 255.36 sayisini int'a ve sonra da olusturdugunuz int sayiyi byte'a cevirip yazdirin

**Soru 5 )** int 2 sayiyi birbirine bolurun ve sonucu yazdirin

**Soru 6 )** int bir sayiyi double bir sayiya bolun ve sonucu yazdirin

**Soru 7 )** Farkli data tipleri ile islem yapip, sonuclarini yazdiralim



## Increment / Bir Variable'in Degerini Artirma Yontemleri

```
int numA = 2 ;  
numA = numA + 3;
```

veya →

```
numA += 3
```

?

```
int numB = 10 ;  
numB = numB * 7;
```

veya →

```
numB *= 7
```

?

```
int numC = 7 ;  
numC++;
```

?

```
int numD = 11 ;  
numD++ ;
```

?



## Decrement / Bir Variable'in Degerini Azaltma Yontemleri

```
int numA = 2 ;  
numA = numA - 3;
```

veya →

```
numA -= 3
```

?

```
int numB = 20 ;  
numB = numB / 5;
```

veya →

```
numB /= 5
```

?

```
int numD = 7 ;  
numD -- ;
```

?

```
int numE = 11 ;  
numE -- ;
```

?



**BATCH** : Batch 59-60

**LESSON** : Java 05

**DATE** : 24.02.2022

**SUBJECT** : Pre & Post Increment

Matematiksel Operatorler  
Modulus

- techproeducation
- techproeducation
- techproeducation
- techproeducation
- techproedu



## Önceki Dersten Aklımızda Kalanlar

1. Data Casting (Veri turunu degistirme): Java her data turunu birbirine cevirmez, ornegin Boolean bir degiskene String bir deger atayamayiz.  
Ancak sayisal veri turlerini birbirlerine cevirebiliriz
  - Eger variable turu(esitligin solu) daha kapsamlı ise, Java bu casting islemi otomatik olarak yapar (Auto Widening)
  - Eger deger'in turu (esitligin sagi) daha kapsamlı ise Java bunu otomatik olarak yapmaz. Cunku daha kapsamlı bir data turunden daha dar kapsamlı bir data turune gecis sirasinda data kayiplari veya farkli deger alma ihtimali olusur.Bu durumda Java sorumluluğu manuel olarak almamizi ister. Sorumluluğu alabilmek icin esitligin sagina parantez icerisinde variable'in data turunu yazariz

```
double sayi1=10.28;  
int sayi2 = (int)sayi1 ;
```
- 2- Increment ve Decrement : artirma veya azaltma demektir
  - sayi1 = sayi1+3; bu çok tercih edilmez
  - sayi1 += 3; genelde bu tercih edilir
  - sayi1++; sadece 1 artirip azaltacaksak bunu kullaniriz



## Pre-Increment & Post Increment

- Pre-Increment ve Post Increment operatorlerinin her ikisi de artirma islemi icin kullanilir
- Pre-Increment isleminde variable statement'da kullanilmadan once artirilir veya azaltilir

```
public static void main(String[] args) {  
    int a=15;  
    int b=++a;  
    System.out.println(b);  
}
```

Output : 16

- Post Increment isleminde variable statement'da kullanilir, sonra artirilir veya azaltilir

```
public static void main(String[] args) {  
    int a=15;  
    int b=a++;  
    System.out.println(b);  
}
```

Output : 15



# Javada Matematiksel Operatorler

- 1- Ustel islemler
- 2- Parantez ici
- 3- Carpma-Bolme
- 4- Toplama-cikarma

Ornek 1 :

$$38 / 2 * ( 4 + 3 ) * 2 =$$

Ornek 2 :

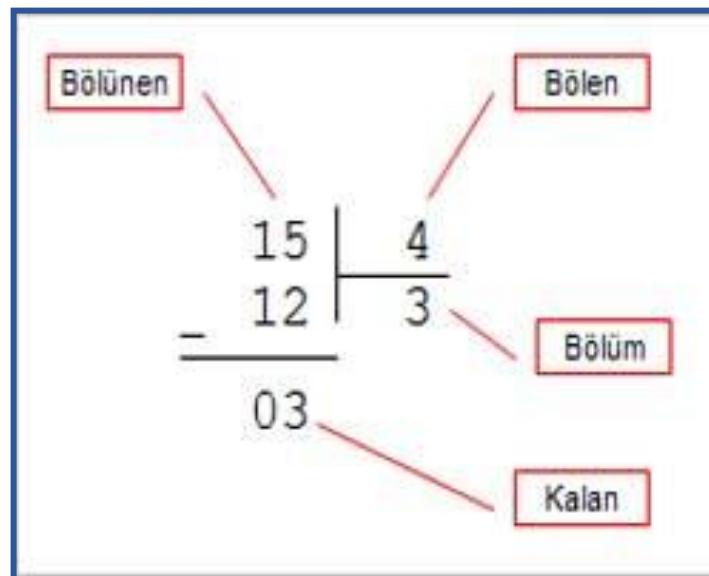
$$8 + 2 * ( 14 - 6 / 2 ) - 12 =$$



# Modulus %

Modulus işlemi bir bolme işleminde kalan sayiyi bize verir

```
public static void main(String[] args) {  
    int a=15 % 4;  
    System.out.println(a);  
}
```





## Modulus %

**Soru** ) Kullanicidan 4 basamakli bir sayi alin ve rakamlar toplamini bulup yazdirin

**Ipucu 1:**

Sayi % 10 => Bize son basamagi verir

$$538 \% 10 = 8$$

**Ipucu 2:**

Int Sayi /10 => Bize son basamak haric sayiyi verir

**int** sayi=538;

**sayi = sayi / 10 =>**

**sayi'**ya 53 degerini atar



## Wrapper Class

Java **primitive** data turleri ile **methodlari** kullanabilmemiz icin **Wrapper class'lari** olusturmustur.

Character,Byte,Integer,Short,Float,Double primitive data turleri icin olusturulan wrapper class'lardir.

```
public class Example {  
    public static void main(String[] args) {  
  
        int num1 = Integer.MIN_VALUE;  
        System.out.println(num1); → -2147483648  
  
        int num2 = Integer.MAX_VALUE;  
        System.out.println(num2); → 2147483647  
  
        int num3 = Byte.MIN_VALUE;  
        System.out.println(num3); → -128  
  
        int num4 = Byte.MAX_VALUE;  
        System.out.println(num4); → 127  
    }  
}
```



**BATCH** : Batch 59-60  
**LESSON** : Java 06  
**DATE** : 25.02.2022  
**SUBJECT** : Concatenation

**Relational Operators**  
**Conditional Operators**

- techproeducation
- techproeducation
- techproeducation
- techproeducation
- techproedu



## Önceki Dersten Aklımızda Kalanlar

1. Wrapper Class : primitive data turlerine ait method yoktur. Java primitive data turundeki variable'larin Wrapper Class'larine yaparak o turler icin de hazir bazi method'lar olusturmustur.
2. Pre Increment- Post Increment : bu ikisi de ++ , veya - icin gecerlidir. Java yukaridan asagiya, soldan saga dogru gider. Eger bir satirda birden fazla islem varsa, once hangisini yapacagini bilmesi gerekdir

```
int sayi2 = sayi1++; once sayi1'in eski degerini sayi2'ye atar, sonra sayi1'i 1 artirir  
sout(++sayi1); once sayi1'in degerini 1 artirip, sonra yeni degerini yazdirir
```

- 3- Modulus % : bir bolme isleminde kalan'I verir (matematikdeki mod islemi gibidir)

Bizim en çok kullandigimiz yerler

- bir sayinin tek mi cift mi oldugunu bulmak
- bir sayinin verilen bir sayıya tam bolunup bolunemedigine bakmaz
- %10 yaparak sayinin birler basamagini almak

- 4- bir sayinin rakamlari toplamini bulmak icin icin, basamak sayisi miktarinca su islemeler tekrar edilir

- %10 ile birler basamagini bulmak
- bulunan basamaktaki rakami rakamlar toplamina eklemek
- toplama ekledigimiz rakamdan kurtulmak icin sayı/10 yapmak



# Concatenation / (String Datalari Birlestirme)

Birden cok String'i + isareti ile topladiginizda Java bu String degiskenleri birlestirerek yeni bir String olusturur

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

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

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

System.out.println(a+b+c); → Hello23
System.out.println(c+b+a); → 5Hello
System.out.println(a+(b+c)); → Hello5
System.out.println(a+b*c); → Hello6
```



# Concatenation

Soru 1 ) Asagida verilen variable'lari kullanarak istenen sonuclari yazdiran programlari yaziniz.

## Variables

```
String str1= "Java";
String str2= "Guzel";
int sayi1=5;
int sayi2=4;
```

## Istenen Yazilar

- 1) Java Guzel 54
- 2) Java 5 Guzel
- 3) Java 94
- 4) Java 19
- 5) 54 Guzel



## Relational Operators /(Karsilastirma Operatorleri)

= Assignment (Atama yapar) operatoru

**int num1=3;** num1 degiskenenine 3 degerini atar

**String str1 = "Ali" + " " + "Can";** str1'e Ali Can degeri atar

**c = c+5;** c'nin degerini 5 artirir ve son degeri c'ye atar

**==** Cift esittir isareti / karsilastirma (Comperison) operatoru

**boolean sonuc1 = 5+2 == 7;** sonuc1 degeri **true** olur

**boolean sonuc2 = 5\*2 == 15;** sonuc2 degeri **false** olur



# Relational Operators /(Karsilastirma Operatorleri)

**!=** Esit degildir isareti

**boolean sonuc1= 5+2 != 7;**

sonuc1 degeri **false** olur

**System.out.println(5\*2 != 15);**

**true** yazdirir

› Buyuktur , **>=** Buyuk veya esittir

**boolean sonuc1= 5+2 >= 7;**

sonuc1 degeri **true** olur

**System.out.println(5\*2 > 15);**

**false** yazdirir

‹ Kucuktur , **<=** Kucuk veya esittir

**boolean sonuc1= 5+2 < 7;**

sonuc1 degeri **false** olur

**System.out.println(5\*2 < 15);**

**true** yazdirir



# Conditional Operators / (Sart Operatorleri)

## && AND (ve) isareti

&& isareti ile birlestirilen tum ifadeler dogru ise sonuc true olur.

Diger tum durumlarda false doner. ( && operatoru mukemmeliyetcidir )

```
boolean sonuc1= (5+2 == 7) && (4+3 !=5) ;  
System.out.println((5*2 != 15) && (5>7));
```

sonuc1 degeri **true** olur  
**false** yazdirir

## || OR (veya) isareti

|| isareti ile birlestirilen tum ifadeler yanlis ise sonuc false olur.

Diger tum durumlarda truee doner. ( || operatoru iyimserdir )

```
boolean sonuc1= (5+2 == 7) || (4+3 !=5) ;  
System.out.println((5*2 == 15) || (5>7));
```

sonuc1 degeri **true** olur  
**false** yazdirir



## & && Arasindaki Fark

**&** isareti kullanildiginda Java isaretin iki yanindaki mantiksal ifadelerin ikisini de kontrol eder. Bu islem kodumuzu yavaslatir

**40<30 & 50==50 & 60>50**

ilk karsilastirma yanlis olmasina ragmen Java tum karsilastirmalari kontrol etmeye devam eder.

**&&** isareti kullanildiginda ise Java en bastan kontrol etmeye baslar, mantiksal ifadelerin birinde yanlisi bulursa sonrakileri kontrol etme ihtiyaci duymaz. Bu islem kodumuzu hizlandirir

**40<30 && 50==50 && 60>50**

ilk karsilastirma yanlis oldugunu gorunce Java diger karsilastirmalari kontrol etmeden alt satira gecer.



**BATCH** : Batch 59-60  
**LESSON** : Java 07  
**DATE** : 26.02.2022  
**SUBJECT** : If Statements

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



## If Statements / (If cümleleri)

Eger hava guzel olursa piknige gidecegiz. ([guzel olmazsa karar yok](#))

Eger (hava guzel olursa) {[piknige gideriz](#)} her durumda alt satira gecer

**If** (boolean şart) {[sart saglanirsa istenen kod](#)} her durumda alt satira gecer

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

    int a = 2;
    int b = 3;

    if (a>b) {
        System.out.println(a+b);
    }
    if (a==b) {
        System.out.println(a*b);
    }
}
```



# If Statements / (If cumleleri)

**Not :** If statement birden fazla olursa hepsi birbirinden bagimsiz olur. If cumlelerini birbirine baglamayi da ogrenecegiz.

Eger hava guzel olursa piknige gidecegiz. ([guzel olmazsa karar yok](#))

Eger Ali ararsa ona kizacagim. ([aramazsa karar yok](#))

Eger aksam mac varsa onu izleriz. ([mac yoksa karar yok](#))

```
8  int a=10;
9  int b=8;
10
11 if (a==b) {
12     System.out.println("iki sayi esit");
13 }
14
15 if (a+b<100) {
16     System.out.println("sayilarin toplami yuzden kucuk");
17 }
18
19 if (a*b>1000) {
20     System.out.println("sayilarin carpimi bin'den buyuk");
21 }
```



## If Statements / (If cümleleri)

Soru 1) Kullanicidan bir tamsayi isteyin ve sayinin tek veya cift oldugunu yazdirin

Soru 2) Kullanicidan gun isimlerinden birinin ilk harfini isteyin ve o harfle baslayan gun isimlerini yazdirin

Ornek: `ilkHarf=P` `output = "Pazar, Pazartesi veya Persembe"`  
`ilkHarf=S` `output = "Sali"`

**\*\*\* Buyuk kucuk harf problem olmaması icin toUpperCase methodunu kullanın**

Soru 3) Kullanicidan gun ismini alin ve haftaici veya hafta sonu oldugunu yazdirin

Ornek: `gun=Pazar` `output = "Hafta sonu"`  
`gun=Sali` `output = "Hafta ici"`

**\*\*\* String icin equals method'unu kullanın**

Soru 4) Kullanicidan dikdortgenin kenar uzunluklarini isteyin ve dikdortgenin kare olup olmadigini yazdirin

Soru 5) Kullanicidan bir gun alin eger gun "Cuma" ise ekran'a "Muslimanlar icin kutsal gun" yazdirin. "Cumartesi" ise ekran'a "Yahudiler icin kutsal gun" yazdirin. "Pazar" ise ekran'a "Hiristiyanlar icin kutsal gun" yazdirin



## If Else Statements

Eger hava guzel olursa piknige gideriz, yoksa evde otururuz.

Eger (hava guzel olursa) {piknige gideriz} yoksa {evde otururuz}

If (boolean şart) {şart saglanırsa istenen kod} else {şart saglanmazsa istenen kod}

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

    int a = 2;
    int b = 3;

    if (a>=b) {
        System.out.println(a+b);
    } else {
        System.out.println(a*b);
    }
}
```



## If Else Statements

### Sorular

- Soru 1) Kullanicidan dikdortgenin kenar uzunluklarini isteyin ve dikdortgenin kare olup olmadigini yazdirin
- Soru 2) Kullanicidan bir karakter girmesini isteyin ve girilen karakterin harf olup olmadigini yazdirin
- Soru 3) Kullaniciya yasini sorun, eger yas 65'den kucuk ise "emekli olamazsin, calismalisin", 65'e esit veya buyukse "Emekli olabilirsin" yazdirin
- Soru 4) Kullanicidan bir ucgenin uc kenar uzunlugunu alin eger uc kenar uzunlugu birbirine esit ise ekrana "Eskenar ucgen" yazdirin. Diger durumlarda ekrana "Eskenar degil" yazdirin.



**BATCH** : Batch 59-60

**LESSON** : Java 08

**DATE** : 28.02.2022

**SUBJECT** : If Statements

Ternary Operator



techproeducation



techproeducation



techproeducation



techproeducation



techproedu



## If Else If ... Statements

Eger soruyu biliyorsa Ali soruyu cozsun , o bilmiyorsa Veli biliyorsa Veli cozsun,  
o da bilmiyorsa Ayse biliyorsa, Ayse cozsun, o da bilmiyorsa Fatma biliyorsa,  
Fatma cozsun, o da bilmiyorsa kim isterse o cozsun.

Eger soruyu biliyorsa Ali soruyu cozsun , o bilmiyorsa Veli biliyorsa Veli cozsun,  
o da bilmiyorsa Ayse biliyorsa, Ayse cozsun, o da bilmiyorsa Fatma biliyorsa,  
Fatma cozsun, o da bilmiyorsa kim isterse o cozsun.

```
If (sart) {sart saglanirsa istenen kod} else if {sart saglanmazsa istenen kod}  
else if {sart saglanmazsa istenen kod} else if ( kac tane durum varsa else if ..... )  
else {sart saglanmazsa istenen kod}
```



## If Else If ... Statements

- Soru 5) Kullanicidan gun ismini yazmasini isteyin. Girilen isim gecerli bir gun ise gun isminin 1.,2. ve 3.harflerini ilk harf buyuk diger ikisi kucuk olarak yazdirin, gun ismi gecerli degilse "Gecerli gun ismi giriniz" yazdirin
- Soru 6) Kullanicidan iki sayı isteyin, sayilarin ikisi de pozitif ise sayilarin toplamini yazdirin, sayilarin ikisi de negative ise sayilarin carpimini yazdirin, sayilarin ikisi farkli işaretlere sahipse "farkli işaretlerde sayilarla islem yapamazsin" yazdirin, sayilardan sifira esit olan varsa "sifir carpmaya gore yutan elemandir" yazdirin.
- Soru 7) Kullanicidan 100 uzerinden notunu isteyin. Not'u harf sistemecevirip yazdirin. 50'den kucukse "D", 50-60 arasi "C", 60-80 arasi "B", 80'nin uzerinde ise "A"
- Soru 8) Kullanicidan maas icin bir teklif isteyin ve asagidaki degerlere gore cevap azdirin.  
Teklif 80.000'in uzerinde ise "Kabul ediyorum" ,  
60 – 80.000 arasında ise "Konusabiliriz" ,  
60.000'nin altinda ise "Maalesef Kabul edemem" yazdirin



## Nested If Else Statements

Eger calisan kadinsa 60 yasindan buyuk oldugunda emekli olabilir, calisan erkekse 65 yasindan buyukse emekli olabilir

Eger (calisan kadinsa) {Kadin yasini kontrol et} ,  
yoksa {erkek yasini kontrol et}

```
If (calisan kadinsa)
    if (yas>60) {emekli olabilirsin} else {emekli olamazsin}
else
    if (yas>65) {emekli olabilirsin} else {emekli olamazsin}
```



## If Else Statements

Soru 11) Nested If kullanarak asagidaki soruyu cozen kodu yaziniz.

Kullanicidan bir sifre girmesini isteyin

Eger ilk harf buyuk harf ise "A" olup olmadigini kontrol edin. Ilk harf A ise "Gecerli Sifre" degilse "Gecersiz Sifre" yazdirin.

Eger ilk harf kucuk harf ise "z" olup olmadigini kontrol edin. Ilk harf z ise "Gecerli Sifre" degilse "Gecersiz Sifre" yazdirin.

Soru12) Kullanıcıdan 4 basamaklı bir sayı girmesini isteyin. Girdiği sayı 5'e bölünüyorsa son rakamını kontrol edin. Son rakamı 0 ise ekrana "5'e bölünen çift sayı" yazdırın. Son rakamı 0 değil ise "5'e bölünen tek sayı" yazdırın. Girdiği password 5'e bölünmüyorsa ekrana "Tekrar deneyin" yazdırın.



# If Else If Statements

## Soru 13) Interview Question

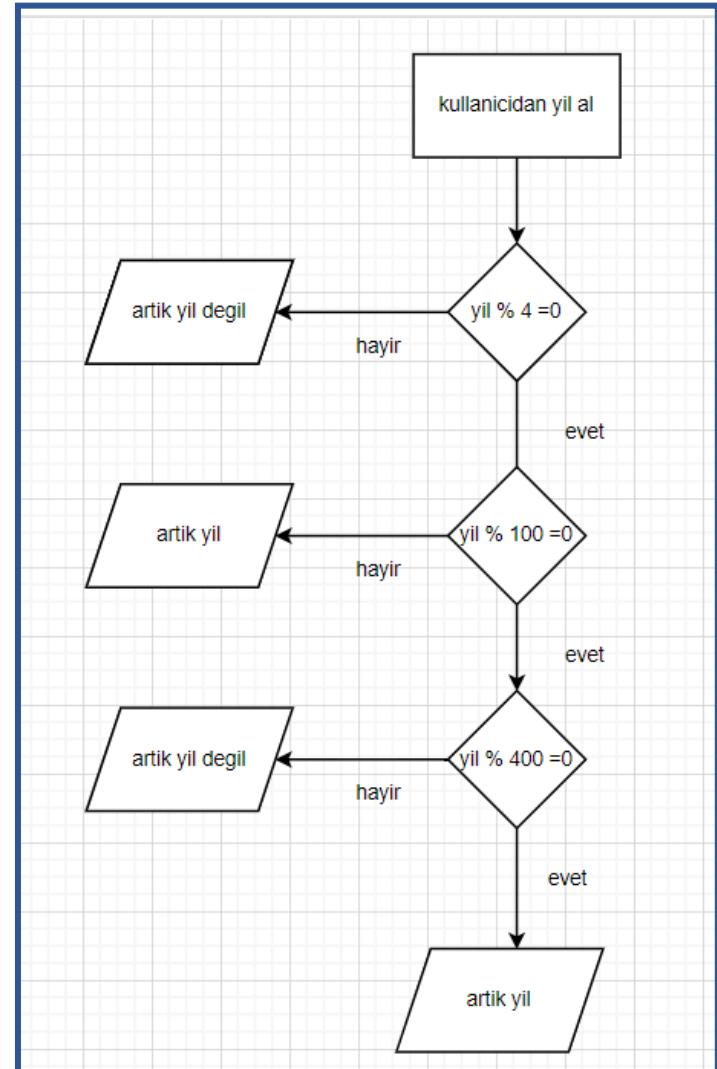
Kullanicidan artik yil olup olmadigini kontrol etmek icin yil girmesini isteyin.

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/>





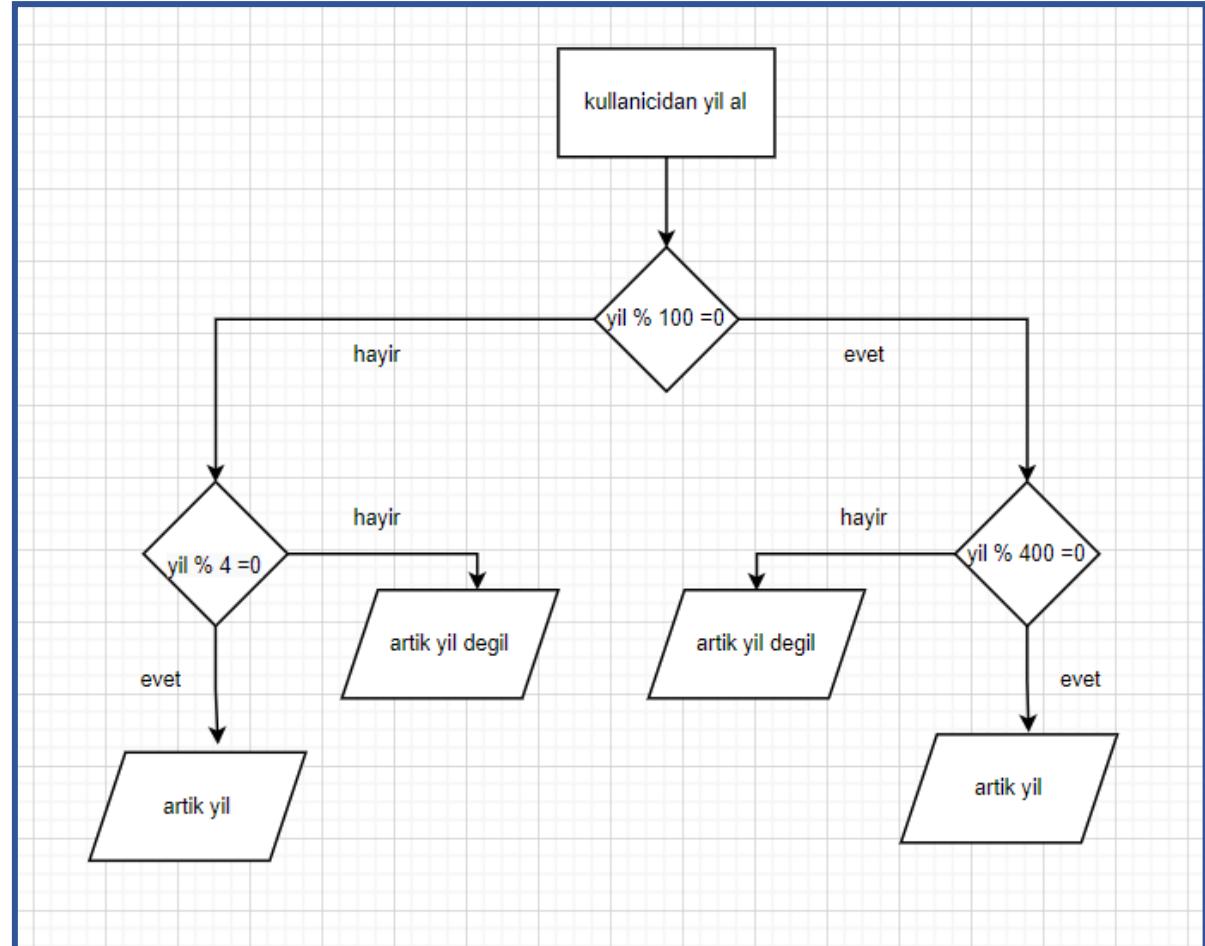
# Nested If Else Statements

## Soru 10) 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'un kati olmasina ragmen 100 ile bolunebilen yillardan sadece 400'un kati olan yillar artik yildir





**BATCH** : Batch 59-60  
**LESSON** : Java 09  
**DATE** : 01.03.2022  
**SUBJECT** : Ternary Operator  
Switch Statement





## Önceki Dersten Aklımızda Kalanlar

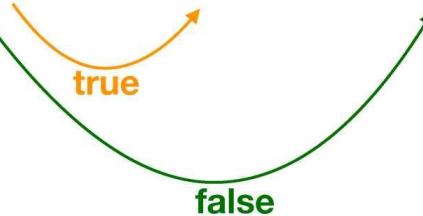
### If Else Statements

- Guluk hayatımızda kullandığımız şart cümleleri gibir.
- If else cümlelerini daha iyi anlayabilmek için 3 farklı formatta gözden geçirdik
  - 1) Bağımsız if cümleleri : tek şart - tek sonuc (şart false olursa ne olacağı belirtmemiş) . Bu durumda tüm if cümleleri birbirinden bağımsız olarak çalışır. Tüm if body'leri çalışabilecegi gibi, hic bir if body'si çalışmaya bilir.
  - 2) If- Else : bir olayda sadece 2 durum söz konusu olduğunda kullanılır. Tek bir şart ve şart doğru olduğunda sonuc / şart false olduğunda sonuc şeklinde 2 sonuc olasılığı çalışır. Sadece biri ve mutlaka biri çalışır
  - 3) If-Else If ... eğer bir olay için belirli sayıda sonuc olasılığı varsa kullanılır, durum sayısından bir eksik sayıda if cümlesi gereklidir, eğer else ile biterse bir tanesi ama sadece bir tanesi mutlaka çalışır, else ile bitmezse o zaman if body'lerinden sadece biri çalışabilir veya hic birisi çalışmaya bilir
- Eğer bizden istenen kod'da birden fazla değişken varsa iç içe (nested) if else blokları kullanılabılır.

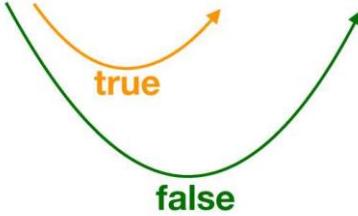


# Ternary Operator

```
if(Condition) { Code 1} else {Code 2}
```



```
Condition ? Code 1 : Code 2
```



**Not1 :** Ternary islemi If Statement ile yapacagimiz islemleri basit olarak yapmamizi saglar

**Not2 :** Ternary islemi bize bir sonuc donecegi icin, bu islemi bir variable'a atamaliyiz.

```
public static void main(String[] args) {  
    int x=10;  
  
    (x/2==0) ? "cift sayi" : "tek sayi";
```

```
public static void main(String[] args) {  
    int x=10;  
  
    String sonuc = (x/2==0) ? "cift sayi" : "tek sayi";  
    System.out.println(sonuc);
```



# Ternary Operator

Ekranda Ne Goruruz ?

Soru1 : int y = 112;

```
System.out.println( (y > 5) ? ("Inek") : ("Koyun") );
```

Soru2 : int y = 112;

```
System.out.println((y < 91) ? 9 : 11);
```

Soru3 : int y = 1;

```
int z = 1;
```

```
int a = y<10 ? y++ : z++;
```

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



## Ternary Operator

**Soru1 )** Kullanicidan iki sayi alin ve buyuk olmayan sayiyi yazdirin

**Soru2 )** Kullanicidan bir tamsayi alin ve sayinin tek veya cift oldugunu yazdirin

**Soru3 )** Kullanicidan bir sayi alin ve sayinin mutlak degerini yazdirin

**Soru4 )** Kullanicidan bir sayi alin. Sayi pozitifse “Sayi pozitif” yazdirin, negatifse sayinin karesini yazdirin



## Nested Ternary

Condition ? (Kod 1) : (Kod 2) ;

Condition1 ? Durum1 : Durum2

Condition2 ? Durum1 : Durum2

**Soru1 :** Kullanicidan bir tamsayi alin ve sayi 10'dan kucukse "Rakam" , 100'den kucukse "iki basamakli sayi"degilse "uc basamakli veya daha buyuk sayi" yazdirin

**Soru2 :** Kullanicidan bir harf isteyin kucuk harf ise consola "Kucuk Harf" , buyuk harfse consola "Buyuk Harf" yoksa "girdiginiz karakter harf degil" yazdirin.



## Nested Ternary

Ekranda Ne Goruruz ?

Soru1 : int y = 8;

(y > 5) ? (y<10 ? 2\*y : 3\*y) : (y>10 ? 2+y : 3+y);

Soru2 : int y = 12;

(y > 5) ? (y<10 ? 2\*y : 3\*y) : (y>10 ? 2+y : 3+y);

Soru3 : int y = 5;

(y > 5) ? (y<10 ? 2\*y : 3\*y) : (y>10 ? 2+y : 3+y);

Soru4 ) Kullanicidan dikdortgenin uzunlugunu ve genisligini alin, girilen degerlere gore dikdorgenin kare olup olmadigini yazdirin.

Soru5 ) Kullanicidan bir sayi alin ve sayi 3 basamakli ise “uc basamakli sayi”, yoksa “Uc basamakli degil” yazdirin



**BATCH** : Batch 59-60

**LESSON** : Java 10

**DATE** : 02.03.2022

**SUBJECT** : **Switch Statement**  
**String Manipulation**

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



# Switch Statement

If else ile cozdugumuz sorularda kontrol etmemiz gereken şart sayısı çok olduğunda switch Statement kullanılır.

```
public static void main(String[] args) {
    int sayı = 3;

    switch(sayı) {
        case 1 :
            System.out.println("sayı = 1");
            break;
        case 2 :
            System.out.println("sayı = 2");
            break;
        case 3 :
            System.out.println("sayı = 3");
            break;
        case 4 :
            System.out.println("sayı = 4");
            break;
        default :
            System.out.println("sayı bunlardan biri değil");
    }
}
```



## Switch Statement

**break** komutu yapacagimiz islem bittiginde switch statement'in sonuna gitmemizi saglar.

**Java** istenen case'e gittikten sonra **break** komutunu gorene kadar tum case'leri calistirir.

**default** komutu basta tanimlanan degisken icin hic bir case calismazsa calistirmak isedigimiz kodlari yazdigimiz bolumdur.

(If else statements da en sonda yazdigimiz else gibi calisir)

Switch Statement'da long,double,float ve boolean **kullanilamaz**



## Switch Statement

**Soru1 :** Kullanicidan haftanin kacinci gunu oldugunu sorun ve gun ismini yazdirin

**Soru2 :** Kullanicidan kacinci ay oldugunu sorun ve ay ismini yazdirin

**Soru3 :** Kullanicidan bir sayi girmesini isteyin

Girilen sayi

10 ise “iki basamakli en kucuk sayi”

100 ise “uc basamakli en kucuk sayi”

1000 ise “dort basamakli en kucuk sayi”

diger durumlarda “Girdigin sayiyi degistir” yazdirin

**Soru4 :** Kullanicidan SDET kisaltmasindaki harflerden birini yazmasini isteyin.

Kullanici S girerse “Software”

D girerse “Developer”

E girerse “Engineer”

T girerse “In Testing” yazdirin

**Soru5 :** Kullanicidan gun ismini alip haftaici veya hafta sonu yazdiralim



# String Manipulation / Methods

## 1- concatenation

Birden fazla String'i birleştirerek tek bir String haline getirmek için kullanılır.

İki şekilde kullanılır.

i) + (toplama) işaretleri ile

```
public static void main(String[] args) {  
    String isim= "Ali";  
    String soyisim="Can";  
  
    System.out.println(isim + " " + soyisim);
```

Output :

Ali Can

ii) concat() methodu kullanarak

```
public static void main(String[] args) {  
    String isim= "Ali";  
    String soyisim="Can";  
  
    System.out.println(isim.concat(soyisim));
```

Output :

AliCan



# String Manipulation / Methods

## 2- charAt()

Istenen indexdeki karakteri (char) dondurur. Index 0'dan baslar, maximum index (String'in uzunlugu - 1) dir.

```
public static void main(String[] args) {  
    String isim= "Techproeducation";  
    System.out.println(isim.charAt(3));
```

Output :

h

Eger method'da index olarak maximum indexden buyuk bir sayi kullanilirsa Java hata verir (**StringIndexOutOfBoundsException**).

```
public static void main(String[] args) {  
    String isim= "Techproeducation";  
    System.out.println(isim.charAt(20));
```

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: 20  
at java.lang.String.charAt(Unknown Source)  
at _00_anlik.asd.main(asd.java:11)
```



**BATCH** : Batch 59-60  
**LESSON** : Java 11  
**DATE** : 03.03.2022  
**SUBJECT** : String Manipulation





## Önceki Dersten Aklımızda Kalanlar

1. Switch - case : If else ile yapabilecegimiz bazi sorulari cok fazla durum varsa switch-case ile yapmayi tercih ederiz
2. En buyuk artisi daha organizeli ve daha Rahat anlasilir bir kod olmasidir, ayrica istedigimiz zaman case'leri gruplandırma da mumkundur
3. Switch parantezine Boolean, long, float ve double yazilmaz
4. Case'lere ise somut degerler veya degeri degismeyecek variable'lar yazabiliriz. Birden fazla case'de ayni deger olamaz
5. Switch icinde yazilan degere gore ilgili case calismaya baslar ve break gorunceye kadar calismaya devam eder
6. Eger case'lerde belirtilen durumların disindaki tum degerler icin ortak bir sonucumuz varsa default : altinda yazabiliriz. Default, if, else if... nin sonundaki else gibidir, geriye kalan tum degerleri icerir



# String Manipulation / Methods

3-toUpperCase()

4-toLowerCase()

Girilen String degiskendeki tum harfleri istenen bicime cevirir.

```
public static void main(String[] args) {  
  
    String isim= "TechProeDucatIon";  
  
    System.out.println(isim.toLowerCase());  
    System.out.println(isim.toUpperCase());
```

Output :

techproeducation  
TECHPROEDUCATION

NOT : toLowerCase(Locale locale)

Girilen String degiskendeki tum harfleri istenen local dilde istenen bicime cevirir.

```
public static void main(String[] args) {  
  
    String isim= "TECHPROEDUCATION";  
  
    System.out.println(isim.toLowerCase(Locale.forLanguageTag("tr")));
```

Output :

techproeducation



## String Manipulation / Methods

### 5-equals

Verilen iki String'in iceriginin birbirine esit olup olmadigini kontrol eder.

Eger verilen Stringlerdeki tum karakterler (bosluk, buyuk harf, kucuk harf, ozel karakter ..) tamamen ayni ise **TRUE** doner, aksi durumda (bir karakter bile farkli olsa) **FALSE** doner.

```
public static void main(String[] args) {  
  
    String isim1= "Ali Can";  
    String isim2= "Ali Can";  
  
    System.out.println(isim1.equals(isim2));
```

Output :

true



# String Manipulation / Methods

equals Vs ==

(Interview Sorusu)

**equals()** methodu verilen iki String'in iceriginin birbirine esit olup olmadigini kontrol eder.

**==** karsilastirma operatoru ise verilen iki String objesinin degerinin yaninda reference(adres)'larine da bakar,

Ayni degere sahip olsa da farkli iki objeyi **==** ile karsilastirdigimizda sonuc **FALSE** olur.

```
public static void main(String[] args) {  
  
    String isim1= "Ali Can";  
    String isim2= isim1+"";  
  
    System.out.println(isim1==isim2);  
  
    System.out.println(isim1.equals(isim2));
```

Output :

false

true



## String Manipulation / Methods

### 6-equalsIgnoreCase

Verilen iki String degiskeni BUYUK HARF / kucuk harf farki gozetmeksizin karsilastirir.

Buyuk / kucuk harf farkliliği disinda herhangi bir karakter farkliliği olduğunda equals methodunda olduğu gibi FALSE dondurur.

```
public static void main(String[] args) {  
    String isim1= "Ali Can";  
    String isim2= "ali can";  
  
    System.out.println(isim1.equalsIgnoreCase(isim2));
```

Output :

true



# String Manipulation / Methods

## 7-length()

Verilen String'deki karakter sayisini dondurur.

```
public static void main(String[] args) {  
    String isim= "Ali Can";  
    System.out.println(isim.length());
```

Output :

7

```
public static void main(String[] args) {  
    String isim= "";  
    System.out.println(isim.length());
```

Output :

0

```
public static void main(String[] args) {  
    String isim= null;  
    System.out.println(isim.length());
```

Exception in thread "main" java.lang.NullPointerException  
at \_00\_anlik.asd.main(asd.java:11)



## String Manipulation / Methods

### 8-indexOf()

Verilen String'de istenen karakterin kullanildigi ilk index'i dondurur.

- 1) char'in index'i sorgulanabilir
- 2) Parametre String olabilir
- 3) Olmayan karakter sorgulanirsa
- 4) Parametre kelime olabilir
- 5) Belli bir index'ten sonrasi sorgulanabilir

```
String str= "Calisirsaniz, Java ogrenmek cok kolay";  
  
System.out.println(str.indexOf('a'));  
  
System.out.println(str.indexOf("a"));  
  
System.out.println(str.indexOf("t"));  
  
System.out.println(str.indexOf("Java"));  
  
System.out.println(str.indexOf('a',11));
```

Output : 1  
: 1  
: -1  
: 14  
: 15



## String Manipulation / Methods

### indexOf( ) Sorular

Soru 1) Kullanicidan bir cümle ve bir harf isteyin, harfin cümlede var olup olmadığını yazdirin

Soru 2) Kullanicidan bir cümle ve bir kelime isteyin, kelimenin cümledeki kullanımına bakarak aşağıdaki 3 cümleden uygun olanı yazdirin

- Girilen kelime cümlede kullanilmamis.
- Girilen kelime cümlede 1 kere kullanilmis.
- Girilen kelime cümlede 1'den fazla kullanilmis.



**BATCH** : Batch 59-60  
**LESSON** : Java 12  
**DATE** : 04.03.2022  
**SUBJECT** : String Manipulation

- techproeducation
- techproeducation
- techproeducation
- techproeducation
- techproedu



## Önceki Dersten Aklımızda Kalanlar

1. String Manipulation : Equal : == operatoru String degerler icin saglikli calismaz, cunku == hem degree hem de referansa bakar, dolayisiyla degeri tamamen ayni olan iki String'i == ile karsilastirirsanz false da verebilir, true'da.

Emin olmak istiyorsak sadece degerleri karsilastiran equals() tercih edilmelidir.

2- equalsIgnoreCase( ) : String case sensitive(buyuk-kucuk g=harf duyarli)dir. Dolayisiyla karsilastirdigimiz kelime ayni kelime am a yazilisi farkli ise equal method'u false doner (Money, MONEY, money farklidir), eger buyuk kucuk harf olmasi bizim icin onemli degilse bu durumda equal yerine equalsIgnoreCase tercih edilir

3- length( ): bize String'in karakter sayisini verir(ozel karakter ve space de dahil). Index ile length( ) karistirilmamalidir, index 0'dan baslarken, length uzunluk ifade ettigi icin 1'den baslar

son harfin index'ini dinamik olarak bulmak icin str.length() - 1 kullanilir

4- indexOf (parameter veya parametreler ) : istedigimiz bir char veya String'in baska bir String icerisinde kacinci index'de oldugunu bize dondurur.

Eger aramaya bastan degil belirledigimiz bir index'den baslamasini istiyorsak iki parametre kullanilir str.indexOf(arananStr , baslangic indexi). Baslangic index'i inclusive'dir.



# String Manipulation / Methods

## 9-lastIndexOf()

Verilen String'de istenen karakterin kullanildigi son index'i dondurur.

- 1) char'in son index'i sorgulanabilir
- 2) Parametre String olabilir
- 3) Olmayan karakter sorgulanirsa
- 4) Parametre kelime olabilir
- 5) Belli bir index'ten oncesi sorgulanabilir

```
String str= "Calisirsaniz, Java ogrenmek cok kolay";
```

```
System.out.println(str.lastIndexOf('a'));
```

: 35

```
System.out.println(str.lastIndexOf("a"));
```

: 35

```
System.out.println(str.lastIndexOf("t"));
```

: -1

```
System.out.println(str.lastIndexOf("Java"));
```

: 14

```
System.out.println(str.lastIndexOf('a',11));
```

: 8



## String Manipulation / Methods

### lastIndexOf() Sorular

Soru 1) Kullanicidan bir cümle ve bir harf isteyin, harfin cümlede var olup olmadığını yazdirin

Soru 2) Kullanicidan bir cümle ve bir kelime isteyin, kelimenin cümledeki kullanımına bakarak asagidaki 3 cümleden uygun olanı yazdirin

- Girilen kelime cümlede kullanilmamis.
- Girilen kelime cümlede 1 kere kullanilmis.
- Girilen kelime cümlede 1'den fazla kullanilmis.



# String Manipulation / Methods

## 10-contains()

Verilen String'in istenen karakter(ler)i icerip icermedigini kontrol eder. Iceriyorsa TRUE, icermiyorsa FALSE dondurur.

- 1) Parametre String olmalidir
- 2) Olmayan karakter sorgulanirsa
- 3) Parametre metin olabilir

```
public static void main(String[] args) {  
  
    String str= "Calisirsaniz, Java ogrenmek cok kolay";  
  
    System.out.println(str.contains("a"));  
  
    System.out.println(str.contains("t"));  
  
    System.out.println(str.contains("Java"));
```

true
false
true

**NOT** contains() methodu char icin kullanilamaz, String kullanmak zorunludur.



## String Manipulation / Methods

**contains( )** sorular

**Soru 1)** Kullanicidan email adresini girmesini isteyin, mail @gmail.com icermiyorsa “lutfen gmail adresi giriniz”, @gmail.com ile bitiyorsa “Email adresiniz kaydedildi ” , @gmail.com ile bitmiyorsa lutfen yazimi kontol edin yazdirin

**Soru 2)** Kullanicidan bir cümle isteyin. Cümle “buyuk” kelimesi içeriyorsa tüm cumleyi buyuk harf olarak, “kucuk” kelimesi içeriyorsa tüm cumleyi kucuk harf olarak yazdirin, iki kelimeyi de icermiyorsa “Cümle kucuk yada buyuk kelimesi icermiyor” yazdirin.



## String Manipulation / Methods

### 11-endsWith()

Verilen String'in istenen karakter(ler) ile bitip bitmediğini kontrol eder. İstenen karakter(ler) ile bitiyorsa TRUE, yoksa FALSE dondurur.

- 1) Parametre String olmalıdır
- 2) Yanlış karakter sorgulanırsa
- 3) Parametre kelime olabilir

```
String str= "Calisirsaniz, Java ogrenmek cok kolay";  
  
System.out.println(str.endsWith("y"));  
  
System.out.println(str.endsWith("t"));  
  
System.out.println(str.endsWith("olay"));
```

true
false
true



## String Manipulation / Methods

### 12-startsWith()

Verilen String'in istenen karakter(ler) ile baslayip baslamadigini kontrol eder. Istenen karakter(ler) ile basliyorsa TRUE, yoksa FALSE dondurur.

- 1) Parametre String olmalidir
- 2) Parametre kelime olabilir
- 3) Belirli karakterden sonrasi olabilir

```
String str= "Calisirsaniz, Java ogrenmek cok kolay";  
  
System.out.println(str.startsWith("C"));           true  
  
System.out.println(str.startsWith("Calis"));        true  
  
System.out.println(str.startsWith("s",4));          true  
  
System.out.println(str.startsWith("Java",14));       true
```



## String Manipulation / Methods

### 13-isEmpty()

Verilen String'in uzunluğu 0(sıfır) ise (Hicbir karakter içermiyorsa) TRUE, yoksa FALSE döndürür.

```
String str= "Calisirsaniz, Java ogrenmek cok kolay";  
  
System.out.println(str.isEmpty());  
  
String str2="";  
  
System.out.println(str2.isEmpty());  
  
String str3=null;  
  
System.out.println(str3.isEmpty());
```

false

true

Hata verir

Exception in thread "main" java.lang.NullPointerException  
at \_00\_anlik.asd.main(asd.java:19)



## String Manipulation / Methods

### 14- replace()

Verilen String'deki istenen karakter(ler)i istenen yeni karakter(ler) ile degistirir.

```
String str= "Java ogrenmek cok kolay";  
  
System.out.println(str.replace("a", "x"));  
  
System.out.println(str.replace("Java", "x"));  
  
System.out.println(str.replace("a", "xxx"));  
  
System.out.println(str.replace("a", ""));  
  
System.out.println(str.replace('a', 'x'));
```

Jvxogrenmek cok kolxy  
x ogrenmek cok kolay  
Jxxxxvxxxx ogrenmek cok kolxxxxy  
Jv ogrenmek cok koly  
Jvxogrenmek cok kolxy

**NOT:** replace() methodu char icin de kullanilabilir



# String Manipulation / Methods

## 15- replaceAll()

replace() methodu ile benzer olarak verilen String'deki istenene karakter(ler)i istenen yeni karakter(ler) ile degistirir. Aralarindaki farklar

- replace() methodunda char kullanilabilir, replaceAll()'da char kullanilamaz
- replaceAll() methodunda Regular Expressions kullanilabilir

**\s** : bosluk (space)

**\S** : bosluk disindaki tum karakterler

**\w** : harfler ve rakamlar (a-z , A-Z, 0-9)

**\W** : harfler ve rakamlar disindaki tum karakterler

**\d** : rakamlar (0-9)

**\D** : rakamlar disindaki tum karakterler



**BATCH** : Batch 59-60  
**LESSON** : Java 13  
**DATE** : 05.03.2022  
**SUBJECT** : String Manipulation





# String Manipulation / Methods

## replaceAll()

```
public static void main(String[] args) {  
  
    String str= "Java'da rakamlar 1234567890";  
  
    System.out.println(str.replaceAll("a", "*"));  
  
    System.out.println(str.replaceAll("\\s", "*"));  
  
    System.out.println(str.replaceAll("\\S", "*"));  
  
    System.out.println(str.replaceAll("\\w", "*"));  
  
    System.out.println(str.replaceAll("\\W", "*"));  
  
    System.out.println(str.replaceAll("\\d", "*"));  
  
    System.out.println(str.replaceAll("\\D", "*"));  
}
```

J\*v\*d\* r\*k\*ml\*r 123456789  
Java'da\*rakamlar\*1234567890  
\*\*\*\*\* \*\*\*\*\* \*\*\*\*\*  
\*\*\*\*\* \*\*\* \*\*\*\*\* \*\*\*\*\*  
Java\*da\*rakamlar\*1234567890  
Java'da rakamlar \*\*\*\*\*  
\*\*\*\*\*1234567890



## String Manipulation / Methods

### 16- replaceFirst()

Verilen String'deki istenen karakter(ler)in ilkini, istenen yeni karakter(ler) ile degistirir

```
public static void main(String[] args) {  
  
    String str= "Java'da rakamlar 1234567890";  
  
    System.out.println(str.replaceFirst("a", "*"));  
  
    System.out.println(str.replaceFirst("lar", "*"));  
  
    System.out.println(str.replaceFirst("\s", "*"));  
  
    System.out.println(str.replaceFirst("\D", "*"));
```

J\*va'da rakamlar 1234567890  
Java'da rakam\* 1234567890  
Java'da\*rakamlar 1234567890  
\*ava'da rakamlar 1234567890



# String Manipulation / Methods

## 17- substring()

Index kullanarak verilen String'in istenen parcasini almamizi saglar.

- Parametre olarak 1 sayi girilirse, girilen index'den String'in sonuna kadar bolumu
- Parametre olarak 2 sayi girilirse, girilen 1.sayidaki indexden (inclusive) baslayip, 2.sayiya kadar (exclusive) karakteri bize dondurur

```
public static void main(String[] args) {  
  
    String str= "Java OOP konsepti kullanir";  
  
    System.out.println(str.substring(0));  
  
    System.out.println(str.substring(10));  
  
    System.out.println(str.substring(26));  
  
    System.out.println(str.substring(29));
```

Java OOP konsepti kullanir

onsepti kullanir

Hata verir

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: -3  
at java.lang.String.substring(Unknown Source)  
at _00_anlik.asd.main(asd.java:17)
```



**BATCH** : Batch 59-60  
**LESSON** : Java 14  
**DATE** : 07.03.2022  
**SUBJECT** : String Manipulation





# String Manipulation / Methods

## substring()

```
public static void main(String[] args) {  
  
    String str= "Java OOP konsepti kullanir";  
  
    System.out.println(str.substring(5,11));  
  
    System.out.println(str.substring(3,4));  
  
    System.out.println(str.substring(8,8));  
  
    System.out.println(str.substring(8,2));
```

OOP ko

a

Hata verir

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: -6  
at java.lang.String.substring(Unknown Source)  
at _00_anlik.asd.main(asd.java:17)
```

**Not :** Java'da iki tur hata mesaji aliriz

- 1- Compile Time Error (CTE) : Kodumuzu yazarken kod altinin kirmizi cizgi olmasi
- 2- Run Time Error (RTE) : Kod calistirildiginda (Execute) karsilastigimiz hatalar



## String Manipulation / Methods

### 18- trim()

Istedigimiz String'in basinda veya sonunda var olan bosluk / "space" leri temizler

```
String str = " Java ogrenmek cok guzel. ";
System.out.println(str);
System.out.println(str.length());
System.out.println(str.trim());
System.out.println(str.trim().length());
```

| Java ogrenmek cok guzel. |

28

|Java ogrenmek cok guzel.|

24



## String Manipulation / Methods

**Soru 1)** String methodlarini kullanarak “ Java ogrenmek123 Cok guzel@ ” String’ini “Java ogrenmek cok guzel.” sekline getirin.

**Soru 2)** String seklinde verlen asagidaki fiyatlarin toplamini bulunuz

String str1 = “\$13.99”

String str2 = “\$10.55”

ipucu : Double.parseDouble() methodunu kullanabilirsiniz.

**Soru 3)** Kullanicidan isim isteyin. Eger

- isim “a” harfi iceriyorsa “Girdiginiz isim a harfi iceriyor”
- isim “Z” harfi iceriyorsa “Girdiginiz isim Z harfi iceriyor”
- ikisi de yoksa “Girdiginiz isim a veya Z harfi icermiyor” yazdirin

**Soru 4)** Kullanicidan isim ve soyismini isteyin ve hangisinin daha uzun oldugunu yazdirin.

**Soru 5)** Kullanicidan 4 harfli bir kelime isteyin ve girilen kelimeyi tersten yazdirin.



## String Manipulation / Methods

**Soru 6)** Kullanicidan bir sifre girmesini isteyin. Asagidaki şartları sagliyorsa “Sifre basari ile tanimlandi”, şartları saglamazsa “Islem basarisiz,Lutfen yeni bir sifre girin” yazdirin

- Ilk harf buyuk harf olmali
- Son harf kucuk harf olmali
- Sifre bosluk icermemeli
- Sifre uzunlugu en az 8 karakter olmali

**Soru 7)** Kullanicidan ismini, soyismini ve kredi karti bilgisini isteyin ve asagidaki gibi yazdirin

isim-soyisim : M\*\*\*\*\* B\*\*\*\*\*

kart no : \*\*\*\* \* \*\*\*\* \* 1234



**BATCH** : Batch 59-60  
**LESSON** : Java 15  
**DATE** : 08.03.2022  
**SUBJECT** : Method Creation

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



## Method Creation / Method Olusturma

**Method :** Istedigimiz islemi bizim adimiza yapan kod bloklaridir(Is yapmak icin tasarlanmis robotlar gibidirler).

Genelde iki amacla method olustururuz

1- Projemiz icerisinde tekrar tekrar kullanacagimiz bir islem icin her seferinde yeniden kod yasmak yerine bir kere yazip ihtiyacimiz oldukca kullanmak

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

**NOT :** Bir method'u olusturmak calismasi icin yeterli degildir, method'un calismasi icin mutlaka cagrılması(method call) gereklidir.



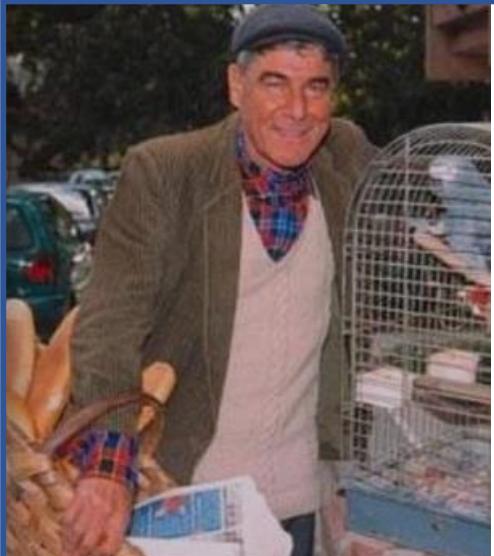


# Method Creation / Method Olusturma

Temelde 2 cesit method vardir

1 : Istedigimiz isi yapip bize bir sonuc dondurmeyen veya sadece konsolda yazi yazdiran method'lar. ( elektrik faturasini yatiran cocugumuz gibi)

Bunların return type'i void olmalıdır.



2 : Istedigimiz isi yapip bize bir sonuc dondurmeyen method'lar. ( bakkaldan alisveris yapip bize getiren kapici gibi)

- Bunların return type'i istedigimiz sonuca uygun olmalıdır.
- Method'un sonunda return keyword'u ve bize dondureceği sonuc olmalıdır
- Dondurdugu sonucu bir uygun bir variable'a atamaliyiz



**BATCH** : Batch 59-60  
**LESSON** : Java 16  
**DATE** : 09.03.2022  
**SUBJECT** : Method Creation

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



# Method Creation / Method Olusturma

## Method Olusutururken Kullanilan Keyword'ler Nelerdir?

```
public int myFirstMethod () {}  
1   2       3       4   5
```

- 1 **public** : Access Modifier (Erisim duzenleyici):method'a kimlerin erisebilecegini belirler  
**protected** : Sadece icinde bulundugu package ve child class'lardan kullanilir  
**default** : Sadece icinde oldugu package  
**private**: Sadece bulundugu class'da kullanilabilir
- 2 **Int** : Return Type, methodun ne urettigini ve bize dondurdugunu belirtir
- 3 **myFirstMethod** :Olusturdugumuz method'un ismidir. Isim mutlaka kucuk harfle baslar, birden fazla kelimededen olusursa sonraki kelimelerin ilk harfleri buyuk harf yazilir (Camel Case)
- 4 **( ) parantez**: Methodlarda isimden sonra parantez kullanilir ve gerektiginde parantez icinde parametre yazilir.
- 5 **Body (Method Body)** : { } arasinda kalan kodlarimizi yazdigimiz bolumdur



# Method Creation / Method Olusturma

```
public int myFirstMethod () {}  
1   2       3       4   5
```

## 1 Access Modifier (Erisim duzenleyici):

**public** : methoda'a kimlerin erisebilecegini belirler

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

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

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

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



## Method Creation / Method Olusturma

2 static (lleride detayli anlatilacak)

Bir method olusturulurken **static** kelimesinin kullanilmasi mecburi degildir.

Main method'umuz static oldugu icin main method'dan cagiracagimiz tum method'lari static yapmamiz gereklidir

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



## Method Creation / Method Oluşturma

- 3 **int (Return Type)** : methodun ne urettigini ve bize ne dondurdugunu belirtir.
- Return Type, primitive veya non-primitive tüm data türlerinden olabilir
  - Eğer method bir şey dondurmeyecekse (örnegin, sadece bir şey hesaplayıp yazdıracaksa) return type olarak **void** seçilir
  - Return Type olarak void dışında bir şey yazdıysak, methodun sonunda mutlaka **return** keyword kullanılmalıdır
  - Return keyword'den sonra return type'a uygun bir **deger veya variable** yazılmalıdır.
  - Return type'a sahip methodlar çağrıdıkları satırda, return keyword'den sonra yazılan değer veya variable'i dondururlar.

```
public static void main(String[] args) {
    int sonuc= topla(15,24);

}

public static int topla(int num1, int num2) {

    return num1 + num2;
}
```



## Method Creation / Method Olusturma

**4 myFirstMethod :**Olusturdugumuz method'un ismidir. Isim mutlaka kucuk harfle baslar, birden fazla kelimededen olusursa sonraki kelimelerin ilk harfleri buyuk harf yazilir (Camel Case)

**5 () parantez :** Methodlarda isimden sonra parantez kullanilir ve gerektiginde parantez icinde parametre yazilir.

**\*\*\*** Eger bir Class'da ayni isme sahip birden fazla method olusturmamiz gerekirse parametreleri farkli yapmamiz gereklidir (Overloading)



# Method Creation / Method Olusturma

6 Body (Method Body) : {} arasında kalan kodlarımızı yazdigimiz bolumdur

\*\*\* Method nerede olusturulmalidir ?

Method Class body'si icinde Main method disinda olusturulmalidir

```
public class asd {  
    public static void main(String[] args) {  
        toplama(5,4);  
    }  
    private static void toplama(int i, int j) {  
        System.out.println(i+j);  
    }  
}
```



# Method Creation / Method Olusturma

Method olusturmak method'u calistirmak icin yeterli degildir.

**Ihtiyac duyuldugunda daha onceden olusturulmus methodu calistirmak icin** Method ismi (parametreler ile birlikte) yazilmalidir.

Bu isleme method cagirma denir

```
public class asd {  
    public static void main(String[] args) {  
        toplama(5,4);  
    }  
    private static void toplama(int i, int j) {  
        System.out.println(i+j);  
    }  
}
```

\*\*\* Method cagirirken parantez icine yazilan degerlere **Arguments (arguman)** denir.

\*\*\* Method cagirirken kullandigimiz argumanlar ile method parametrelerinin uyumlu olması gereklidir.

\*\*\* Sayi parametreleri icin char degerler de arguman olarak kullanilabilir



# Method Overloading

## Interview Sorusu

1) **Overloading nedir ?** Eger bir Class'da ismi ayni fakat parametreleri farkli olan methodlar olusturursak buna **Overloading** denir.

2) **Overloading nasıl yapılır ?** Java ayni isim ve ayni parametrelerle birden fazla method olusturulmasına izin vermez. Ayni isimle birden fazla method olusturmak isterseniz **method signature (metot imzasi)**'nin degistirilmesi gerekir

3) **method signature (metot imzasi) nasıl degistirilir?**

Method signature'i degistirmek icin 3 yontem kullanilabilir

- parametrelerin data tipleri degistirilebilir
- parametrelerin sayisi degistirilebilir
- parametre sayisi ayni olmak zorunda ise farkli data tipindeki parametrelerin sirasi degistirilir

\*\*\* method'un return type'ini degistirmek, access modifier'ini degistirmek veya static kelimesi eklemek method signature'i degistirmez



**BATCH** : Batch 59-60  
**LESSON** : Java 16  
**DATE** : 09.03.2022  
**SUBJECT** : For Loop

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



# Socrative Quiz

- 1) <https://b.socrative.com/login/student/> adresine gidin
- 2) Room Name **BULUTLUOZ** yazın
- 3) Isminizi yazın
- 4) **Done** butonuna basin

Sure : 13 Dakika



## For Loop

Belirli bir koşul sağlandığı sürece tekrarlanması gereken işler için kullanılan kod bloklarına LOOP(Dongu) denir. Tekrar sayısı belirli olan durumlarda for loop kullanılması tercih edilir.



```
for ( int i=4; i>1; i- - ) {  
    System.out.println( i );  
}
```



## For Loop

```
Start → STOP ↑ ↓  
for(Starting Value ; Ending Condition ; Increasing or Decreasing the Value) {  
}
```

- Eger **Ending Condition** hep **true** verirse loop sonsuz donguye girer
- Eger Loop'ta **Ending Condition** hic true olmazsa loop body hic devreye girmez
- Artis degeri 1 olmak zorunda degil, farkli da olabilir ( $i+=2$  vb..)



## For Loop

**Soru 1)** Ekrana 10 kez “Java guzeldir” yazdirin

**Soru 2 )** 10 ile 30 arasindaki(10 ve 30 dahil) sayiları aralarında virgül olarak aynı satırda yazdirin

**Soru 3)** 100'den baslayarak 50'ye(dahil) kadar olan sayıları aralarında virgül olarak aynı satırda yazdirin

**Soru 4)** Kullanıcıdan 100'den küçük bir tamsayı isteyin. 1'den baslayarak girilen sayıya kadar 3'un katı olan sayıları yazdirin.

**Soru 5)** Kullanıcıdan 100'den küçük bir tamsayı isteyin. 1'den baslayarak girilen sayıya kadar 3'un veya 5'in katı olan sayıları yazdirin.

**Soru 6) Interview Question** Kullanıcıdan 100'den küçük bir tamsayı isteyin. 1'den baslayarak girilen sayıya kadar tüm sayıları yazdirin. Ancak;

- Sayı 3'un katı ise sayı yerine “Java” yazdirin.
- Sayı 5'in katı ise sayı yerine “Guzeldir” yazdirin.
- Sayı hem 3'un hem 5'in katı ise sayı yerine “Java Guzeldir” yazdirin.



## For Loop

**Soru 7 ) Interview Question** Kullanicidan bir String isteyin ve Stringi tersten yazdirin.

**Soru 8 ) Interview Question** Kullanicidan bir String isteyin ve Stringi tersine ceviren bir method yazin.

**Soru 9 ) Interview Question** Kullanicidan bir String isteyin. Kullanicinin girdigi String'in palindrome olup olmadigini kontrol eden bir program yazin.

**Soru 10 )** Kullanicidan iki sayı isteyin. Girilen sayılar ve aralarındaki tüm tamsayıları toplayıp, sonucu yazdırın bir program yazınız

**Soru 11 ) Interview Question** Kullanicidan 10'dan küçük bir tamsayı isteyin ve girilen sayının faktoryel'ini bulun. ( $5!=5*4*3*2*1$ )



**BATCH** : Batch 59-60

**LESSON** : Java 17

**DATE** : 09.03.2022

**SUBJECT** : Nested For Loop

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



## Nested For Loop

Bazen tek bir loop ile istedigimiz sonuclarla ulasamayiz.

Ozellikle iki boyutlu sekiller cizdirmek veya carpim tablosu gibi sayı ikilileri olusturmak icin nested loop kullanmamiz gereklidir.

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

```
for (int i = 1; i <= 4; i++) {  
  
    for (int j = 1; j <= 4; j++) {  
  
        System.out.print("(" + i + "," + j + ") ");  
    }  
  
    System.out.println();  
}
```

(1,1)	(1,2)	(1,3)	(1,4)
(2,1)	(2,2)	(2,3)	(2,4)
(3,1)	(3,2)	(3,3)	(3,4)
(4,1)	(4,2)	(4,3)	(4,4)



## Nested For Loop

**Soru 12 )** Kullanicidan pozitif bir rakam girmesini isteyin ve girilen rakama gore asagidaki sekli cizdirin

\*

\* \*

\* \* \*

\* \* \* \*

**Soru 13 )** Kullanicidan pozitif bir rakam girmesini isteyin ve girilen rakama gore carpim tablosu olusturun. Ornek,kullanici 3 girerse,

1 2 3

2 4 6

3 6 9



# While Loop

```
while(condition) {           Code          }
```

true

After running the code check the condition again

```
while(condition) {           Code          }
```

false

Break the loop and proceed to the next line

```
int i = 0;  
while (i < 5) {  
    System.out.println(i);  
    i++;  
}
```



**BATCH** : Batch 59-60

**LESSON** : Java 19

**DATE** : 12.03.2022

**SUBJECT** : While Loop

Do While Loop  
Scope

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



## While Loop

**Soru 1 )** While loop kullanarak 3 den 13 e kadar tum tek tamsayıları ekranaya yazdırınız.

**Soru 2 )** For loop ve while Loop kullanarak 3 basamaklı sayılarından 15, 20 ve 90'na tam bolunebilen sayıları yazdırın.

**Soru 3 )** Kullanıcıdan başlangıç ve bitiş değerlerini alın. Başlangıç değeri ve bitiş değeri dahil aralıklarındaki tüm çift sayıları while loop kullanarak ekranaya yazdırınız.

**Soru 4 )** Kullanıcıdan başlangıç ve bitiş harflerini alın ve başlangıç harfinden başlayıp bitiş harfinde biten tüm harfleri büyük harf olarak ekranaya yazdırın. Kullanıcının hata yapmadığını farz edin.



## While Loop

**Soru 5)** Kullanicidan bir rakam alin ve bu rakam icin carpim tablosunu ekrana yazdirin. Kullanicinin hata yapmadigini farz edin.

Ornegin kullanici 3 girerse;

$$3 \times 1 = 3 \quad 3 \times 2 = 6 \quad 3 \times 3 = 9 \quad 3 \times 4 = 12 \quad 3 \times 5 = 15 \quad 3 \times 6 = 18 \quad 3 \times 7 = 21 \quad 3 \times 8 = 24 \quad 3 \times 9 = 27 \quad 3 \times 10 = 30$$

**Soru 6 )** Kullanicidan bir sayi alin ve bu sayiyi tam bolen sayilari ve toplam kac tane olduklarini ekranda yazdirin

**Soru 7 )** Kullanicidan bir sayi alin ve bu sayinin rakamlari toplamini yazdirin



# Do While Loop

**do { Code } while(condition)**

true

Run the code then check the condition

**do { Code } while(condition)**

false

Run the code then check the condition

Break the loop and proceed to the next line

```
public static void main(String[] args) {  
    int i = 0;  
  
    do {  
        System.out.println(i);  
        i++;  
    }  
    while (i<5);  
}
```



# Do While Loop Vs While Loop

```
public static void main(String[] args) {  
    int i = 10;  
  
    do {  
        System.out.println(i);  
        i++;  
    }  
    while (i<5);  
}
```

```
public static void main(String[] args) {  
    int i = 10;  
  
    while (i<5){  
        System.out.println(i);  
        i++;  
    }  
}
```

**Fark :** While Loop, dongunun başlangıcında kosulu kontrol eder ve kosul sağlanırsa body icindeki kodları çalıştırır.  
Do-while loop'ta ise , kosul body içerisindeki kodlar 1 kere çalıştırıldıktan sonra kontrol edilir.

**Sonuc :** Bir while loop'daki kosul yanlışsa, loop hiç çalışmaz 'do-wile' loop'ta ise , kosul yanlışsa kodlar 1 kere çalışır



## Do While Loop

**Soru 1)** 9 den 190 e kadar 7 nin kati olan tum tamsayilari ekrana yazdiriniz.

**Soru 2 )** 'm' harfinden baslayarak 'c' harfine kadar tum harfleri yazdirin.

**Soru 3)** Kullanicidan toplamak üzere pozitif sayilar isteyin, islemi bitirmek icin 0'a basmasini soleyin.

Kullanici 0'a bastiginda toplam kac pozitif sayi girdigini ve girdigi pozitif sayilarin toplaminin kac oldugunu yazdirin.

**Soru 4)** Kullanicidan toplamak üzere pozitif sayilar isteyin, islemi bitirmek icin 0'a basmasini soleyin.

Kullanici yanlislikla negative sayi girerse o sayiyi dikkate almayin ve "Negatif sayi giremezsiniz" yazdirip basa donun

Kullanici 0'a bastiginda toplam kac pozitif sayi girdigini, yanlislikla kac negative sayi girdigini ve girdigi pozitif sayilarin toplaminin kac oldugunu yazdirin.



## Do While Loop

**Soru 5 )** Kullanicidan bir sifre girmesini isteyin. Girilen sifreyi asagidaki sartlara gore kontrol edin ve sifredeki hatalari yazdirin.

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

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

**Soru 6 )** Kullanicidan toplamak icin sayi isteyin ve toplam 500'e ulasincaya kadar devam istemeyi ettirin. Toplam 500'e ulastiginda veya gectiginde toplami ve kac sayi girildigini yazdirin



# Scope (Instance, Class ve Local Variables )





# Object Nasıl Kullanılır ?



Ogretnen



Personel



Ogrenci

09:00	TÜRKÇE-1
09:30	MATEMATİK-1
10:00	TÜRKÇE-2
10:30	MATEMATİK-2
11:00	TÜRKÇE-3
11:30	MATEMATİK-3
12:00	TÜRKÇE-4
12:30	MATEMATİK-4
13:00	İYEP TÜRKÇE

Dersler



Notlar



**BATCH** : Batch 59-60  
**LESSON** : Java 20  
**DATE** : 14.03.2022  
**SUBJECT** : Scope  
Arrays

- techproeducation
- techproeducation
- techproeducation
- techproeducation
- techproedu



## Scope (Instance, Class ve Local Variables )

- Bir Class icerisinde olusturulan variable'lar icin Scope, o variable'a nereden, nasil ulasabilecegini ve nerede gecerli oldugunu ifade eder.
- Scope'a uymayan bir kullanimda Java Compile Time Error verir.
- Java'da olusturulan variable'lar icin 4 Scope mevcuttur
  - 1) Instance (Object) Variables // ogretmenin adi gibi, ogrencinin notu gibi
  - 2) Static (Class) Variables // okul adi, adresi gibi
  - 3) Local (Method) Variables
  - 4) Loop Variables



# Scope (Instance, Class ve Local Variables )

## Instance (Object) Variable

Class'in icinde ancak main method'un  
disinda olmalidir

Static olmamalidir

Olusturulmasi yeterlidir, deger  
atanmasi şart degildir.

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

## Default Value

Eger instance bir variable olusturur ama deger atamazsaniz, Java otomatik olarak  
default degerleri assign eder. (String icin null, sayisal data turleri 0, boolean false)



# Scope (Instance, Class ve Local Variables )

## Instance (Object) Variable

class icerisinde veya baska class'larda direkt kullanilamaz, kullanmak istedigimizde MUTLAKA object olusturmali ve object uzerinden ulasilmalidir.

```
public class Example {  
  
    int sayi;  
    char ilkHarf;  
    String isim;  
    boolean ogrenciMi;  
  
    public static void main(String[] args) {  
  
        Example ex1=new Example();  
        System.out.println(ex1.sayi);  
        System.out.println(ex1.ilkHarf);  
        System.out.println(ex1.isim);  
        System.out.println(ex1.ogrenciMi);  
    }  
}
```

Outputs

0  
null  
false

## Ornek :

Bir okul uygulamasi yaptigimizi dusundugumuzde, ogretmenIsmi, ogrenciIsmi, matematiNotu gibi degiskenler bir kisi ile ilişkilendirilmedikce anlamlı olmaz



# Scope (Instance, Class ve Local Variables )

## Class (static) Variable

Class'in icinde ancak main method'un disinda olmalidir.

Static olmalıdır

Olusturulmasi yeterlidir, deger atanmasi şart değildir.

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



# Scope (Instance, Class ve Local Variables )

**Class (static) Variable**, class içerisinde direkt kullanılır, başka class'larda kullanmak istedigimizde object olusturmaya ihtiyac duymadan classIsmi.variableIsmi ile variable'a ulaşabilir ve kalıcı olarak değiştirebiliriz.

```
public class Example {  
  
    static int okulId;  
    static String okulIsmi;  
    static boolean acikMi;  
  
    public static void main(String[] args) {  
  
        System.out.println(okulId);  
        System.out.println(okulIsmi);  
        System.out.println(acikMi);  
    }  
}
```

Outputs

0  
null  
false

**Ornek :** Bir okul uygulaması yaptığımızı düşünün okulIsmi, okulId, acikMi gibi değişkenler bir kişiyi değil okulla ilgili herkesi ilgilendirir ve bir kişi okul ismini veya okul telefon numarasını değiştirdiğinde okulla ilgili herkes için okul ismi değişir.



# Scope (Instance, Class ve Local Variables )

## Instance Vs Class Variables

**Instance (Object) Variable**, class içerisinde veya başka class'larda direk kullanılamaz, kullanmak istedigimizde MUTLAKA object olusturmali ve object uzerinden ulasılabilir.

**Class (static) Variable**, class içerisinde direkt kullanılabılır, başka class'larda kullanmak istedigimizde object olusturmaya ihtiyac duymadan classIsmi.variableIsmi ile variable'a ulaşabilir ve kalıcı olarak degistirebiliriz.

Static variable'lar herkes için ortaktır (okul ismi gibi) , instance variable'lar ise objeye bağlıdır (matematikNotu, ogrenciIsmi gibi)

Static variable yetkisi olan herkes tarafından degistirilebilir ve bu değişim her obje için geçerlidir. Instance variable da yetkisi olan herkes tarafından degistirilebilir ancak yapılan değişiklik sadece o obje ile ilgilidir, geneli kapsamaz.



# Scope (Instance, Class ve Local Variables )

## Local Variable

- Herhangi bir method içerisinde oluşturulan variable'lardır (main method dahil).
- Sadece o method içerisinde geçerlidir
- Baska methodlarda da kullanılacak variable'lari, local oluşturmak yerine **class level**'da oluşturmak gereklidir.
- Class level'da oluşturulacak variable, main method'da kullanılacaksa static olarak oluşturulmalıdır. Bu durumda bu variable kullanacak, diger method'lar da static olmalıdır.

```
public class Example {  
  
    public static void main(String[] args) {  
  
        int sayi;  
  
    }  
  
    public void add() {  
  
        String isim;  
    }  
}
```



# Scope (Instance, Class ve Local Variables )

## Local Variable

- Java local variable'lara default değer atamaz.
- Sadece olusturdugunuzda Java şikayet etmez. ( variable olusturuldu method icerisinde değer atanacak diye bekler.)
- Olusturulan local variable'lara değer atamadan kullanmaya calisirsaniz Java şikayet eder(CTE)

```
5 public class Example {  
6  
7  
8     public static void main(String[] args) {  
9  
10        int sayi;  
11        sayi++;  
12    }  
13  
14  
15  
16    public void add() {  
17        String isim;  
18        System.out.println(isim);  
19    }  
20  
21}
```



# Scope (Instance, Class ve Local Variables )

## Loop Variables

- Bir loop icinde olusturulan variable'lар sadece o loop icerisinde gecerlidir.
- Loop icerisinde olusturulan variable'lara loop disindan ulasilamaz ve loop disinda kullanilamaz.
- Loop icerisinde olusturulan local variable'lari disarida kullanmaya calisirsaniz Java sikayet eder(CTE)

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



# Scope (Instance, Class ve Local Variables )

```
public class MyClass{  
    int num1;  
    String name = "Ali";  
    public static void main(String args){  
        add();  
        product (5);  
    }  
    public static add(){  
        num1++;  
        int num2 = 6;  
        char letter;  
        System.out.println("Do addition ");  
    }  
    public product(int num3){  
        name = "Veli";  
        num2++;  
        System.out.println(num3 * num3);  
    } }
```

- 1) Hangileri instance variable'dir ?
- 2) Hangileri local variable'dir?
- 3) num1 icin default value nedir ?
- 4) Java hangi satirlarin altini kirmizi cizer?
- 5) Kac satir compile time error verir?



# Scope (Ozet)

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

Temel olarak 4 Scope'dan bahsedebiliriz

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

- 1- static olarak tanimlanan variable'lara tum method'lardan ulasilabilir
- 2- static olarak tanimlanmayan (instance) variable'lara sadece static olmayan method'lardan ulasilabilir
- 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    }}}
```



**BATCH** : Batch 59-60  
**LESSON** : Java 21  
**DATE** : 15.03.2022  
**SUBJECT** : Arrays

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



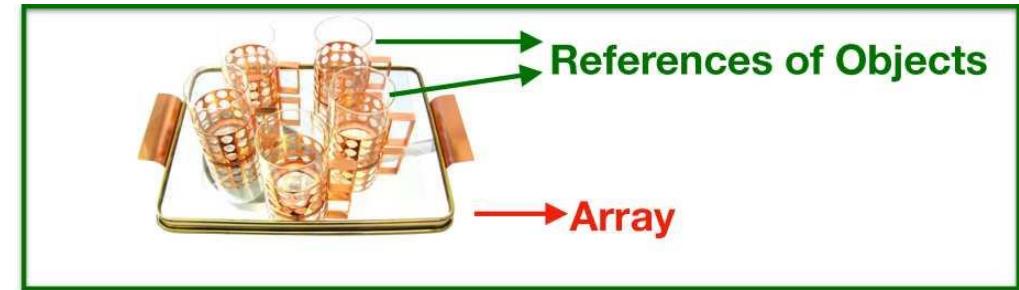
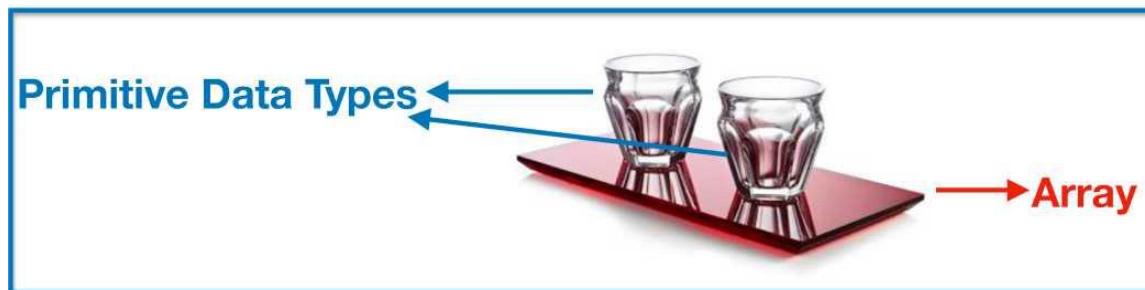
## Önceki Dersten Aklımızda Kalanlar

- 1- Scope : bir Class'daki variable'larin nerelerde kullanabilecegini belirleyen kurallardir.
- 2- temelde 4 scope'dan bahsedebiliriz
  - Class level scope'lar
    - Instance(object) variable : static olmayan class level variable'lardir. Static olmadiklari icin static method'larda direkt kullanilamazlar. Static method'larda kullanmak isterseniz obje olusturmaniz gereklidir. Her objenin degeri kendisine bagli oldugundan, instance variable'larin degeri sorulursa objeyi takip etmeliyiz. Statik olmayan method ise instance variable'lari direkt kullanabilirler
    - Static(class) variable: class icerisindeki static olan veya olmayan tum method'lardan kullanilabilirler. Class'a bagli oldugu icin static variable'in son degeri soruldugunda, main method'dan baslayip deger istenen satira kadar tum kodu takip etmeliyiz.
  - Local scope'lar
    - Local(method) variable : Method icerisinde olusturulan variable'lar sadece o method'da gecerlidir. Diger method'lardan erisilmez ve kullanilamaz. Local variable'lar icin default deger olmaz. Atama yapmadigimiz variable'l kullanmamiza Java izin vermez
    - Loop variable : Loop icerisinde olusturulan variable'lar loop disinda kullanilamaz. Bundan dolayi loop icerisinde kullanacagimiz variable'lari loop'tan once olustururuz.



# Arrays

Arrays birden fazla variable depolamak için kullanılabilen object (non-primitive data)'lerdir.



- 1) Arrays'de sadece primitive datalar veya non-primitive datalara ait referans'lar depolanabilir
- 2) Arrays içindeki tüm variable'lar aynı data type'inde olmalıdır.

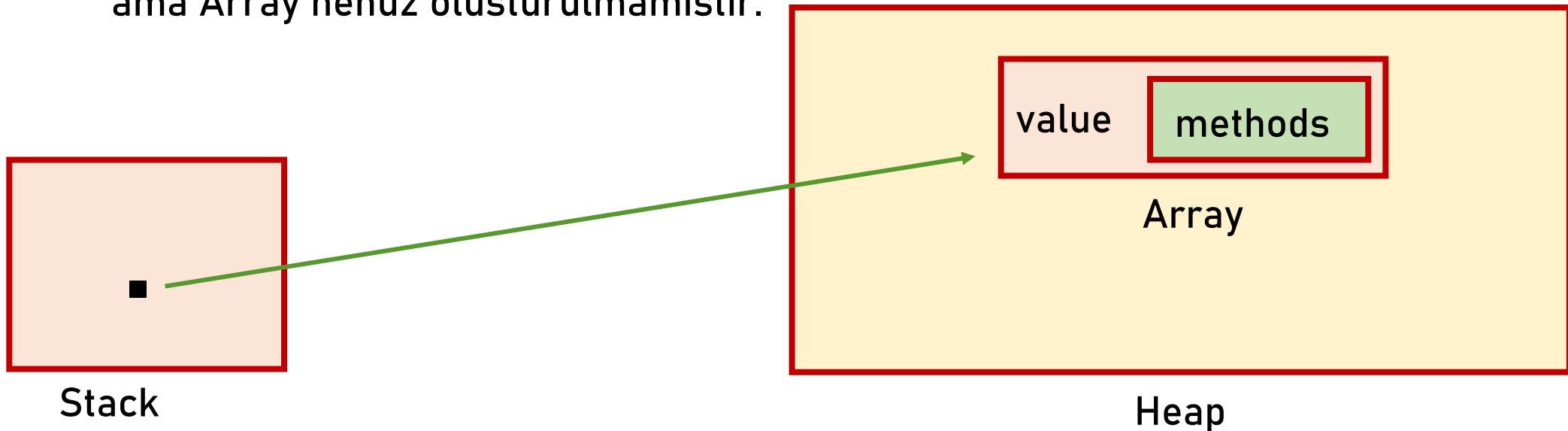


## Arrays

5) Array'ler object (non-primitive) 'tir. Bu yuzden

- Heap Memory'de depolanirlar.
- Value ile birlikte method'lara da sahiptirler
- runtime'da olusturulurlar.

Bir Array declare edildiginde stack memory'de referans olusturulur  
ama Array henuz olusturulmamistir.





# Arrays

## 6) Bir Array nasıl declare edilir?

Array declare etmek için iki yol vardır :

- int myArray[ ] ; // Bu daha çok kullanılır
- int [ ] myArray;

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

## 7) Bir Array nasıl oluşturulur

```
int myArray[ ] = new int[6];
```

- Yukarıdaki kod **length'i** 6 olan bir array oluşturur.
- Biz array'e eleman eklemezsek Java elemanlar için data type'ına uygun default değerler atar.
- Eğer yukarıdaki array'i yazdırırsanız ekranda {0, 0, 0, 0, 0, 0} gorursunuz

**NOT :** Array oluştururken **length'i** yazmazsanız compile time error alırsınız.



# Arrays

## 8) Array'e degerler nasil atanir

```
int myArray[ ] = new int[3];
```

```
myArray[0] = 9;  
myArray[1] = 10;  
myArray[2] = 11;
```

Once olusturup, sonra istedigimiz indexler icin deger atayabiliriz

Veya

```
int myArray[ ] = {9, 10, 11};
```

Olusturma ve tum indexler icin deger atamayı tek satırda yaparız.

**Soru 1:** Elemanlari “Ali” , “Veli”, “Ayse” ve “Fatma” olan bir array olusturun ve bu array'i yazdirin.



## Arrays

9) Array'in elemanlarına nasıl ulaşılır ve nasıl update edilir ?

```
int myArray[ ] = {9, 10, 11};
```

Array elemanlarına index'ler kullanılarak ulaşılır.

myArray[0] ==> 9,

myArray[1] ==> 10,

myArray[2] ==> 11,

NOT 1 : "n" array'in length'i olmak üzere myArray[n-1] son elemani gösterir

NOT 2 : Bir Array'de olmayan index'i kullanmak isterseniz  
"ArraysIndexOutOfBoundsException" alırsınız.

Soru 2: Soru 1'deki elemanlardan "Ali" yerine "Can", "Ayse" yerine "Gul" atayın.



# Arrays

10) Bir Array'in uzunlugu nasil bulunur?

```
int myArray[] = {9, 10, 11};
```

```
int size = myArray.length;
```

NOT : String ve Array icin length method'larinda dikkatli olmak gereklidir.

Strings ==> `length()`

Arrays ==> `length`



# Arrays

11) Bir Array'in tum eleamanlari nasil yazdirilir?

```
int myArray[ ] = {9, 10, 11};
```

```
for(int i=0; i<size; i++) {  
    System.out.println(myArray[i]);  
}
```

```
System.out.println(Arrays.toString(myArray));
```

**Soru 1:** Verilen 3 elemanli bir array'in tum elemanlarini bir soldaki konuma tasiyacak bir program yazin. Ornek; array [1,2, 3] ise output [2, 3, 1] olacak.

**Soru 2:** Verilen bir array'in tum elemanlarini toplayan bir program yazalim.



**BATCH** : Batch 59-60  
**LESSON** : Java 22  
**DATE** : 16.03.2022  
**SUBJECT** : Arrays

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



## Önceki Dersten Aklımızda Kalanlar

- 1- Javada birden fazla elementi store edebilecegimiz data turlerindendir.
- 2- Array'e primitive data turunden degerler veya non-primitive data turlerindeki objelerin referanslarini koyabiliriz.
- 3- Arrayi deklare etmek icin icine koyacagimiz elementlerin data turu, arrayin ismi ve [ ] yazmamiz gerekir. Deklare ederken kullandigimiz data turu arrayin icine koyacagimiz datalari sinirlandirir, belirttigimiz data turu disinda baska data turunden element koyamayiz
- 4- Atama yapmak icin iki yontem vardir
  - esitligin sagina suslu parantez icerisinde uygun elementleri yazabiliriz. Array'e bu sekilde atama yaptigimizda Java yazdigimiz elemnt sayisini array'in uzunlugu olarak kabul eder. Int arr[ ] = {1,4,6,7};
  - Esitligin sagina new keyword, olusturacagimiz arrayin data turu ve boyutunu belirtirsek java, icinde default degerlerin oldugu bir array olusturur. new int[5];
- 5- Array'in uzunlugu degistirilemez, dolayisiyla element ekleme veya cikarmaya uygun degildir.
- 6- Array non-primitive oldugu icin direk yazdirmak istedigimizde referans bilgisini yazdirir, array'in tum elementlerini yazdirmak icin Arrays class'indan toString() kullaniriz



# Arrays

12) Bir Array'in tum elemanlari nasil siralanir?

```
int myArray[ ] = {9, 15, 11};
```

```
Arrays. sort (myArray);
```

Siralama buyukten kucuge nasil yapilir ?

- Once sort methodu kullanilir
- Sonra siralamayı ters cevirmek icin loop kullanilir



## Arrays

13) Bir Array'de istenen bir elemanın varlığı nasıl kontrol edilir?

`binarySearch()` method'u belli bir elemanın bir array'de olup olmadığını kontrol etmek için kullanılır.

Ancak, `binarySearch()` methodunu kullanmadan önce mutlaka `sort()` methodu kullanılmalıdır.

```
int[ ] numbers = { 2, 8, 6, 4 };
Arrays.sort(numbers);
System.out.println( Arrays. binarySearch(numbers, 2)); //===== 0
System.out.println( Arrays. binarySearch(numbers, 4)); //===== 1
```

Eğer bir eleman array'de yoksa output negatif olur.

- 1) 0 eleman var olsaydı sıra numarası kaç olurdu, buluruz.
- 2) Bulduğumuz sıra numarasının negatif hali, `binarySearch()`'un outputu olur.

```
System.out.println( Arrays. binarySearch(numbers, 1)); // ===== -1
System.out.println( Arrays. binarySearch(numbers, 3)); // ===== -2
System.out.println( Arrays. binarySearch(numbers, 9)); // ===== -5
```



# Arrays

Output nedir ?

```
int[ ] numbers = { 2, 1, 7, 6 };
Arrays.sort(numbers);
System.out.println(Arrays.binarySearch(numbers, 2));
System.out.println(Arrays.binarySearch(numbers, 7));
System.out.println(Arrays.binarySearch(numbers, 3));
System.out.println(Arrays.binarySearch(numbers, 9));
```

→ 1  
→ 3  
→ -3  
→ -5

```
String[ ] letters = { "A", "N", "F", "C" };
Arrays.sort(letters);
System.out.println(Arrays.binarySearch(letters, "A"));
System.out.println(Arrays.binarySearch(letters, "C"));
System.out.println(Arrays.binarySearch(letters, "E"));
System.out.println(Arrays.binarySearch(letters, "G"));
```

→ 0  
→ 1  
→ -3  
→ -4



## Arrays

14) İki array'in eşit olup olmadığı nasıl kontrol edilir?

`equals()` method'u değerleri ve indexleri birlikte kontrol edip, boolean bir değer return eder.

```
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

### 16) Bir String nasıl array'e cevrilir ?

`split()` method'u String'e ait bir method'dur ve belirledigimiz ayırac'a göre String'i parçalara ayırip bir Array'e çevirir.

```
String str = "Java ogrenmek, IT alaninda yer edinmek demektir.";  
  
String arr1[] = str.split(",");  
System.out.println(Arrays.toString(arr1));  
→ [Java ogrenmek, IT alaninda yer edinmek demektir.]  
  
String arr2[] = str.split(" ");  
System.out.println(Arrays.toString(arr2));  
→ [Java, ogrenmek,, IT, alaninda, yer, edinmek, demektir.]  
  
String arr3[] = str.split("");  
System.out.println(Arrays.toString(arr3));  
→ [J, a, v, a, , o, g, r, e, n, m, e, k, , , , I, T, , a, l, a, n, i, n, d,  
a, , y, e, r, , e, d, i, n, m, e, k, , d, e, m, e, k, t, i, r, .]
```



# Arrays

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.



**BATCH** : Batch 59-60  
**LESSON** : Java 23  
**DATE** : 17.03.2022  
**SUBJECT** : Arrays

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



## Önceki Dersten Aklımızda Kalanlar

1- `Arrays.sort(arr);` arr array'ini natural order'a gore siralar. Array'de Descending siralama yoktur ama istersek biz once sort yapip sonra bir for-loop yardimiyla siralamayi degistirebiliriz

2- `Arrays.binarySearch(arr, istenenElement );` binarySearch method'u bir array'de istedigimiz elementin var olup olmadigini bize doner. Ancak binarySearch'un saglikli calismasi icin once sort method'u calistirilmalidir. Eger sort calistirilmazsa sonuc ongoreulemez.

- Eger aradigimiz element array'de varsa binarySearch bize index'ini dondurur
- Eger aradigimiz element array'de yoksa, olmadigini belirtmek icin - işaretini verdikten sonra, eger o element olsaydi hangi sirada olacagini dondurur(index degil sira)

3- `str.split("istenenString");` verilen str'i istenenString'ı kullanarak parcalara ayirir. Bunun icin istenenString'ı bulur, onu silip geriye kalanları bir array'in elementleri olarak dondurur. Bu method bir array dondurdugu icin sonucu mutlaka bir array'e assign etmeliyiz.

Eger split method'unda "" kullanırsak, verilen String'ı karakterlere bolmuş oluruz

4- `arr1.equals(arr2);` verilen arr1 ve arr2'nin tum elementlerini gozden gecirir,

- eger tum elementler ve indexleri ayni ise true dondurur
- elementlerde veya siralamada fark varsa false dondurur



# Multi Dimensional Arrays

Eger bir Array ic ice Array'lerden olusuyorsa buna Multi Dimensional Array denir

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

*child arrays*

[ [ {1,2},{3,4},{5,6} ] ] [ [ {7,8},{9,1},{2,3} ] ]

*parent array*                           *parent array*

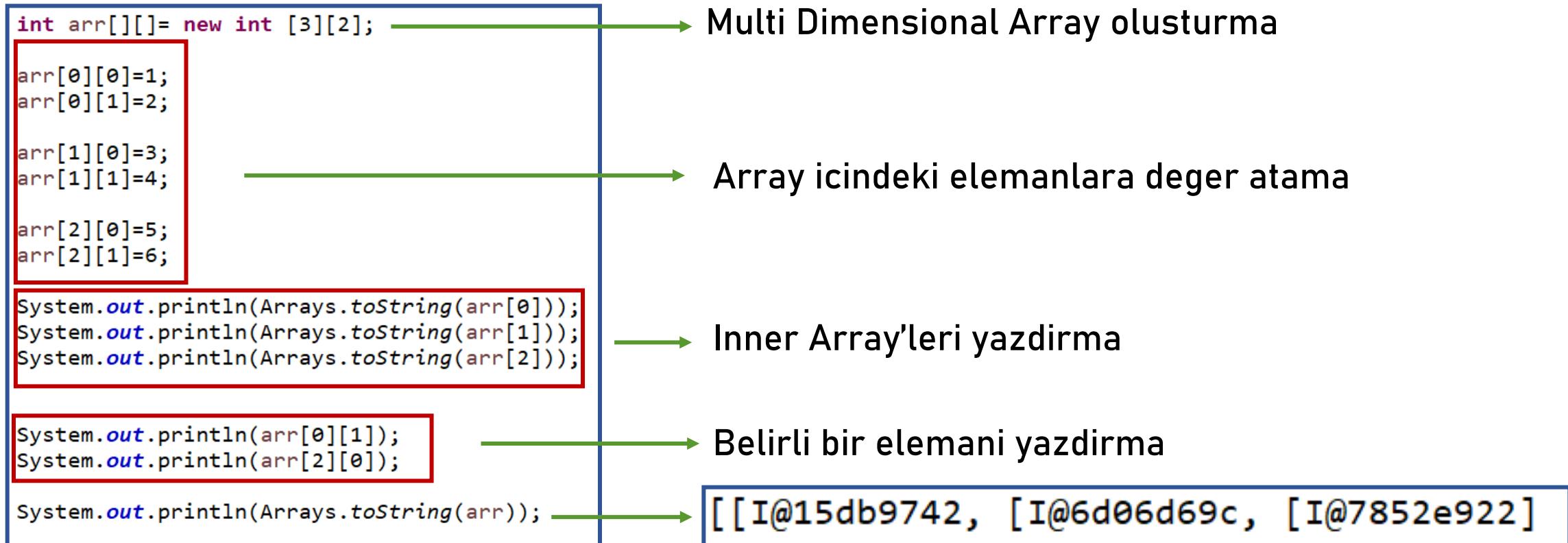
*grand parent array*



# Multi Dimensional Arrays

Array'i tanimlarken (declaration), her bir kat icin bir [ ] kullanilir.

```
Int arr[ ][ ] = { {1,2} , {3,4}, {5,6}};
```





# Multi Dimensional Arrays

Multi Dimensional Array'in tüm elemanları nasıl yazdırılır ?

```
public static void main(String[] args) {  
  
    int arr[][] = { {1,2} , {3,4}, {5,6}};  
  
    for (int i = 0; i < arr.length; i++) {  
  
        for (int j = 0; j < arr[i].length; j++) {  
  
            System.out.print(arr[i][j] + " ");  
        }  
    }  
  
    System.out.println(Arrays.deepToString(arr));  
}
```

→ Nested For Loop kullanabilir

→ Arrays Class'ından method kullanabilir



# Multi Dimensional Arrays

- Soru 1)** Asagidaki multi dimensional array'in tum elemanlarinin carpimini ekrana yazdiran bir method yaziniz. { {1,2,3}, {4,5,6} }
- Soru 2)** Asagidaki multi dimensional array'in ic array'lerindeki son elemanlarin carpimini ekrana yazdiran bir program yaziniz { {1,2,3}, {4,5}, {6} }
- Soru 3)** Asagidaki multi dimensional array'lerin ic array'lerinde aynı index'e sahip elemanların toplamini ekrana yazdiran bir program yaziniz. (Zor soru) arr1 = { {1,2}, {3,4,5}, {6} } ve arr2 = { {7,8,9}, {10,11}, {12} }
- Soru 4)** Asagidaki multi dimensional array'in ic array'lerindeki tum elemanların toplamini birer birer bulan ve herbir sonucu yeni bir array'in elemani yapan ve yeni array'i ekrana yazdiran bir program yaziniz { {1,2,3}, {4,5}, {6,7} }
- Ornek; { {1,2,3}, {4,5}, {6,7} } ==> 1+2+3=6 4+5=9 6+7=13 ==> output: {6, 9, 13}
- Soru 5)** Kullanicidan bir cumle isteyin ve cumledeki kelime sayisini yazdirin
- Soru 6)** Verilen bir Array'den isten degere esit olan elamanlari kalsdirip, kalanları yeni bir Array olarak yazdiran bir method yaziniz



**BATCH** : Batch 59-60  
**LESSON** : Java 24  
**DATE** : 17.03.2022  
**SUBJECT** : ArrayList

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



# ArrayList

## ArrayList nedir?

ArrayList length'i esnek olan bir Array'dir

## ArrayList'e neden ihtiyac duyuyoruz?

- Biz array olustururken length'in en basta belirlemek zorundayız ve daha sonra length'ini degistirememiz.Bu durum bizim esnek calismamiza engel olur.
- Bir array'in uzunlugunu degistirmek istedigimizde yeni bir array olusturmamız gerekir, ArrayList de gerekmeyez.
- Bir array'den bir eleman silmek istedigimizde yeni bir array olusturmamız gerekir, ArrayList de gerekmeyez.



# ArrayList

## ArrayList olusturma

```
ArrayList<String> list1 = new ArrayList<String>();
```

```
ArrayList<String> list2 = new ArrayList<>();
```

```
List<String> list3 = new ArrayList<>(); En çok bu kullanılır
```

```
ArrayList<String> list4 = new List<>();
```

Compile Time Error verir, eşitliğin sağ tarafında ArrayList kullanmak zorundayız

## ArrayList'i nasıl yazdırırız?

ArrayList'i ekrana yazdirmak çok kolaydır.

```
System.out.println(list3);
```



# ArrayList Method'lari

## 1) add()

add() method ArrayList'e eleman eklemek icin kullanilir

Ornek :

```
List<String> hayvan = new ArrayList<>();
```

A) add( ) method'u index olmadan calisabilir

```
hayvan.add("kedi"); // [kedi]
```

```
hayvan.add("yilan"); // [kedi, yilan]
```

B) add( ) method'u index ile de calisabilir

```
hayvan.add(1, "kartal"); // [kedi, kartal, yilan]
```

```
hayvan.add(0, "sinek"); // [sinek, kedi, kartal, yilan]
```

```
hayvan.add(1, "aslan"); // [sinek, aslan, kedi, kartal, yilan]
```

```
System.out.println(hayvan); // [sinek, aslan, kedi, kartal, yilan]
```



## ArrayList Method'lari

### 2) size()

size() method ArrayList'de kaç eleman olduğunu gösterir.

Ornek :

```
List<String> hayvan = new ArrayList<>();
```

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

```
hayvan.add("kedi"); // [kedi]
```

```
hayvan.add("yilan"); // [kedi, yilan]
```

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

### 3) isEmpty()

isEmpty() method'u ArrayList boş ise true, boş değilse false döndürür



## ArrayList Method'lari

### 4) remove()

remove() method'u ArrayList'den belli bir elemani silmek icin kullanilir.

A) remove(index) kullanarak. Size'dan buyuk index yazilrsa exception verir.  
Index'li remove() methodu ArrayList'de verilen index'deki elemani siler.

```
List<String> hayvan = new ArrayList<>();  
hayvan.add("kedi"); // [kedi]  
hayvan.add("yilan"); // [kedi, yilan]  
hayvan.remove(1); // index'i 1 olan elemani siler  
System.out.println(hayvan); // [kedi]
```

**NOT:** remove(index) method'u silinen elemani dondurur. Yani method'u  
**System.out.println()** icinde kullanırsak silinen elemani ekrana yazdırır.

```
System.out.println(hayvan.remove(1)); //yilan
```



# ArrayList Method'lari

B) `remove("eleman")` index'i degil elemani kullanırsak kullandığımız elemanın ilk kullanıldığı yeri bulur ve siler.

```
List<String> hayvan = new ArrayList<>();  
hayvan.add("kedi"); // [kedi]  
hayvan.add("yilan"); // [kedi, yilan]  
hayvan.add("kedi"); // [kedi, yilan, kedi]  
hayvan.remove("kedi");  
System.out.println(hayvan); // [yilan]
```

**Not:** Index'siz remove() method'u true veya false dondurur.

```
System.out.println(hayvan.remove("kedi")); //true yani kedi eleman olarak vardi ve sildim
```

```
System.out.println(hayvan.remove("tavsan")); // false yani tavsan eleman olarak yoktu  
dolayisiyla silemedim
```



## ArrayList Method'lari

### 5) set()

set() methodu ArrayList'de var olan bir elemani degistirmeye yarar

```
List<String> hayvan = new ArrayList<>();
```

```
hayvan.add("kedi"); // [kedi]
```

```
hayvan.add("yilan"); // [kedi, yilan]
```

```
hayvan.set(1, "tavsan");
```

```
System.out.println(hayvan); // [kedi, tavsan]
```

**NOT:** set() method'u add() method'u yerine kullanilamaz .  
Olmayan bir index ile set() kullanilirsa exception verir.

```
hayvan.set(2, "aslan"); // IndexOutOfBoundsException
```



## ArrayList Method'lari

### 6) get(index)

get( ) methodu ArrayList'deki istenen indexdeki elemani dondurur.

```
List<String> hayvan = new ArrayList<>();
```

```
hayvan.add("kedi"); // [kedi]
```

```
hayvan.add("yilan"); // [kedi, yilan]
```

```
System.out.println(hayvan.get(0)); // kedi
```

```
System.out.println(hayvan.get(1)); // yilan
```



**BATCH** : Batch 59-60  
**LESSON** : Java 25  
**DATE** : 19.03.2022  
**SUBJECT** : ArrayList

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



## ArrayList Method'lari

### 7) contains()

contains() methodu ArrayList'de bir elemanın var olup olmadığını kontrol eder. Eleman varsa true, yoksa false return eder.

```
List<String> hayvan = new ArrayList<>();  
hayvan.add("kedi"); // [kedi]  
hayvan.add("yilan"); // [kedi, yilan]
```

```
System.out.println(hayvan.contains("kedi")); // true
```

```
System.out.println(hayvan.contains("tavsan")); // false
```



## ArrayList Method'lari

8) **Collections.sort()** : sort() methodu ArrayList'deki elemanlari kucukten buyuge veya alfabetik siraya gore dizer.

```
List<String> hayvan = new ArrayList<>();
hayvan.add("yilan"); // [yilan]
hayvan.add("kedi"); // [yilan, kedi]
hayvan.add("tavsan"); // [yilan, kedi, tavsan]
```

```
System.out.println(hayvan); // [yilan, kedi, tavsan]
```

```
Collections.sort(hayvan);
```

```
System.out.println(hayvan); // [kedi, tavsan, yilan]
```



## ArrayList Method'lari

### 9) equals()

equals( ) methodu iki listteki ayni indexteki elemanların ayni olup olmadigini kontrol eder. Ayni indexteki tum elemanlar ayni ise true return eder, farkli ise false return eder

```
List<String> first = new ArrayList<>();  
List<String> second = new ArrayList<>();  
System.out.println(first.equals(second)); // true
```

```
first.add("a"); // [a]  
System.out.println(first.equals(second)); // false
```

```
second.add("a"); // [a]  
System.out.println(first.equals(second)); // true
```

```
first.add("b"); // [a,b]  
second.add(0,"b"); // [b,a]  
System.out.println(first.equals(second)); // false
```



## ArrayList Method'lari

### 10) clear()

clear() methodu ArrayList'teki tum elemanlari siler.  
Return type'i void'dir, hic bir sey donmez

```
List<String> hayvan = new ArrayList<>();  
hayvan.add("yilan"); // [yilan]  
hayvan.add("kedi"); // [yilan, kedi]  
  
System.out.println(hayvan.isEmpty()); // false  
System.out.println(hayvan.size()); // 2  
  
hayvan.clear();  
System.out.println(hayvan.isEmpty()); // true  
System.out.println(hayvan.size()); // 0
```



## ArrayList Sorular

- 1) Elemanlari A, C, E, ve F olan bir String ArrayList olusturup ekrana yazdiriniz.
- 2) indexsiz **add()** methodunu kullanarak, B'yi ekleyiniz.  
index'li **add()** methodunu kullanarak, L'yi 1 numarali index'e ekleyiniz.  
ArrayList'i ekrana yazdiriniz, list goyle olmali; A, L, C, E, F, B.
- 3) **set()** methodu kullanarak, E'yi D yapiniz.  
ArrayList'i ekrana yazdiriniz, list goyle olmali; A, L, C, D, F, B.
- 4) **remove()** methodu kullanarak, F'yi siliniz.  
ArrayList'i ekrana yazdiriniz, list goyle olmali; A, L, C, D, B.
- 5) **sort()** methodu kullanarak, elemanlari alfabetik siraya diziniz.  
ArrayList'i ekrana yazdiriniz, list goyle olmali; A, B, C, D, L.
- 6) **contains()** methodu kullanarak, L'nin list'de var oldugunu ve M'nin list'de var  
olmadigini dogrulayiniz.
- 7) **size()** methodu kullanarak, list'in kag eleman oldugunu ekrana yazdiriniz.
- 8) **clear()** methodu kullanarak, list'deki tum elemanlari siliniz.
- 9) **isEmpty()** methodu kullanarak, list'deki tum elemanların silindigini dogrulayiniz



## Array'i ArrayList'e Cevirmek

```
String [] arr = {"tavsan", "serce" };
```

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

Uzunlugu degistirilemeyen bir list'e cevirir. Yani; yeni olusturulan listte add(), remove() ve clear() methodlarini kullanamazsiniz. Exception

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

```
System.out.println(list); // [tavsan, serce]
```

**NOT:** Eger array'deki bir elemani degistirirseniz list'teki eleman da otomatik olarak degisir. Listteki bir elemani degistirirseniz array de otomatik olarak degisir.

```
list.set(1, "test"); // [tavsan, test]
```

```
arr[0] = "new"; // [new, test]
```

```
System.out.println(Arrays.toString(arr)); // [new, test]
```

```
System.out.println(list); // [new, test]
```



# ArrayList'i Array'e Cevirmek

```
List<String> list = new ArrayList<>();  
list.add("tavsan");  
list.add("horoz");  
System.out.println(list); // [tavsan,horoz]
```

## 1.yontem

```
String arr[ ] = list.toArray(new String[0]);
```

```
System.out.println(arr.length); // 2  
System.out.println(Arrays.toString(arr)); // [tavsan,horoz]
```

## 2.yontem

```
Object arr[ ] = list.toArray();
```

```
System.out.println(arr.length); // 2  
System.out.println(Arrays.toString(arr)); // [tavsan,horoz]
```



**BATCH** : Batch 59-60

**LESSON** : Java 26

**DATE** : 24.03.2022

**SUBJECT** : For Each Loop

Constructor

- techproeducation
- techproeducation
- techproeducation
- techproeducation
- techproedu



## For Each Loop / Enhanced (Gelistirilmis) For Loop

### Faydalari:

Kodun daha okunabilir olmasini saglar. Hata yapma ihtimalini azaltir.

```
public static void main(String args[ ]){  
  
    int arr[ ]={12,13,14,44};  
  
    for( int i: arr) {  
        System.out.print(i + " ");  
    }  
  
}
```

```
public static void main(String args[ ]){  
    ArrayList<String> list=new  
    ArrayList<String>();  
    list.add("Ali");  
    list.add("Veli");  
    list.add("Can");  
  
    for( String s : list) {  
        System.out.print(s + " ");  
    }  
}
```



## For Each Loop

### Soru 1:

Bir integer array olusturunuz ve bu array'deki tum sayilarin carpimini For-each loop kullanarak bulunuz. Sonucu ekrana yazdiriniz.

### Soru 2:

Bir integer list olusturunuz ve bu list'deki tum sayilarin karesinin toplamini For-each loop kullanarak bulunuz. Sonucu ekrana yazdiriniz.

### Soru 3:

iki String array olusturunuz ve bu array'lerdeki ortak elemanlari For-each loop kullanarak bulunuz. Sonucu ekrana yazdiriniz.

Ortak eleman yoksa ekrana "Ortak eleman yok" yazdiriniz

### Soru 4:

Bir String olusturunuz, bu String'deki character'leri for-each loop kullanarak yazdiriniz. *ipucu: split()*



# Constructor (Java object'leri nasıl oluşturur ?)

**Constructor** Java'da object oluşturmak için kullanılan kod blogudur.

**Constructor** çalışmadan object oluşturulması mümkün degildir

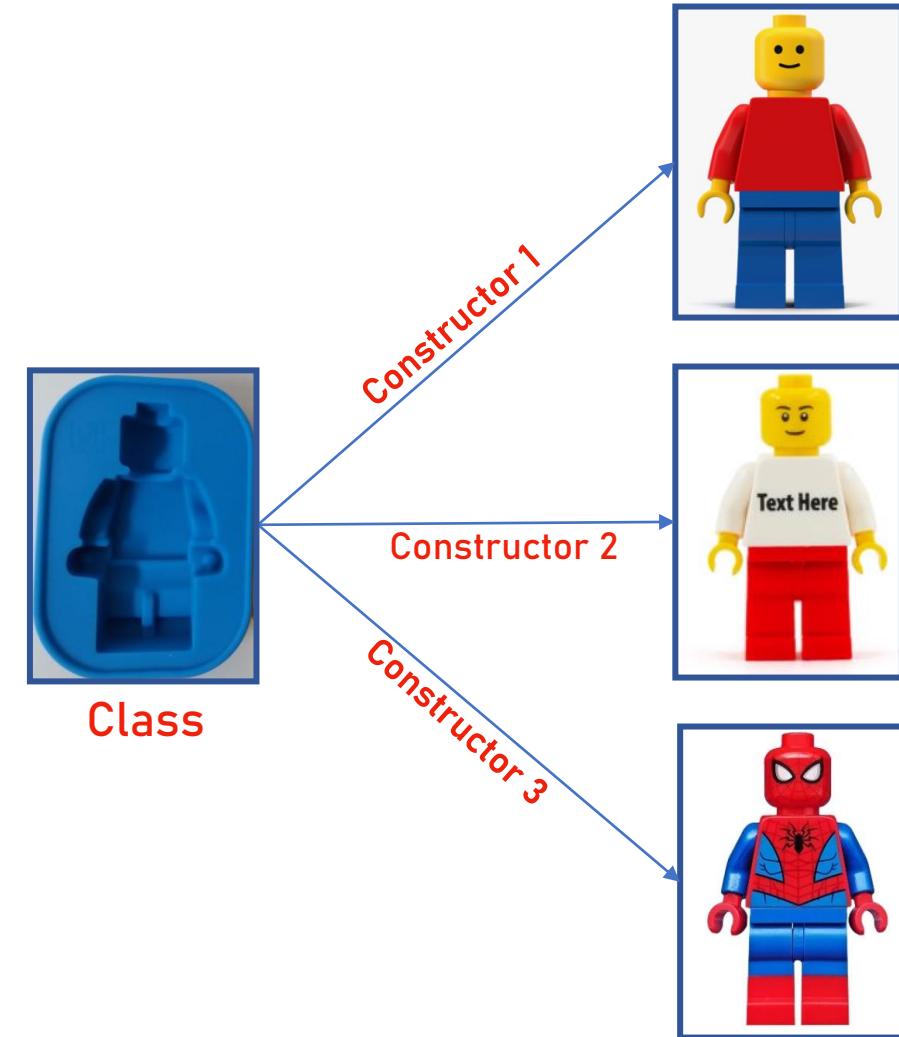
**Constructor** nedir ?

**Constructor** Method degildir.

**Constructor** variable degildir.

**Constructor** su ana kadar öğrendiklerimizden farklıdır

**Constructor** constructor'dır.





# Constructor (Java object'leri nasıl oluşturur ?)

Bana tisort uret

```
public Uret();
```

Bana yesil tisort uret

```
public Uret("yesil");
```

Bana yesil, v yaka tisort uret

```
public Uret("yesil", "V yaka");
```

Bana yesil, v yaka, medium tisort uret

```
public Uret("yesil", "V yaka", "medium");
```

Bana yesil, v yaka, medium tisortlerden 100 tane uret

```
public Uret("yesil", "V yaka", "medium",100);
```





# Constructor (Java object'leri nasıl oluşturur ?)



Class(Object Kalibi)

Field      Method  
(Variables) (Functions)



Object

Birden fazla Obje birleştirilir



Application



**BATCH** : Batch 59-60  
**LESSON** : Java 27  
**DATE** : 25.03.2022  
**SUBJECT** : Constructor

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



# Constructor (Java object'leri nasıl oluşturur ?)

Java'da Class'lar object üretmek için Constructor kullanılır

Java'da bir Class oluşturduğumuzda, Java object oluşturabilmemiz için default constructor oluşturur.

Default constructor Class içinde görülmeyecektir

Kullanıcı yeni bir Constructor oluşturursa Java default constructor'i siler.

Constructor **nasıl** ve **nerede** oluşturulur ?

- Constructor Class içerisinde oluşturulur.
- Constructor'in ismi Class ismi aynı olmalıdır, dolayısıyla isim büyük harfle başlar
- Constructor'ların return type'ları olmaz

```
public class MyClass {  
    MyClass() {  
    }  
  
    public static void main(String args) {  
    }  
}
```



# Constructor

Bir Car Class'i olusturalim.

Arabalari olustururken ozelliklerini  
yukleyebilmek icin variable ve method'lar  
olusturabiliriz

Bizim class'imizda

- Ilan no
- Marka
- Model
- Yıl
- Fiyat gibi variable'lar ve
- hiz(120) ve yakin(dizel) gibi iki de method olsun.

2020 BMW 3.20İ FIRST EDITION M SPORT 900 KM'DE BOYASIZ

647.900 TL

KONYA / KARATAY / FEVZİÇAKMAK MAHALESİ

İlan No: 16126048  
İlan Tarihi: 24 Şubat 2021  
Marka: BMW  
Seri: 3 Serisi  
Model: 320i First Edition M Sport  
Yıl: 2020  
Yakıt Tipi: Benzin  
Vites Tipi: Otomatik  
Motor Hacmi: 1597 cc  
Motor Gücü: 170 hp  
Kilometre: 900 km  
Boya-değisen: Tamamı orjinal  
Takasa Uygun: Takasa Uygun  
Kimden: Galeriden

#16126048 6/25

#16126048

Karşılaştır Favori Paylaş



# Constructor

```
public class Car {  
  
    int ilanNo;  
    String marka;  
    String model;  
    int yil;  
    int fiyat;  
  
    public static void main(String[] args) {  
  
    }  
  
    public void hiz(int maxHiz) {  
        System.out.println("Araba max. " + maxHiz + " km hiz yapar");  
    }  
  
    public void yakit (String yakitTuru) {  
        System.out.println("Araba yakit olarak " + yakitTuru + " kullanir");  
    }  
}
```



# Constructor

Baska bir Class'da olusturdugumuz Car class'indan bir object olusturaim ve degerler atayalim.

```
public static void main(String[] args) {  
  
    Car car1=new Car();  
    car1.ilanNo=1001;  
    car1.marka="Toyota";  
    car1.model="Corolla";  
    car1.yil=2010;  
    car1.fiyat=15000;  
  
    System.out.println(car1.ilanNo +"," + car1.marka +"," +car1.model +"," +  
                      car1.yil +"," + car1.fiyat);  
    car1.hiz(150);  
    car1.yakit("Benzin");
```

```
1001,Toyota,Corolla,2010,15000  
Araba max. 150 km hiz yapar  
Araba yakit olarak Benzin kullanir
```



# Constructor

```
public class Car {  
    int ilanNo;  
    String marka;  
    String model;  
    int yil;  
    int fiyat;  
  
    public static void main(String[] args) {  
    }  
  
    public void hiz(int maxHiz) {  
        System.out.println("Araba max. " + maxHiz + " km hiz yapar");  
    }  
  
    public void yakit (String yakitTuru) {  
        System.out.println("Araba yakit olarak " + yakitTuru + " kullanir");  
    }  
}
```

Class icinde gorunur Constructor yok  
(Default Constructor) calisiyor

Default Constructor

Default Constructor

Default Constructor

1001,Toyota,Corolla,2010,15000  
Araba max. 150 km hiz yapar  
Araba yakit olarak Benzin kullanir

1002,Opel,Astra,2012,10000  
Araba max. 130 km hiz yapar  
Araba yakit olarak Dizel kullanir

1003,Audi,A4,2015,20000



# Parametrized (Parametreli) Constructor

- Default constructor ile olusturdugumuz obje icin default deger disinda deger atamak istersek once objeyi olusturmali, sonra tum ozelliklere tek tek Assignment yapmaliyiz.
- Deger atama islemini obje olustururken yapmak istersek atama yapacagimiz variable'lari iceren constructor olusturabiliriz.
- Bu durumda this ile yolladigimiz degerleri instance variable'lara assign etmeliyiz.

```
public class Car {  
    public int ilanNo;  
    public String marka;  
    public String model;  
    public int yil;  
    public int fiyat;  
  
    public Car() {  
    }  
  
    public Car(int ilanNo, String marka, String model, int yil, int fiyat) {  
        this.ilanNo=ilanNo;  
        this.marka=marka;  
        this.model=model;  
        this.yil=yil;  
        this.fiyat=fiyat;  
    }  
  
    public static void main(String[] args) {  
    }  
}
```



# Constructor

Class icinde birden fazla Constructor olursa ...

Java obje create ederken kullandigimiz argument'lere gore uygun constructor'i kullanir

```
public class Car {  
  
    public int ilanNo;  
    public String marka;  
    public String model;  
    public int yil;  
    public int fiyat;  
  
    1 public Car() {  
    }  
    2 public Car(int ilanNo) {  
        this.ilanNo=ilanNo;  
    }  
    3 public Car(int ilanNo, String marka, String model,  
                int yil, int fiyat) {  
        this.ilanNo=ilanNo;  
        this.marka=marka;  
        this.model=model;  
        this.yil=yil;  
        this.fiyat=fiyat;  
    }  
  
    public static void main(String[] args) {  
    }  
}
```

Car car1 = new Car();  
Default degerlere sahip olur

Car car24 = new Car(1024);  
ilan no 1024 olur,  
diger variable'lar Default degerlere sahip olur

Car car84 = new Car(1834, opel, astra, 2020, 15000);  
Argument olarak girilen degerlere sahip olur



**BATCH** : Batch 59-60  
**LESSON** : Java 28  
**DATE** : 26.03.2022  
**SUBJECT** : Constructor  
Static Keyword





# Constructor

Asagidaki class calistirildiginda output ne olur ?

```
public class Student {  
  
    String name = "Emily";  
    int age = 20;  
  
    Student(String name, int age) {  
        this.name = name;  
        this.age = 22;  
    }  
  
    public static void main(String args) {  
  
        Student st = new Student("Oliver", 21);  
  
        System.out.print(st.name);  
  
        System.out.print(", " + st.age);  
    }  
}
```



# Constructor

Asagidaki class calistirildiginda output ne olur ?

```
public class Student {  
    String name;  
    int age;  
    String phone;  
  
    Student() {  
    }  
    Student(String name, int age, String phone) {  
        this.phone = phone;  
        this.name = name;  
    }  
  
    public static void main(String[] args) {  
        Student s1 = new Student();  
  
        Student s2 = new Student("John",25,"029-998877");  
  
        System.out.print(s2.name + ", " + s2.age + ", " + s2.phone);  
    }  
}
```



## Constructor Call

- Obje olusturulurken constructor calisir.
- Bazi durumlarda constructor icinden baska bir constructor cagirmamiz gerekebilir
- Bir constructor'un icinden diger bir constructor'i cagirmak icin  
**this(parametreler);** kullanilir.

**Soru :** Yandaki class calistirildiginda output ne olur ?

**this(parametreler);** kullanirken

**Kural 1 :** **this(parametre);** cagrildigi constructor'in ilk satirinda yazilmalidir

**Kural 2:** Kural 1'den dolayi bir constructor icinde sadece 1 constructor cagrilabilir

```
public class MyConstructor {  
    int x =5;  
  
    MyConstructor(){  
        System.out.println("-x" + x );  
    }  
  
    MyConstructor(int x){  
        this();  
  
        System.out.println("-x" + x );  
    }  
  
    public static void main(String[] args) {  
        MyConstructor mc1 = new MyConstructor(4);  
  
        MyConstructor mc2 = new MyConstructor();  
    }  
}
```



# Constructor Call

Yandaki class calistirildiginda output ne olur ?

```
public class MyConstructor {  
  
    int x =3;  
    int y =5;  
  
    MyConstructor(){  
        x+=1;  
        System.out.println("-x" + x );  
    }  
  
    MyConstructor(int i){  
        this();  
  
        this.y= i;  
        x += y;  
        System.out.println("-x" + x );  
    }  
    MyConstructor(int i , int i2){  
        this(3);  
  
        this.x -= 4 ;  
        System.out.println("-x" + x );  
    }  
  
    public static void main(String[] args) {  
        MyConstructor mc1 = new MyConstructor(4,3);  
    }  
}
```



# Constructor Call

Yandaki class calistirildiginda output ne olur ?

Which are true of the following code? (Choose all that apply)

```
1: public class Rope {  
2:     public static void swing() {  
3:         System.out.print("swing ");  
4:     }  
5:     public void climb() {  
6:         System.out.println("climb ");  
7:     }  
8:     public static void play() {  
9:         swing();  
10:    climb();  
11: }  
12:    public static void main(String[] args) {  
13:        Rope rope = new Rope();  
14:        rope.play();  
15:        Rope rope2 = null;  
16:        rope2.play();  
17:    }  
18: }
```

- A. The code compiles as is.
- B. There is exactly one compiler error in the code.
- C. There are exactly two compiler errors in the code.
- D. If the lines with compiler errors are removed, the output is climb climb.
- E. If the lines with compiler errors are removed, the output is swing swing.
- F. If the lines with compile errors are removed, the code throws a NullPointerException.



# Constructor Call

```
public class MyClass{  
    int num1;  
    String name = "Ali";  
  
    MyClass(){  
        char letter = 'c';  
    }  
  
    MyClass (int num1){  
        this();  
        this.num1 = num1;  
    }  
  
    void MyClass(){  
        num1++;  
    }  
  
    increase(int num1){ name++;  
    } }
```

Boslukları “True” veya “False” olarak doldurun.

- 1) Turquoise'lar instance variable'lardır.....
- 2) Orange un-parameterized constructor'dır. ....
- 3) Pembe parameterized constructor'dır. ....
- 4) Yesil un-parameterized constructor'dır. ....
- 5) Mavi parameterized constructor'dır. ....
- 6) “letter” local variable'dır ....
- 7) Instance variable'lara mutlaka atama yapılmalıdır ....
- 8) Verilen code block'da , sadece 1 tane compile time error vardır. ....
- 9) “this” keyword, instance variable'lar ile ilişkilidir. ....
- 10) this() yesil MyClass()'i çağırır ....



**BATCH** : Batch 59-60

**LESSON** : Java 29

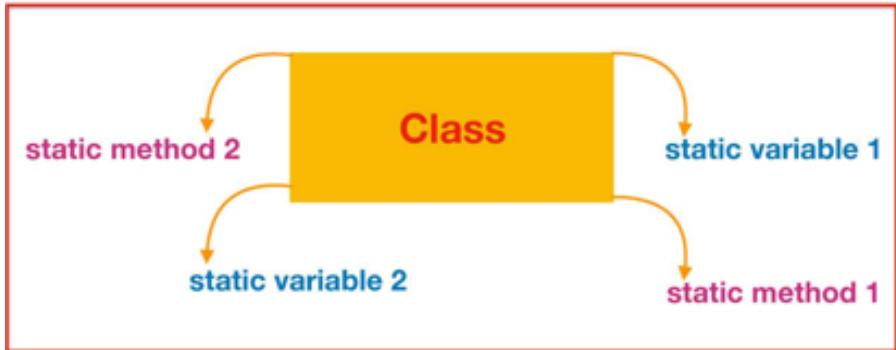
**DATE** : 28.03.2022

**SUBJECT** : Static Keyword

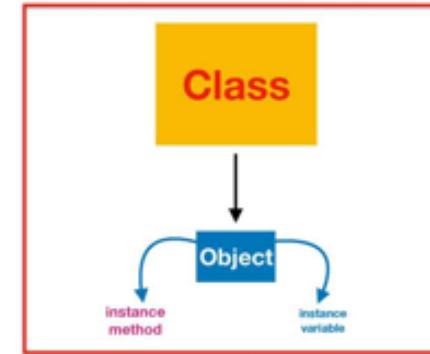
-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu



# Static Keyword



Static variables / methods



Non-static variables / methods

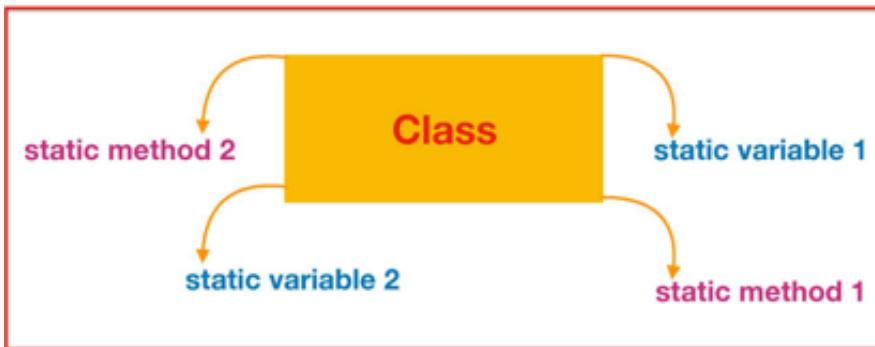
**static** kelimesi bir variable'i veya Method'u Class'a baglamak icin kullanilir.

Bir variable veya Method static olarak etiketlendiginde o artık class'in elemani olur ve ona ulasmak icin **object olusturmamiza gerek kalmaz**.

Instance variable'a ulasmak icin ise **MUTLAKA object olusturmaliyiz**



# Static Variables



Static variables / methods

- 1) Class yuklendiginde, memory'de static variable'lar olusturulur.
- 2) Static variable'lar bir tane olusturulur ve class'daki tum objeler onu gorur ve kullanir.
- 3) Memory kullanimi static variable'lar icin sadece bir kere olur.
- 4) Static variable'lar static veya static olmayan methodların içinde kullanilabilir.
- 5) Static variable'lara baska classlardan sadece class ismi kullanilarak ulasilabilir, obje olusturmaya gerek yoktur.



# Static Variables

Asagidaki class calistirildiginda output ne olur ?

```
public class Deneme {  
  
    static int count=0;  
  
    public void increment() {  
        count++;  
    }  
  
    public static void main(String[] args) {  
        Deneme obj1=new Deneme();  
  
        Deneme obj2=new Deneme();  
  
        obj1.increment();  
  
        obj2.increment();  
  
        System.out.println("Obj1: count is="+obj1.count);  
        System.out.println("Obj2: count is="+obj2.count);  
    }  
}
```



# Static Variables

Asagidaki class calistirildiginda output ne olur ?

```
public class Deneme {  
  
    int x;  
    static int y;  
  
    Deneme(int i){  
        x+=i;  
        y+=i;  
    }  
  
    public static void main(String[] args) {  
  
        new Deneme(2);  
  
        Deneme dnm=new Deneme(3);  
  
        System.out.println(dnm.x + "," + dnm.y);  
    }  
}
```



# Static Methods

- 1) Return Type'dan once **static keyword** kullanarak, **static method** olusturabiliriz

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



## Static Methods

- 2) Static Method'lar **static variable** (class variables) lari direkt kullanabilirler

```
public class Deneme {  
  
    static int sayi1=10;  
    int sayi2=20;  
  
    public static void main(String[] args) {  
        System.out.println(sayi1);  
  
        System.out.println(sayi2);  
    }  
  
    public static void add() {  
        System.out.println(sayi1);  
  
        System.out.println(sayi2);  
    }  
}
```

ama static olmayanlari object olusturmadan kullanamazlar



## Static Methods

3) Static Method'lar **static** ve **non-static** method'lardan cagrilabilir.

```
public class Deneme {  
  
    public static void main(String[] args) {  
        add();  
    }  
  
    public static void add() {  
    }  
  
    public void concat() {  
        add();  
    }  
  
}
```



# Static Methods

Asagidaki class calistirildiginda output ne olur ?

```
public class Counter {  
    int count;  
    static int stCount;  
    public Counter() {  
        count ++ ;  
        stCount ++ ;  
    }  
    public int getCount(){  
        return count;  
    }  
    public static int getStCount(){  
        return stCount;  
    }  
  
    public static void main(String[] args) {  
  
        Counter cs1 = new Counter();  
        Counter cs2 = new Counter();  
        Counter cs3 = new Counter();  
        Counter cs4 = new Counter();  
        Counter cs5 = new Counter();  
        Counter cs6 = new Counter();  
        System.out.println("count is: " + cs6.getCount());  
        System.out.println("stCount is: " + cs6.getStCount());  
    }  
}
```



BATCH : Batch 59-60

LESSON : Java 30

DATE : 29.03.2022

SUBJECT : Static Keyword

Pass By Value & Pass By  
Reference

- techproeducation
- techproeducation
- techproeducation
- techproeducation
- techproedu



## Önceki Dersten Aklımızda Kalanlar

- 1- static keyword kullandığımız class üyeleri direkt class'a bağlı olurlar. (Objelere göre degismez)
- 2- Static bir class üyesine class içerisinde heryerden ulaşılabilir ve değiştirilebilir.
- 3- Baska class'lardan static class üyelerine ulaşmak istediğimizde classIsMi.staticUye yazarak tüm static class üyelerine ulaşabilir ve değiştirebiliriz
- 4- Program çalışmaya başlayınca, Java öncelikle static variable'ların oluşturulmasını ve varsa ilk değer atamasını yapar, sonra kod çalıştığı surece static variable'larda yapılan değer değişiklikleri kalıcı olur.
- 5- static bir method içerisinde non-static bir class üyesi kullanılamaz



## Static Methods

Asagidaki class calistirildiginda output ne olur ?

```
public class StaticMember {  
    static int x;  
    int y;  
  
    StaticMember(){  
        x+=2;  
        y++;  
    }  
    static int getSquare(){  
        return x * x;  
    }  
    public static void main(String[] args) {  
        StaticMember sm1 = new StaticMember();  
  
        StaticMember sm2 = new StaticMember();  
  
        int z = sm1.getSquare();  
  
        System.out.print("-x" + z + "-y" + sm2.y);  
    }  
}
```



## Static Methods

Asagidaki class calistirildiginda output ne olur ?

```
class Counter {
    int count=0;

    Counter(){
        count++;
        System.out.println(count);
    }

    public static void main(String args[]){
        Counter c1=new Counter();
        Counter c2=new Counter();
        Counter c3=new Counter();
    }
}
```



# Static Methods

Asagidaki class calistirildiginda output ne olur ?

```
class Student{  
  
    int number;  
    String name;  
    static String college ="ITS";  
  
    Student(int r, String n, String college){  
        this.number = r;  
        this.name = n;  
        this.college = college;  
    }  
  
    public static void main(String args[]){  
  
        Student s1 = new Student(111,"Karan", "MIT");  
        Student s2 = new Student(222,"Aryan", "Harvard");  
  
        System.out.println(s1.number);  
        System.out.println(s2.number);  
  
        System.out.println(s1.name);  
        System.out.println(s2.name);  
  
        System.out.println(s1.college);  
        System.out.println(s2.college);  
    }  
}
```



# Static Methods

What is the result of the following program?

```
1: public class Squares {  
2:     public static long square(int x) {  
3:         long y = x * (long) x;  
4:         x = -1;  
5:         return y;  
6:     }  
7:     public static void main(String[] args) {  
8:         int value = 9;  
9:         long result = square(value);  
10:        System.out.println(value);  
11:    } }
```

- A. -1
- B. 9
- C. 81
- D. Compiler error on line 9.
- E. Compiler error on a different line.



# Instance Variables vs Static Variables

## Instance Variable

- 1) instance variables .....'in icinde ama .....'in disinda olusturulur
- 2) Instance variables bir .....'e baglidir. Dolayisiyla, bir ..... olusturuldugunda olusur ve ..... silindiginde silinirler.
- 3) Instance variables ..... ismi ile cagrılabilirler.
- 4) instance variable icin ilk deger atamasi yapmak .....dir. Eger ilk atama yapilmazsa default deger alir.
- 5) Her yeni obje olusturuldugunda, instance variables ilk atanan degere esit olur. **True / False**
- 6) Bir class'i kullanarak 2 instance variable'a sahip 6 obje olusturursak, 12 instance variables olusturmus oluruz. **True / False**

## Static Variable

- 1) Static variables .....'in icinde ama .....'in disinda olusturulur
- 2) Static variables bir .....'a baglidir. Dolayisiyla, bir ..... olusturuldugunda olusur ve ..... silindiginde silinirler.
- 3) Static variables ..... ismi ile cagrılabilirler.
- 4) Static variable icin ilk deger atamasi yapmak ..... dir. Eger ilk atama yapilmazsa default deger alir.
- 5) Class variable'a her yeni deger atamasi oldugunda, degeri tum objeler icin degisir. **True / False**
- 6) Bir class'i kullanarak 2 static variable'a sahip 6 obje olusturursak, 2 static variables olusturmus oluruz. **True / False**



# Static Blocks

- 1) Static block static variable'lara deger atamasi yapmak icin kullanilir.
- 2) Static block, class ilk calistirilmaya baslandiginda calisir ve static variable'lara ilk deger atamasi yapar (initialize)
- 3) Static block'lar constructor'lardan, tum method'lardan ve main method'dan once calisir.
- 4) Eger 1'den fazla static block varsa ustteki blok daha once calisir.

```
public class StaticBlock {  
    public static int age;  
  
    static {  
        System.out.println("Static block 2 calisti");  
        age = 24;  
    }  
  
    static {  
        System.out.println("Static block 1 calisti");  
        age = 23;  
    }  
  
    public StaticBlock() {  
        System.out.println("Constructor calisti");  
        System.out.println(++age);  
    }  
  
    public static void main(String[] args) {  
  
        System.out.println("Main method calisti 1");  
        System.out.println(++age);  
        StaticBlock obj = new StaticBlock();  
        System.out.println("Main method calisti 2");  
    }  
}
```



# Pass By Value & Pass By Reference

Programlama dillerinde bir method cagrildiginda original data'nin nasil kullanilacagi iki sekilde belirlenebilir.

**Pass By Value** : Primitive bir data'yi parametre olarak bir method'a gonderdigimizde Java original variable yerine ayni degere sahip kopya bir variable olusturur ve method icerisinde kopya variable uzerinden islem yapilir.

**Pass By Reference** : de ise method cagrildiginda, data'nin original degeri ile islem yapilir. Eger method icerisinde data degistirilirse original deger de degismis olur.

Bu iki alternative gozonune alindiginda Java Pass By Value ozelligini kullanmaktadır.

<https://www.youtube.com/watch?v=wWh4U4Np05w>



# Pass By Value & Pass By Reference

**Soru :** Verilen bir fiyat icin %10 indirim yapan bir method olusturun.

- Method'da indirim uygulanan fiyati yazdirin
- Method Call sonrasi original fiyati yazdirin ve method'da yapılan degisikligin orginal degeri degistirip degistirmedigini kontrol edin.

```
public class StaticBlock {  
  
    public static void main(String[] args) {  
  
        int fiyat = 100;  
  
        System.out.println("Method'da hesaplanan fiyat : " + indirim(fiyat));  
        System.out.println("Method call sonrasi fiyat : " + fiyat);  
    }  
  
    private static int indirim(int fiyat) {  
        fiyat *= 0.90;  
        return fiyat;  
    }  
}
```



# Pass By Value & Pass By Reference

**Soru2 :** Verilen bir fiyat icin %10 , %20, %25 indirim yapan uc method olusturun.

- Method'da indirim uygulayip fiyat main method'da yazdirin
- Method'lari arka arkaya cagirip dogru calistiklarini kontrol edin

```
public static void main(String[] args) {  
    int fiyat = 100;  
  
    System.out.println("Method10'da hesaplanan fiyat : " + indirim10(fiyat));  
    System.out.println("Method20'da hesaplanan fiyat : " + indirim20(fiyat));  
    System.out.println("Method25'da hesaplanan fiyat : " + indirim25(fiyat));  
    System.out.println("Method call sonrasi fiyat : " + fiyat);  
}  
  
public static int indirim10(int fiyat) {  
    fiyat *= 0.90;  
    return fiyat;  
}  
  
public static int indirim20(int fiyat) {  
    fiyat *= 0.80;  
    return fiyat;  
}  
public static int indirim25(int fiyat) {  
    fiyat *= 0.75;  
    return fiyat;  
}
```



# Pass By Value & Pass By Reference

**Soru3 :** Bir list olusturalim. Eleman olarak 10,11,12 ekleyelim. Iki method olusturup list elemanlarini artirmayi deneyelim

- 1. Method'da elemanlari for each loop kullanarak artirin
- 2. Method'da elemanlari set method'u kullanarak artirin
- Method'lari arka arkaya cagirip artislarin kalici olup olmadiklarini kontrol edelim.

**NOT :** Java'da list veya array'in elemanlarini update ettiginizde elemanlar kalici olarak degisir.

List veya array'in kendisi degismez ama elemanlari kalici olarak degisir

```
public static void main(String[] args) {  
    List<Integer> list =new ArrayList<Integer>();  
    list.add(10);  
    list.add(11);  
    list.add(12);  
  
    degistir(list);  
    System.out.println(list);  
    degistir2(list);  
    System.out.println(list);  
}  
public static void degistir(List<Integer> list) {  
    for (Integer each : list) {  
        each=each+3;  
        System.out.print(each + " ");  
    }  
    System.out.println();  
    System.out.println(list);  
}  
public static void degistir2(List<Integer> list) {  
  
    for (int i = 0; i < list.size(); i++) {  
        list.set(i, list.get(i)+3);  
        System.out.print(list.get(i)+" ");  
    }  
    System.out.println();  
    System.out.println(list);  
}
```



# Pass By Value & Pass By Reference

**Soru4 :** Bir list ve bir array olusturalim ve eleman olarak 10,11,12 ekleyelim. İki method olusturup list ve array'i degistirmeyi deneyelim

- 1. Method'da array'e yeni bir array assign edelim ve yeni halini yazdiralim
- 2. Method'da list'e yeni bir list assign edelim ve yeni halini yazdiralim
- Method call'dan sonra main method'da yeniden yazdirip degisip degismediklerini kontrol edelim.

```
public static void main(String[] args) {
    int num[] = {10, 11, 12};
    degistirArray(num);
    System.out.println("method'dan sonra main method'un icinde array: " + Arrays.toString(num)); // [10, 11, 12]
    List<Integer> list = new ArrayList<>();
    list.add(10);
    list.add(11);
    list.add(12);
    degistirList(list);
    System.out.println("method'dan sonra main method'un icinde list : " + list); // [10, 11, 12]
}
public static void degistirList(List<Integer> list) {
    list = new ArrayList<>();
    list.add(40);
    System.out.println("list methodda : " + list); // [40]
}
public static void degistirArray(int num[]) {

    num = new int[5];

    System.out.println("array methodda : " + Arrays.toString(num)); // [0, 0, 0, 0, 0]
}
```



**BATCH** : Batch 59-60  
**LESSON** : Java 31  
**DATE** : 30.03.2022  
**SUBJECT** : Mutable & Immutable  
Classes

- techproeducation
- techproeducation
- techproeducation
- techproeducation
- techproedu



# Mutable & Immutable Classes

**Immutable** (değişmez) class'lar, objeleri bir kez oluşturuluktan sonra değiştiremediğimiz class'lardır.

**Mutable** (değişebilir) class'lar ise tam tersi olarak, oluşturduğumuz objeleri değiştirebildiğimiz class'lardır.

**NOT :** Immutable class'dan oluşturulan **objeler de immutable** olurlar.

Bu tur bir object'i oluşturabiliriz, fakat onları değiştiremeyiz.

Immutable bir objeyi değiştirmek istersek, Java o objeyi klonlar ve yapılan değişiklikleri klonlanmış yeni obje üzerinde gerçekleştirir.

Peki böyle bir duruma niçin ihtiyacımız olur diye bir soru sorarsak,

cevabı **thread safe (guvenli es zamanlı çalışma)** konusudur.

Immutable nesnelerin değerleri değişimeyeceği için üzerinde kaç tane thread çalışırsa çalışın hep aynı değerler üzerinden işlem yapılacaktır.



# Mutable & Immutable Classes

Java'da yaygın olanlarından örnek verecek olursak

**String** ve tüm Wrapper (Integer, Long, Double, Byte....) class'lar immutable sınıflardır.

Date, StringBuilder, StringBuffer, Arrays ve ArrayList mutable Class'lardandır.

Asagidaki ornekte String methodu kullanildiginda str'in degeri degismezken method kullanilan list'in degeri degismektedir.

```
public static void main(String[] args) {  
  
    String str= "Mehmet";  
    str.toUpperCase();  
    System.out.println(str); // Mehmet  
  
    List <String> list= new ArrayList<>();  
    list.add("Mehmet");  
    list.add("Ayse");  
    System.out.println(list); // [Mehmet, Ayse]  
}
```



# Mutable & Immutable Classes

String'de yaptigimiz degisikligin kalici olmasi icin assignment yapabiliriz.

Bu durumda da String immutable oldugu icin Java atadigimiz yeni deger icin klon bir variable olusturur ve yeni degeri yeni klonlanmis String'e assign eder, referansin isaret ettigi object de yeni klon object olur.

**Ornek :** Yandaki ornekte str String'i olusturulduktan sonra loop icerisinde 100 tane klon str olusturulur ve yazdirilir, loop sonuna gelip alt satira indigimizde Java son olusturulan klon'u refere eder.

```
public static void main(String[] args) {  
  
    String str = "";  
    for (int i = 0; i < 100; i++) {  
        str = i + ". deger";  
        System.out.println(str);  
    }  
  
    System.out.println("son deger : " + str);  
}
```



# Mutable & Immutable Classes

Java'da bir String iki şekilde olusturulabilir.

- 1- `String str = "Mehmet";` şeklinde (her zaman yaptigimiz şekilde) oluşturulduğunda, Java Virtual Machine öncelikle o değişkeni **String Havuzunda** arar ve bir karşılaşma bulursa, aynı referansı verir yeni String'e.

Bu yüzden aşağıdaki soruda `str3==str4` true dondurur,

- 2- `String str = new String("Mehmet");` şeklinde yeni bir obje olarak oluşturulduğunda, Java önce yeni bir Object oluşturur ve sonra istenen değeri assign eder.

Bu yüzden yukarıdaki ornekte  
`str1==str2` false dondurur

```
public static void main(String[] args) {  
  
    String str1=new String("mehmet");  
    String str2=new String("mehmet");  
  
    System.out.println("new == " + (str1 == str2));  
    System.out.println("new equals " + (str1.equals(str2)));  
  
    String str3="mehmet";  
    String str4="mehmet";  
  
    System.out.println("klasik == " + (str3 == str4));  
    System.out.println("klasik equals " + (str3.equals(str4)));  
}
```



# Mutable & Immutable Classes

What is the result of the following code? (Choose all that apply)

```
13: String a = "";
14: a += 2;
15: a += 'c';
16: a += false;
17: if ( a == "2cfalse") System.out.println("==");
18: if ( a.equals("2cfalse")) System.out.println("equals");
```

- A. Compile error on line 14.
- B. Compile error on line 15.
- C. Compile error on line 16.
- D. Compile error on another line.
- E. ==
- F. equals
- G. An exception is thrown.



## Mutable & Immutable Classes

What is the result of the following statements?

```
6: List<String> list = new ArrayList<String>();  
7: list.add("one");  
8: list.add("two");  
9: list.add(7);  
10: for(String s : list) System.out.print(s);
```

- A.** onetwo
- B.** onetwo7
- C.** onetwo followed by an exception
- D.** Compiler error on line 9.
- E.** Compiler error on line 10.



# Mutable & Immutable Classes

What is the result of the following statements?

```
3: ArrayList<Integer> values = new ArrayList<>();  
4: values.add(4);  
5: values.add(5);  
6: values.set(1, 6);  
7: values.remove(0);  
8: for (Integer v : values) System.out.print(v);
```

- A. 4
- B. 5
- C. 6
- D. 46
- E. 45
- F. An exception is thrown.
- G. The code does not compile.



**BATCH** : Batch 59-60  
**LESSON** : Java 32  
**DATE** : 31.03.2022  
**SUBJECT** : Date & Time

- techproeducation
- techproeducation
- techproeducation
- techproeducation
- techproedu



# Socrative Quiz

- 1) <https://b.socrative.com/login/student/> adresine gidin
- 2) Room Name **BULUTLUOZ** yazın
- 3) Isminizi yazın
- 4) **Done** butonuna basin

Sure : 13 Dakika



# Date & Time

Java'da tarih ve zaman için **3 Class** vardır. Bunlardan kendimize uygun olanı seçip object oluşturarak kullanabiliriz.

## 1) Local Date

```
LocalDate currentDate1 = LocalDate.now();
```

## 2) Local Time

```
LocalTime currentTime1 = LocalTime.now();
```

## 3) Local Date Time

```
LocalDateTime currentTime1 = LocalDateTime.now();
```



# Date & Time

## 1) Local Date

```
public static void main(String[] args) {
    LocalDate tarih = LocalDate.now();

    System.out.println(tarih); // 2021-03-13

    System.out.println(tarih.plusDays(5)); // 2021-03-18
    System.out.println(tarih.plusMonths(3)); // 2021-06-13
    System.out.println(tarih.plusYears(2)); // 2023-03-13

    System.out.println(tarih.plusDays(3).plusMonths(2).plusYears(1)); // 2022-05-16

    System.out.println(tarih.getYear()); // 2021
    System.out.println(tarih.getMonth()); // MARCH
    System.out.println(tarih.getMonthValue()); // 3
    System.out.println(tarih.getDayOfMonth()); // 13
    System.out.println(tarih.getDayOfYear()); // 72
    System.out.println(tarih.getDayOfWeek()); // SATURDAY

    System.out.println(tarih.minusDays(12)); // 2021-03-01
    System.out.println(tarih.minusMonths(5)); // 2020-10-13
    System.out.println(tarih.minusYears(2)); // 2019-03-13

    System.out.println(tarih.minusYears(2).plusMonths(3).minusDays(5)); // 2019-06-08

    System.out.println(tarih.isLeapYear()); // false
    LocalDate tarih2 = LocalDate.of(2019, 03, 05);
    System.out.println(tarih.isAfter(tarih2)); // true
    System.out.println(tarih.isBefore(tarih2)); // false
}
```



## Date & Time

### 2) Local Time

```
public static void main(String[] args) {  
  
    LocalTime currentTime1 = LocalTime.now();  
    System.out.println(currentTime1); //12:38:19.828  
  
    System.out.println(currentTime1.plusHours(3)); // 15:38:19.828  
  
    System.out.println(currentTime1.minusMinutes(6)); // 12:32:19.828  
  
    System.out.println(currentTime1.getSecond()); // 19  
  
    System.out.println(currentTime1.now(ZoneId.of("Japan"))); // 18:38:19.829  
    System.out.println(currentTime1.now(ZoneId.of("Turkey"))); // 12:38:19.830  
    System.out.println(currentTime1.now(ZoneId.of("Europe/Moscow"))); // 12:38:19.831  
}
```



## Date & Time

### 3) Local Date Time

```
public static void main(String[] args) {  
  
    LocalDateTime dateTime1 = LocalDateTime.now();  
    System.out.println(dateTime1); //2021-03-13T12:43:08.188  
  
    String dt=dateTime1.toString();  
    System.out.println(dt.startsWith("2021")); // true  
  
}
```



# Date & Time

## Date Time Formatter

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

    DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MMM/yy");
    // M==>Months, m==>minutes
    // MMM==>ilk 3 karakter
    // MM==>kacinci ay oldugu 2 rakam (03-04-vs.)
    // M==>kacinci ay oldugu 1 rakam (3-4-etc.)
    // MMMM==>Tum isim

    LocalDate tarih = LocalDate.now();
    System.out.println(tarih); // 2021-03-13
    System.out.println(dtf.format(tarih)); // 13/Mar/21
    System.out.println(dtf.format(tarih.plusMonths(9))); // 13/Dec/21

    LocalTime saat = LocalTime.now();
    DateTimeFormatter dtf2 = DateTimeFormatter.ofPattern("hh:mm");
    // hh==> am-pm formatinda
    // HH==> 24 saat formatinda

    System.out.println(dtf2.format(saat)); // 01:07
```



# Date & Time

Iki Tarih Arasındaki Zamani Bulmak

```
public static void main(String[] args) {  
  
    LocalDate d1 = LocalDate.now();  
    LocalDate bd1 = LocalDate.of(1997, 5, 23);  
  
    //yas bulmak icin yil,ay ve gun'u bulmak isterseniz  
    Period age = Period.between(bd1, d1);  
    System.out.println(age); // P23Y9M18D  
  
    //Yas bulmak icin sadece yili ogrenmek isterseniz  
    int ageYear = Period.between(bd1, d1).getYears();  
    System.out.println(ageYear); // 23  
}
```



**BATCH** : Batch 59-60  
**LESSON** : Java 33  
**DATE** : 1.04.2022  
**SUBJECT** : Varargs  
String Builder





## Varargs

Varargs tek method'a istediğimiz kadar parametre yollayarak sonuç almamizi sağlar.

Yani parametre sayımız değişken ancak method'un yapacağı iş sabitse Varargs kullanarak tek method'la kodumuzu yazabiliriz

Varargs ozellik olarak list gibi calisir (uzunlugu esnektir) fakat varargs'in arka planında Java Array ile calisir.

Varargs'i declare etmek icin data type'dan sonra ... kullaniriz. (int... sayi vb..)

Bir method'da varargs disinda parametre varsa varargs parametre olarak en sona yazilmalidir. (aksi durumda varargs nerede duracagini bilemez)

Bir method'da sadece 1 varargs kullanilabilir

```
public static void main(String[] args) {  
  
    add(5,7); //12  
    add(5,7,-15); //-3  
    add(5,7,-15,20); // 17  
    add(5,7,-15,20,30); // 47  
}  
  
public static void add(int... sayi ) {  
  
    int toplam=0;  
    for (int i : sayi) {  
        toplam+=i;  
    }  
    System.out.println("girilen sayilarin toplami : "+ toplam);  
}
```



## Varargs

Which of the following compile? (Choose all that apply)

- A.** public void moreA(int... nums) {}
- B.** public void moreB(String values, int... nums) {}
- C.** public void moreC(int... nums, String values) {}
- D.** public void moreD(String... values, int... nums) {}
- E.** public void moreE(String[] values, ...int nums) {}
- F.** public void moreF(String... values, int[] nums) {}
- G.** public void moreG(String[] values, int[] nums) {}



# Varargs

```
public class Go {  
  
    public static void main(String[] args) {  
        new Go().Go(1, "hello");  
        new Go().Go(2, "hello", "hi");  
    }  
}  
TopJavaTutorial.com  
  
public void Go(int x, String... y){  
    System.out.print(y[y.length-x] + " ");  
}  
}
```



# String Builder

- StringBuilder “**mutable**” yani değiştirilebilir String elde etmemize olanak tanır.
- Böylece hafızada her seferinde yeni bir alan açılmadan var olan alan üzerinde değişiklik yapılabilir. Bu da StringBuilder sınıfını hafıza kullanımı olarak String sınıfının önüne geçirir.
- StringBuilder thread-safe değildir. Yani synchronized değildir. Thread'lı bir işlem kullanılacaksa StringBuilder kullanılması güvenli değildir.
- **Not:** StringBuffer, StringBuilder'a benzer. StringBuilder, StringBuffer'dan hızlıdır. Multi-thread için StringBuffer kullanılır.



# String Builder

- 1) `StringBuilder sb1 = new StringBuilder()` **====> Bos bir StringBuilder olusturur**
- 2) `StringBuilder sb2 = new StringBuilder("animal");` **====> Belli bir degeri olan StringBuilder olusturur**
- 3) `StringBuilder sb3 = new StringBuilder(5);` **====> Ilk uzunlugu tahmin edilen bir StringBuilder olusturur.**

```
StringBuilder sb = new StringBuilder(5);
```

0	1	2	3	4

```
sb.append("anim");
```

a	n	i	m	
0	1	2	3	4

```
sb.append("als");
```

a	n	i	m	a	l	s		...
0	1	2	3	4	5	6	7	...



# String Builder

1) append( ); StringBuilder'a ekleme yapar

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

    StringBuilder sb = new StringBuilder();
    sb.append("Mehmet");
    System.out.println(sb); // Mehmet
    sb.append(" Hoca");
    System.out.println(sb); // Mehmet Hoca
    sb.append(" Java").append(" anlatir.");
    System.out.println(sb);
    // Mehmet Hoca Java anlatir.
    // append ile arka arkaya ekleme yapilabilir
    sb.append("Java cok guzel",0,4);
    System.out.println(sb);
    // Stringin tumu degil bir bolumu eklenebilir
    // Mehmet Hoca Java anlatir.Java
}
```



# String Builder

2) **length( );** StringBuilder'in uzunlugunu verir

```
StringBuilder sb = new StringBuilder();
sb.append("Mehmet");
System.out.println(sb.length()); // 6
```

3) **capacity( );** StringBuilder'un kapasitesini verir

```
public static void main(String[] args) {
    StringBuilder sb = new StringBuilder(5);
    System.out.println(sb.length()); // 0
    System.out.println(sb.capacity()); // 5
    sb.append("Kemal"); // Kemal
    System.out.println(sb.length()); // 5
    System.out.println(sb.capacity()); // 5
    sb.append(" Can"); // Kemal Can
    System.out.println(sb.length()); // 9
    System.out.println(sb.capacity()); // 12
}
```



# String Builder

4) **charAt( );** StringBuilder'da istenen index'deki karakteri verir

5) **delete(4,7);** StringBuilder'da istenen index'ler arasindaki karakterleri siler.

6) **deleteCharAt(7);** StringBuilder'da istenen index'deki tek karakteri siler

7) **equals( );** İki StringBuilder'in değerlerinin karşılaştırır.

**NOT 1:** equals( ) method'unda parantez içine String yazarsak hata vermez ama false doner.

**NOT 2:** equals( ) method'u == gibi çalışır



## String Builder

8) **indexOf( );** StringBuilder'da istenen karakterin index'ini verir.

9) **insert(3, "Java ");** StringBuilder'da istenen indexden baslayarak istenen karakteri ekler.

10) **insert(3, "Java ",1,2);** StringBuilder'da istenen indexden baslayarak verilen String'in istenen parcasını ekler.

11) **replace(3, 8, " Ali ");**  
StringBuilder'da istenen index'ler arasındaki bolumun yerine verilen String'i ekler.



# String Builder

12) **reverse( );** StringBuilder'i tersine cevirir.

13) **setCharAt(3, 'k');** StringBuilder'da istenen index'deki karakteri istedigimiz karakter yapar.

14) **subSequence(3,7);** StringBuilder'da istenen indexler arasindaki karakterleri dondurur.

15) **toString( );** StringBuilder'i String'e cevirir.  
toString( )'den sonra nokta koyup String method'lari kullanilabilir.



# String Builder

16) **trimToSize( );** StringBuilder'in capacity ile length'ini esitler.

17) **compareTo( );** 2 StringBuilder'in tum karakterlerinin esitligini kontrol eder. (0 ise esit)

**Soru :** For loop ile 1000 defa bir islem yapalim. Oncesinde ve sonrasinda zamani kontrol edip StringBuilder ve String class'larinin performanslarini karsilastiralim.

**Ipucu:** long TimeSb = System.nanoTime(); kullanalim



# String Builder

## Soru 1:

What is the result of the following code?

```
7: StringBuilder sb = new StringBuilder();
8: sb.append("aaa").insert(1, "bb").insert(4, "ccc");
9: System.out.println(sb);
```

- A. abbaaccc
- B. abbaccca
- C. bbaaaccc
- D. bbaaccca
- E. An exception is thrown.
- F. The code does not compile.



# String Builder

## Soru 2:

What is the result of the following code?

```
2: String s1 = "java";
3: StringBuilder s2 = new StringBuilder("java");
4: if (s1 == s2)
5:     System.out.print("1");
6: if (s1.equals(s2))
7:     System.out.print("2");
```

- A. 1
- B. 2
- C. 12
- D. No output is printed.
- E. An exception is thrown.
- F. The code does not compile.



# String Builder

## Soru 3 :

Which are the results of the following code? (Choose all that apply)

```
String numbers = "012345678";
System.out.println(numbers.substring(1, 3));
System.out.println(numbers.substring(7, 7));
System.out.println(numbers.substring(7));
```

- A. 12
- B. 123
- C. 7
- D. 78
- E. A blank line.
- F. An exception is thrown.
- G. The code does not compile.



# String Builder

## Soru 4:

What is the result of the following code?

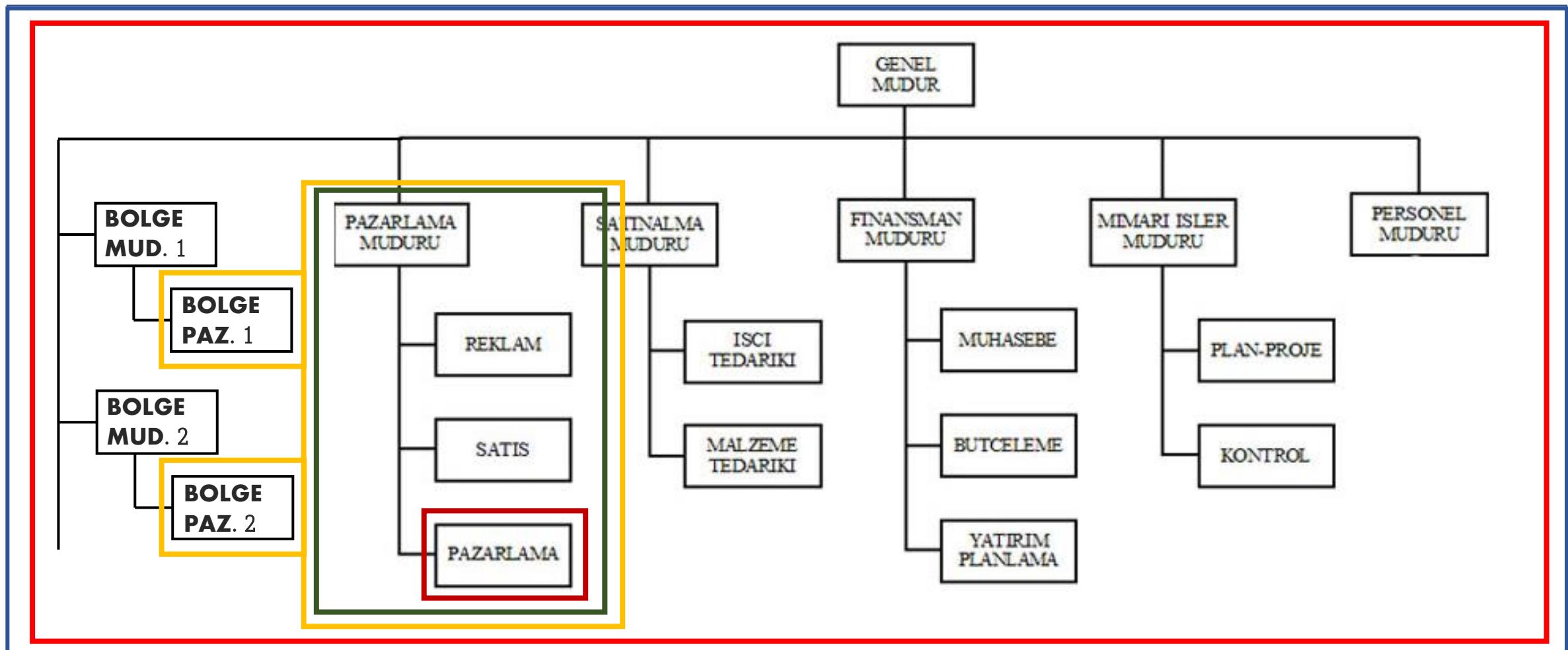
```
4: int total = 0;  
5: StringBuilder letters = new StringBuilder("abcdefg");  
6: total += letters.substring(1, 2).length();  
7: total += letters.substring(6, 6).length();  
8: total += letters.substring(6, 5).length();  
9: System.out.println(total);
```

- A. 1
- B. 2
- C. 3
- D. 7
- E. An exception is thrown.
- F. The code does not compile.



# Access Modifier

Java bir method'u, variable'i ya da class'i oluştururulurken bu öğelere kimlerin erişebileceğini belirtme olanağı tanır.





# Access Modifier

Java bir method'u, variable'i ya da class'i oluştururulurken bu öğelere kimlerin erişebileceğini belirtme olanağı tanır.

Bu eylemi gerçekleştirebilmek için kullanılan anahtar kelimelere **Access Modifiers** (Erişim Belirleyiciler) adını veririz.

Java'da 4 Farklı access modifier vardır

- 1) Private
- 2) Default (Eğer herhangi bir Access Modifier yazmazsak, Java Access Modifier'i default olarak kabul eder.)
- 3) Protected
- 4) Public

**NOT:** Class'lar için sadece public veya default kullanılabilir. Class'lar private veya protected olamazlar.



# Access Modifier

## 1) Private : Sadece olusturuldugu Class'da kullanilabilir

The screenshot shows a Java code editor with a project structure on the left and code on the right.

**Project Structure:**

- JavaProject C:\Users\Lenovo\IdeaProjects\JavaProject
- .idea
- src
  - package1
    - Class1
    - Class2
    - ClassChild1
    - ClassChild2
  - package2
    - Class1
    - Class2
    - ClassChild1
    - ClassChild2
  - package3
    - Class1
    - Class2
    - ClassChild1
    - ClassChild2
  - package4
- JavaProject.iml
- External Libraries
- Scratches and Consoles

**Code:**

```
1 package package1;
2
3 public class Class1 {
4
5     private static String privateVariable;
6     private static void privateMethod(){
7
8
9     }
10
11
12
13
14     public static void main(String[] args) {
15
16
17         }
18
19
20
21
22     public static void method1(){
23
24         }
25
26
27     public void method2(){
28
29         }
```

A red X is drawn over the entire package structure on the left, indicating that private members are only accessible within their own class, not across packages. A blue arrow points from the first line of code to the class definition. A green checkmark is placed over the main() method, which is a public static method and therefore accessible from other classes within the same package.



# Access Modifier

2) Default : Sadece olusturuldugu Class'in ait oldugu Package icinde kullanilabilir

The screenshot shows a Java project structure and a code editor. The project tree on the left has packages package1, package2, package3, package4, and JavaProject.iml. package1 contains Class1, Class2, ClassChild1, and ClassChild2. package2 contains Class1, Class2, ClassChild1, and ClassChild2. package3 contains Class1, Class2, ClassChild1, and ClassChild2. package4 is empty. The code editor on the right shows a class definition:

```
1 package package1;
2
3 public class Class1 {
4
5     static String defaultVariable;
6     static void defaultMethod(){
7
8 }
9
10
11
12
13
14     public static void main(String[] args) {
15
16         System.out.println("Hello, World!");
17     }
18
19     public static void method1(){
20
21     }
22
23 }
```

A green checkmark is placed over the package1 folder in the project tree and over the main() method in the code editor. A red 'X' is placed over the package2, package3, and package4 folders in the project tree.



# Access Modifier

- 3) Protected : Olusturuldugu Class'in ait oldugu Package icinde ve baska package icindeki Child Class'larda kullanilabilir

The screenshot shows a Java code editor with a project structure on the left and code on the right.

**Project Structure:**

- JavaProject C:\Users\Lenovo\IdeaProjects\JavaProject
- .idea
- src
  - package1
    - Class1
    - Class2
    - ClassChild1
    - ClassChild2
  - package2
    - Class1
    - Class2
    - ClassChild1
    - ClassChild2
  - package3
    - Class1
    - Class2
    - ClassChild1
    - ClassChild2
  - package4
- JavaProject.iml
- External Libraries
- Scratches and Consoles

**Code:**

```
1 package package1;
2
3 public class Class1 {
4
5     protected static String protectedVariable;
6     protected static void protectedMethod(){
7
8
9     }
10
11
12
13
14     public static void main(String[] args) {
15
16
17     }
18
19
20
21
22     public static void method1(){
23
24
25     }
26
27
28     public void method2(){
29
30
31     }
32 }
```

Annotations in the code editor:

- A green box highlights the package declaration and the protected variable/method definitions.
- A blue arrow points from the package1 folder in the project tree to the package declaration in the code.
- Red crossed-out boxes highlight package2 and package3, with green checkmarks next to them indicating they are not accessible.
- A large green checkmark is placed over the main() method definition.



# Access Modifier

4) Public : Her yerden erisilebilir. Hicbir sinirlama (restriction) icermez.

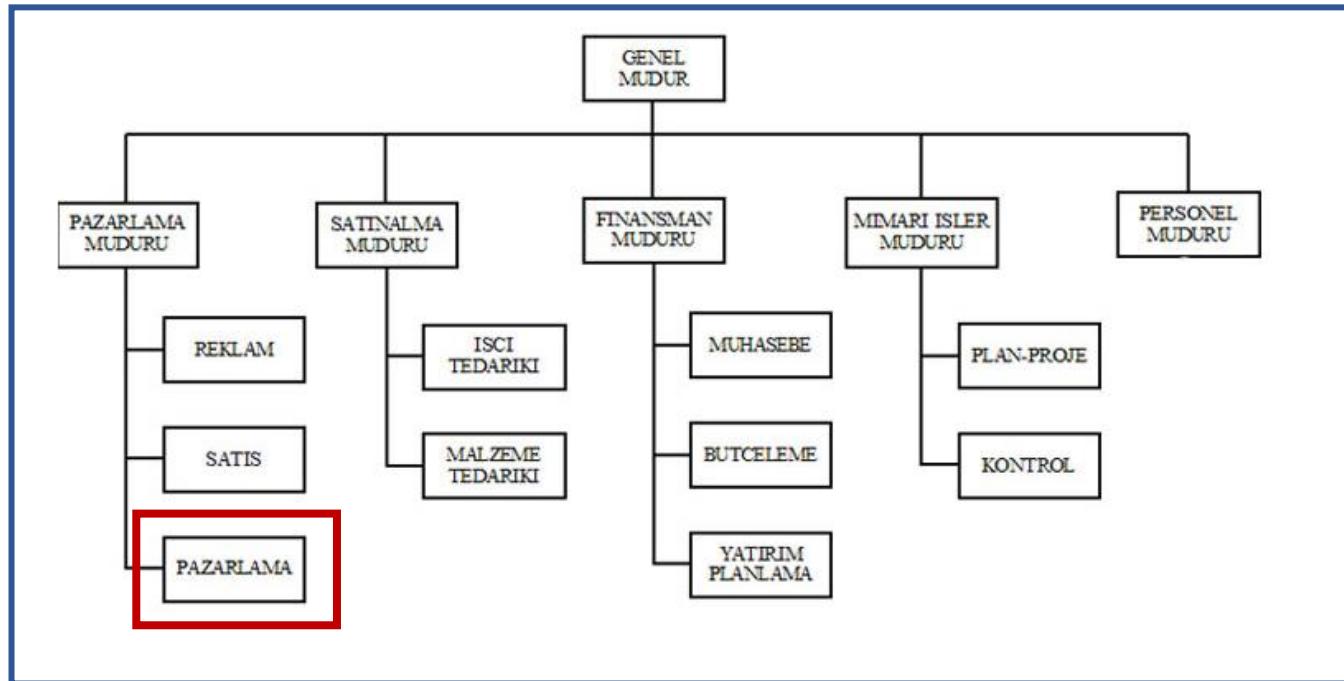
The screenshot shows a Java project structure and its corresponding code in an IDE. The project tree on the left shows a package named 'package1' containing classes 'Class1', 'Class2', 'ClassChild1', and 'ClassChild2'. Below 'package1' is another package 'package2' containing classes 'Class1', 'Class2', 'ClassChild1', 'ClassChild2', and a package named 'package3' which contains classes 'Class1', 'Class2', 'ClassChild1', 'ClassChild2', and a package named 'package4'. The code editor on the right displays the following Java code:

```
1 package package1;
2
3 public class Class1 {
4
5     public static String publicVariable;
6     public static void publicMethod(){
7
8
9         public static void main(String[] args) {
10
11
12             public static void method1(){
13
14
15                 }
16
17
18             public void method2(){
19
20
21
22         }
23     }
24 }
```

A blue arrow points from the 'public' keyword in the first line of the code to the 'Class1' class in the project tree. A large green checkmark is placed over the entire code block, indicating that the code correctly demonstrates the use of the 'public' access modifier.



# Encapsulation (Data Hiding)



Pazarlama birimi olarak rapor düzenliyoruz  
İhtiyaclarımız

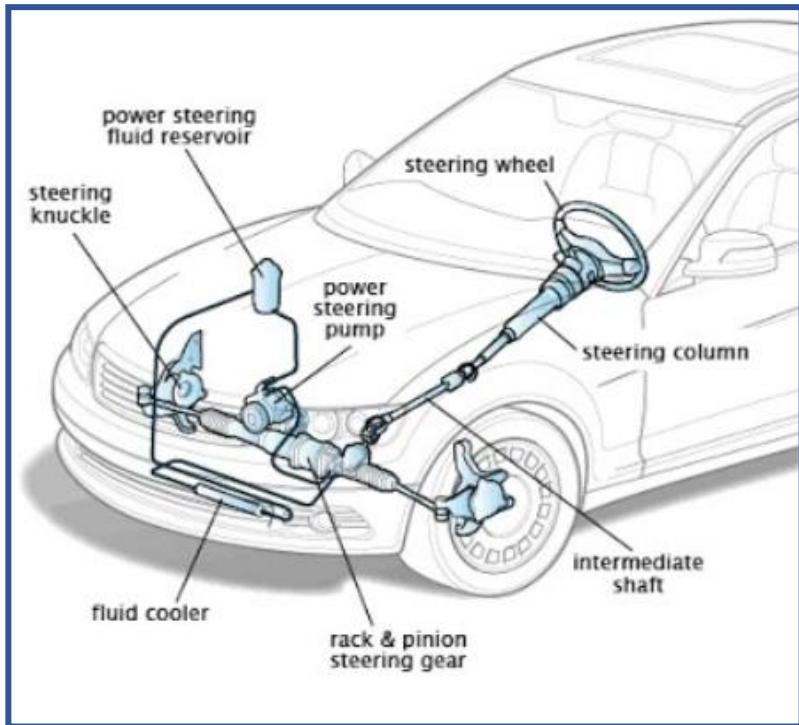
- 1- Satis bolumundeki personel rapor hazırlayacagimiz verileri girsin ama rapor sonuclarini goremesin
- 2- Raporu gormesine izin verilenler raporu gorsun ama degistiremesin

Java bir method'u ya da variable'i oluşturulurken class disindan bu ögелere erisimi kisitlama veya belirlenmis dataları degistirebilme olanağı tanır.



# Encapsulation (Data Hiding)

Before Encapsulation



After Encapsulation (Data Gizleme)





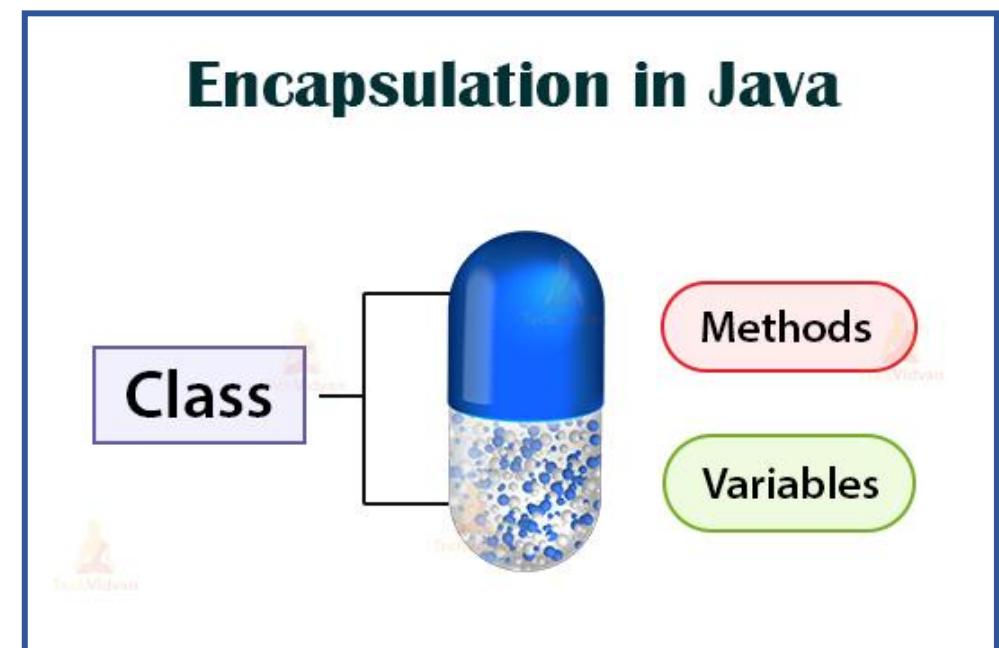
# Encapsulation (Data Hiding)

**Encapsulation**, onemli Class'ın üyelerini korumak için uygulanan data saklama yöntemidir.

Farklı Class'lardan erişilerek ya da yanlış kullanım sonucu kodunuzun veya onemli datalarınızın değişmesini istemiyorsanız Encapsulation ile datalarınızı koruyabilirsiniz.

Encapsule edilen variable ve method'lara sadece sizin verdığınız oranda erişilebilir.

Encapsule edilen variable ve method'lara izin verdığınız kişiler ulaşabilir ama DEĞİŞTIREMEZ.





## Encapsulation (Data Hiding)

Datalarımızı korumak için data'larımızı private yapabiliriz ama private yaptığımız dataları başka Class'lar kullanamaz. Bu durum OOP concept'ine uygun olmaz.

Private yaparak KORUMA ALTINA aldığımız Class üyeleri ninin sadece OKUNMASINI istiyorsak getter(), DEGER ATANMASINA da izin vermek istiyorsak setter() method'larını oluştururuz.

Encapsulation iki adımda yapılır:

- 1) Class üyelerini (*variable, method*) private yapmalısınız.
- 2) public olan getter() ve setter() methodlar üretmelisiniz.

**Not:** getter() data'yı sadece okumamiza yarar, data'da değişiklik yapamaz.

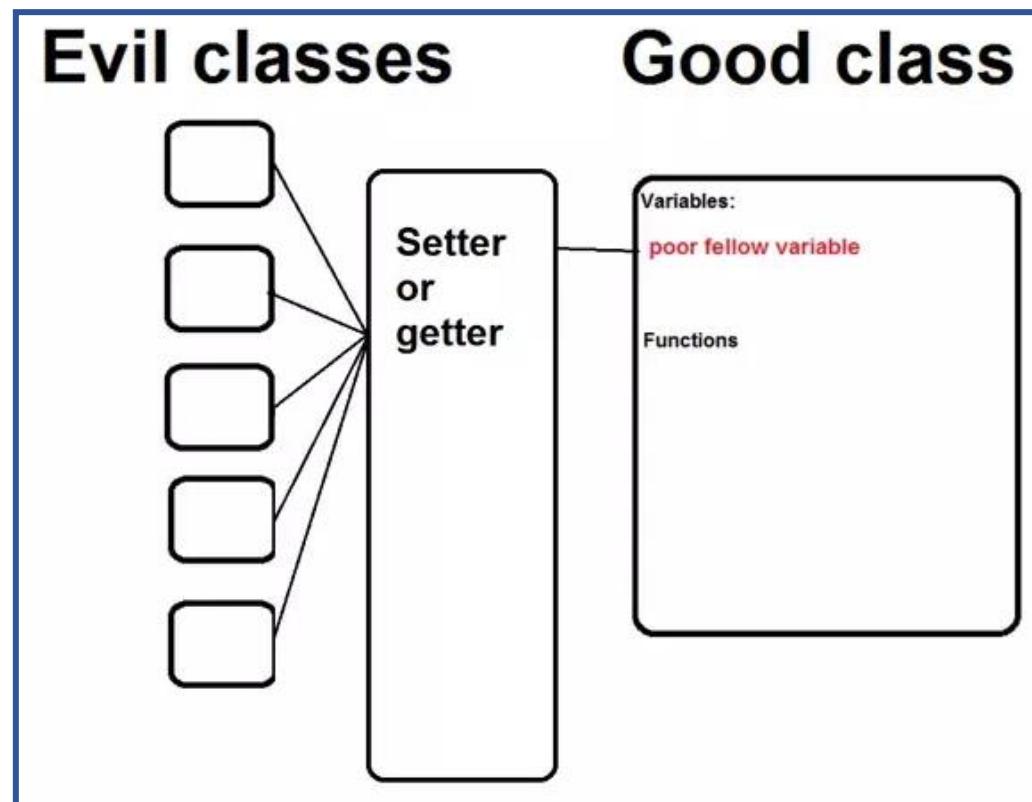
**Not:** setter() başka Class'larda oluşturulan objeler için data değerini değiştirmemizi sağlar.



# Getters & Setters



Getters and setters  
lead to the dark side...

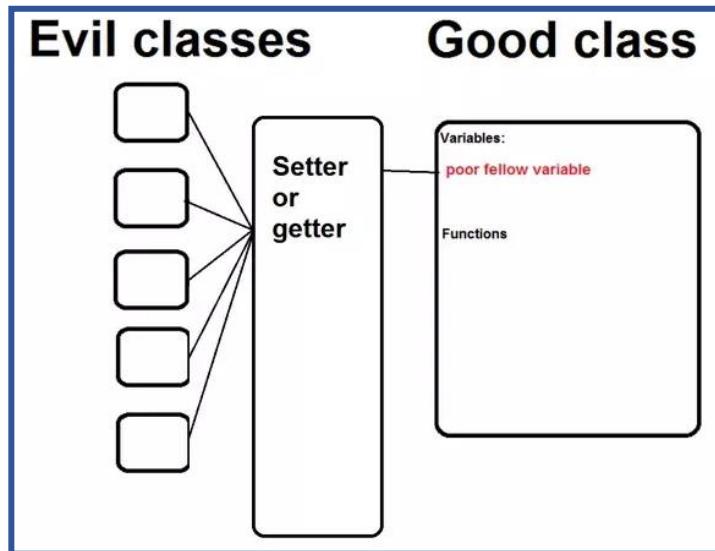




# Getters & Setters



Getters and setters  
lead to the dark side...



**Setter Methods :** Encapsule ettigimiz class uyelerinin,baska class'lardan obje uretilerek deger atanmasina izin verir.

```
public class Example {  
  
    private String tcNo= "12345678901";  
  
    public static void main(String[] args) {  
  
    }  
  
    public void setTcNo(String tcNo) {  
        this.tcNo = tcNo;  
    }  
}
```

Eger sadece **setter** method olusturulursa data degerleri degistirilebilir ama ilk atanan deger veya bizim atadigimiz yeni deger baska class'lardan okunamaz.



# Getters & Setters (Java Beans)

Getter and setter method'lari “**Java Beans**” olarak da adlandirilir.

## Isim verme kurallari (Naming convention)

- 1) Data type'lari **boolean** olan variable'larin getter metod isimleri “**is**” ile baslar.

```
private boolean happy = true;

public boolean isHappy() {
    return happy;
}
```

```
private int a=10;

public int getA() {
    return a;
}
```



# Getters & Setters (Java Beans)

## İsim verme kuralları (Naming convention)

```
private int a=10;
private boolean happy;

public void setA(int a) {
    this.a = a;
}

public void setHappy(boolean happy) {
    this.happy = happy;
}
```



# Tekrar Sorulari

1) Encapsulation nedir?

Encapsulation, hassas datalari korumak icin kullanılan data saklama yontemidir

2) Datalari nasıl saklariz?

Data'lari private access modifier kullanarak saklariz.

3) Saklanan datalara diger class'lardan ulasabiliriz?

Getter ve setter method'larini kullanarak ulasabiliriz.

4) getter() method'u ne yapar ?

Saklanan datalari okumamizi saglar.

5) setter () method'u ne yapar ?

Saklanan datalari obje uzerinden update edebilmemizi saglar

6) immutable class nedir?

Encapsule edilen bir class'da sadece getter method'u olusturursak datalari okuyabiliriz ama degistiremeyiz. Bu tur class'lara immutable class denir.

7) setter() method'lari icin naming convention nedir?

Tum data turleri icin isimler "set" ile baslar.

8) getter() method'lari icin naming convention nedir?

Boolean data turu icin "is" ile, diger data turleri icin "get" ile baslar.



# Getters & Setters

Which are methods using JavaBeans naming conventions for accessors and mutators?  
(Choose all that apply)

- A. public boolean getCanSwim() { return canSwim; }
- B. public boolean canSwim() { return numberWings; }
- C. public int getNumWings() { return numberWings; }
- D. public int numWings() { return numberWings; }
- E. public void setCanSwim(boolean b) { canSwim = b; }



## Getters & Setters

Which of the following are true? (Choose all that apply)

- A. Encapsulation uses package private instance variables.
- B. Encapsulation uses private instance variables.
- C. Encapsulation allows setters.
- D. Immutability uses package private instance variables.
- E. Immutability uses private instance variables.
- F. Immutability allows setters.



# Inheritance (Kalitim - Miras)



## Hayvanlar

(Hareket eder, nefes alır  
Beslenir, Cogalır, ölürl)



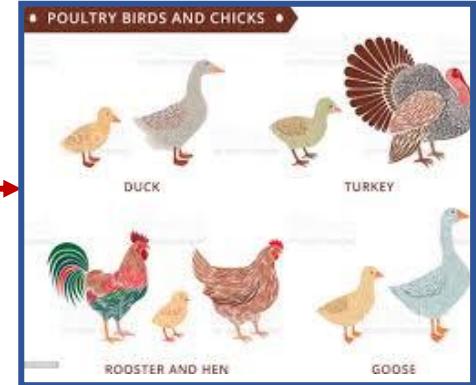
## Balıklar

(Denizde yasar, solungacla nefes alır,  
yuzerek hareket eder)

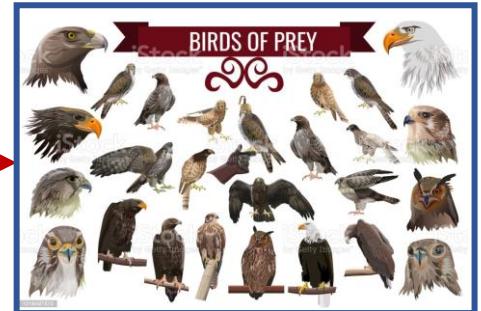


## Kuslar

(Kanatlari vardir,  
akcigerle nefes alırlar,  
gagalari vardir)



**Kumes Hayvanları**  
(ucamazlar,  
yuruyerek har.ederler)



## Avcı Kuslar

(ucarlar, et yerler,  
penceleri vardir)



# Inheritance (Kalitim - Miras)



Baba

Anne



Ali Can



Mert



Merve



Can



Ayse



yusuf

Kiz Cocugu  
Kiz Kardes  
Abla  
Hala  
Anne  
Teyze



# Inheritance (Kalitim - Miras)

Personel  
Kisisel bil.  
Departman  
Izin



Calisan  
Mesai  
Avans  
Rapor  
Std.Ucret



Isciler

Beyaz Yakalilar

Yan Hizmetler

Usta Basi  
saat ucreti 20  
Mesai : is bitene kadar

Surekli isciler  
saat ucreti 15  
Mesai : gunluk 8 saat

Gecici isciler  
saat ucreti 12  
Mesai : haftalık 25 saat

Disardan hizmet alimi  
Kendi Personelimiz

Muhasebe



## Inheritance (Kalitim - Miras)

- Java'da inheritance, bir objenin/class'in başka bir objenin/class'in tüm özelliklerini ve davranışlarını elde ettiği bir mekanizmadır.
- Inheritance,OOP'lerin (Nesne Yönelimli programlama sistemi) önemli bir parçasıdır.
- Java'da Inheritance'in arkasındaki fikir, daha once'den olusturulmus Class'larin üzerine yeni Class'lar oluşturabilmemizdir.
- Inheritance sayesinde yeni olusturdugumuz bir class'in var olan bir class'in tum methodlarini ve variable'larini kullanmasini saglayabiliriz.
- Inheritance bu islemin adidir. Inheritance sayesinde **child class**, **parent class**'daki public veya protected primitive datalari, objectleri, veya metodlari problemsiz bir sekilde kullanabilir.



## Inheritance (Kalitim - Miras)

Inheritance sayesinde yazılan bir code'un tekrar tekrar kullanılabilmesi (**reusability**) mümkün olur.

Geneli kapsayan class üyeleri parent class'a, daha spesifik olanlar ise child class'larda olusturulur.

**NOT 1:** Child classlar public ve protected data'lari problemsiz bir sekilde inherit edebilir.

**NOT 2:** Private data'lar inherit edilemez.

**NOT 3:** Default data'lar child ve parent aynı package'da oldukları zaman inherit edilebilirler.

**NOT 4:** Static Methods veya variable'lar inherit edilemezler.



# Inheritance (Kalitim - Miras)

## Interview Question

Nicin Inheritance kullaniriz ?

Inheritance sayesinde parent olarak tanımlanan class(ve onun parent class'larindaki)  
**protected/public** class üyelerini kullanabiliriz(reusability).

Inheritance'in faydalari nelerdir?

- 1:** Tekrarlardan kurtuluruz
- 2:** Daha az kod yazarak islemlerimizi yapabiliriz
- 3:** Kolayca update yapabiliriz
- 4:** Application'in bakimi ve surdurulmesi (maintenance) kolaylasir



# Inheritance (Kalitim - Miras)

```
public class Personel {  
  
    public static int sayac=1000;  
    public int id;  
    public String isim;  
    public String soyisim;  
    public String adres;  
    public String tel;  
  
    public int idAtama() {  
        this.id=sayac;  
        sayac++;  
        return id;  
    }  
}
```

Parent Class  
(Super)

```
public class Muhasebe extends Personel {  
  
    public int saatUcreti;  
    public String statu;  
    public int maas;  
  
    public int maasHesapla() {  
  
        int maas = saatUcreti*8*30;  
        return maas;  
    }  
}
```

Child Class (Sub) Parent Class (Super)

```
public class Memur extends Muhasebe{  
  
    public static void main(String[] args) {  
        Memur memur1=new Memur();  
        memur1.isim="Ali";  
        memur1.soyisim="Can";  
        memur1.tel="5521245789";  
        memur1.saatUcreti=20;  
        memur1.maas=memur1.maasHesapla();  
        memur1.id=memur1.idAtama();  
  
        Memur memur2=new Memur();  
        memur2.isim="Aliye";  
        memur2.soyisim="Canli";  
        memur2.tel="5521545789";  
        memur2.saatUcreti=25;  
        memur2.maas=memur2.maasHesapla();  
        memur2.id=memur2.idAtama();  
  
        System.out.println(memur1.id + " " + memur1.maas);  
        System.out.println(memur2.id + " " + memur2.maas);  
    }  
}
```

```
public class Isci extends Muhasebe{  
  
    public static void main(String[] args) {  
  
        Isci isci1=new Isci();  
        isci1.isim="Mehmet";  
        isci1.soyisim="Bulutluoz";  
        isci1.tel="5551234567";  
        isci1.saatUcreti=10;  
        isci1.maas=isci1.maasHesapla();  
        isci1.id=isci1.idAtama();  
  
        Isci isci2=new Isci();  
        isci2.isim="Ayse";  
        isci2.soyisim="Bulut";  
        isci2.tel="5557654321";  
        isci2.saatUcreti=15;  
        isci2.maas=isci2.maasHesapla();  
        isci2.id=isci2.idAtama();  
  
        System.out.println(isci1.id + " " + isci1.maas);  
        System.out.println(isci2.id + " " + isci2.maas);  
    }  
}
```

Child  
Class  
(Sub)

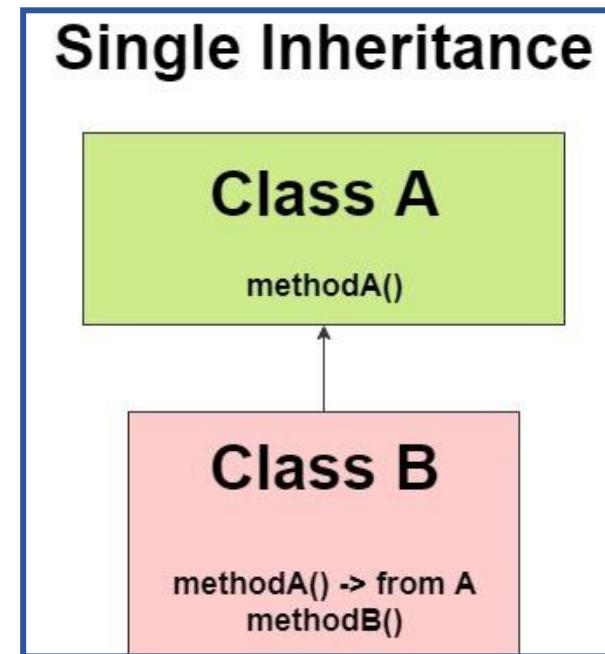
Child  
Class  
(Sub)



# Inheritance Türleri

## Single Inheritance

Java Single Inheritance kabul eder. Bir child class'in sadece bir tane parent class'i olabilir .



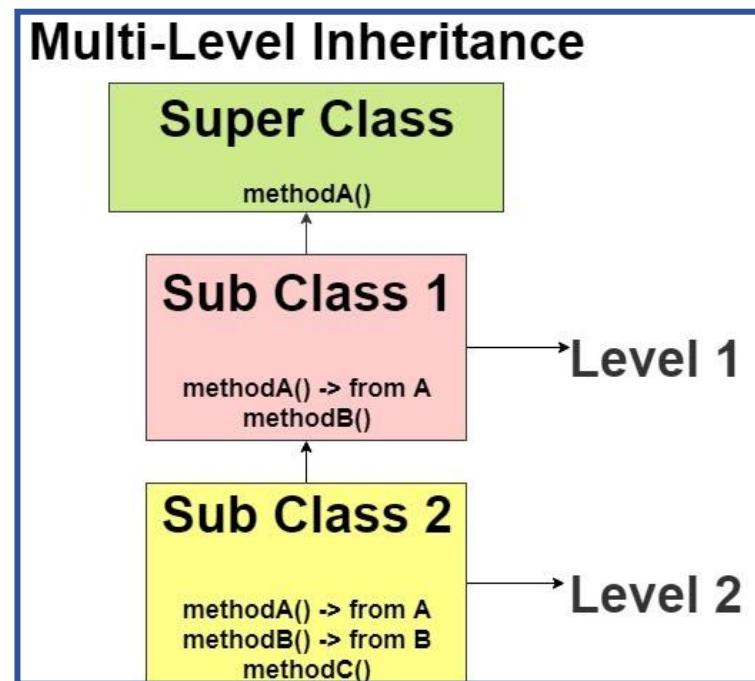
Bir cocugun ailesi bir tane olur



# Inheritance Türleri

## Multilevel Inheritance

Java Inheritance zincirini kabul eder. Bir child class'in sadece bir tane parent class'i olabilir (ve onun parent class zinciri).



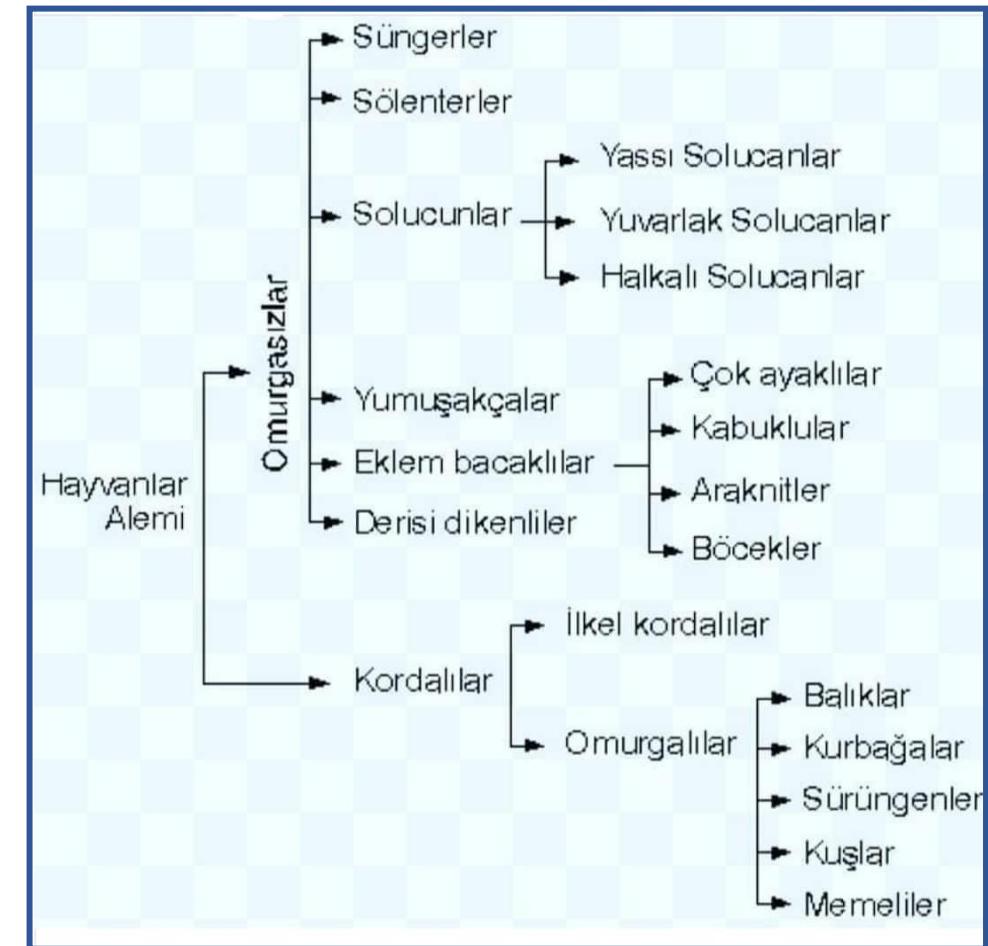
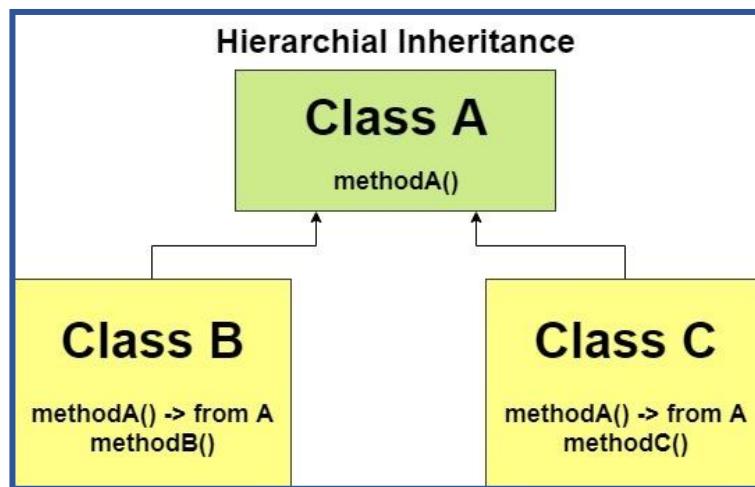
İnsanlardaki soy ağacı gibi, child class'in parent'i ve grand parent'leri olabilir.



# Inheritance Türleri

## Hierarchical Inheritance

Birden fazla class aynı class'i parent olarak kullanabilir.

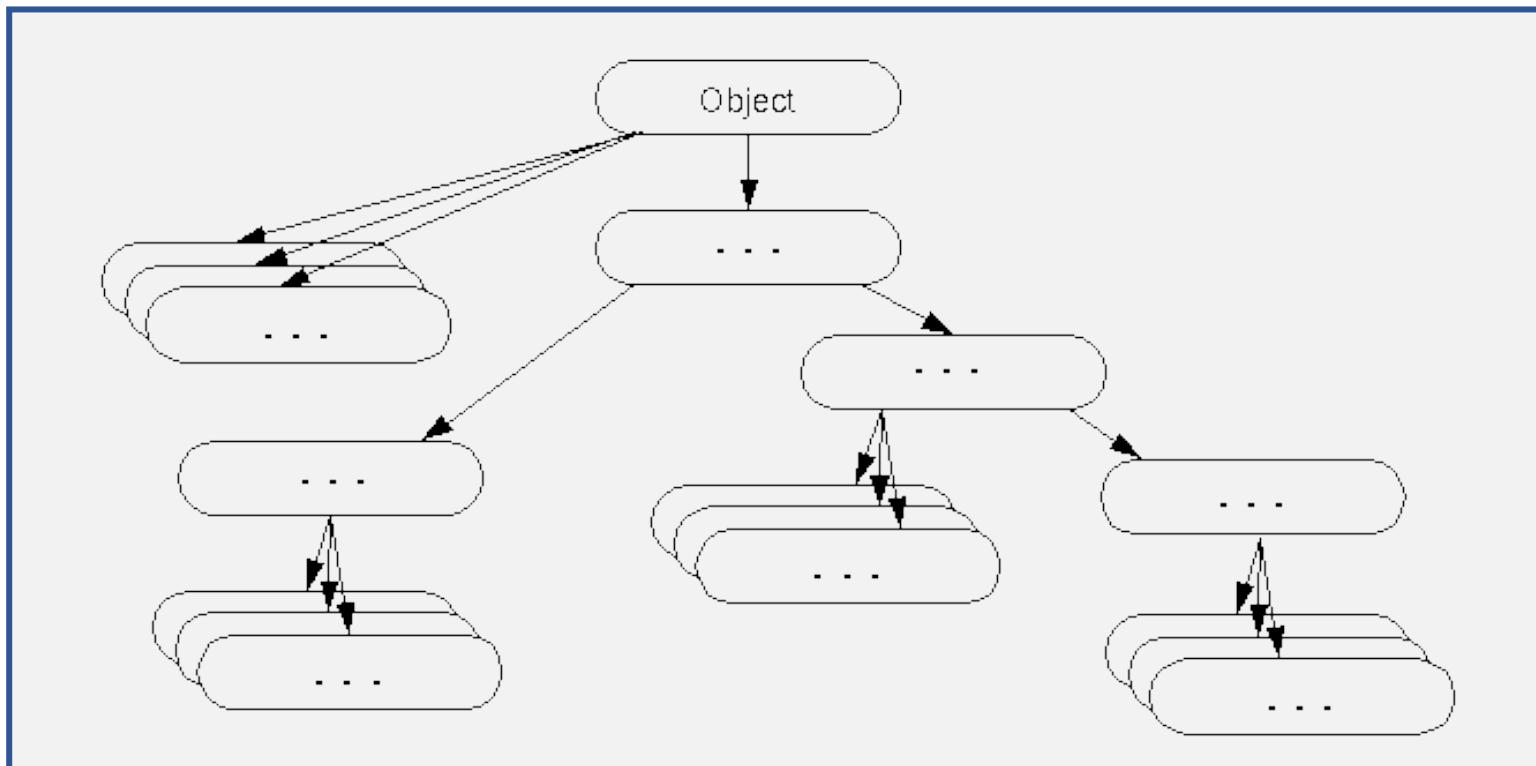




## Inheritance Türleri

Java'da, bütün class'lar **Object Class**'dan inherit ederler.

**Object Class** bütün class'ların parent'idir ve **Object Class** parent'i olmayan tek class'dır.

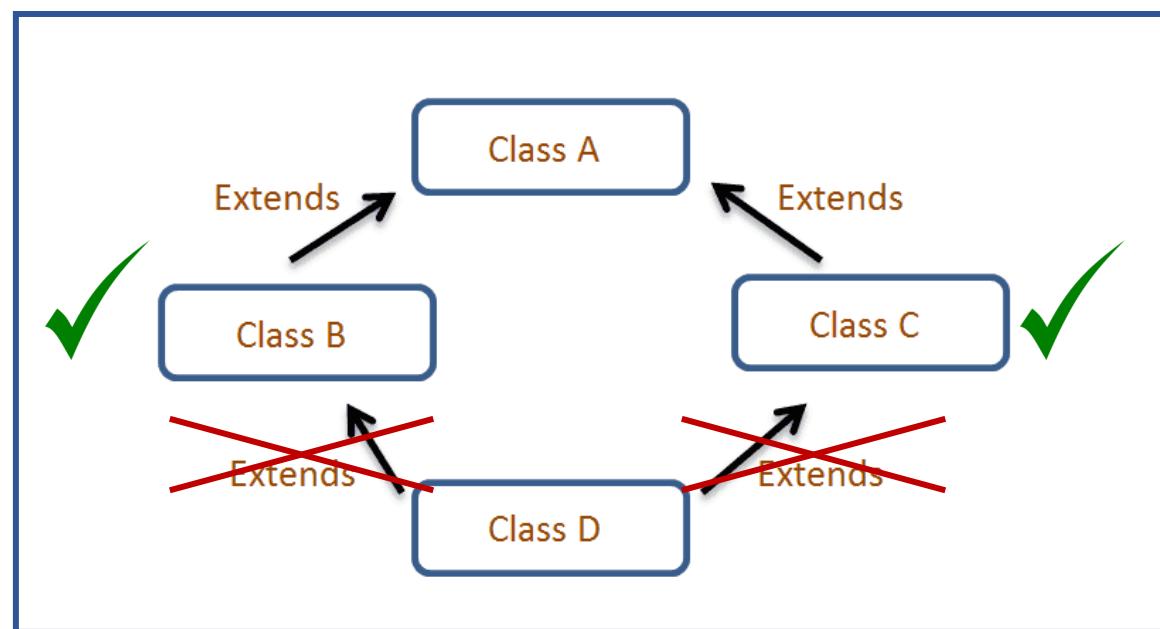




# Inheritance Türleri

## Multiple Inheritance

Bir class'in birden fazla class parent olarak kabul etmesi demektir,  
ancak Java multiple inheritance **KABUL ETMEZ**

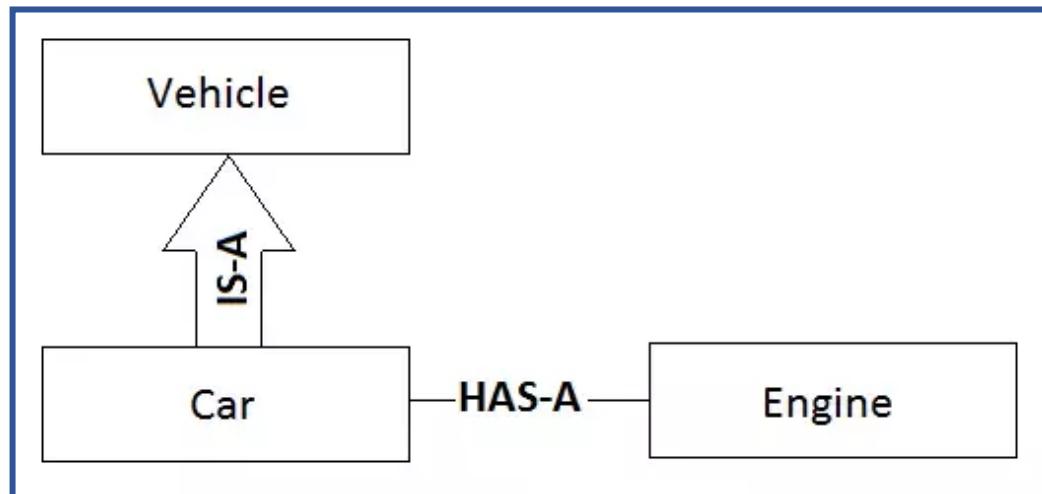




## IS-A () & HAS-A () Relationship

IS-A ilişkisini kolayca tanımlayabileceğinizi unutmamak önemli bir noktadır. Bir extends anahtar sözcüğünü gördüğünüz her yerde, bu sınıfın IS-A ilişkisine sahip olduğu söylenebilir.

( BMW IS-A Car, Car IS-A Vehicle vb..)



HAS-A ilişkisi Java'da kodun yeniden kullanılabilirliği için kullanılır.

Java'da Has-A ilişkisi, basitçe, bir sınıfın bir örneğinin başka bir sınıfın bir örneğine veya aynı sınıfın başka bir örneğine başvurusu olduğu anlamına gelir.

(Apartman HAS-A daire, daire HAS-A mutfak vb..)



# Inheritance'da Constructor Çağırma

- 1) Bir class'da constructor çalıştırıldığımızda önce parent class'daki constructor çalışır. Çünkü her constructor'in ilk satırında super() keyword vardır(görünmese bile).

```
public class Personel {  
    Personel(){  
        System.out.println("Personel constructor calisti");  
    }  
}
```

```
public class Muhasebe extends Personel{  
    Muhasebe(){  
        System.out.println("Muhasebe constructor calisti");  
    }  
}
```

extends

extends

```
public class Isci extends Muhasebe{  
    Isci(){  
        System.out.println("Isci constructor'i calisti");  
    }  
  
    public static void main(String[] args) {  
        Isci isci1=new Isci();  
    }  
}
```

output

```
Personel constructor calisti  
Muhasebe constructor calisti  
Isci constructor'i calisti
```



# Inheritance'da Constructor Çağırma

Aşağıdaki 3 class birbiriyile aynıdır.

```
public class Zebra extends Hayvanlar {  
}
```

```
public class Zebra extends Hayvanlar{  
    Public Zebra(){  
    }  
}
```

```
public class Zebra extends Hayvanlar {  
    Public Zebra(){  
        super();  
    }  
}
```



# Inheritance'da Constructor Çağırma

- 2) Eğer parent (super) class'da super() ile çağrıdığınız constructor yoksa Java Compile time Error verir.

Ornek 1:

```
public class Muhasebe extends Personel{  
    Muhasebe(){  
        System.out.println("Muhasebe costructor calisti");  
    }  
}
```

extends  
↑

```
2  
3 public class Isci extends Muhasebe{  
4     Isci(){  
5         super(5);  
6         System.out.println("Isci constructor'i calisti");  
7     }  
8  
9     public static void main(String[] args) {  
10  
11         Isci isci1=new Isci();  
12  
13     }  
14  
15 }
```



# Inheritance'da Constructor Çağırma

Ornek 2:

```
public class Muhasebe extends Personel{  
      
    Muhasebe(String isim){  
          
    }  
}
```

extends

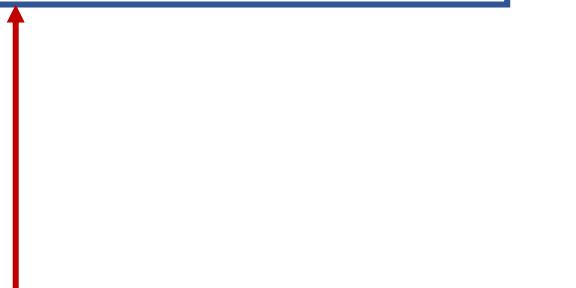
```
3 public class Isci extends Muhasebe{  
x 4     Isci(){  
5           
6             System.out.println("Isci constructor'i calisti");  
7     }  
8  
9     public static void main(String[] args) {  
10  
11         Isci isci1=new Isci();  
12     }  
13  
14 }  
15 }
```



# Inheritance'da Constructor Çağırma

- 2) super(); parent class'dan constructor çağirmak için, this(); içinde olunan class'da başka bir constructor çağirmak için kullanılır.

```
public class Muhasebe extends Personel{  
  
    Muhasebe(String isim){  
        System.out.println("Parametreli muhasebe constructor'i calisti");  
    }  
  
    Muhasebe(){  
        this("a");  
        System.out.println("Parametresiz muhasebe constructor'i calisti");  
    }  
}
```



```
public class Isci extends Muhasebe{  
    Isci(){  
        System.out.println("Isci constructor'i calisti");  
    }  
  
    public static void main(String[] args) {  
  
        Isci isci1=new Isci();  
  
    }  
}
```



# Inheritance'da Constructor Çağırma

Output nedir?

```
public class Okul {  
    public Okul() {  
        System.out.println("Parent class cons.");  
    }  
  
}
```

```
class Sinif extends Okul {  
    public Sinif(int age) {  
        super();  
        System.out.println("child class parametreli cons.");  
    }  
    public Sinif() {  
        this(11);  
        System.out.println("child class parametresiz cons.");  
    }  
  
    public static void main(String[] args) {  
        Sinif sinif1=new Sinif();  
    }  
}
```



# Inheritance'da Constructor Çağırma

1) Aşağıdaki programdaki CTE'ler nasıl düzelttilir ve düzelttilip çalıştırıldığında konsolda ne yazdırır?

```
6 class Derived {  
7     public Derived(String temp) {  
8         System.out.println("Derived class " + temp);  
9     }  
10  
11    public class Test01 extends Derived {  
12        public Test01 (String temp) {  
13            System.out.println("Test class " + temp);  
14        }  
15    }  
16    public static void main(String[] args) {  
17        Test01 obj = new Test01();  
18    }  
19}  
20}  
21
```



## Inheritance'da Constructor Çağırma

2) Aşağıdaki programdaki CTE'ler nasıl düzelttilir ve düzelttilip çalıştırıldığında konsolda ne yazdırır?

```
class Derived {  
    public Derived(String temp) {  
        System.out.println("Derived class " + temp);  
    }  
  
    public class Test01 extends Derived {  
        public Test01 (String temp) {  
            super("Hoscakal");  
            System.out.println("Test class " + temp);  
        }  
        public static void main(String[] args) {  
            Test01 obj = new Test01("Merhaba");  
        }  
    }  
}
```

```
Derived class Hoscakal  
Test class Merhaba
```



## Inheritance'da Class Üyelerini Çağırma

- "super." keyword parent class'dan variable çağırma için kullanılır. "this." keyword içinde bulunan class'dan variable çağırma için kullanılır.
- Esasında "this" keyword parent class'dan variable çağırma için de kullanılabilir; fakat tavsiye edilmez. Cunku, child ve parent class'larda aynı isimli iki variable varsa, "this" parent class'dan variable çağrıramaz.

- super() ve this() constructor çağırma için kullanılırlar ve constructor'in ilk satırında olmalıdır. Bu durumda bir constructor'da ikisinin birden olması mümkün değildir.
- super. ve this. variable çağırma için kullanılırlar. İlk satırda olma şartı olmadığı için ikisi birlikte kullanılabılırler.



# Inheritance'da Class Uyelerini Çağırma

```
class Class2 {  
    protected int num1=10;  
    protected int num2=11;  
    protected String name="Ali Can";  
    protected String name2="Veli Cem";  
  
    Class2(){  
  
        System.out.println("Parent Class constructor calisti");  
    }  
}
```

extends

```
public class Deneme extends Class2{  
    int num1=20;  
    int num3=21;  
    String name2="Hakan San";  
    String name3="Kemal";  
    Deneme(){  
        System.out.println("Child Constructor calisti");  
  
        System.out.println(this.num1); → 20  
        System.out.println(super.num1); → 10  
        System.out.println(this.num2); → 11  
        System.out.println(super.num2); → 11  
        System.out.println(this.num3); → 21  
        // System.out.println(super.num3);  
        super.name1="Hatice Sen";  
        System.out.println(this.name1); → Hatice Sen  
        System.out.println(super.name1); → Hatice Sen  
        this.name2="Kadir Naz";  
        System.out.println(this.name2); → Kadir Naz  
        System.out.println(super.name2); → Veli Cem  
        System.out.println(this.name3); → Kemal  
        //System.out.println(super.name3);  
    }  
  
    public static void main(String[] args) {  
        Deneme deneme=new Deneme();  
    } }
```

Parent Class constructor calisti

Child Constructor calisti

20

10

11

11

21

Hatice Sen

Hatice Sen

Kadir Naz

Veli Cem

Kemal

outputs



# Inheritance'da Class Üyelerini Çağırma

```
public class Zebra extends Animal {  
    public Zebra() {  
        System.out.println("Child cons. runs at the end");  
        super();  
    }  
}
```

Does not compile

```
public class Zebra extends Animal { public Zebra() {  
    super();  
    System.out.println("Child cons. runs at the end");  
}
```

compile



# Inheritance'da Class Üyelerini Çağırma

Output nedir?

```
class Okul {  
    public void getDetails() {  
        System.out.println("Derived class ");  
    }  
}  
  
public class Test03 extends Okul {  
    public Test03() {  
        System.out.println("Test class ");  
        super.getDetails();  
    }  
  
    public static void main(String[] args) {  
        Test03 obj = new Test03();  
        obj.getDetails();  
    }  
}
```

Test class  
Derived class  
Derived class



# Inheritance'da Data Type Kullanimi

```
public class Personel {  
    public String isim;  
    public String soyisim;  
    public String statu;  
}
```

↑ extends

```
public class Isci extends Personel{  
    String bolum;  
    int isBasYili;  
  
    public static void main(String[] args) {  
    }  
}
```

↑ extends

```
public class UstaBasi extends Isci {  
    String sorumluOlduguBirim;  
    int sorumluOlduguIsciSayisi;  
  
    public static void main(String[] args) {  
    }  
}
```

UstaBasi Class'inda 3 data turu ile Usta Basi objesi olusturulabilir

```
public static void main(String[] args) {  
    UstaBasi ub1 = new UstaBasi();  
    ub1.sorumluOlduguBirim="tamirhane"; // Ustabasidan  
    ub1.bolum="Tamirhane"; // Isciden  
    ub1.isim="Mehmet"; // Personelden  
  
    Isci ub2=new UstaBasi();  
    ub2.bolum="Atolye"; //Isciden  
    ub2.statu="Iisci"; //Personelden  
  
    Personel ub3=new UstaBasi();  
    ub3.soyisim="Bulut"; // Personelden  
}
```

Bir obje olustururken data turunu parent(lar)'dan secebiliriz

**Avantaj :** Daha genis tanimlama yapilabilir

**Dezavantaj :** o class ve parent class'lara ait olan variable'lar kullanilabilir.

Ayni isimde **iki method varsa** Data Turu'ne bakilir.



# Tekrar Sorulari

1- Inheritance'in avantajlari nelerdir ?

- A) Reusability B) Maintenance C) Less Code

2- Bir Class'a Parent Class olusturmak icin Syntax nedir?

public class ChildClassIsmi **extends** ParentClassIsmi

3- Hangi access modifier'lar inherit edilebilir ?

**public** ve **protected** olanlar her yerden, **default** olanlar ayni paketten inherit edilebilir.

4- super() ile this()'in farki nedir?

super() parent class'dan, this() ise içinde bulunan class'dan constructor cagirmak icin kullanilir

5- super() ile super.'nin farki nedir?

super() parent class'dan constructor, super. ise variable veya method cagirmak icin kullanilir

6- this() ile this.'nin farki nedir?

this() constructor, this. ise class variable veya method'u cagirmak icin kullanilir



## Tekrar Sorulari

7- super. ile this.'nin farki nedir?

**super** parent class'dan variable veya method cagirmak icin kullanilir, this ise icinde bulunulan class'da class level variable veya method'lari cagirmak icin kullanilir.

**this** ile parent class'dan da variable veya method cagrılabilir ancak ayni isimde bir variable/method hem icinde bulunulan class'da hem de parent class'da olursa this parent class'da olani degil icinde bulunulan class'dakini cagirir.

Emin olmak icin parent class icin super kullaniriz.

8- super() ve this() bulunduklari constructor'da ilk sirada olmalıdır. **True** / ~~False~~

9- super() ve this() bir constructor'da sadece 1 kere kullanilabilir. **True** / ~~False~~

10- super() ve this() birlikte ayni constructor'da kullanilabilir. **True** / ~~False~~



# Overriding

Aynı isimde farklı iki method oluşturmanın iki yolu vardır

**1- Method Signature'ini degistirerek aynı isimde farklı iki method yapmak**

**Overloading**

**2- Method Signature'ini degistirmeden iki method'dan sadece birinin çalışmasını sağlamak**

**Overriding**

**NOT 1:** Method Signature'ini degistirmezsek Java her iki method'u aynı method olarak görür ve bir class içerisinde aynı method'u iki kez oluşturmaya İZİN VERMEZ.

Biz parent ve child class'da signature'l aynı olan iki method oluşturursak Java ikisinden sadece birini çalıştırır

**NOT :** Her iki yöntemde dikkat edilirse Method Body'nin değişmesi şart değildir.

Ancak 2.yöntemde signature zaten değişmediği için, Body değişmezse 2.method farklı bir method olmaz.



# Overriding

## Method Signature

Method signature, method ismi ve parametrelerden olusur.

Signature'i degistirmek icin bilesenlerinden isim veya parametrelerle ilgili degisiklikler yapilmalidir.

- 1) Isim ayni kalsa da parametre sayisi degistirildiginde signature degisir

```
public void toplama(int a, int b) {  
    System.out.println(a+b);  
}
```

```
public void toplama(int a, int b,int c) {  
    System.out.println(a+b+c);  
}
```

```
public void toplama(int a, int b,int c,int d) {  
    System.out.println(a+b+c+d);  
}
```

- 2) Farkli data turlerine sahip parametrelerin yerleri degistirildiginde **method signature** degisir.

```
public void toplama(int a, String b, boolean c) {  
    System.out.println("Merhaba");  
}
```

```
public void toplama(int a, boolean c,String b) {  
    System.out.println("Hosgeldin");  
}
```

```
public void toplama(String b, int a, boolean c) {  
    System.out.println("Hoscakal");  
}
```



# Overriding

**Method Overriding nedir ?**

Parent class'da varolan bir methodu **method signature'ini degistirmeden**, method body'sini degistirerek kullanmaya **Method Overriding** denir.

**Method Overriding nisin kullanilir ?**

Overriding kullanarak, child class'in parent class'daki methodu **kendine uyarlayarak (implement)** kullanmasini saglamis oluruz.

Overriding yapildiginda parent class'daki methoda **Overridden Method**, child class'daki methoda **Overriding Method** denir.

Eclipse menu'den Source sekmesinde bulunan **Override/Implement methods** secenegiyle otomatik olarak overriding method'u olusturabiliriz. Bu sekilde yapılan islemde Java @Override annotation'i kullanir.

@Override kullanmak zorunda degiliz, istersek silebiliriz. Ancak kodun anlasilabilir ve okunabilir olmasi icin degil, overridden method'da degisiklik yapildiginda Java'nin rapor etmesi icin kullanilmasi tercih edilir.



# Overriding

```
public class Isci extends Personel{  
  
    public static void main(String[] args) {  
  
        Isci isci=new Isci();  
        isci.isim="Mehmet";  
        isci.soyisim="Bulut";  
        isci.status="isci";  
        System.out.println(isci.isim + " "+isci.soyisim+" "+  
        isci.status+ " "+isci.maasHesapla());  
  
    }  
  
    public int maasHesapla() {  
  
        return (30*8*15) ;  
    }  
  
    public void calismaSaati() {  
        System.out.println("Isciler gunluk 8 saat calisir");  
    }  
}
```

Overridden Methods

```
public class UstaBasi extends Isci {  
  
    public static void main(String[] args) {  
  
        UstaBasi ub1 = new UstaBasi();  
        ub1.isim = "Seher";  
        ub1.soyisim = "Boss";  
        ub1.status = "Usta Basi";  
        System.out.println(ub1.isim + " "+ub1.soyisim+" "+  
        ub1.status+ " "+ ub1.maasHesapla());  
        ub1.calismaSaati();  
    }  
  
    public int maasHesapla() {  
  
        return (30 * 8 * 20);  
    }  
  
    public void calismaSaati() {  
        System.out.println("Ustabasi is bitene kadar calisir");  
    }  
}
```

```
public class GeciciIsci extends Isci {  
  
    public static void main(String[] args) {  
  
        GeciciIsci gi1 = new GeciciIsci();  
        gi1.isim = "Faruk";  
        gi1.soyisim = "Yanik";  
        gi1.status = "GeciciIisci";  
        System.out.println(gi1.isim + " "+gi1.soyisim+" "+  
        gi1.status+ " "+ gi1.maasHesapla());  
        gi1.calismaSaati();  
  
    }  
  
    public int maasHesapla() {  
  
        return (25 * 8 * 10);  
    }  
  
    public void calismaSaati() {  
        System.out.println("Gecici isciler haftalık 25 saat calisir");  
    }  
}
```

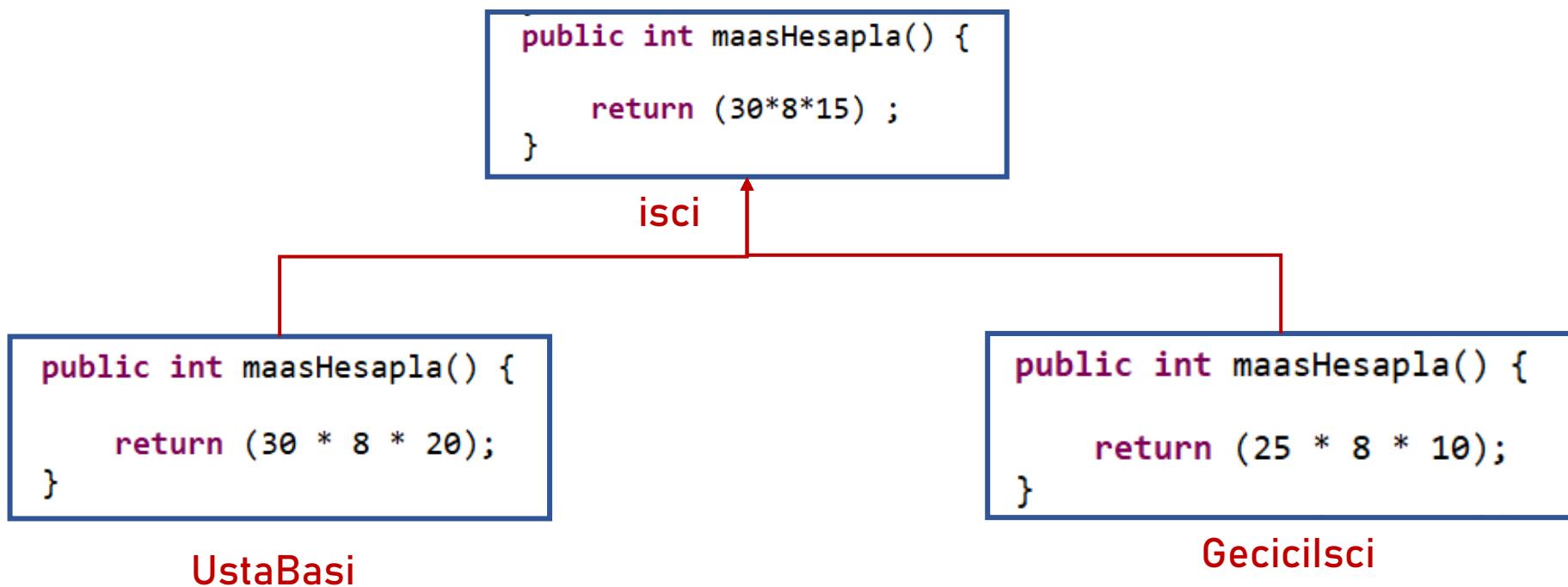
Overriding  
Methods



# Method Overiding'i Nicin Kullaniriz ?

Overriding parent class'daki genel method'u degistirmeden child class'in kendine uygun method uretmesini saglar

Ornegimizdeisci maasi hesaplanirken genel bir formul varken, child class olan Ustabasi ve Gecicilsci Class'lari kendilerine uygun maas hesaplama method'larina sahiptirler.



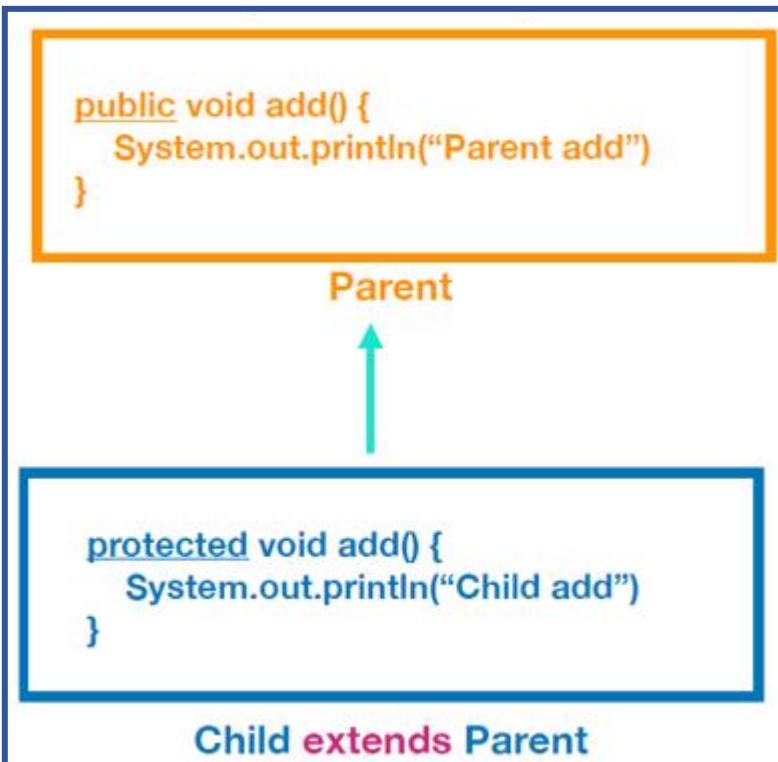


# Overriding Kurallari

- 1) Method Signature'i (*isim ve parametreler*) ayni olmalıdır.
- 2) Child class'daki method'un (overriding method) Access Modifier'i parent class'daki method'un (overridden) modifier'ından daha dar olamaz.
- 3) Overriding method **covariant** return type kullanmalıdır.
- 4) private, static and final method'lar overriding yapılamazlar  
**5 ve 6 sonra açıklanacak**
- 5) Child class'daki method (overriding method), parent class'daki method'un (overridden method) throw edip etmedigine bakmaksızın **compile time exception throw** edebilir. Ancak parent class'da throw edilen exception'dan daha geniş olamaz
- 6) Eger abstract olmayan bir class **abstract class'a** extend ediyorsa veya **bir interface'i** implement ediyorsa abstract method'ların tamamı override edilmelidir

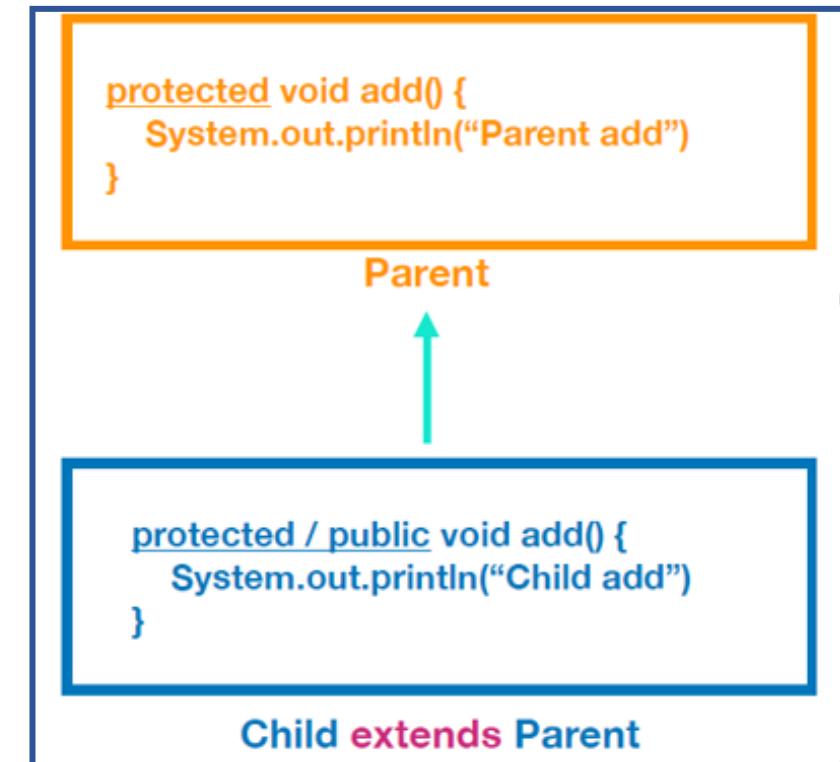


# Overriding Kurallari ? Return Type



Yanlis

Child Parent'i sinirlayamaz



Dogru

Child'in access modifier'i Parent ile  
ayni veya daha genis olmalıdır

**NOT :** Private method'lar override edilemez..



## Overriding Kurallari

Overriden ve overriding method'larin ikisini de kullanmak istersek child class'da (overriding method) **super** keyword'unu kullanabiliriz.

```
public class Lamb extends Animal {  
  
    public void eat(){  
        super.eat();  
        System.out.println("Lambs eat grass");  
    }  
  
    public static void main(String[] args) {  
  
        Lamb lamb = new Lamb();  
        lamb.eat();  
    }  
}
```



# Polymorphism

Polymorphism = Overloading + Overriding

- Poly "çok" morph ise "form", "biçim" anımlarını taşır. Bu ikisinin birleşimiyle oluşan "**polymorphism**" sözcüğü "çok biçimlilik" anlamına gelir.
- Özetle, oluşturulan nesnelerin gerekiğinde kılıktan kılığa girip başka bir nesneymiş gibi davranışabilmesine polymorphism diyebiliriz. Bunlar program kodlarının yeniden kullanılabilmesi veya var olan kodun geliştirilebilmesi açısından çok önemlidir.



## Polymorphism Türleri

- Method **Overloading** bir **compile time (static)** polymorphism'dir. Method **Overloading** sayesinde aynı isme, aynı body'e, farklı parametrelere sahip bir çok method üretip kullanabiliriz.
  
- Method **Overriding** bir **run time (dynamic)** polymorphism'dir. Method **Overriding** sayesinde aynı isme, aynı parametrelere'e, farklı body'e sahip bir çok method üretip kullanabiliriz.



# Overloading vs Overriding

- 1) Overloading'de method signature degisir, Overriding'de degismez.
- 2) Overloading'de body istenirse degistirilebilir, Overriding'de body %99 degistirilir.
- 3) final, static ve private methodlar Overload edilebilir, ama Override edilemezler.
- 4) Overloading Compile Time Polymorphism (static)'dir, Overriding is Run Time Polymorphism'(dynamic)'dir.
- 5) Overloading'de inheritance gerekmez, Overriding'de gerekir.
- 6) Overloading'de istedigimiz sekilde access modifier ve return type kullanabiliriz ama Overriding'de access modifier ve return type kullanma belli kurallara baglidir.



# Polymorphism

1) "method signature" nedir? Hangi method'lar Java'ya gore aynidir ?

Method signature "method ismi" ve "parameter listesi"nden olusur.

Signature'l ayni olan method'lar Java'ya gore ayni method'dur.

2) Polymorphism nedir?

Polymorphism "overloading" ve "overriding"in birlesimidir.

3) "Overloading" ve "Overriding"in farki nedir ?

Overloading'de sadece parametreler degisir, overriding'de signature'a dokunulmaz sadece body degisir.

4) "Overriding"in faydasi nedir?

Coklu uygulama, reusability



# Polymorphism

1) Asagidaki programdaki CTE nasıl giderilebilir ? Program düzelttilip calistirildiginda konsolda ne yazdirir?

```
3 class ParentClass {  
4     public void getDetails(String temp) {  
5         System.out.println("Derived class " + temp);  
6     }  
7 }  
8  
9 public class Test01 extends ParentClass {  
10  
11     public int getDetails(String temp) {  
12         System.out.println("Test class " + temp);  
13         return 0;  
14     }  
15  
16     public static void main(String[] args) {  
17         Test01 obj = new Test01();  
18         obj.getDetails("GFG");  
19     }  
20 }
```

- a) Derived class GFG
- b) Test class GFG
- c) Compilation error
- d) Runtime error



# Polymorphism

2) Asagidaki programdaki CTE nasil giderilebilir ? Program düzelttilip calistirildiginda konsolda ne yazdirir?

```
3 class Derived {  
4     public void getDetails() {  
5         System.out.println("Derived class");  
6     }  
7 }  
8  
9 public class Test02 extends Derived {  
10    protected void getDetails() {  
11        System.out.println("Test class");  
12    }  
13  
14    public static void main(String[] args) {  
15        Derived obj = new Test02();  
16        obj.getDetails();  
17    }  
18 }
```

- a) Test class
- b) Compilation error due to line xyz
- c) Derived class
- d) Compilation error due to access modifier



# Polymorphism

2) Asagidaki programdaki CTE nasil giderilebilir ? Program düzelttilip calistirildiginda konsolda ne yazdirir?

Cevap :

```
3 class Derived {  
4     public void getDetails() {  
5         System.out.println("Derived class");  
6     }  
7 }  
8  
9 public class Test02 extends Derived {  
10    public void getDetails() {  
11        System.out.println("Test class");  
12    }  
13  
14    public static void main(String[] args) {  
15        Derived obj = new Test02();  
16        obj.getDetails();  
17    }  
18 }
```

- a) Test class
- b) Compilation error due to line xyz
- c) Derived class
- d) Compilation error due to access modifier



# Polymorphism

3) Asagidaki program calistirildiginda konsolda ne yazdirir?

```
class Derived03 {  
    public void getDetails() {  
        System.out.printf("Derived class ");  
    }  
  
}  
  
public class Test03 extends Derived03 {  
    public void getDetails() {  
        System.out.printf("Test class ");  
        super.getDetails();  
    }  
  
    public static void main(String[] args) {  
        Derived03 obj = new Test03();  
        obj.getDetails();  
    }  
}
```

- a) Test class Derived class
- b) Derived class Test class
- c) Compilation error
- d) Runtime error



# Polymorphism

4) Asagidaki programdaki CTE nasıl giderilebilir ? Program düzelttilip calistirildiginda konsolda ne yazdirir?

```
3 class Derived04 {  
4     protected final void getDetails() {  
5         System.out.println("Derived class");  
6     }  
7 }  
8  
9 public class Test04 extends Derived04 {  
10    protected final void getDetails() {  
11        System.out.println("Test Class");  
12    }  
13  
14    public static void main(String[] args) {  
15        Derived04 obj = new Derived04();  
16        obj.getDetails();  
17    }  
18 }
```

- a) Derived class
- b) Test class
- c) Runtime error
- d) Compilation error



# Polymorphism

4) Asagidaki programdaki CTE nasil giderilebilir ? Program düzelttilip calistirildiginda konsolda ne yazdirir?

Cevap :

```
3 class Derived04 {  
4     protected void getDetails() {  
5         System.out.println("Derived class");  
6     }  
7 }  
8  
9 public class Test04 extends Derived04 {  
10    protected final void getDetails() {  
11        System.out.println("Test Class");  
12    }  
13  
14    public static void main(String[] args) {  
15        Derived04 obj = new Derived04();  
16        obj.getDetails();  
17    }  
18 }
```

- a) Derived class
- b) Test class
- c) Runtime error
- d) Compilation error



# Polymorphism

5) Asagidaki program calistirildiginda konsolda ne yazdirir?

```
class Person {
    public void talk() {
        System.out.println("First Program");
    }
}

class Student extends Person {
    public void talk() {
        System.out.println("Second Program");
    }
}

public class Test05 {

    public static void main(String[] args) {
        Person p = new Student();
        p.talk();
    }
}
```



# Polymorphism

6) Asagidaki program calistirildiginda konsolda ne yazdirir?

```
public class Test06 {
    public static void main(String[] args) {
        new C().create();
        new D().update();
        new R().read();
        new D().delete();
    }
    class C {
        public void create() { System.out.print("c"); }
    }
    class U {
        private void update() { System.out.print("u"); }
    }
    class R extends C {
        public void create() { System.out.print("C"); }
        protected void read() { System.out.println("R"); }
    }
    class D extends U {
        void update() { System.out.println("U"); }
        void delete() { System.out.println("D"); }
    }
}
```



# Polymorphism

7) Aşağıdaki program çalıştırıldığında konsolda ne yazdırır?

```
class Super {  
    public Integer getLength() {  
        return new Integer(4);  
    }  
}  
  
public class Test07 extends Super {  
    public Integer getLength() {  
        return (5);  
    }  
  
    public static void main(String[] args) {  
        Super sooper = new Super();  
        Test07 sub = new Test07();  
        System.out.println(sooper.getLength().toString() + ", " + sub.getLength().toString());  
    }  
}
```



# Polymorphism

8) Asagidaki program calistirildiginda konsolda ne yazdirir?

```
public class Test08 {

    public static void main(String[] args) {
        X x = new X();
        Y y = new Y();
        y.m2();
        x.m1();
        y.m1();
        x=y;
        x.m1();
    }

    class X{
        public void m1() {
            System.out.println("m1, X class");
        }
    }
    class Y extends X{
        public void m1() {
            System.out.println("m1, Y class");
        }
        public void m2() {
            System.out.println("m2, Y class");
        }
    }
}
```



# Polymorphism

9) Asagidaki program calistirildiginda konsolda ne yazdirir?

What will be the output of the following program?

```
public class Outer {
    public static void main(String args[]) {
        Computer mouse = new Laptop();
        System.out.println(mouse.getValue(100, 200));
    }
}
class NoteBook {
    int getValue(int a, int b) {
        if (a > b)
            return a;
        else
            return b;
    }
}
class Computer extends NoteBook {
    int getValue(int a, int b) {
        return a * b;
    }
}
class Laptop extends Computer {
    int getValue(int a, int b) {
        return b - a;
    }
}
```



# Polymorphism

10) Asagidaki program calistirildiginda konsolda ne yazdirir?

What will be the output of the following program?

```
public class Product {
    public static void main(String[] args) {
        M m = new M();      M n = new N();
        M o = new O();      O oo = new O();
        m.product(3);      n.product(3);
        oo.product(3);
    }
    class M {
        int product(int i) {
            int result = i * i;
            System.out.print("(" + i + ", " + result + ")~");
            return result;
        }
    }
    class N extends M {
        int product(int i) {
            int result = i + i;
            System.out.print("[ " + i + ", " + result + "]~");
            return result;
        }
    }
    class O extends M {
        int product(int i) {
            int result = i * 2;
            System.out.print("(" + i + ", " + result + ")~");
            return result;
        }
    }
}
```



# Inheritance'da Data Type Kullanımı

Output nedir?

```
class Person {
    public Person() {
        System.out.println("Person Constructor");
    }

    public void talk() {
        System.out.println("First Program");
    }
}

class Student extends Person {
    public void talk() {
        System.out.println("Second Program");
    }
}

public class Test04 {

    public static void main(String[] args) {
        Person p = new Student();
        p.talk();
    }
}
```

Person Constructor  
Second Program



Exception