

23 EKIM 2021  
DERS 1

**Genel Hatırlatmalar**  
**Java Giriş**

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## 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 Hatırlatmalar

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  
<https://class.techproed.com/>
12. Teknik destek slack @technical support
13. Ders esnasında canlı destek  
Haluk Bilgin, Fatih Kilic, Ebubekir Sahin
14. Customer service +1 917 768 74 66

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

~ CHINESE PROVERB

## 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....

---

# 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

## Ders İsleyisi 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



Google Classroom

3. Google Clasroom

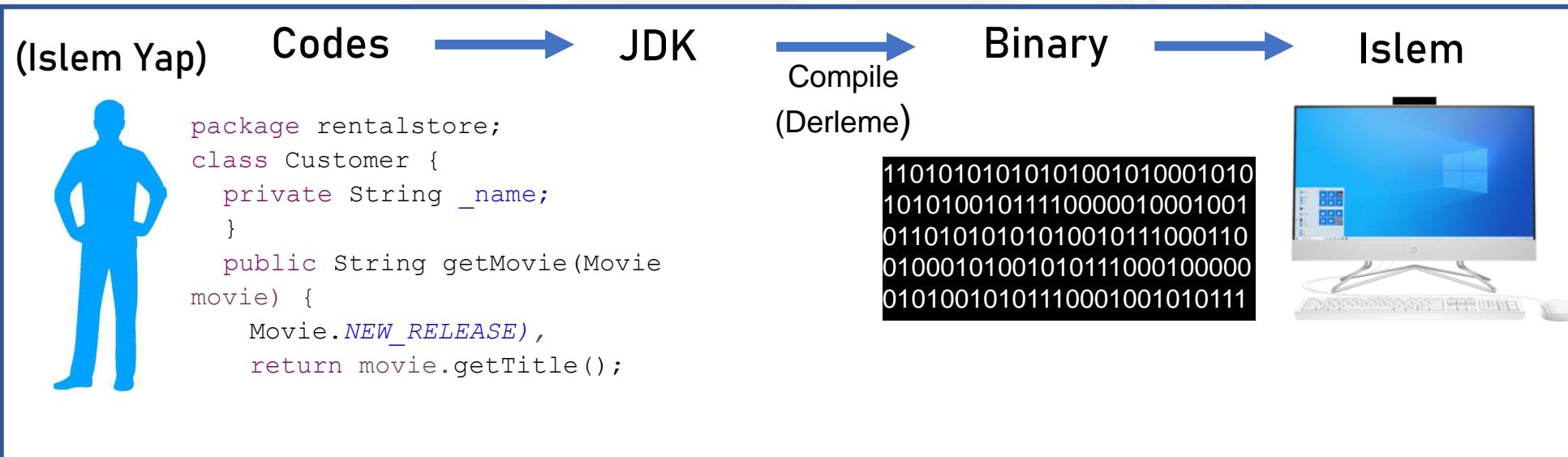
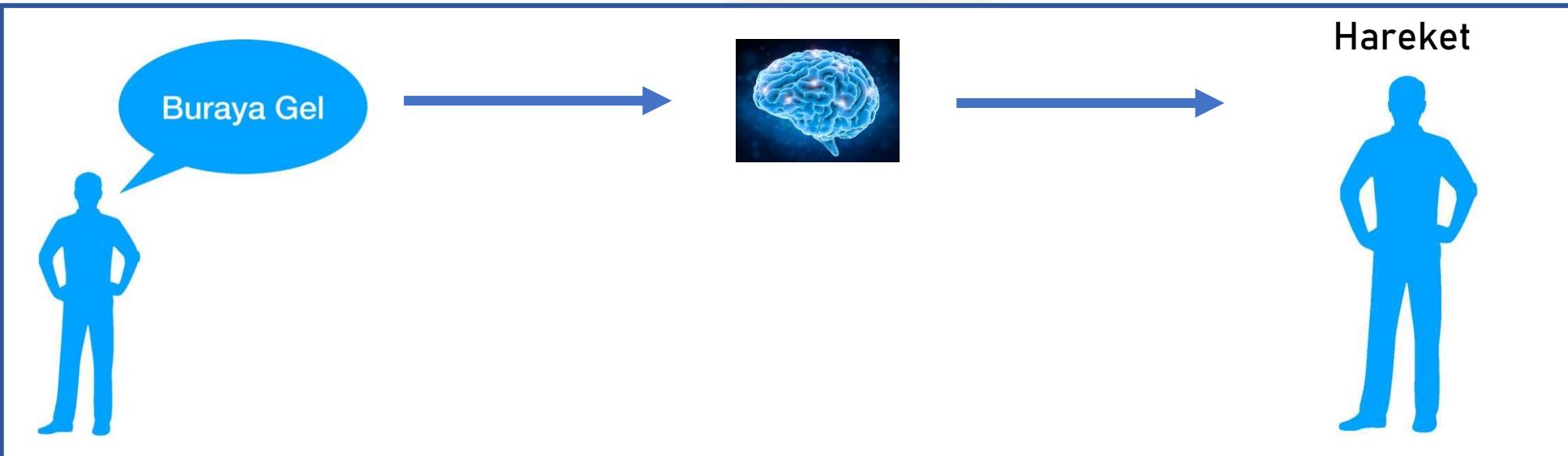
- Tüm ders notları, zoom linki ve videolar Google Classroom'dan paylaşılacak
- Maillerinize davetiye gönderildi

## 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 cozulerek konu tekrarı yapılır



# Programlama Dili Nedir ?



# Nicin Java ?

1- Öğrenmesi kolay

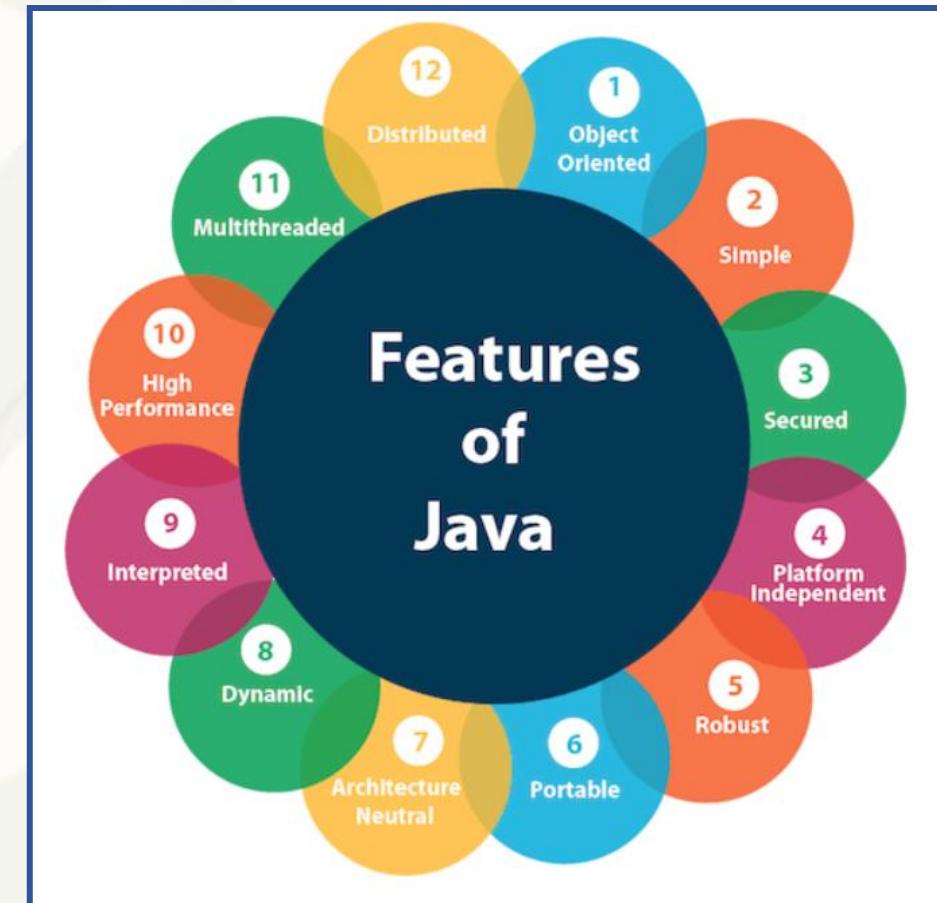
2- Dünyada en çok kullanılan programlama dili

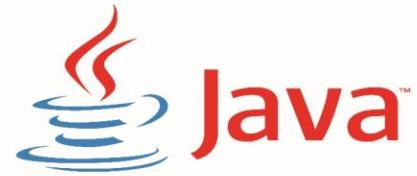
Sun'a göre 3 milyar cihaz Java kullanıyor. Şu anda Java'nın kullanıldığı birçok cihaz var.

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)



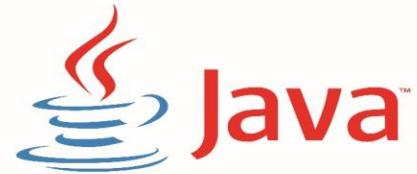
Application (Urun)

1- Feature (Fields veya Variables)

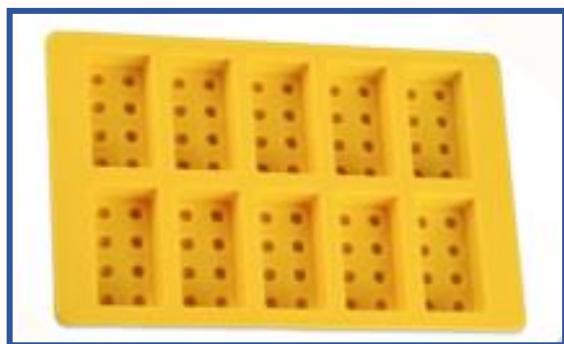
Pasif ozellik (renk,sekil,isim)

2- Functionality (Method)

Aktif ozellik (tasima,degistirme)



## Bir Object Nasil Olusturulur?



Class(Object Kalibi)

Field      Method  
(Variables)    (Functions)



Object

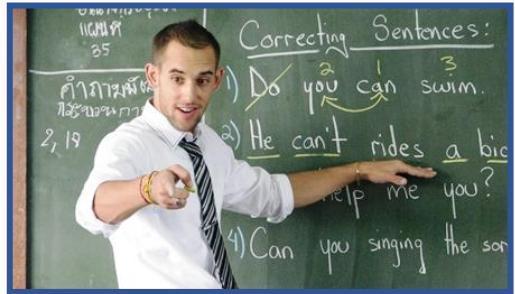
Birden fazla Obje birlestirilir

Application





## Object Nasıl Kullanılır ?



Ogretmen

Dersler

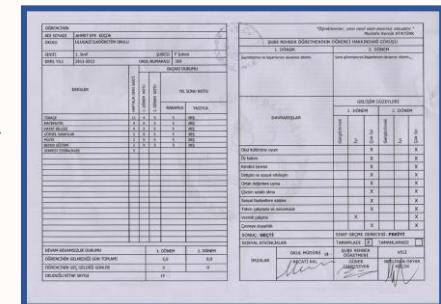
09:00	TÜRKÇE-1
09:30	MATEMATİK-1
10:00	TÜRKÇE-2
10:30	MATEMATİK-2
11:00	TÜRKÇE-3
11:30	MATEMATİK-3
12:00	TÜRKÇE-4
12:30	MATEMATİK-4
13:00	İYEP TÜRKÇE



Personel



Ogrenci



Notlar



## Bir Class Hangi Bolumlerden Olusur?

```
public class C2_MethodCreation2 {
```

1

2

3

2

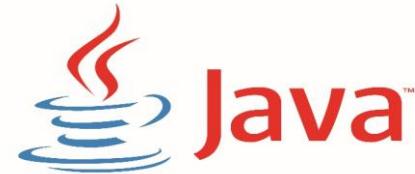
```
}
```

// Class sonu

1 – Class Declaration

2 – Curly braces : Suslu parantez

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



## Bir Class'in Icinde Neler Bulunur?

```
public class C2_MethodCreation2 {
```

```
    private double ortalama;  
    public int sonuc;
```

1

1 – Field / Variables

```
    public static void main(String[] args) {  
        ortalama(85.2 ,90.3); // method call  
    }
```

2

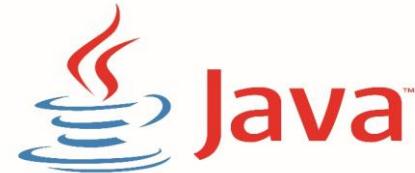
2 – Main Method

```
        public static void ortalama(double sayi1, double sayi2) {  
            System.out.println("girdiginiz iki sayinin ortalamasi : " + (sayi1+sayi2)/2);  
        }
```

3

3 – Method

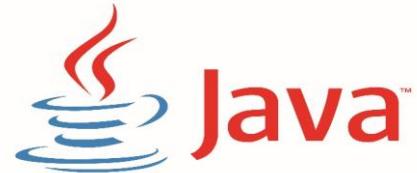
```
} // Class sonu
```



## Class Olustururken (Declaration) Kullanilan Keyword'ler Nelerdir?

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

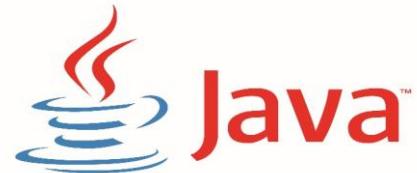
- 1 **public** : Access Modifier (Erisim duzenleyici) : class'a kimlerin erisebilecegini belirler. Public olursa her yerden erisilebilir  
**default** : Sadece bulundugu Package'den kullanilabilir
- 2 **class** : Yazdigimiz kodun class oldugunu belirtir
- 3 **MyFirstClass** : Olusturdugumuz class'in 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 Nelerdir?

```
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 kelimedenden 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



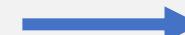
## Main Method

```
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

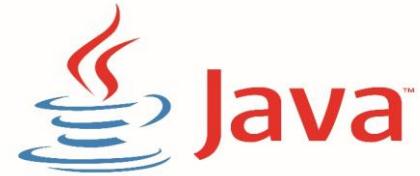


Motor

Java Project



Main Method



## Yorum Cumlesi (Comment) Nasil Eklenir ?

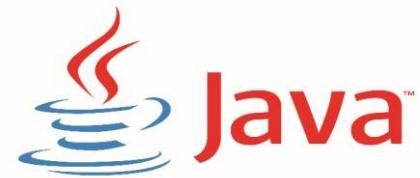
```
public class Example {  
    // Bir satiri comment haline getirmek icin // kullanilir  
    String isim = "Mehmet";  
  
    /*  
     * Eger birden fazla  
     * satiri yorum haline  
     * getirmek istiyorsak  
     * kullanilir  
     */  
    int sayi=10;  
    double not=75.70;  
}  
  
boolean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

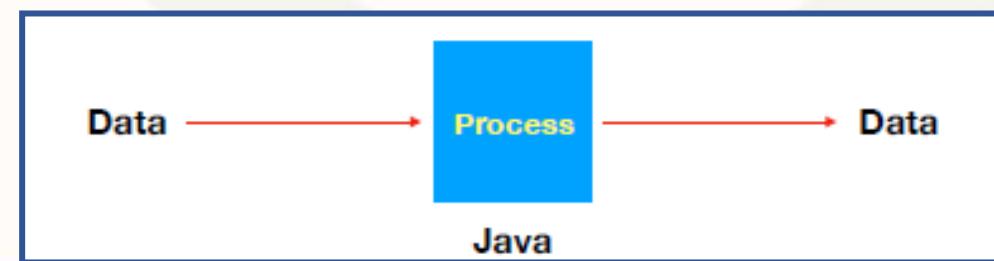


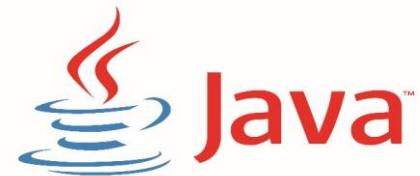
## Data

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

Joe, Smith, 1234 Daire, SLC, UT, 8404,8015553211
0143 0157 0155 0160 0165 0164 0145 0162 0040 0150 0157 0160 0145
01100011011011110110110101110000011101010111010001100101011100100010000001101000000 101

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

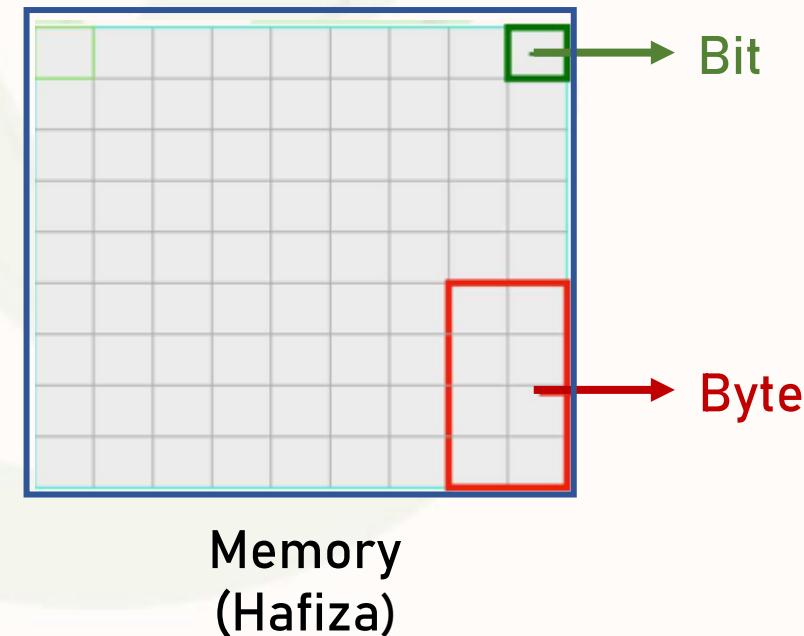
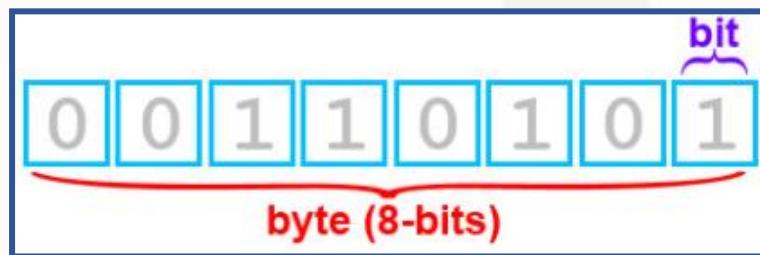


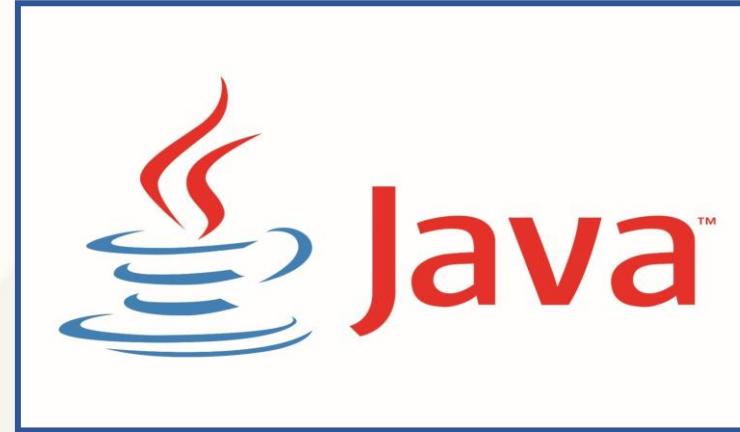


## Bit

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

Note: 8 bit =1 byte

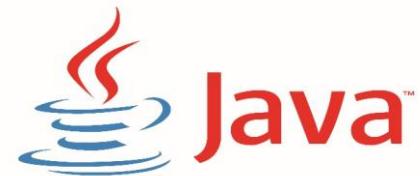




23 EKIM 2021  
DERS 2

Java Giriş  
Variables

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.



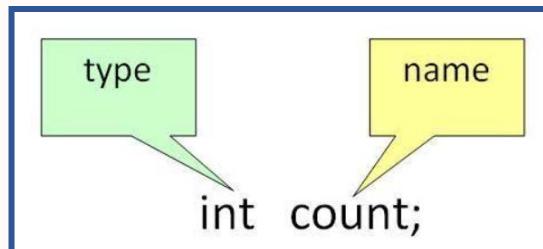
## Variables (Degisken) Olusturma Declaration

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

Variable içinde değer saklayan bir konteynirdir (**container**).

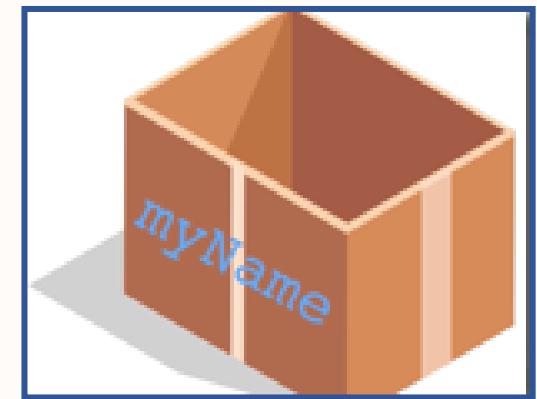
Bir değişkende saklanan değer, program yürütülürken değiştirilebilir.

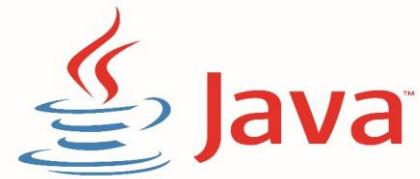
Java'da, tüm değişkenler kullanılmadan önce deklare edilmelidir (**variable declaration**)



Variable declaration için iki seyi belirtmemiz gerekiyor

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



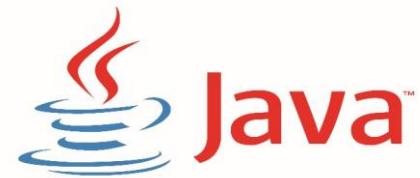


## Variables Deger Atama Assignment

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

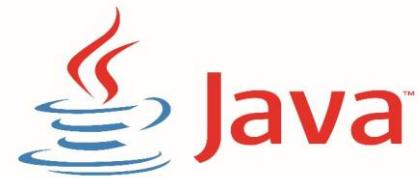
2- İlk önce declaration, daha sonra atama yapılabilir.

```
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    level=2;
11
12    level=3;
13
14
15
16
17
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.

```
8  
9  int level=5, yas=20, maas=10000;  
10
```

# Eclips Kullanim

## 1- Proje olusturma

File -- New -- Project -- (Java Project) Next -- java2021FallTr -- finish

## 2- Package (paket) olusturma

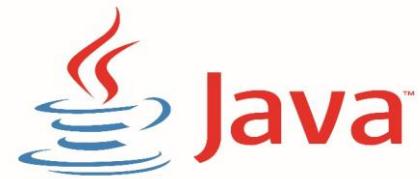
src dosyasina sag click -- New -- Package -- day01variables -- finish

## 3- Class olusturma

day01variables dosyasina sag click -- New -- Class -- C01\_Variables01 -- finish

## 4- Main method olusturma

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

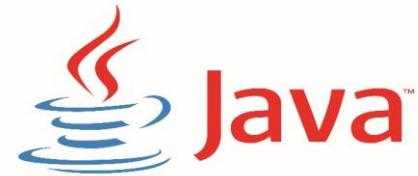


## Data Types

Java'da iki data tipi kullanılmaktadır

- **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

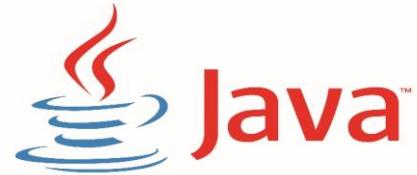
Harf, sayi veya simbol bakilmaksizin sadece 1 karakter kullanacak variable'larda kullanilir

```
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;
```

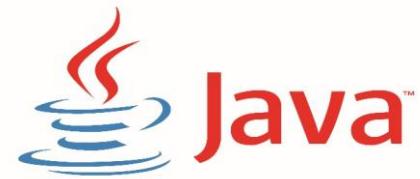
4) **short** Data Type: -32.768 den 32.767'e (dahil) tamsayilar icin kullanilabilir. Hafizada **16 bit** kullanir

```
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

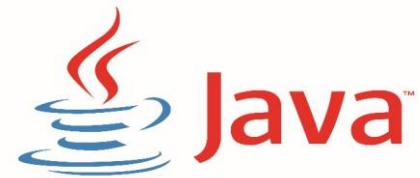
7) **float** Data Type: Kucuk ondalik sayilar icin kullanilabilir. Hafizada **64 bit** kullanir

```
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

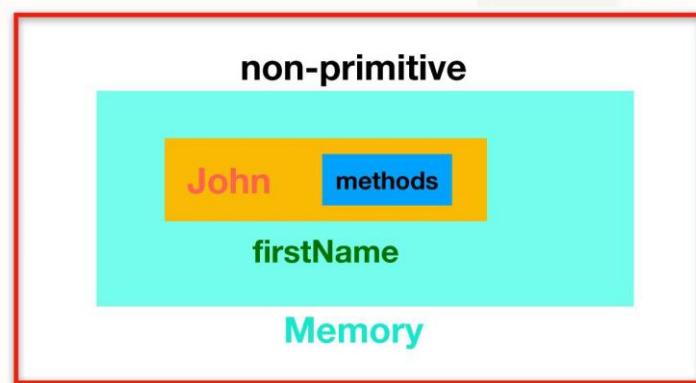
```
double doubleVar2 = -2.1234567907800000000123
```



## Non-Primitive Data Type

### String Data Type:

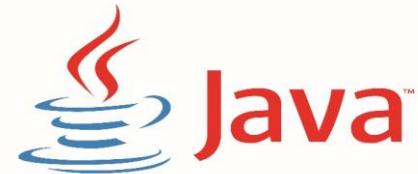
String pes pese dizilmis char'lardan olusur. Kelimeler, cumleler, matematiksel islem yapilmayacak sayisal degerler de String olarak tanimlanabilir



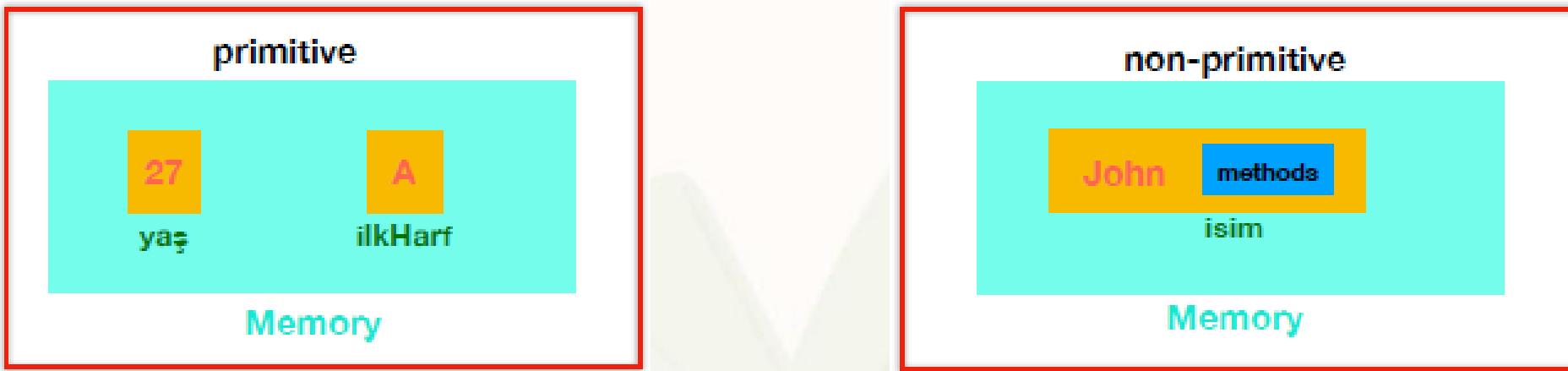
```
String okulAdi = "Yildiz Koleji, Cankaya Ankara #";  
String telNo      = "5321234567";  
String ilkHarf    = "A";
```

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

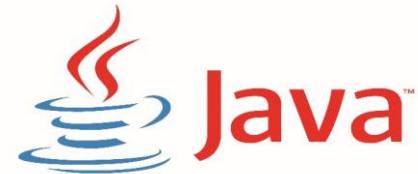
**Note:** Baska non-primitive data type'lar da var, daha sonra ogrenecegiz.



## Primitive VS Non-Primitive Data Types

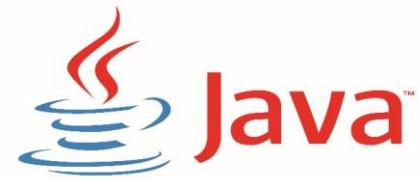


- 1) Primitive'ler sadece value icerir, non-primitive'ler value ve methodlar icerir.
- 2) Primitive'ler kucuk harf ile, non-primitive'ler buyuk harf ile baslar.
- 3) Primitive'leri Java olusturur biz primitive data turu olusturamayiz.  
Non-primitive'leri biz de olusturabiliriz, Java da olusturabilir.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



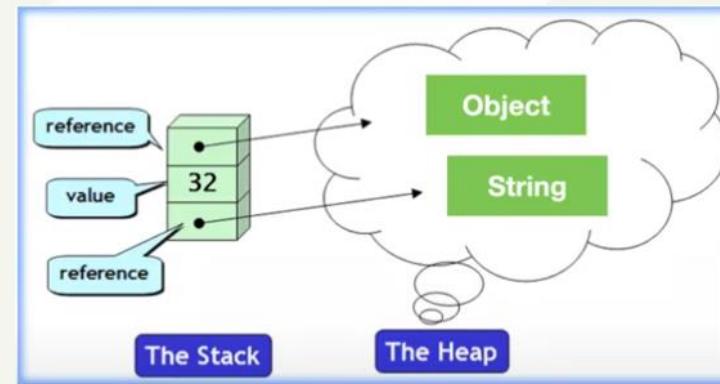
## Variable ve Method'lar Nasıl Adlandırılır

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



## Memory (Hafiza) Kullanimi

Javada kullanılan iki hafıza vardır

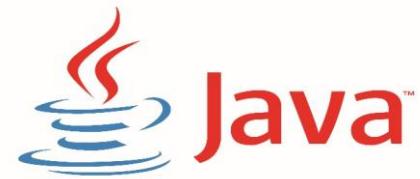


Stack => small

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

byte                      String                      long

Object

int

boolean

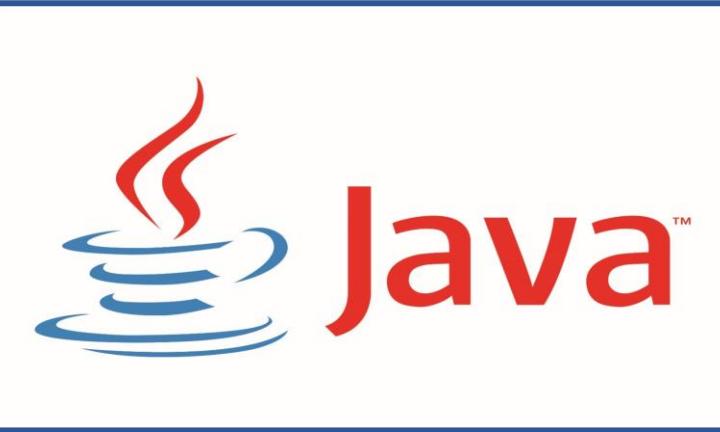
Array



Stack



Heap



24 EKİM 2021  
DERS 3

**Scanner  
Data Casting**

Mehmet BULUTLU0Z  
Elk.Elektronik Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Data : bilgisayarda islenen(processed) veya depolanan (stored) herseye data denir
- 2) Compile : derleme, bilgiyi islemek demektir. Java'nin kullanicinin yazdigı kodları bilgisayarin anlayacagi binary kodlara cevirmesidir.
- 3) OOP Concept : Object oriented Programming , java'da Class'lari kullanarak objeler üretiriz, sonra bu objeleri kullanarak application'lar üretiriz. (Lego gibi)
- 4) Class'larda neler bulunur? : fields (pasif,hareketsiz) , methods (dinamik,hareketli)
- 5) Variable nedir ? : Data değerlerini saklamak (store) için kullanılan container'dır.
- 6) Variable neden kullanılır ? : Datayı program içerisinde kullanabilmek için variable'lara atarız. Programımız içinde ne zaman variable ismini yapsak, Java bize o variable'a atanmış son değeri getirir.

## Onceki Dersten Aklimizda Kalanlar

7) Variable nasıl olusturulur? : Variable olusturmak için declaration (tanımlama) yapılır.  
Veri turu (data type) ve variable ismini yazılmalıdır

8) Variable değer atama : Assignment denir. Variable isminin karşısına = işaretini konularak istenilen değer assign edilir.

9 ) Variable declaration ve assignment için nelere dikkat etmeliyim ?

Declaration ve assignment sırasıyla yapılmalıdır.

Once declaration, sonra assignment olmalı.

Istersek tek satırda ikisini birden yapabiliriz

`int sayı = 20 ;`

İstersek de once declare edip sonra değer atayabiliriz

`int sayı;`

`sayı = 10;`

## Onceki Dersten Aklimizda Kalanlar

10 ) Class olusturmak icin hangi keyword'lar kullanilir ? :

```
public class ClassIsmi { Class Body }
```

11 ) Method olusturmak icin hangi keyword'lar kullanilir ? :

```
public String methodIsmi (parametre) { method Body}
```

12 ) Main method Nedir ?

Java'nin kodlari calistirmaya basladigi giris noktasidir (Entry Point).

13 ) Main method olusturmak icin kullanilan syntax nedir ? :

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

## Onceki Dersten Aklimizda Kalanlar

14 ) Java'da kac cesit data type vardir ? Java'da temel iki data tipi vardir.

- primitive (ikel) data types : boolean (true /false) ,  
char (tek karakter),  
byte, short, int, long(tam sayilar),  
float, double (ondalikli sayilar)
- non-primitive (object): String (simdilik)

15 ) Java'da kac cesit hafiza vardır ? :

stack- (small) primitive datalar ve non-primitive data tiplerinin reference'larinin  
store edildigi hafizadir.

Heap memory (huge) : non-primitive datalarin store edildigi hafizadir.

## 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 degisken olusturun ve bunların toplamini yazdirin.

## 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

# ASCII Table

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

## 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.

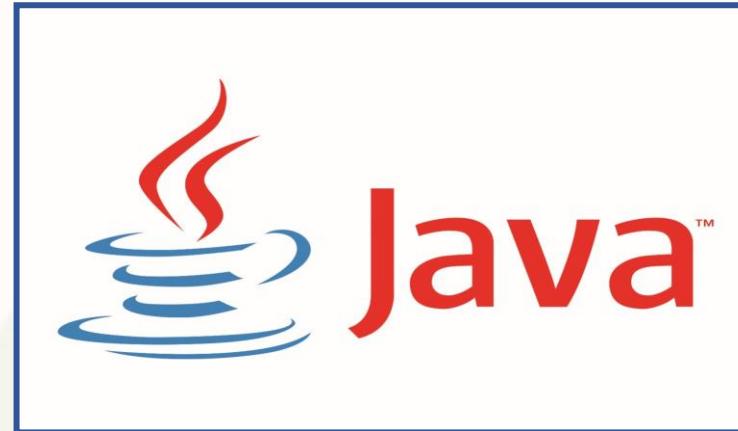
Burada aciklama olarak ne yazdirlsa kodumuz calisir, hatta birsey yazdirmasak da calisir ancak kullanici kendisinden ne istedigimizi bilmezse deger girmesi gerektigini veya ne tur bilgi girmesi gerektigini bilemez

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



25 EKİM 2021  
DERS 4

Data Casting  
Increment-Decrement

Mehmet BULUTLU0Z  
Elk.Elektronik Muh.

## Onceki Dersten Aklimizda Kalanlar

1) Scanner (Kullanicidan deger alma) : 3 adimda kullanicidan deger alabiliriz.

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

scan bu objenin ismidir, baska isim de verebiliriz ama bunu tercih ediyoruz.

- Kullaniciya bir mesaj veriyoruz (Kullaniciya ondan ne bekledigimizi soyluyoruz)

- scan objesi ile ilgili method'u kullanarak kullanicinin girdigi degeri alip, olusturdugumuz uygun bir variable'a assign ediyoruz

- String degiskenler icin next() ve nextLine() method'lari kullanilabilir. Next() sadece bir kelime alir (ilk space'e kadar olan kisim), nextLine() ise satirin tamamini alir

2) Scanner objesi olusturdugumuzda Java itiraz edip altini kirmizi cizer. (Cunku class'imizda scanner ile ilgili bilgiler yoktur) Java.util kutuphanesinden Scanner class'ini import edince bu sikayet ortadan kalkar.

## 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 alinmistir,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.

# Data Casting

## Veri Sınıfı Değiştirme

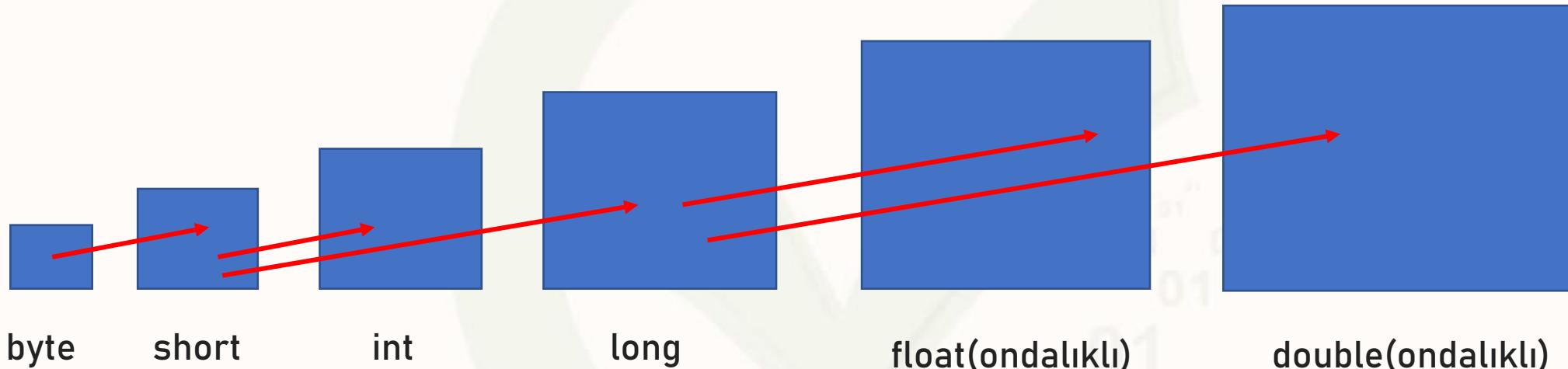
- Java'da kod yazarken bir veri tipinden diğer bir veri tipine aktarım yapmamız gerekebilir.
- Veri tiplerinde bir variable'a , olusturuldugu data tipinden farkli bir data turunden deger atanmasina Data Casting denir.
- Data casting yaparken aklımızdan cikarmamamız gereken konu data tiplerinin sınırlarıdır.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

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

# Data Casting

## 1) Auto Widening (Otomatik Genisletme)

Dar veri tipinden daha geniş bir very tipine gecmek istedigimizde Java donusumu otomatik olarak yapacaktır.



Orn : byte num1 = 12;

short num2 = num1; // yazdirırsak 12 olarak yazdırır

int num3 = num2; // yazdirırsak 12 olarak yazdırır

float num4=num3; // yazdirırsak 12.0 olarak yazdırır

double num5=num4; // yazdirırsak 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 boldurun ve sonucu yazdirin
- Soru 6 )** int bir sayiyi double bir sayıya bolun ve sonucu yazdirin
- Soru 7 )** Farklı data tipleri ile işlem 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 - - ;
```

?

## 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 azaltilar

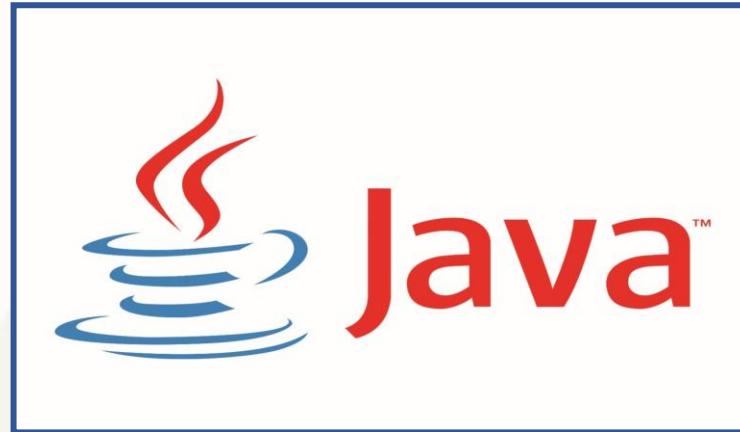
```
public static void main(String[] args) {  
    int a=15;  
    int b=++a;  
    System.out.println(b);  
}
```

Output : 16

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

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

Output : 15



26 EKİM 2021  
DERS 5

**Matematiksel Operatorler  
Modulus**

Mehmet BULUTLU0Z  
Elk.Elektronik Muh.

# Javada Matematiksel Operatorler

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

Ornek 1:

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

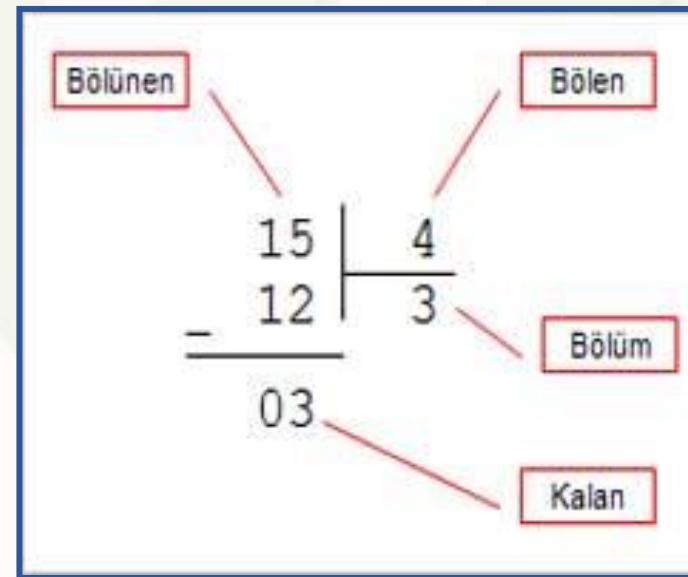
Ornek 2 :

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

# Modulus %

Modulus işlemi bir bolme işleminden kalan sayiyi bize verir

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



## Modulus %

**Soru 1)** Kullanicidan 4 basamakli bir sayı alın ve rakamlar toplamini bulup yazdırın

**Ipucu 1:**

Sayı % 10 => Bize son basamagi verir

$$538 \% 10 = 8$$

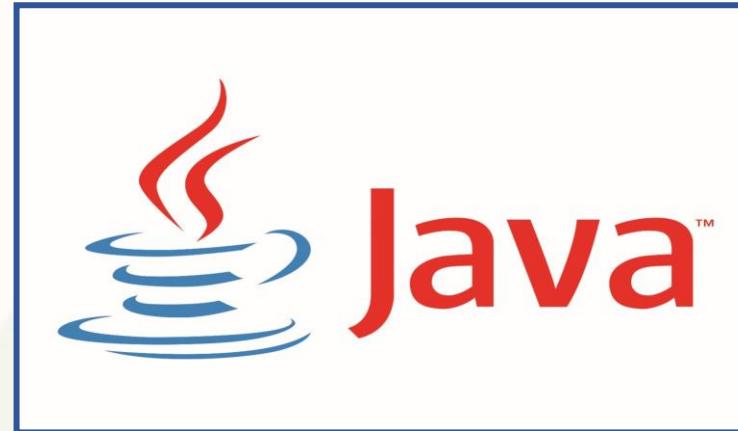
**Ipucu 2:**

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

**int** sayı=538;

**sayı = sayı / 10 =>**

**sayı'ya** 53 degerini atar



30 EKİM 2021  
DERS 6

**Wrapper Class**  
**Concatenation**  
**Relational Operators**

Mehmet BULUTLU0Z  
Elk.Elektronik Muh.

## Onceki Dersten Aklimizda Kalanlar

1- matematiksel operatorler : kodlarimizi yazarken dogru sekilde yazmazsak yanlis sonuclara ulasabiliriz.

“Java” + 30 + 20 => Java3020

“Java” + (30 + 20 ) => Java50

20+30+ “Java” => 50Java

”+20+30+ “Java” => 2030Java

A- us ve parantez : yazim sekline gore us veya parantez once olabilir

B- carpma ve bolme

C- toplama ve cikarma

Egit oncelikli islemlerde once soldaki islem yapilir

2- Modulus : bir bolme islemindeki kalan degerini bize verir

sayi % 2 ye bakariz sonuc sifirsa sayi cift, sonuc 1 ise sayi tek

sayi % 10 sayinin birler basamagini verir

sayi/10 (sayi tam sayi olmak uzere) : birler basamagini yok edip kalan kismi bize verir

627 % 10 => 7

627 / 10 => 62

# Wrapper Class

Java **primitive** data turleri ile **methodlari** kullanabilmemiz icin **Wrapper class'lar**i 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);  
  
        int num2 = Integer.MAX_VALUE;  
        System.out.println(num2);  
  
        int num3 = Byte.MIN_VALUE;  
        System.out.println(num3);  
  
        int num4 = Byte.MAX_VALUE;  
        System.out.println(num4);  
  
    }  
}
```

```
-2147483648  
2147483647  
-128  
127
```

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

HelloWorld  
Hello World

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

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

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

## Concatenation (String Datalari Birlestirme)

Soru 1) Asagida verilen variable'lari kullanarak istenen sonucları yazdırın programları yazınız.

### 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 degiskenine 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

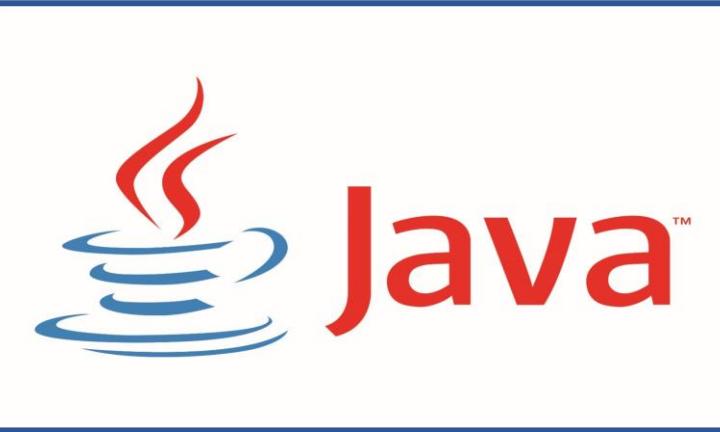
### || OR (veya) isareti

|| 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



30 EKİM 2021  
DERS 7

If Statements  
If Else Statements

Mehmet BULUTLU0Z  
Elk.Elektronik Muh.

## 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) {şart saglanırsa 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
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 (Sorular)

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 olmamasi icin toUpperCase methodunu kullanin**

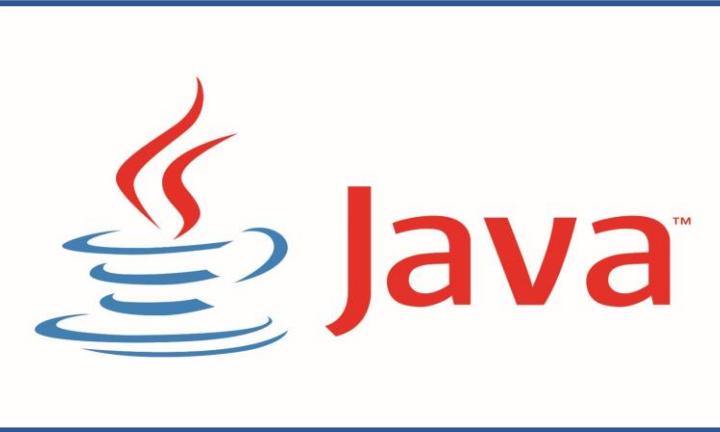
Soru 3) Kullanicidan gun ismini alin ve haftaici veya hafta sonu oldugunu yazdirin

Ornek: gun=Pazar output = “Hafta sonu”  
gun=Sali output = “Hafta ici”

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

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

Soru 5) Kullanicidan bir gun alin eger gun “Cuma” ise ekranra “Muslimanlar icin kutsal gun” yazdirin. “Cumartesi” ise ekranra “Yahudiler icin kutsal gun” yazdirin. “Pazar” ise ekranra “Hiristiyanlar icin kutsal gun” yazdirin



31 EKİM 2021  
DERS 8

## If Else Statements

Mehmet BULUTLU0Z  
Elk.Elektronik Muh.

## Onceki Dersten Aklimizda Kalanlar

1- Wrapper class : Java primate data turleri ile method'lar kullanabilmemiz icin her primitive data turune bir tane non-primitive data turu hazirlamistir ve bunlara ozel olarak wrapper class denir

2- Mantiksal karsilastirmalar && ve OR

- && mukemmeliyetcidir, sonucun true olmasi icin hepsinin true olmasi gerekir, matematikteki carpma gibidir, bir tane sifir sonucu sifir yapmaya yeter, && bir tane false sonucu false yapmaya yeter

- OR ise iyimserdir, sonucun false olmasi icin tum bilesenlerin false olmasi gerekir. Matematikteki toplama gibidir, 1 tane bile 1 olursa toplam sifir olmaz, OR icin de 1 tane bile TRUE olursa sonuc false olmaz

3- = isareti assignment isaretidir ve karsilastirma yapmaz,

- Java = isareti gordugunde bir atama oldugunu anlar ve oncelikle esitligin sag tarafindaki degeri hesaplar, esitligin sag tarafindaki islem bitince, bulunan sonucu esitligin solundaki variable'a assign eder

- eger karsilastirma yapmak istersek != kullanmalıyiz. Java == gordugunde bu cift esittir isaretinin saginda ve solundaki ifadelerin esit olup olmadigina bakar, esit ise true, esit degilse false dondurur.

- eger esit olmadiklarini control etmek istiyorsak != isareti kullanilir. != isareti mantiksal bir ifadenin onune konulursa tersine cevirir. !true=>false gibi..

- String ile esitligi control etmek istedigimizde == yerine equals() kullanilir

## 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 ekran'a "Eskenar ucgen" yazdirin. Diger durumlarda ekran'a "Eskenar degil" yazdirin.

## & Ile && 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.

## If Else If ... Statements

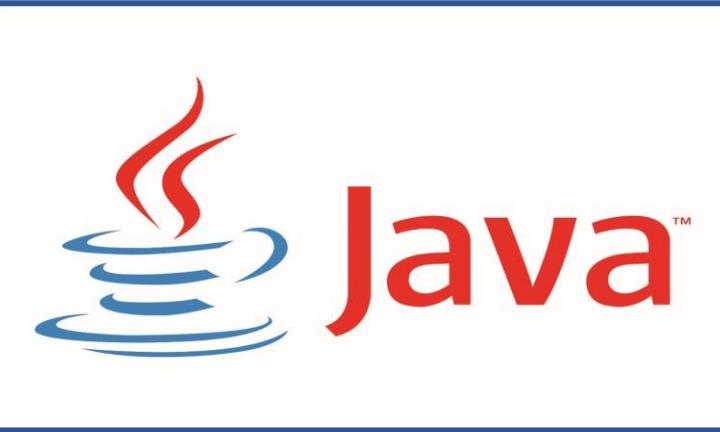
Eger soruyu biliyorsa Ali soruyu cozsun , o bilmiyorsa Veli biliyorsa Veli cozsun,  
o da bilmiyorsa Ayse biliyorsa, Ayse cozsun, o da bilmiyorsa Fatma biliyorsa,  
Fatma cozsun, o da bilmiyorsa kim isterse o cozsun.

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 (Sorular)

- 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 sayi isteyin, sayilarin ikisi de pozitif ise sayilarin toplamini yazdirin, sayilarin ikisi de negative ise sayilarin carpimini yazdirin, sayilarin ikisi farkli isaretlere sahipse "farkli isaretlerde 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 arasinda ise "Konusabiliriz",  
60.000'nin altinda ise "Maalesef Kabul edemem" yazdirin



1 KASIM 2021  
DERS 9

Nested If Else Statements  
Ternary

Mehmet BULUTLU0Z  
Elk.Elektronik Muh.

## Onceki Dersten Aklimizda Kalanlar

1- basit if cumleleri sadece bir şart ve bir sonuctan oluşur, şart gerçekleşirse if body çalışır, şartın gerçekleşmemesi durumunda if body'si çalışmaz, kod bir alttan devam eder.

Basit if cumleleri art arda yazılısa da bağımsız olarak çalışır. Örneğin 5 basit if cümlesi alt alta yazılısa hepsinin if body'si devreye girebileceği gibi, hiçbir devreye girmeyebilir

2- If else : eğer bir şart cümlesinin sonuçları sadece 2 durum ise, başka hiç bir ihtimal yoksa if else kullanırız. If içine yazılan koşul true ise if body'si , koşul false ise else body'si devreye girer.

if ve else body'leri birbirini tamamladığı için sadce ve sadece iki body'den biri çalışır

3- If Else If ... : Eğer bir soru için sınırlı sayıda durum mevcut ise ( örneğin not durumuna göre A,B,C,D not alınması gibi) arkaya arkaya if else cümleleri eklenebilir.

if else if şartları arkaya arkaya geldiğinden dolayı birbirini bütünlüler, yani ilk if de şart olarak yazdığımız durumu bir daha sorgulamamıza gerek kalmaz.

4-Nested If cümleleri : bazen sorguladığımız durum birden fazla değişkene bağlı olabilir ( cinsiyet ve yaşı' gore emekliliğin değişmesi gibi)

Bu durumda ilk önce bir değişkene göre if else if ... yapısı oluşturulur, sonra uygun bölmelere ikinci değişkene göre nested if else if .. Cumleleri oluşturulur.

## 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

### Sorular

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 (Sorular)

## Soru 9) Interview Question

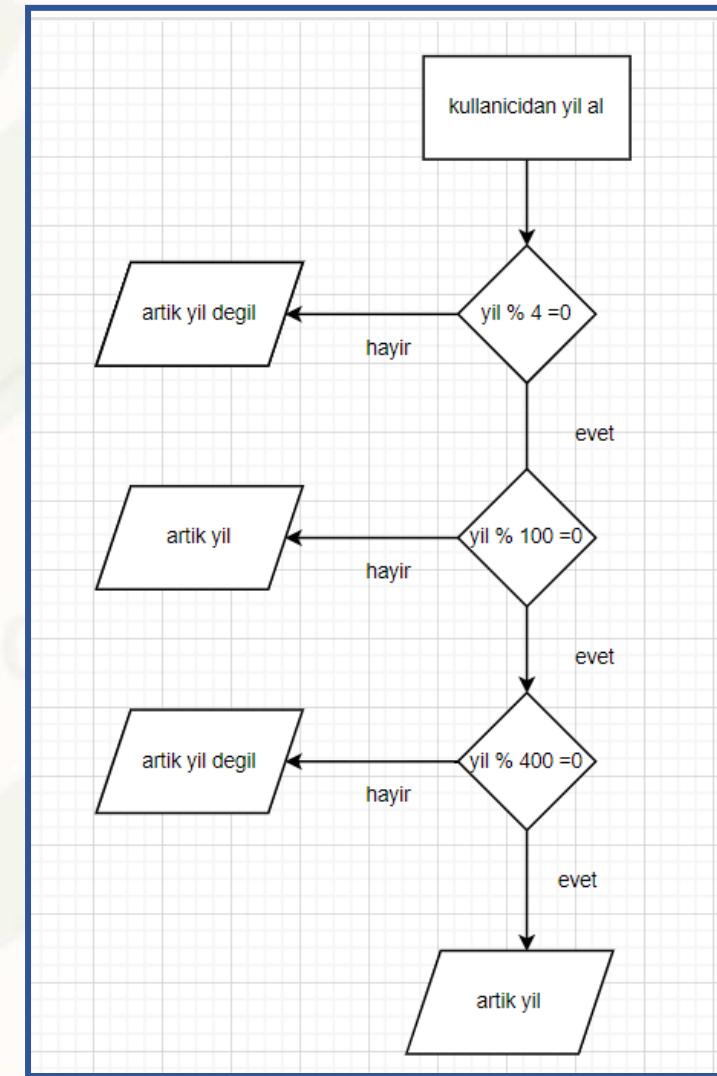
Kullanicidan artik yil olup olmadigini kontrol etmek icin yil girmesini isteyin.

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

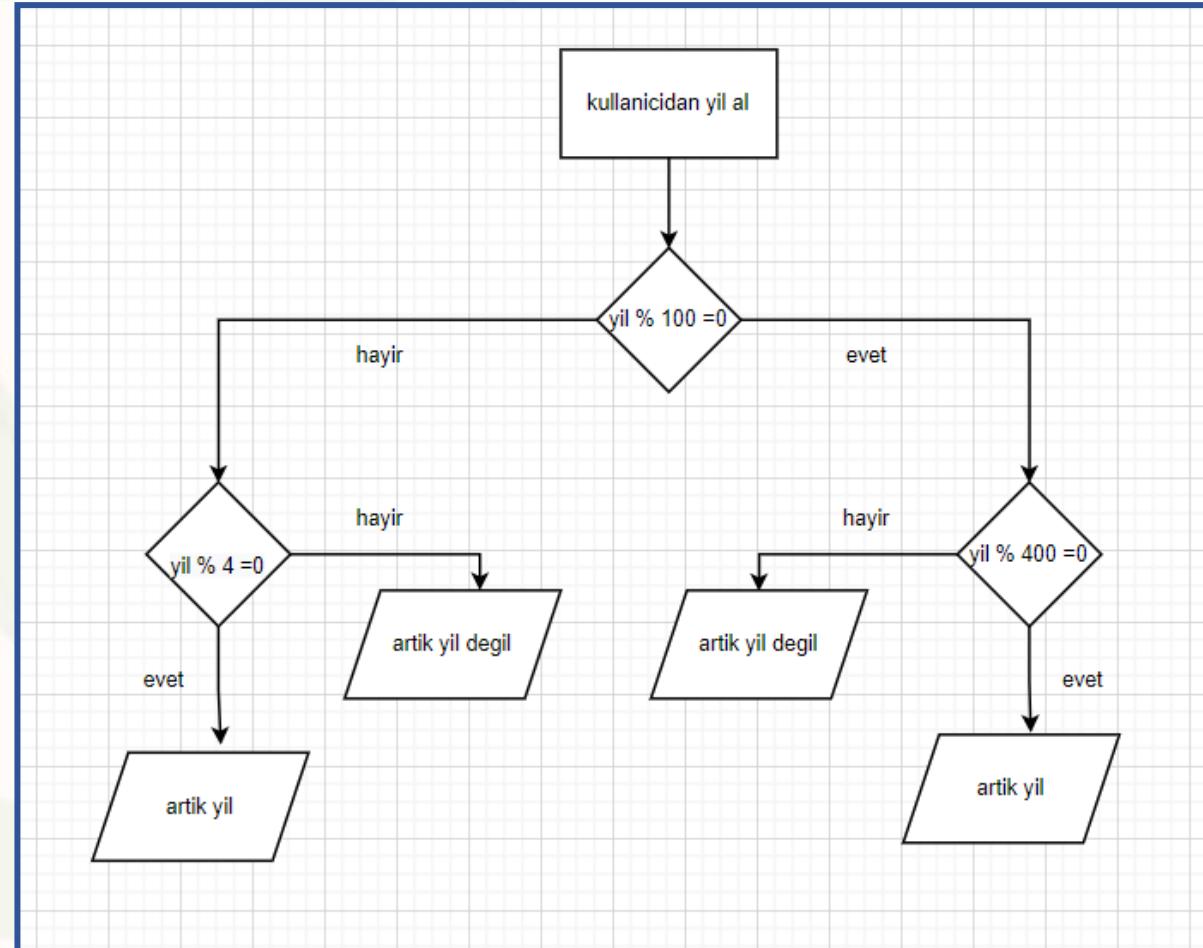
## Sorular

### Soru 10) Interview Question

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

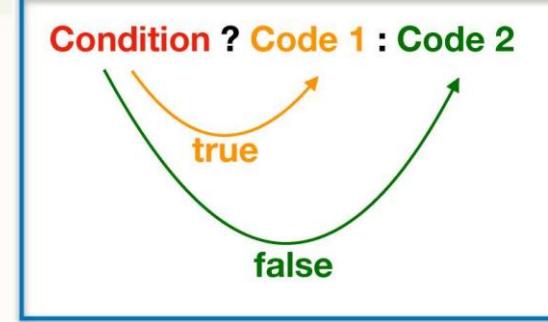
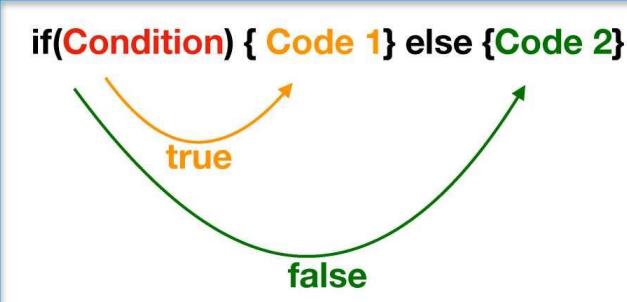
Kural 1: 4 ile bolunemeyen yillar artik yil degildir

Kural 2: 4'un kati olmasina ragmen 100 ile bolunebilen yillardan sadece 400'un kati olan yillar artik yildir



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

# Ternary Operator



**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;

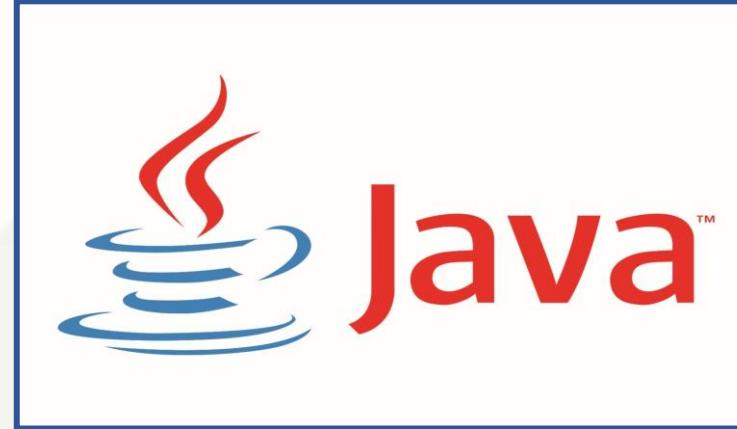
```
System.out.println((y < 91) ? 9 : 11);
```

Soru3 : int y = 1;

```
int z = 1;
```

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

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



2 KASIM 2021  
DERS 10

Ternary  
Switch Case

Mehmet BULUTLU0Z  
Elk.Elektronik Muh.

## Onceki Dersten Aklimizda Kalanlar

1) Ternary : If-Else ile yazabildigimiz bazi kodlari ternary ile daha basit olarak yazabiliriz.

(Condition) ? (condition true ise) : (condition false ise)

Condition true veya false oldugunda calisacak kisim basit sonuc veya islemler olmalıdır.

Her iki durumda da java bize sonuc dondurdugu icin ya ternary'i uygun bir variable'a atamali veya direktly sysout icine yazarak ekraninda cikti olarak gormeliyiz

eger true veya false oldugunda donecek sonuclarin data turleri farkli ise atama yapamayacagimiz icin sadece sysout opsiyonu kullanilabilir

# Ternary Operator

## Sorular

**Soru1 )** Kullanicidan iki sayı alın ve büyük olmayan sayiyi yazdırın

**Soru2 )** Kullanicidan bir tamsayı alın ve sayının tek veya çift olduğunu yazdırın

**Soru3 )** Kullanicidan bir sayı alın ve sayının mutlak değerini yazdırın

**Soru4 )** Kullanicidan bir sayı alın. Sayı pozitifse “Sayı pozitif” yazdırın, negatifse sayının karesini yazdırın

## 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 basamaklı sayı" degilse "uc basamaklı veya daha buyuk sayı" 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

## 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 Sorular

**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 sayı”

100 ise “uc basamakli en kucuk sayı”

1000 ise “dort basamakli en kucuk sayı”

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

## 1- 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)
```

# 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

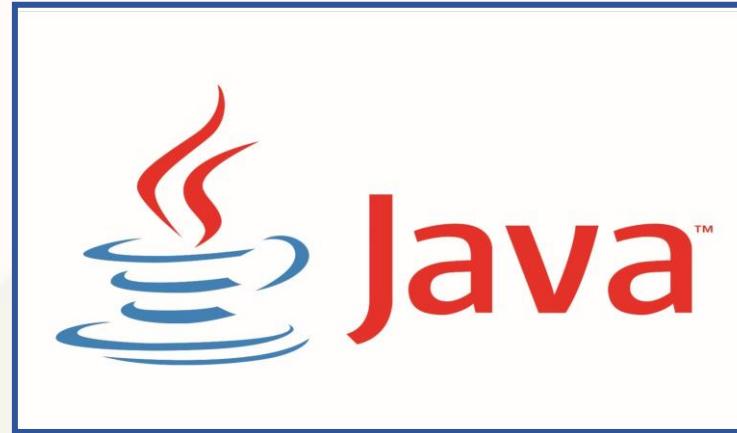
NOT : toLowerCase(Locale locale)

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



6 KASIM 2021  
DERS 11

## String Manipulations

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Nested ternary : bazen kodumuz basit olmakla beraber iki tercihten fazlasini gerektirebilir. Bu durumda tek ternary ile kodu yazmamız mümkün olmaz, ilk ternary de false kismina yeni bir ternary yazarak ihtimalleri 3'e çıkarabiliriz.

Eger true yerine de nested ternary yazarsak ihtimalleri 4 'e çıkarabiliriz.

Ancak ternary'yi tercih etme sebebimiz basit yapida olmasidir, karmaşık durumlarda ternary yerine if-else tercih edilmelidir

- 2) Switch -case : Eger if-else if ... cozebilecegimiz bir sorunda tercih sayisi çok ise if-else 'leri cogaltmak kodumuzu karmaşık hale getirebilir.

Bu durumda if-else yerine switch-case tercih edilir.

Switch parantezinin içine Boolean bir ifade yazamyız, aynı şekilde parantez içine long, double ve float da yazılmaz.

Switch parantezinin içine yazdigim degree gore Java uyan case'e gider ve break gorené kadar kodu calistirmaya devam eder.

Eger birden fazla case icin calisacak kod aynı ise break yazmadan case'leri alt alta siralayabiliriz.

Eger yazilan case'ler disinda kalan tum durumlar icin ortak bir kod varsa, default : yazip ilgili kodumuzu ekleyebiliriz.

# 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 oldugunda equals methodunda oldugu 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

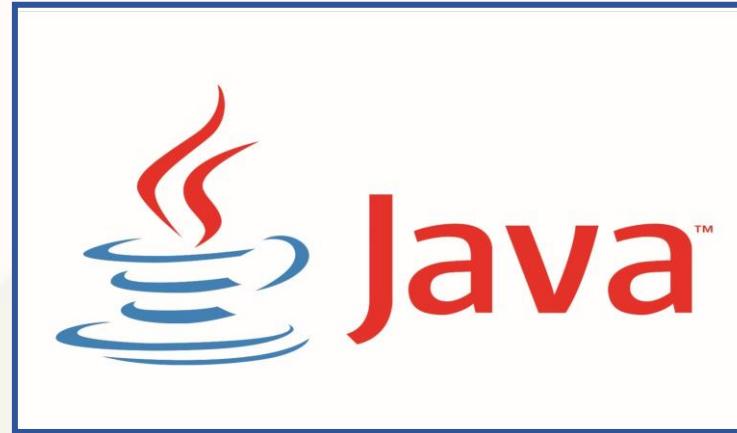
## 8-indexOf()

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



6 KASIM 2021  
DERS 12

## String Manipulations

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# 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.

# 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'));  
System.out.println(str.lastIndexOf("a"));  
System.out.println(str.lastIndexOf("t"));  
System.out.println(str.lastIndexOf("Java"));  
System.out.println(str.lastIndexOf('a',11));
```

: 35 : 35 : -1 : 14 : 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 kelime olabilir

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

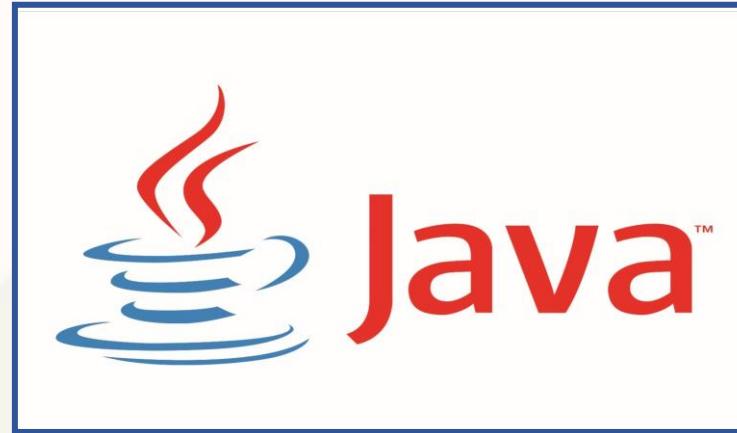
**NOT** contains() methodu char icin kullanilamaz, String kullanmak zorunludur.

## 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 iceriyorsa tum cumleyi buyuk harf olarak, “kucuk” kelimesi iceriyorsa tum cumleyi kucuk harf olarak yazdirin, iki kelimeyi de icermiyorsa “Cumle kucuk yada buyuk kelimesi icermiyor” yazdirin.



7 KASIM 2021  
DERS 13

## String Manipulations

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) str.indexOf(str2) : str icinde str2 nin ilk kullanildigi yerin index'ini verir.  
str2 yerine char da kullanilabilir.  
eger str.indexOf(str2, int a) a.index'ten sonra str2 arar  
eger bir str2 icin ilk kullanilan index'i bulduysak ve ikinci bir kullanım var mi diye control etmek istiyorsak str.indexOf(str2, ilk index+1) seklinde kullanabiliriz, boylece java ilk kullanım index'inden sonra ikinci bir kullanım var mi diye kontrol etmis olur
- 2) str1.equals(str2) str1 ve str2'nin iceriginin ayni olup olmadigini kontrol eder, ikisinin içerikleri esit ise true, esit degilse false doner.  
== ile equals() nun temel farki , == hem degree hem referansa bakar dolayisiyla ancak ayni objeyi kendisiyle karsilastirdigimizda true doner, equal() ise referans'a degil sadece icerige bakar, case sensitive olarak ayni icerige sahiplerse true doner
- 3) equalsIgnoreCase() kullanildiginda ise case sensitive olmadan içeriklerin ayni olup olmadigina bakar
- 4) str1.lastIndexOf(str2) aramaya sondan baslayarak, basa dogru gelir ve buldugu ilk str2'nin index'ini verir, digger ozellikleri indexOf(0'a benzer
- 5) str1.contains(str2) : str1'in icinde str2 varsa true, yoksa false doner. char kullanilamaz  
\*\*\* indexOf ve lastIndexOf bize int index dondurur, equals() ve contains() ise Boolean bir sonuc dondurur

## 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'));
```

Jxvx ogrenmek cok kolxy  
x ogrenmek cok kolay  
Jxxxxxx ogrenmek cok kolxxxxy  
Jv ogrenmek cok koly  
Jxvx ogrenmek 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

# 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)
```

## String Manipulation / Methods

### substring()

```
public static void main(String[] args) {  
  
    String str= "Java OOP konsepti kullanır";  
  
    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 çizgi 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

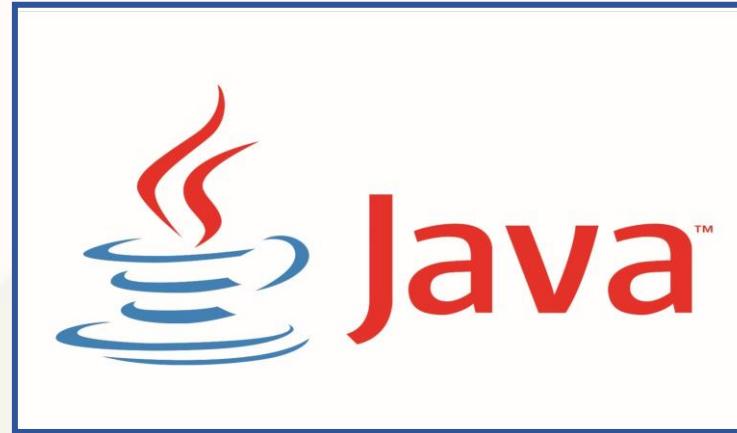
**Soru 6)** Kullanicidan bir sifre girmesini isteyin. Asagidaki şartlari sagliyorsa "Sifre basari ile tanimlandi", şartlari saglamazsa "Islem basarisiz,Lutfen yeni bir sifre girin" yazdirin

- 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



8 KASIM 2021  
DERS 14

Method Creation  
Method Call

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# Socrative Quiz

- 1) <https://www.socrative.com/> adresine gidin
- 2) **Login** butonuna basin
- 3) **Student Login** butonuna basin
- 4) Room Name **BULUTLUOZ** yazın
- 5) Isminizi yazın
- 6) **Done** butonuna basin

Sure : 15 Dakika

# 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) : { } arasında kalan kodlarimizi yazdigimiz bolumdur

# Method Creation

## Method Oluşturma

```
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
no modifier	Y	Y	N	N
private	Y	N	N	N

# Method Creation

## Method Oluşturma

2 static (İlleride detaylı anlatılacak)

Bir method oluştururken **static** kelimesinin kullanılması mecburi degildir.

Main method'umuz static oldugu icin main method'dan cagiracagimiz tum method'lari static yapmamız 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 tum data turlerinden olabilir
  - Eger method bir sey dondurmeyecekse (ornegin, sadece bir sey hesaplayip yazdiracaksa) return type olarak **void** secilir
  - Return Type olarak void disinda bir sey yazdiysak, methodun sonunda mutlaka **return** keyword kullanilmalidir
  - Return keyword'den sonra return type'a uygun bir **deger veya variable** yazilmalidir.
  - Return type'a sahip methodlar cagrildiklari satira, return keyword'den sonra yazilan deger veya variable'i dondururler.

```
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 kelimedenden 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 yazdığımız 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 Call

### Method Cagirma

Method olusturmak method'u calistirmak icin yeterli degildir.

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

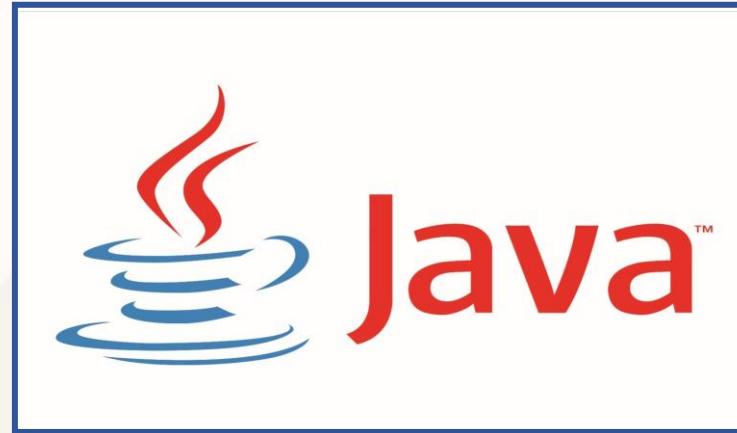
Bu isleme method cagirma denir

```
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



**9 KASIM 2021  
DERS 15**

**Method Creation  
Method Call**

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Method Creation

**Soru 1 )** Kullanicidan bir sayi alin. Bu sayinin tek mi cift mi oldugunu, sifirdan buyuk mu kucuk mu oldugunu, ayrica ve 100'den buyukse birler,onlar ve yuzler basamagindaki rakamlarin toplamini,100'den kucukse sadece 1'ler basamagini yazdiran 3 method olusturun.

**Soru 2 )** Kullaniciya kac sayi toplamak istedigini sorun. Kullanici 2,3 veya 4 degerini girerse, kullanicidan bu sayilari girmesini isteyin ve sayilarin toplamini yazdirin. Kullanici toplamak istedigi sayi adedini 4'den buyuk girerse "Cok sayi girdiniz, ben toplayamam" yazdirin.

**Soru 3)** Email kontrolu yapan bir program yazin.Kullanicinin girdigi sifre

- @ isareti icermiyorsa gecersiz email yazdirin
- @gmail.com icermiyorsa "lutfen gmail adresinizi girin" yazdirin
- @gmail.com ile bitmiyorsa "Yazimda bir sorun var, maili kontrol ediniz"

**Soru 4)** Kullanicidan ismini, soyismini ve bosluk birakmadan 16 hane olarak kredi karti numarasini alin. Isim ve soyismi ilk harfleri buyuk diger harfler kucuk olacak sekilde, KK numarasini ise 4 rakamlik 4 blok ve aralarinda bosluk olacak sekilde duzelten 2 method yazin, ve programda kullanabilmek icin disenlenmis hallerini geri dondurun.

# Method Overloading

## Interview Sorusu

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

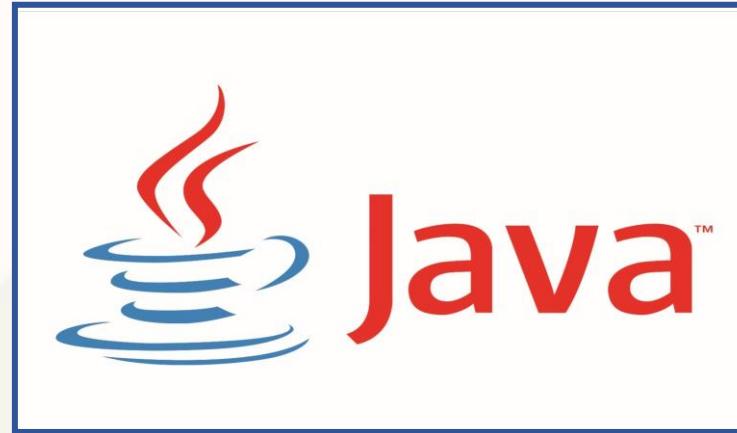
2) Overloading nasıl yapılır ? Java ayni isim ve ayni parametrelerle birden fazla method olusturulmasına izin vermez. Ayni isimle birden fazla method olusturmak isterseniz **method signature (metot imzası)**'nin degistirilmesi gereklidir

3) method signature (metot imzası) 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



10 KASIM 2021  
DERS 16

## For Loop

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Method : bizim icin tekrar tekrar kullanabilecegimiz functionality'leri yapan kod bloklaridir.(belli isleri yapmaya programlanmis robotlar gibi)
- 2) Method'lari bir kere olusturur, hert ihtiyacimiz oldugunda method'u cagirarak(method call) kullaniriz
- 3) Method olusturmak, calismasi icin yeterli degildir, method'un calismasi icin method call yapmamiz gerekir
- 4) Method'lar yaptiklari ise gore bize bir sonuc dondurebilir (return) veya sadece yazdirma gibi islemler yapip bize hic bir sey dondurmeye bilir(void).
- 5) Eger method main method'a bir sonuc dondurecekse, main method'da bu sonunu ya direkt yazdiririz, veya data turune uygun bir variable'a assign ederek program icinde return eden bilgiyi kullanabiliriz
- 6) Ayni method ismine sahip birden fazla method olusturulabilir, bu durumda ayni isme birden fazla method yuklendiği için buna overloading denir.
- 7) Overloading yapabilmek icin method ismi ayni olup, method signature'l farkli olmaliydi.
- 8) Method signature : method ismi, parameter sayisi, parameter data turleri ve parameter siralamasi demektir.

Overloading yapmak istedigimizde method ismi degisemeyeceginden parameter sayisi, parameter data turleri veya farkli data turundeki parametrelerin siralamasi degistirilmelidir.

Java method call yapildiginda gidecegi method'u parameter sayisi ve parameter data turlerinden optimal olanı secer

## 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.

The diagram illustrates the structure of a for loop. It features a green arrow pointing right labeled "Start", a red octagonal STOP sign, and two orange arrows pointing up and down labeled "Increasing or Decreasing the Value". The code structure is as follows:

```
for(Starting Value ; Ending Condition ; Increasing or Decreasing the Value) {  
}
```

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

# For Loop



- Eger **Ending Condition** hep **true** verirse loop sonsuz donguye girer
- Eger Loop'ta **Ending Condition** hic **true** olmazsa loop body hic devreye girmez
- loop'da artis degeri pozitif oldugu gibi negatif de olabilir (i-- vb)
- 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

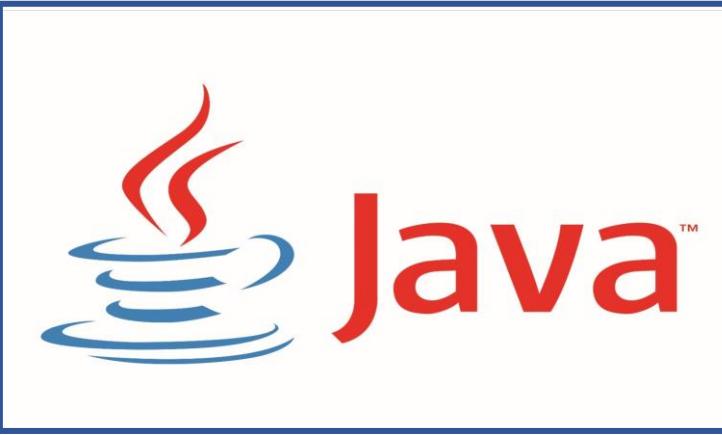
**Soru 3)** 100'den baslayarak 50'ye(dahil) kadar olan sayıları aralarında virgül olarak aynı satırda yazdirin

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

**Soru 5)** Kullanicidan 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** Kullanicidan 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.



13 KASIM 2021  
DERS 17

## For Loop

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## 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 sayi isteyin. Girilen sayilar ve aralarindaki tum tamsayilari toplayip, sonucu yazdiran bir program yaziniz

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

## 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)

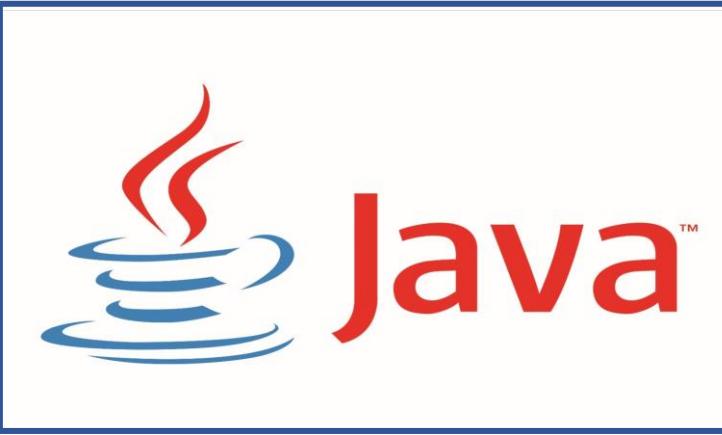
## Nested For Loop

**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
```



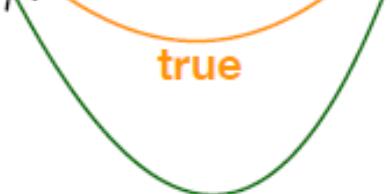
14 KASIM 2021  
DERS 18

Nested For Loop  
While Loop

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# While Loop

`while(condition) {      Code      }`



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++;  
}
```

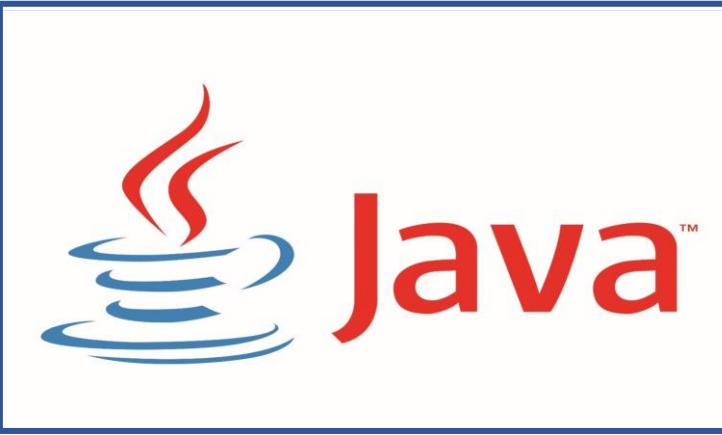
## While Loop

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

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

**Soru 3 )** Kullanicidan baslangic ve bitis degerlerini alin. Baslangic degeri ve bitis degeri dahil aradalarindaki tum cift tamsayıları while loop kullanarak ekrana yazdiriniz.

**Soru 4)** Kullanicidan baslangic ve bitis haflerini alin ve baslangic harfinden baslayip bitis harfinde biten tum harfleri buyuk harf olarak ekrana yazdirin. Kullanicinin hata yapmadigini farz edin.



15 KASIM 2021  
DERS 19

While Loop  
Do While Loop

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## 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;

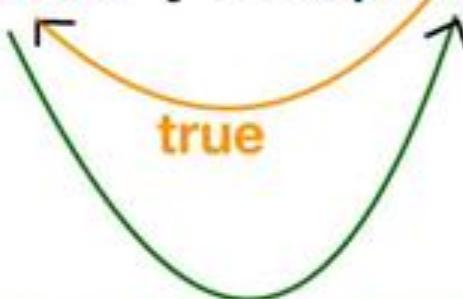
$3 \times 1 = 3$   $3 \times 2 = 6$   $3 \times 3 = 9$   $3 \times 4 = 12$   $3 \times 5 = 15$   $3 \times 6 = 18$   $3 \times 7 = 21$   $3 \times 8 = 24$   $3 \times 9 = 27$   $3 \times 10 = 30$

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

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

## Do While Loop

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



Run the code then check the condition

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

Run the code then check the condition

false

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 katı olan tüm tamsayıları ekrana yazdırınız.

**Soru 2)** 'm' harfinden başlayarak 'c' harfine kadar tüm harfleri yazdırın.

**Soru 3)** Kullanıcıdan toplamak üzere pozitif sayılar isteyin, işlemi bitirmek için 0'a basmasını söyleyin.

Kullanıcı 0'a bastığında toplam kaç pozitif sayı girdigini ve girdiği pozitif sayıların toplamının kaç olduğunu yazdırın.

**Soru 4)** Kullanıcıdan toplamak üzere pozitif sayılar isteyin, işlemi bitirmek için 0'a basmasını söyleyin.

Kullanıcı yanlışlıkla negative sayı girerse o sayıyı dikkate almayın ve "Negatif sayı giremezsiniz" yazdırıp basa donun

Kullanıcı 0'a bastığında toplam kaç pozitif sayı girdigini, yanlışlıkla kaç negative sayı girdigini ve girdiği pozitif sayıların toplamının kaç olduğunu yazdırın.

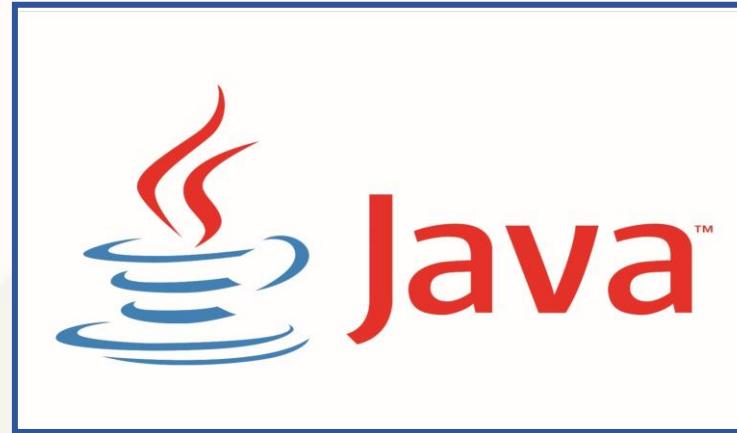
## Do While Loop

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

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

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

**Soru 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



15 KASIM 2021  
DERS 20

Scope

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# Socrative Quiz

- 1) <https://www.socrative.com/> adresine gidin
- 2) **Login** butonuna basin
- 3) **Student Login** butonuna basin
- 4) Room Name **BULUTLUOZ** yazın
- 5) Isminizi yazın
- 6) **Done** butonuna basin

Sure : 12 Dakika

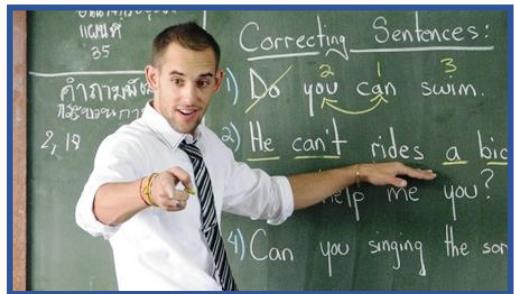
# Scope

## Instance, Class ve Local Variables





## Object Nasıl Kullanılır ?



Ogretmen

Dersler

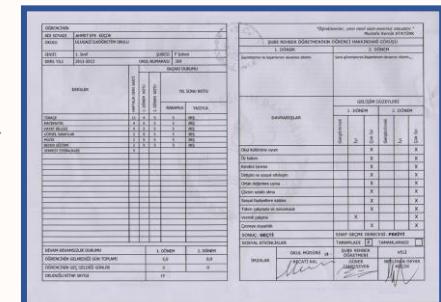
09:00	TÜRKÇE-1
09:30	MATEMATİK-1
10:00	TÜRKÇE-2
10:30	MATEMATİK-2
11:00	TÜRKÇE-3
11:30	MATEMATİK-3
12:00	TÜRKÇE-4
12:30	MATEMATİK-4
13:00	İYEP TÜRKÇE



Personel



Ogrenci



Notlar

# Scope

## Instance, Class ve Local Variables

### Scope (Kapsam)

- 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 içinde ancak main method'un  
disında olmalıdır

Static olmamalıdır

Oluşturulması yeterlidir, değer  
atanması şart değildir.

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

### Default Value

Eğer instance bir variable oluşturur ama değer atamazsanız, Java otomatik olarak  
default değerleri assign eder. (String için null, sayısal data türleri 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 olmalidir

Olusturulmasi yeterlidir, deger atanmasi şart degildir.

```
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 degistirebiliriz.

```
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 okulları, okullar, açık mı 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.

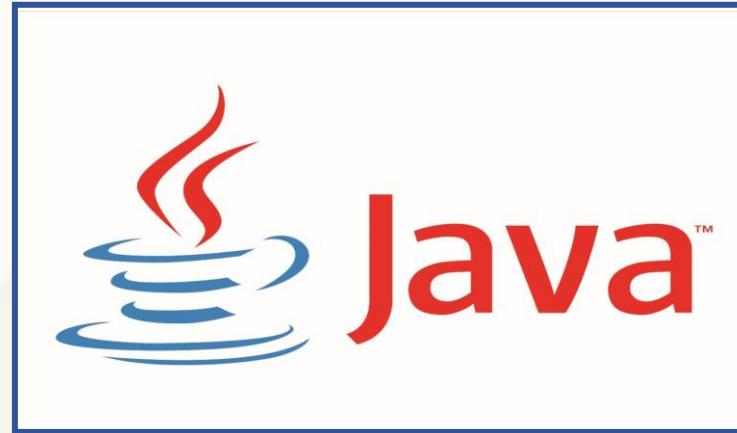
## Scope Instance Vs Class Variables

**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 classIsimi.variableIsimi ile variable'a ulasabilir 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 degisim her obje için gecerlidir. Instance variable da yetkisi olan herkes tarafından degistirilebilir ancak yapılan degisiklik sadece o obje ile ilgilidir, geneli kapsamaz.



20 KASIM 2021  
DERS 21

Scope  
Arrays

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# 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, diğer 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  
18           String isim;  
19           System.out.println(isim);  
20  
21 }
```

# Scope

## Instance, Class ve Local Variables

### Loop Variables

- Bir loop icinde olusturulan variable'lar sadece o loop icerisinde gecerlidir.
- Loop icerisinde olusturulan variable'lara loop disindan ulasilmaz 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);  
}
```

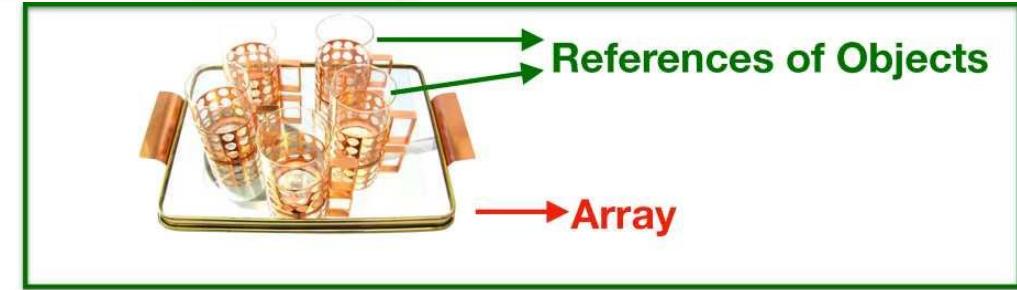
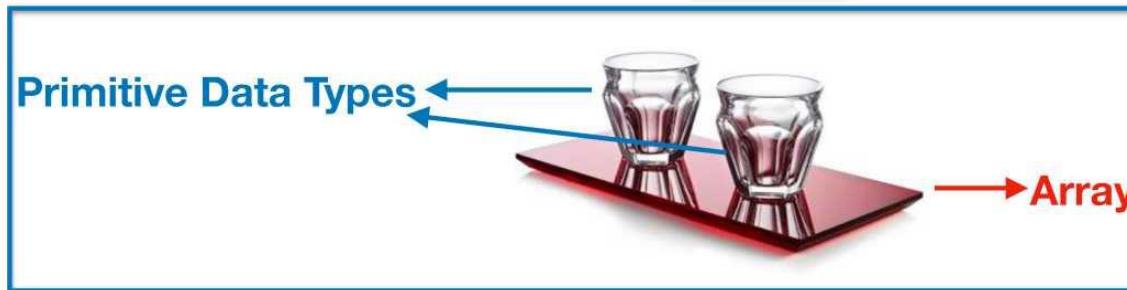
## Scope Instance, Class ve Local Variables

```
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?

# 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

- 3) Bir Array olusturmadan once o Array'in icine kac variable koyacagimiza karar vermeliyiz.
- 4) Bir Array icine koyabilecegimiz variable sayisina o Array'in "length" i denir. O Array icine length'den fazla variable koyamayiz.



Maximum capacity (length) = 2



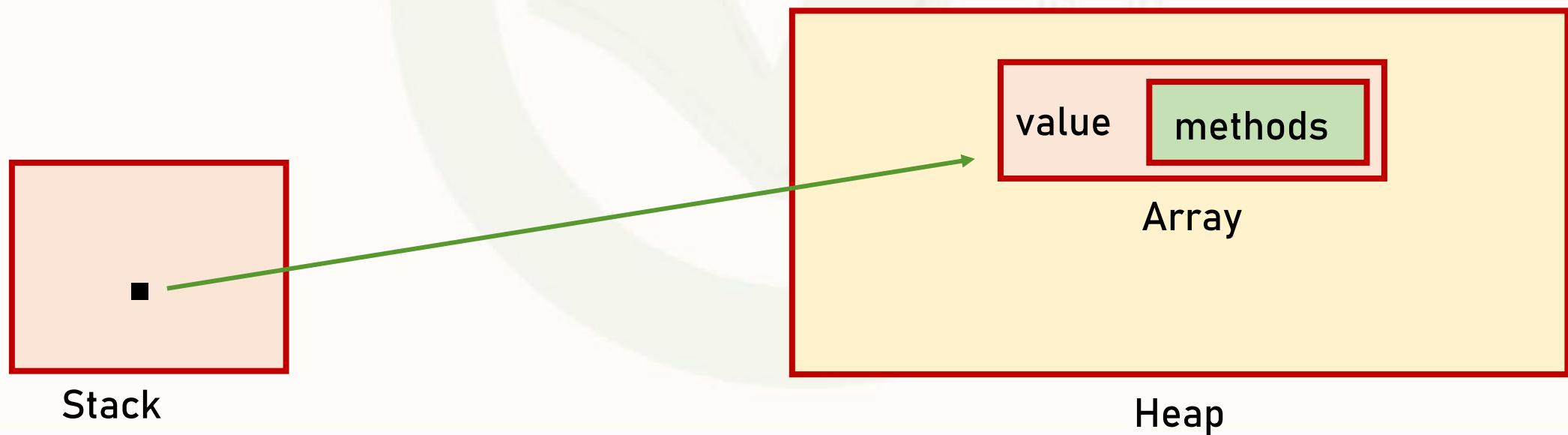
Maximum capacity (length) = 5

# 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 satirda yapariz.

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

## Arrays

9) Array'in elemanlarina nasil ulasilir ve nasil update edilir ?

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

Array elemanlarina index'ler kullanilarak ulasilir.

myArray[0] ==> 9,

myArray[1] ==> 10,

myArray[2] ==> 11,

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

NOT 2 : Bir Array'de olmayan index'i kullanmak isterseniz  
“**ArraysIndexOutOfBoundsException**” alirsiniz.

Soru 2: Soru 1'deki elemanlardan “Ali” yerine “Can”, “Ayse” yerine “Gul” atayin.

# 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 gerekir.

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.

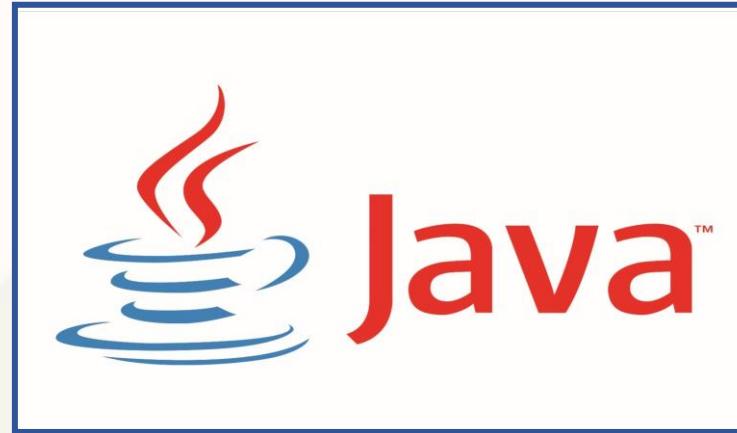
# Arrays

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 siralamayi ters cevirmek icin loop kullanilir



21 KASIM 2021  
DERS 22

## Arrays

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Scope : Bir variable'in class icerisinde ulasılıbildiği ve ulasılmadığı alanlara o variable'in scope'u denir.
- 2) Temelde iki tur scope vardır
  - Class level olanlar

Class level'da oluşturulan variable'lara değer atamak zorunda değiliz, biz değer atamışsa Java o değeri oluşturulan tüm objeler için ilk değer olarak kabul eder(initialize), biz değer atamazsa Java data turune uygun olarak default değerler atar

\* Instance variables (object variables) : class icerisindeki static method'lardan (main method da dahil) instance variable'lara object oluşturarak ulaşabiliriz, ve object üzerinden yapılan atamalar sadece o objeyi bağlar, diğer objeleri etkilemez. ( okul orneginde öğretmenin dersi, öğrencinin matematik notu gibi..).

instance variable'lara static olmayan method'lardan direkt ulaşabiliriz.

\* static variables (class variables) : static variable'lara class icerisinde her yerden ulaşılabilir (static veya static olmayan tüm method'lardan) .

static variable'ların değeri tüm objeler ve tüm method'lar için ortaktır. Projemiz çalışmaya başladıkta nm sonra static variable'larda yapılan her değişiklik kalıcı olur. Bir method'da static variable'in değerini değiştirdiğimizde, diğer method'lar ve objeler için de static variable'in değeri değişmiş olur.

Okul ornegini düşünürsek okul'un adı, adresi gibi herkes için ortak olan tanımlamalardır

## Onceki Dersten Aklimizda Kalanlar

- local olan scope'lar : Local variable'lar olusturuldugunda deger atamasi olmasa da java CTE vermez. Ancak Java local variable'lara default deger atamadigi icin, biz emniyette kalmak adina local variable'lara olusturulugu anda kod akisini engellemeyecek bir deger atamayı tercih ederiz.

Local variable'lar static olarak tanimlanamazlar, static tanimlamak istedigimiz variable'lari class level'da olusturmak zorundayiz.

\*Local variables : local variable'lar bir method icinde olup, loop icinde olmayan variable'lardir ve sadece olusturulduklari method icerisinde gecerlidirler.

\*Loop variables : Loop variable's adindan anlasilacagi gibi bir loop'un icerisinde olusurulan variable'lardir ve sadece o loop icinde gecerlidirler.

Icinde bulunduklari loop'un disindan erisilemezler.

- 3- Arrays : Array String disinda ogrendigimiz ikinci non-primitive data turudur. Non-primitive oldugu icin bircok method'u vardir. Ayrca array'ler ile kullanilmak üzere bir çok faydalı method icin kullanabilecegimiz Arrays class'ı da mevcuttur.

- Array'ı olusturmak icin iki tur yontem var

- 1) once array'in uzunlugunu ve data turunu declare ederek array'ı olusturur, sonra istedigimiz index'deki degerleri kendimiz atariz. Deger atamadigimiz index'lere java default degerleri atar

- 2) esitligin sol tarafinda array'ı declare eder, saginda ise {icinde} direkt degerleri yazariz

## 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.

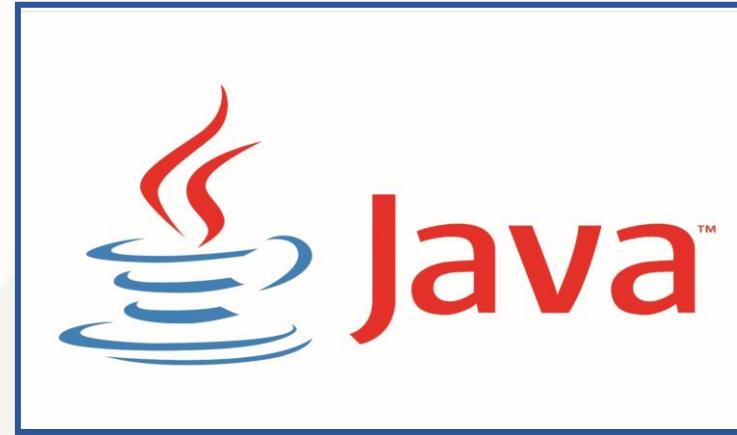
```
String str = "Java ogrenmek, IT alaninda yer edinmek demektir.";  
  
String arr1[] = str.split(",");  
System.out.println(Arrays.toString(arr1));  
                                → [Java ogrenmek, IT alaninda yer edinmek demektir.]  
String arr2[] = str.split(" ");  
System.out.println(Arrays.toString(arr2));  
                                → [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.



22 KASIM 2021  
DERS 23

## Multi Dimensional Arrays

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Multi Dimensional Arrays (Cok Katli Array'ler)

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*  
*parent array*           *parent array*  
*grand parent array*

# Multi Dimensional Arrays (Cok Katli Array'ler)

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

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

```
int arr[][]= new int [3][2];  
  
arr[0][0]=1;  
arr[0][1]=2;  
  
arr[1][0]=3;  
arr[1][1]=4;  
  
arr[2][0]=5;  
arr[2][1]=6;  
  
System.out.println(Arrays.toString(arr[0]));  
System.out.println(Arrays.toString(arr[1]));  
System.out.println(Arrays.toString(arr[2]));  
  
System.out.println(arr[0][1]);  
System.out.println(arr[2][0]);  
  
System.out.println(Arrays.toString(arr));
```

Multi Dimensional Array olusturma

Array icindeki elemanlara deger atama

Inner Array'leri yazdirma

Belirli bir elemani yazdirma

[[I@15db9742, [ I@6d06d69c, [ I@7852e922]

# Multi Dimensional Arrays

## (Cok Katli Array'ler)

Multi Dimensional Array'in tum elemanlari nasıl yazdirilir ?

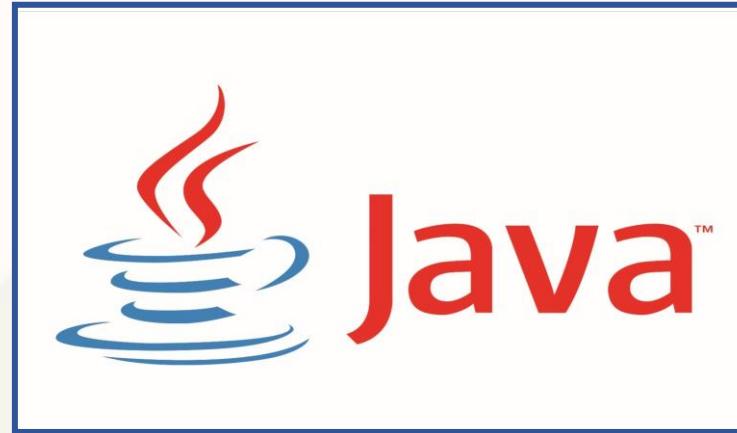
```
public static void main(String[] args) {  
  
    int arr[][] = { {1,2} , {3,4}, {5,6}};  
  
    for (int i = 0; i < arr.length; i++) {  
  
        for (int j = 0; j < arr[i].length; j++) {  
  
            System.out.print(arr[i][j] + " ");  
        }  
    }  
  
    System.out.println(Arrays.deepToString(arr));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

→ Nested For Loop kullanilabilir

→ Arrays Class'indan method kullanilabilir

## Multi Dimensional Arrays (Cok Katli Array'ler)

- 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 elemanların 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 cümle 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



23 KASIM 2021  
DERS 24

## ArrayLists

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# 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 gereklidir, ArrayList de gerekmeyez.
- Bir array'den bir eleman silmek istedigimizde yeni bir array olusturmamız gereklidir, 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 dondurur

## 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

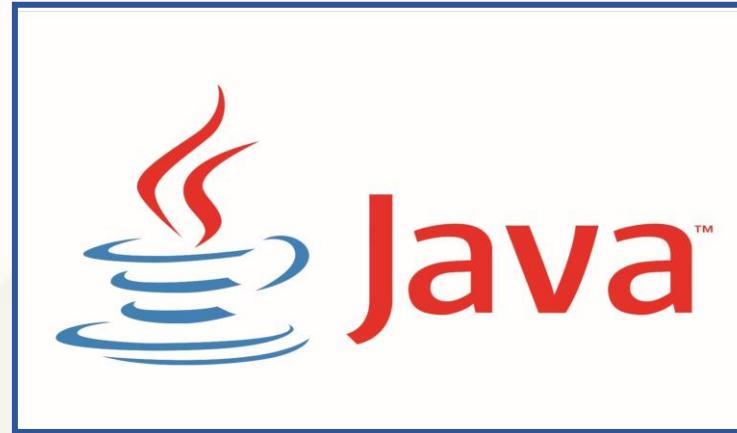
B) `remove("eleman")` index'i degil elemani kullanırsak kullandığımız elemanın ilk kullanıldığı yeri bulur ve siler.

```
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, kedi]
```

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

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

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



24 KASIM 2021  
DERS 25

## ArrayLists

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## 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
```

## 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 olmadığını kontrol eder. Aynı indexteki tüm elemanlar aynı ise true return eder, farklı 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
```

## Sorular

## ArrayList

- 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

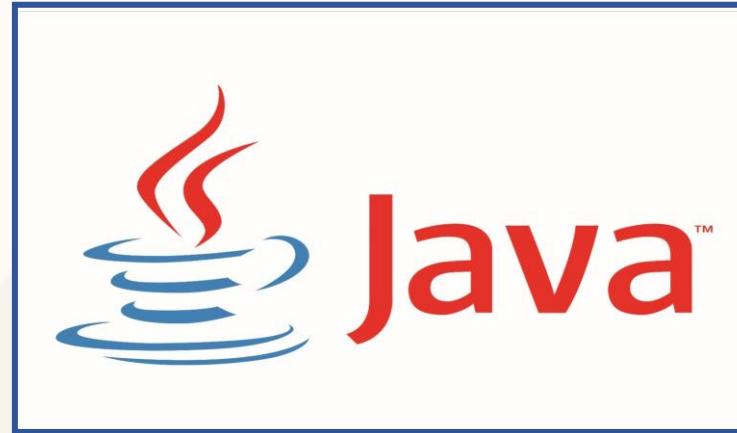
```
String arr[ ] = list.toArray(new String[0]);
```

```
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]
```



26 KASIM 2021  
DERS 26

For-Each Loop  
Constructor

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## 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  
mumkun değildir

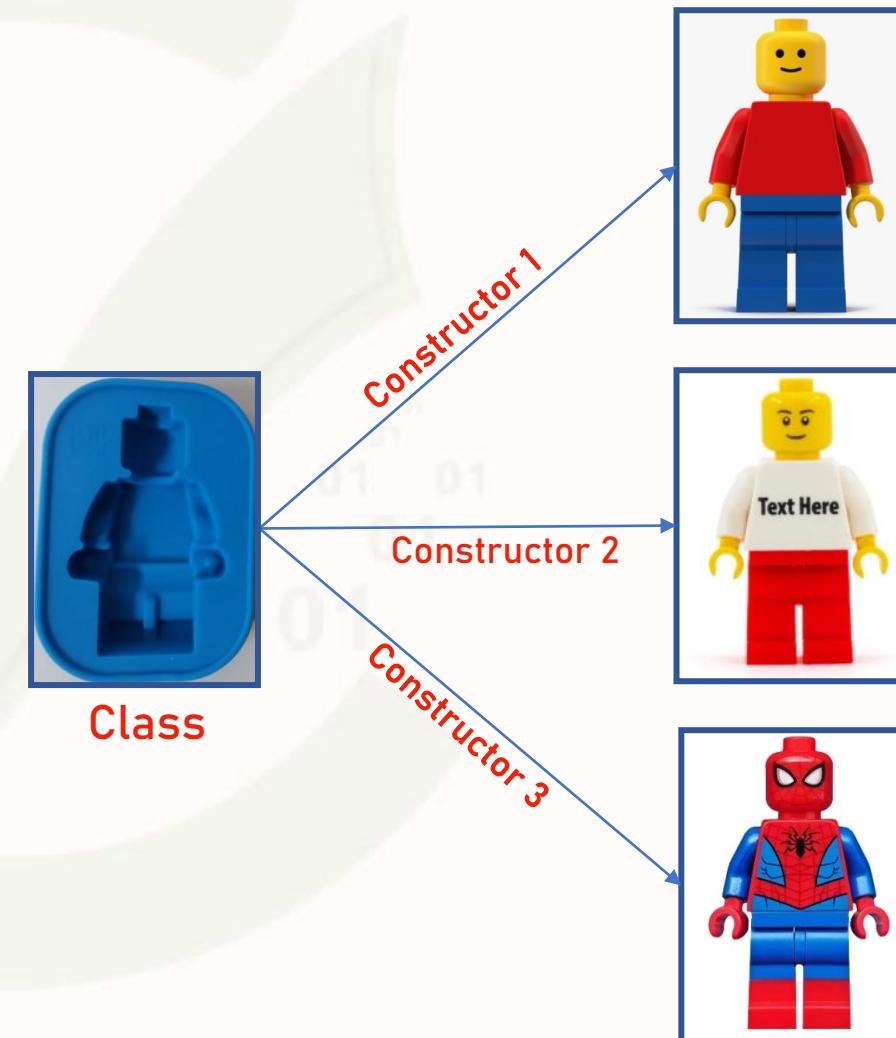
**Constructor** nedir ?

**Constructor** Method değildir.

**Constructor** variable değildir.

**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");
```

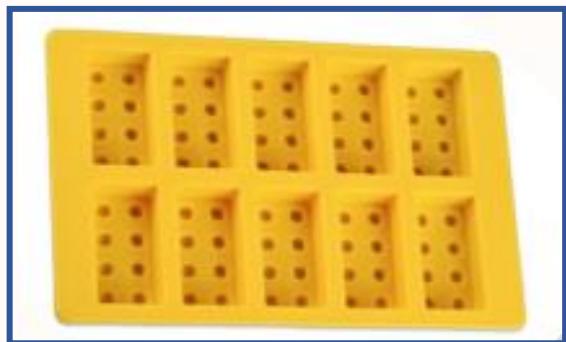
Bana yesil, v yaka, medium tisortlerden 100 tane uret

```
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



# 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.

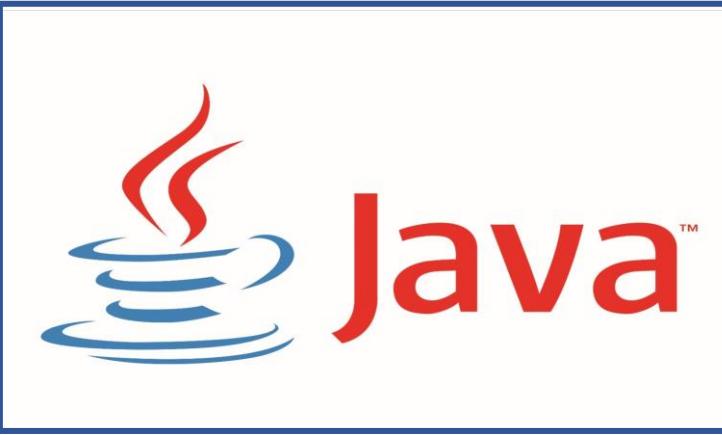
Default constructor Class içinde görülmeyecektir

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) {  
    }  
}
```



27 KASIM 2021  
DERS 27

**Constructor**  
**Constructor Call**

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Constructor : Java'da obje uretmemizi saglayan kod bloklaridir.
- 2) Constructor bizim su ana kadar ogrendigimiz yapilardan farklidir. En çok method'a benzer fakat 2 farkliliği vardır
  - Return type yoktur
  - Class ismi ile ayni isimde olma mecburiyetinden dolayi ismi buyuk harfle baslar
- 3) Bir class olusturdugumuzda Java default constructor (parametresiz constructor) olusturur ve class'a koyar. Ancak default constructor gorunmez.
- 4) Eger biz parametreli veya parametresiz herhangi bir constructor olusturursak Java default constructor'i siler.  
Eger herhangi bir constructor olusturmamız gerekirse, default constructor silindiğinden dolayi onun yerine parametresiz bir constructor da yazmamız tavsiye edilir
- 5) Biz baska class'dan parametresiz constructor ile bir obje uretirsek objemiz atanmis degerler varsa o degerleri, eger deger atanmamis ise default degerleri alır.  
Parametresiz constructor ile olusturdugumuz objeye istedigimiz degerleri atamak icin her bir ozellik icin assignment yapmam gerekir.  
Eger obje uretilirken istedigim degerleri atayarak olusturulsun diyorsaniz parametreli constructor kullanmalisiniz.

# Constructor

## (Java object'leri nasıl oluşturur ?)

Bir Car Class'i oluşturalim.

Arabaları oluştururken özelliklerini yükleyebilmek için variable ve method'lar oluşturabiliriz

Bizim class'imızda

- İlan no
- Marka
- Model
- Yıl
- Fiyat gibi variable'lar ve
- hız(120) ve yakit(dizel) gibi iki de method olsun.

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

# 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 yakin 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 Constructor (Parametreli Constructorlar)

- 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

# Constructor

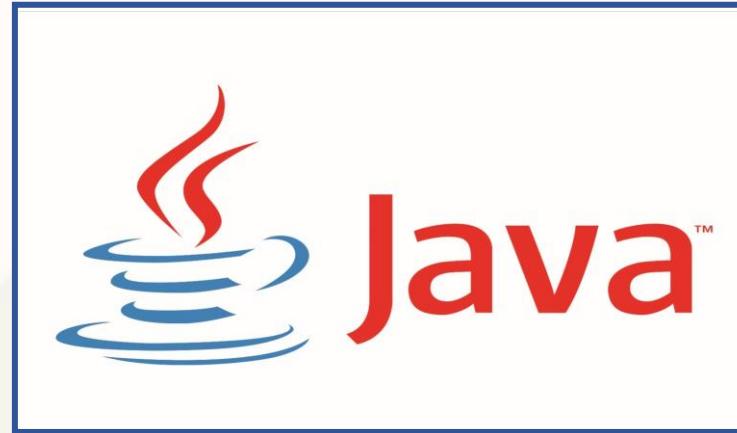
Asagidaki class calistirildiginda output ne olur ?

```
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);  
    }  
}
```



**29 KASIM 2021  
DERS 28**

**Constructor Call  
Static Keyword**

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## 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.

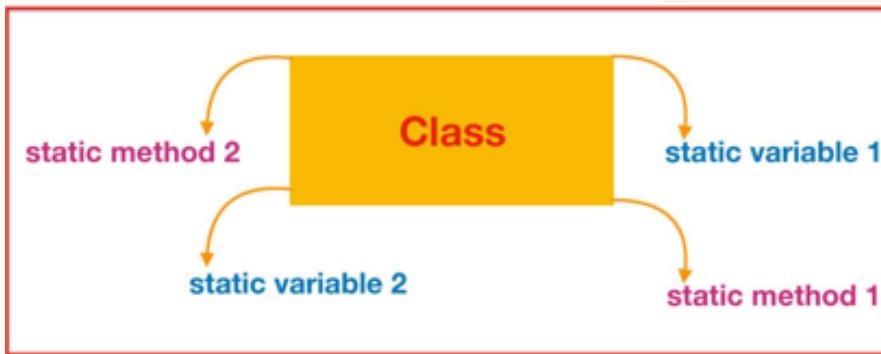
## Constructor Call

```
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++;  
    } }
```

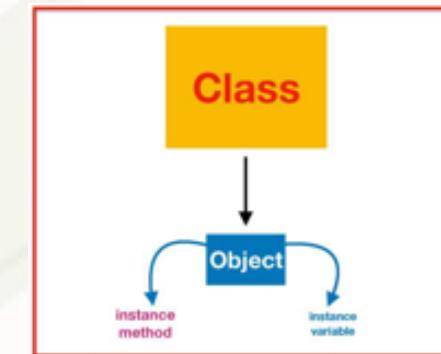
Boslukları "True" veya "False" olarak doldurun.

- 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 ....

# Static Keyword



Static variables / methods



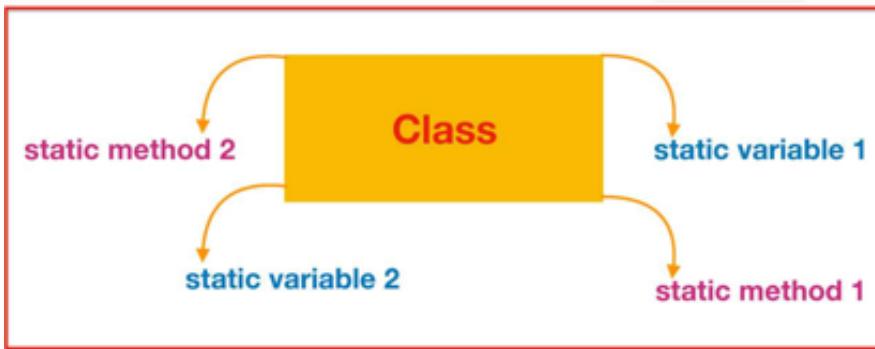
Non-static variables / methods

**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 olmayanları object olusturmadan kullanamazlar

## Static Methods

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

```
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());  
    }  
}
```

## 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 ?

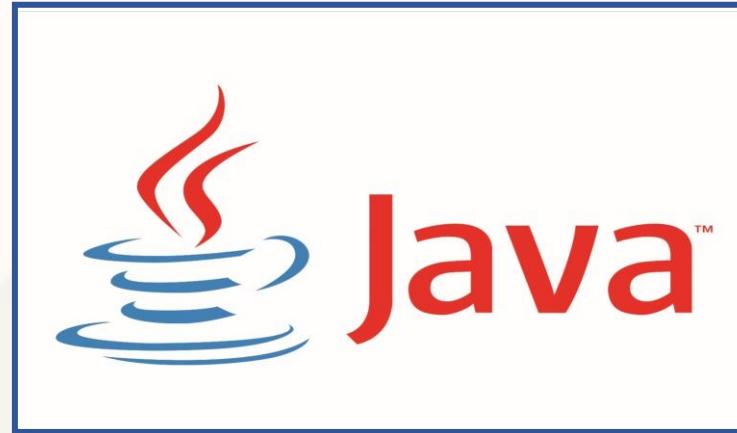
```
class Student{  
  
    int number;  
    String name;  
    static String college ="ITS";  
  
    Student(int r, String n, String college){  
        this.number = r;  
        this.name = n;  
        this.college = college;  
    }  
  
    public static void main(String args[]){  
  
        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.



30 KASIM 2021  
DERS 29

**Static Blocks**  
**Pass By Value & Pass By Reference**  
**Mutable & Immutable Classes**

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## 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 atanen 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 nasıl 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 fiyati 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 artirmayı 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

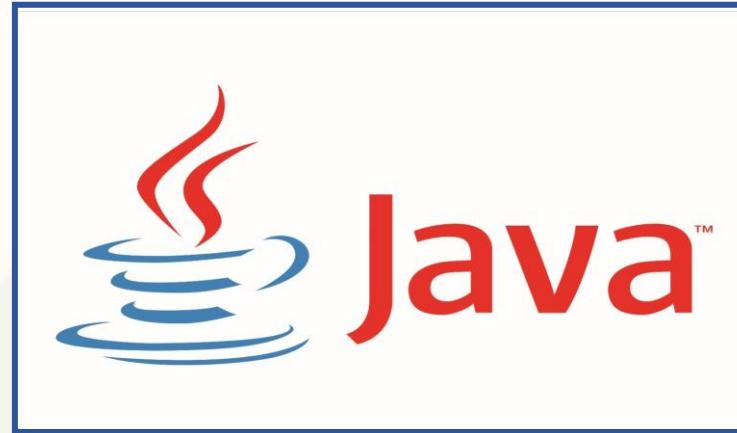
```
public static void main(String[] args) {  
    List<Integer> list =new ArrayList<Integer>();  
    list.add(10);  
    list.add(11);  
    list.add(12);  
  
    degistir(list);  
    System.out.println(list);  
    degistir2(list);  
    System.out.println(list);  
}  
public static void degistir(List<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]
}
```



1 ARALIK 2021  
DERS 30

**Mutable & Immutable Classes  
Date && Time**

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# Mutable & Immutable Classes

**Immutable** (değişmez) class'lar, objeleri bir kez oluşturulduktan 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 niacin ihtiyacımız olur diye bir soru sorarsak,

cevabı **thread safe (guvenli es zamanlı calisma)** 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 kullanican 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 sekilde 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

## Sorular

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

## Sorular

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);
```

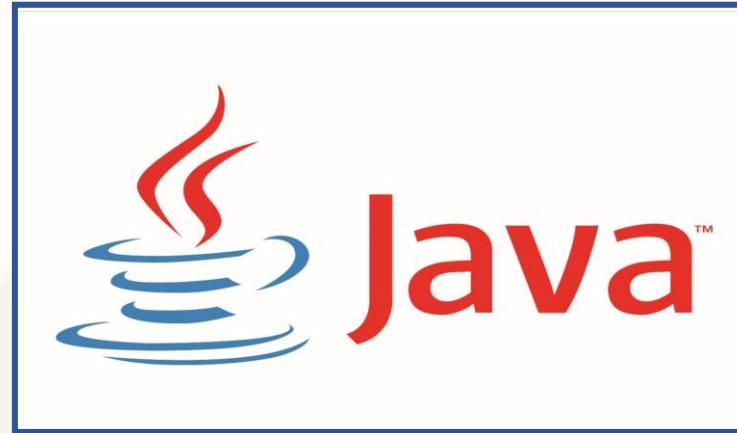
- A.** onetwo
- B.** onetwo7
- C.** onetwo followed by an exception
- D.** Compiler error on line 9.
- E.** Compiler error on line 10.

# Mutable & Immutable Classes

## Sorular

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.



2 ARALIK 2021  
DERS 31

Date & Time  
Varargs

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Pass By Value ve Pass By Reference : Bu bizim verecegimiz bir karar degil, her yazılım dili kendisinin PBV veya PBR olmasi kopnusunda bir karar verir ve bunu kullanicilara deklare eder.

Burada bize dusen kullandigimiz dilin tercihini bilip kodlarimizi ona gore yazmaktir.

Java PBV'u tercih etmistir.

PBV : Java main methodla diger method'lar arasında veya method'larin kendi aralarinda bilgi alis verisi yapmak istedigi zaman variable in kendisini degil degerini(value) gonderir. Boylece asil variable'in degeri degismemis olur

Eger gonderilen data non-primitive (list ve array gibi) ise objenin kendisine yeni deger atayamayiz ama List ve Array 'in elamanlarini degistirebilirsiniz

- 2- Mutable ve Immutable Class'lar : Bazi classlar ve o class'lardan uretilen objelerin degerleri degistirilemez.

Bu objelerde yeni deger atamasi yaparsak Java eski objenin degerini degistiremedigi icin yeni bir obje olusturup, yeni degeri yeni objeye atar, eski objenin referans ile bagini kesip, yeni objeyi referans ile baglar. Bu class'lara immutable denir

Immutable class'larin en meshurlari : String ve Wrapper class'lardır

mutable class'larin en meshurlari : list, array,

String havuzu : Java'da yeni String olusturmak icin new kelimesini kullanırsak, Java her seferinde oncelikle bir obje olusturur.

Ama simdiye kadar yaptigimiz gibi String str="Seyfullah"; yazarsak, Java once hafizada olan String'lerden bu degerde olan var mi diye bakar ve bulursa yeni bir obje olusturmak yerine buldugu objenin referansina yeni olusturulan objeyi de ekler. Bu durumda == true doner, digger durumlarda == false doner

## 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

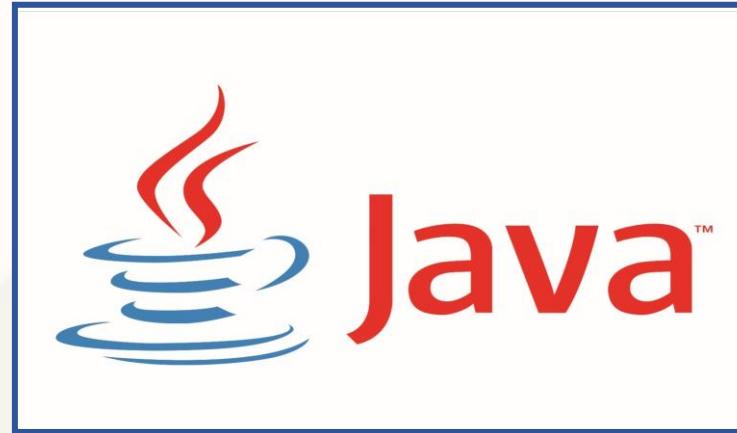
## Date Time Formatter

```
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  
}
```



3 ARALIK 2021  
DERS 32

**Varargs**  
**String Builder**

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## 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 espektir) 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

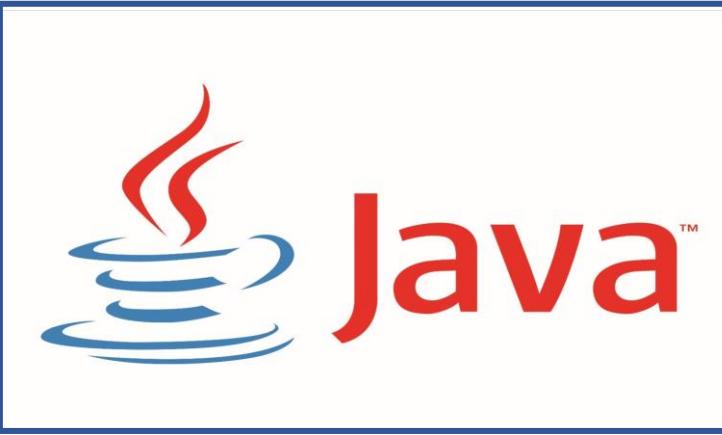
Which of the following compile? (Choose all that apply)

- 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");  
    }  
}  
  
public void Go(int x, String... y){  
    System.out.print(y[y.length-x] + " ");  
}  
}
```

TopJavaTutorial.com



4 ARALIK 2021  
DERS 33

**String Builder**

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# 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 degerlerinin karsilastirir.

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

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

# 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.

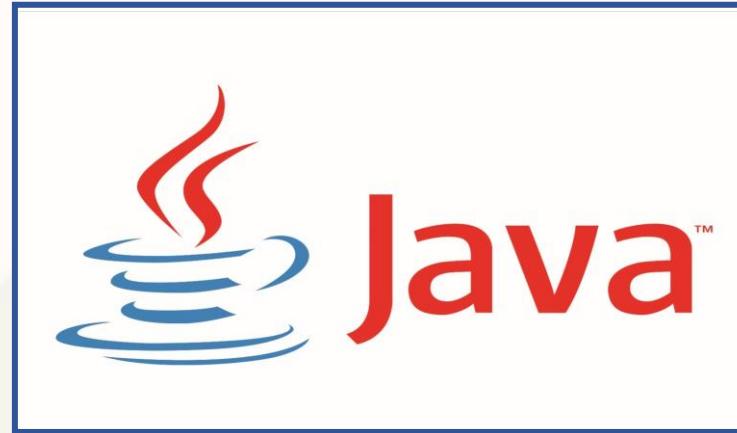
# String Builder

## 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.



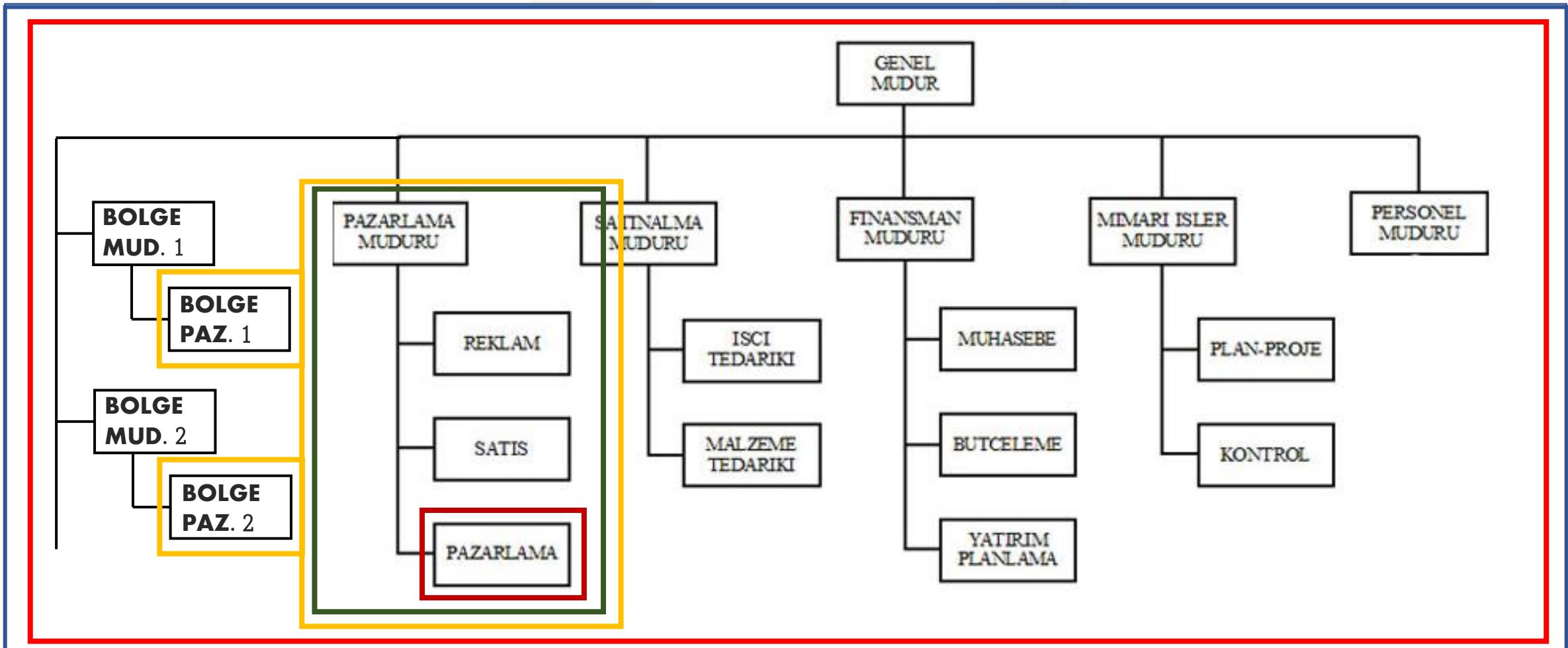
6 ARALIK 2021  
DERS 34

Access Modifier  
Encapsulation

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## 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.

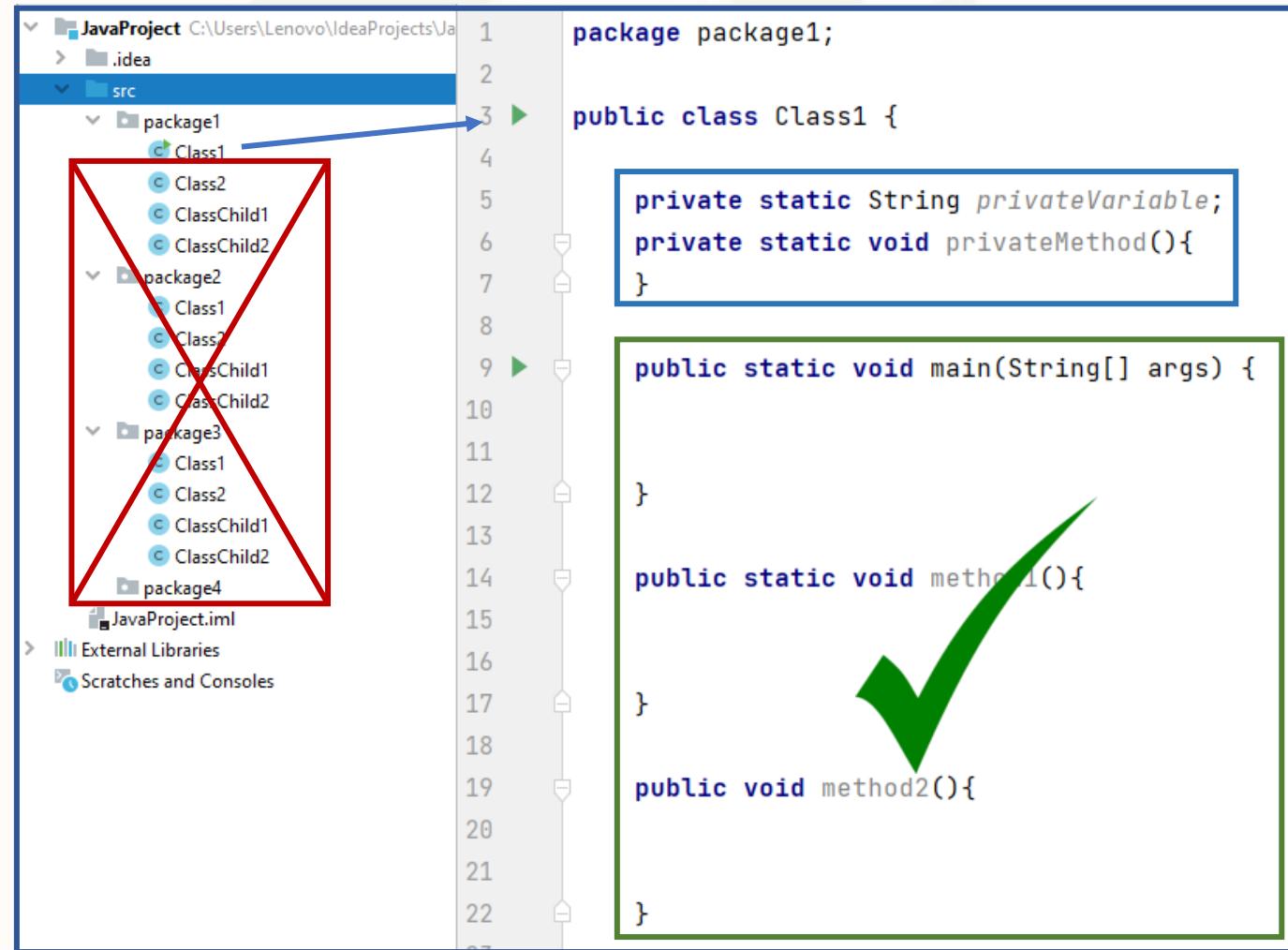
Java'da 4 Farklı access modifier vardır

- 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 olusturulduyu Class'da kullanilabilir



# Access Modifier

2) Default : Sadece olusturuldugu Class'in ait oldugu Package icinde kullanilabilir

The screenshot shows a Java project structure and a code editor side-by-side. On the left, the project tree displays packages package1, package2, package3, and package4 under the src folder. package1 contains Class1, Class2, ClassChild1, and ClassChild2. package2 contains Class1, Class2, ClassChild1, and ClassChild2. package3 contains Class1, Class2, ClassChild1, and ClassChild2. package4 is empty. A green box highlights package1, and a red diagonal line with a red X marks package2, package3, and package4, indicating that default access is limited to package1. On the right, the code editor shows:

```
1 package package1;
2
3 public class Class1 {
4
5     static String defaultVariable;
6     static void defaultMethod(){
7
8
9     public static void main(String[] args) {
10
11
12
13
14     public static void method1(){
15
16
17
18     public void method2(){
19
20
21
22     }
23 }
```

A blue box highlights the static members (defaultVariable and defaultMethod) and the main() method, while a green box highlights the method2() method, demonstrating that default access is limited to package1.

## Access Modifier

- 3) Protected : Olusturuldugu Class'in ait oldugu Package icinde ve baska package icindeki Child Class'larda kullanilabilir

The screenshot shows a Java project structure and a code editor. The project tree on the left has packages package1, package2, package3, and package4. package1 contains Class1, Class2, ClassChild1, and ClassChild2. package2 contains Class1 and Class2. package3 contains Class1, Class2, ClassChild1, and ClassChild2. package4 is empty. The code editor on the right shows a class named Class1 with the following code:

```
1 package package1;
2
3 public class Class1 {
4
5     protected static String protectedVariable;
6     protected static void protectedMethod(){
7
8
9     public static void main(String[] args) {
10
11
12
13
14     public static void method1(){
15
16
17     }
18
19     public void method2(){
20
21
22
23 }
```

Annotations in the code editor highlight specific parts:

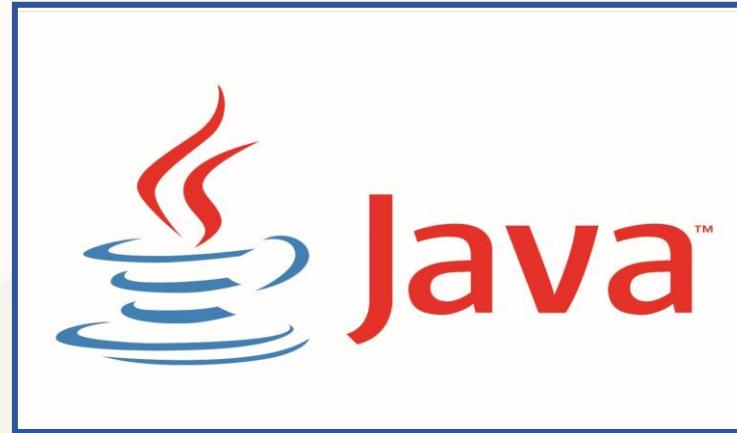
- A green box highlights the package declaration and the protected variable definition.
- A blue box highlights the protected method definition.
- A green box highlights the main method definition.
- A blue box highlights the method1 method definition.
- A green box highlights the method2 method definition.
- A red box with a slash through it highlights package2, indicating that Class1 and Class2 in package2 cannot access the protected members of Class1.
- A red box with a slash through it highlights package3, indicating that Class1 and Class2 in package3 cannot access the protected members of Class1.
- A green checkmark is placed next to package1, indicating that Class1 and ClassChild1/ClassChild2 in package1 can access the protected members of Class1.
- A green checkmark is placed next to package3, indicating that ClassChild1 and ClassChild2 in package3 can access the protected members of Class1.
- A green checkmark is placed next to the method2 definition, indicating that it is a valid declaration.

# Access Modifier

4) Public : Her yerden erisilebilir. Hicbir sinirlama (restriction) icermez.

The screenshot shows a Java code editor with a project structure on the left and code on the right. A green box highlights the package declaration, and a green checkmark points to the 'public' keyword in the class declaration. Another green box highlights the main method, and another green checkmark points to the 'public' keyword in the method declaration.

```
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
13
14     public static void method1(){
15
16
17
18
19     public void method2(){
20
21
22
23 }
```



7 ARALIK 2021  
DERS 35

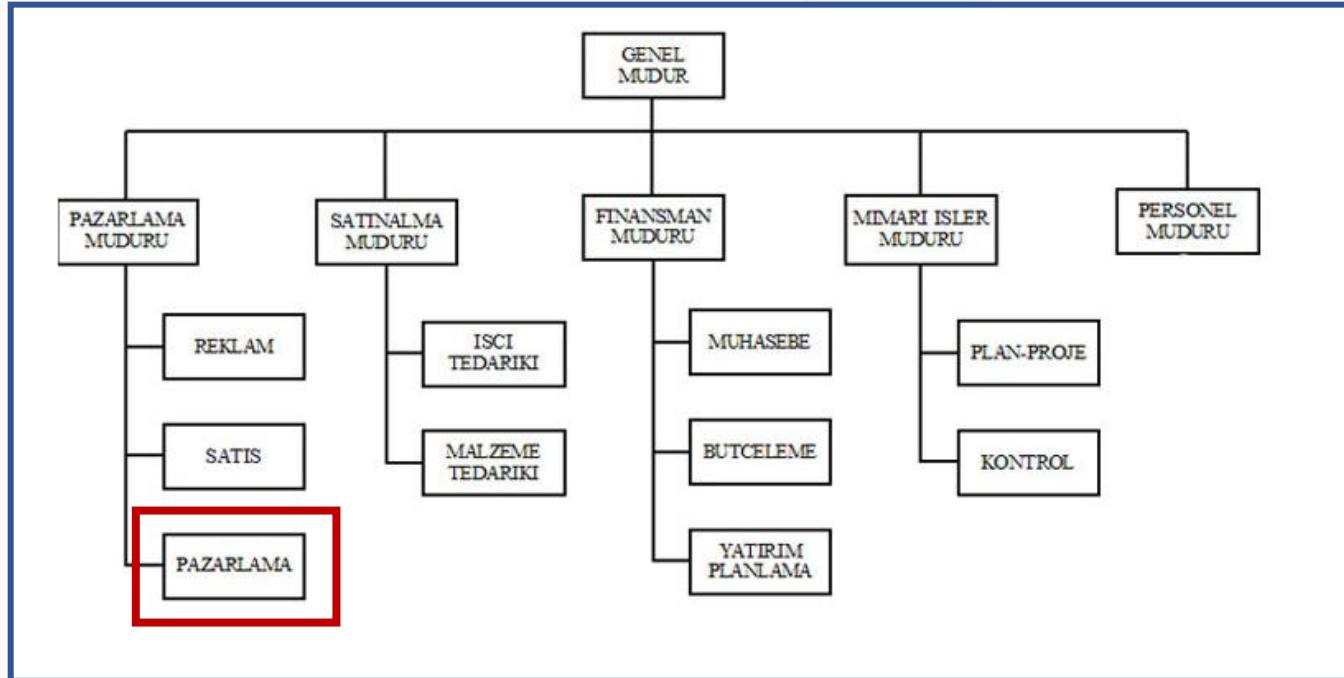
Encapsulation  
Inheritance

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) AccessModifiers : Bir Class'a veya class uyesine kimlerin ulasabilecegini veya hangi oranda ulasabilecegini sinirlasmak isterseniz access modifiers kullaniriz
- 2) Access modifier ile bir sinirlama yaptigimizda erisim ve degistirme yetkisini birlikte dusunuruz.
- 3) Eger Erisim ve degistirme konusunda birbirinden bagimsiz olarak yetkilendirme yapmak isterseniz getter ve setter method'lарine kullaniyoruz.
- 4) Getter ve setter method'lariyla yaptigimiz duzenleme tum proje kapsami icin aynidir. Class'lara veya package'lara gore bir yetkilendirme olmaz

# Encapsulation (Data Hiding)



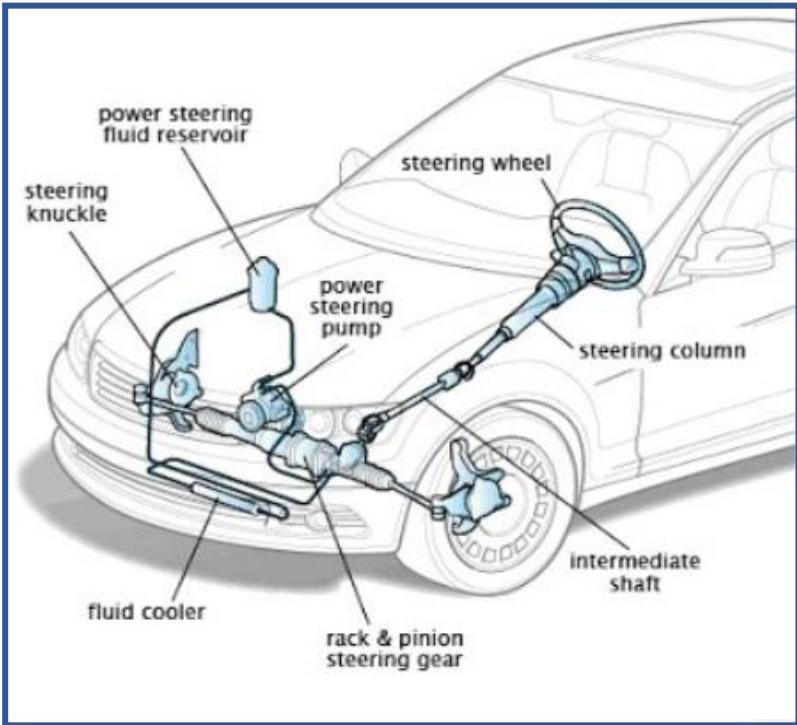
Pazarlama birimi olarak rapor düzenliyoruz  
İhtiyaclarımız

- 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 oluştururken class disindan bu öğelere erisimi kisitlama veya belirlenmis dataları degistirebilme olanağı tanır.

# Encapsulation (Data Hiding)

Before Encapsulation



After Encapsulation (Data Gizleme)



# Encapsulation

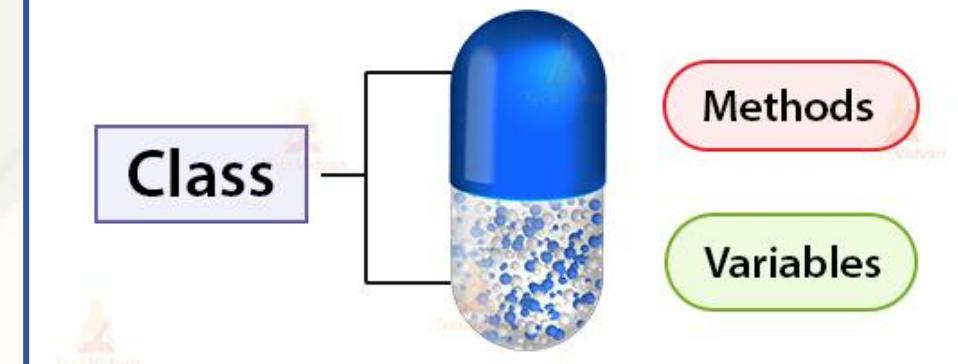
**Encapsulation**, onemli Class'ın üyelerini korumak için uygulanan data saklama yöntemidir.

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 in Java



# Encapsulation

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 üretmelısınız.

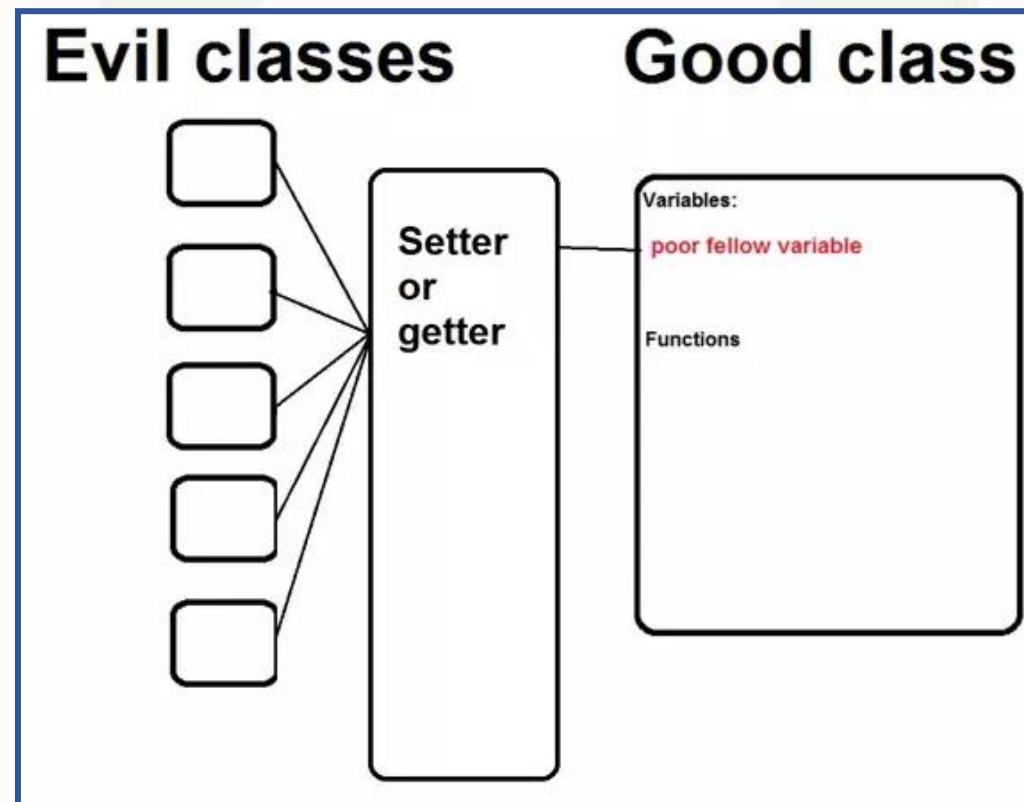
**Not:** getter() data'yı sadece okumamiza yarar, data'da değişiklik yapamaz.

**Not:** setter() başka Class'larda oluşturulan objeler için data değerini değiştirmemizi sağlar.

# Getters & Setters



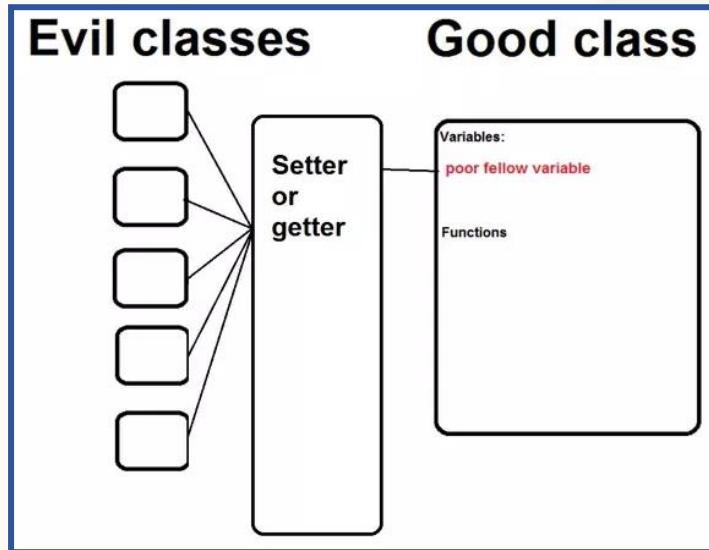
Getters and setters  
lead to the dark side...



# Encapsulation



Getters and setters  
lead to the dark side...



**Getter Methods :** Encapsule ettigimiz class  
uyelerinin okunabilmesine izin verir.

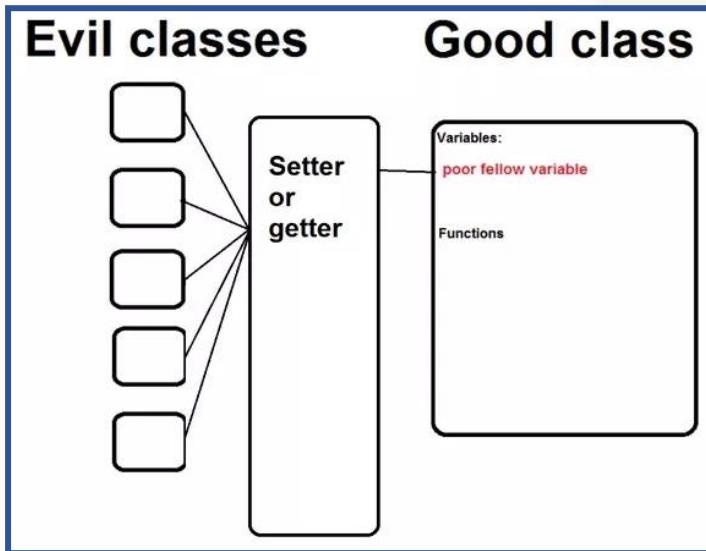
```
public class Example {  
  
    private String tcNo= "12345678901";  
  
    public static void main(String[] args) {  
    }  
  
    public String getTcNo() {  
        return tcNo;  
    }  
}
```

Eger sadece getter method olusturulursa data  
degerleri degistirilemez sadece okunabilir. Bu  
tarz class'lara **immutable** class denir.

# Encapsulation



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;
}
```

- 2) Data type'lari boolean olmayan variable'larin getter metod isimleri “get” ile baslar.

```
private int a=10;

public int getA() {
    return a;
}
```

# Getters & Setters (Java Beans)

İsim verme kuralları (Naming convention)

- 3) Setter method isimleri her zaman “set” ile baslar

```
private int a=10;
private boolean happy;

public void setA(int a) {
    this.a = a;
}

public void setHappy(boolean happy) {
    this.happy = happy;
}
```

# Tekrar Soruları

1) Encapsulation nedir?

Encapsulation, hassas dataları korumak için kullanılan data saklama yöntemidir

2) Dataları nasıl saklarız?

Data'ları private access modifier kullanarak saklarız.

3) Saklanan datalara diğer class'lardan ulaşabiliriz?

Getter ve setter method'larını kullanarak ulaşabiliriz.

4) getter() method'u ne yapar ?

Saklanan dataları okumamızı sağlar.

5) setter() method'u ne yapar ?

Saklanan dataları obje üzerinden update edebilmemizi sağlar

6) immutable class nedir?

Encapsule edilen bir class'da sadece getter method'u oluşturursak dataları okuyabiliriz ama degistiremeyiz. Bu tur class'lara immutable class denir.

7) setter() method'ları için naming convention nedir?

Tüm data türleri için isimler "set" ile baslar.

8) getter() method'ları için naming convention nedir?

Boolean data türü için "is" ile, diğer data türleri için "get" ile baslar.

# Getters & Setters (Java Beans)

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 (Java Beans)

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)

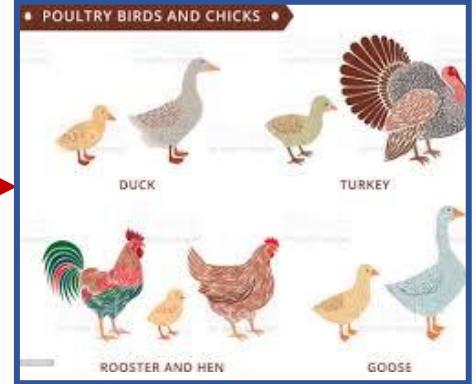


**Balıklar**

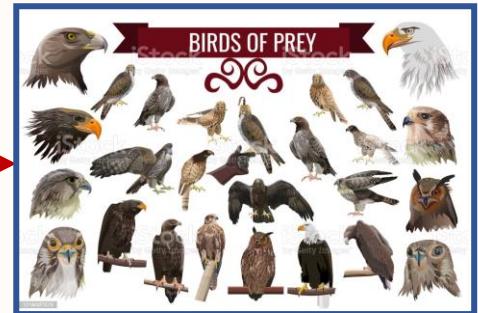
(Denizde yasar, solungacla nefes alır,  
yuzerek hareket eder)



**Kuşlar**  
(Kanatlari vardır,  
akcigerle nefes alırlar,  
gagalari vardır)



**Kumes Hayvanları**  
(ucamazlar,  
yuruyerek har.ederler)



**Avcı Kuşlar**

(ucarlar, et yerler,  
penceleri vardır)

# Inheritance (Kalitim - Miras)



Baba

Anne



Ali Can



Mert



Merve



Can



Ayse



yusuf

# Inheritance (Kalitim - Miras)

**Personel**  
Kisisel bil.  
Departman  
Izin

**Calisan**  
Mesai  
Avans  
Rapor  
Std.Ucret



Isciler

Beyaz Yakalilar

Yan Hizmetler

- Usta Bası  
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

# 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 dataları, objectleri, veya metodları 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

## Interview Question

Nicin Inheritance kullanırız ?

Inheritance sayesinde parent olarak tanımlanan class(ve onun parent class'larındaki) **protected/public** class üyelerini kullanabiliriz(reusability).

Inheritance'in faydalari nelerdir?

- 1:** Tekrarlardan kurtuluruz
- 2:** Daha az kod yazarak işlemlerimizi yapabiliriz
- 3:** Kolayca update yapabiliriz
- 4:** Application'in bakımı ve sürdürülmesi (maintenance) kolaylaşır

# Inheritance

```
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;  
    }  
}
```

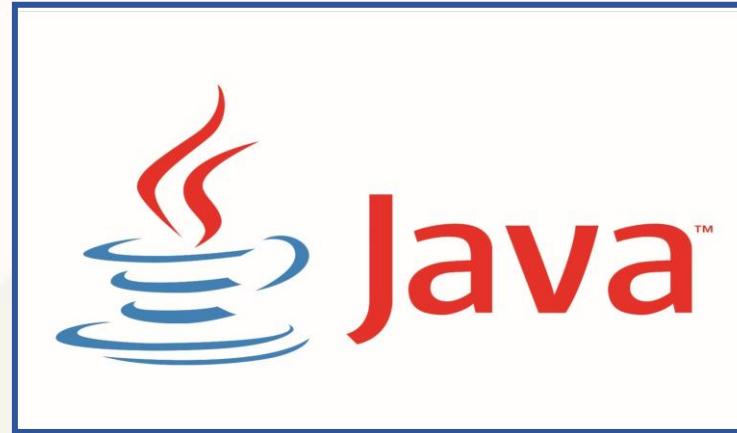
Child Class (Sub)    Parent Class (Super)

```
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);  
    }  
}
```

Child  
Class  
(Sub)

```
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)



8 ARALIK 2021  
DERS 36

Inheritance

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

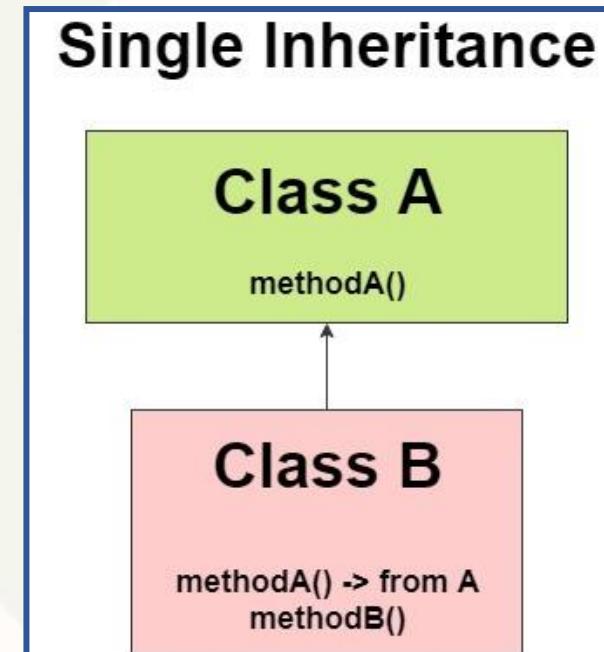
## Onceki Dersten Aklimizda Kalanlar

- 1) Inheritance : miras veya kalitim demekti
- 2) Java parent ve child class'lar arasında isleyisi inheritance kuralları ile belirlenmiştir.
- 3) Inheritance'in temel özellikleri
  - Child Class'lar parent class'daki protected veya public olarak tanımlanan tüm class üyelerini direk kullanabilirler
  - Obje'ler oluşturdukları class'a göre üst class'lardaki özellikleri inherit ederler
  - Child class'lar bazı özellikleri kendilerine göre adapte edebilirler
- 4) Bir class'i child yapmak için class isminin sonuna extends keyword ve parent class ismi yazılır
- 5) Inheritance'in faydalari nelerdir ?
  - Reusability
  - Az kod yazma
  - Update ve maintenance'i kolaylaştırır
  - Kodlarımızın anlaşılabilir olmasını sağlar

# Inheritance Türleri

## Single Inheritance

Java Single Inheritance kabul eder. Bir child class'in sadece bir tane parent class'i olabilir .

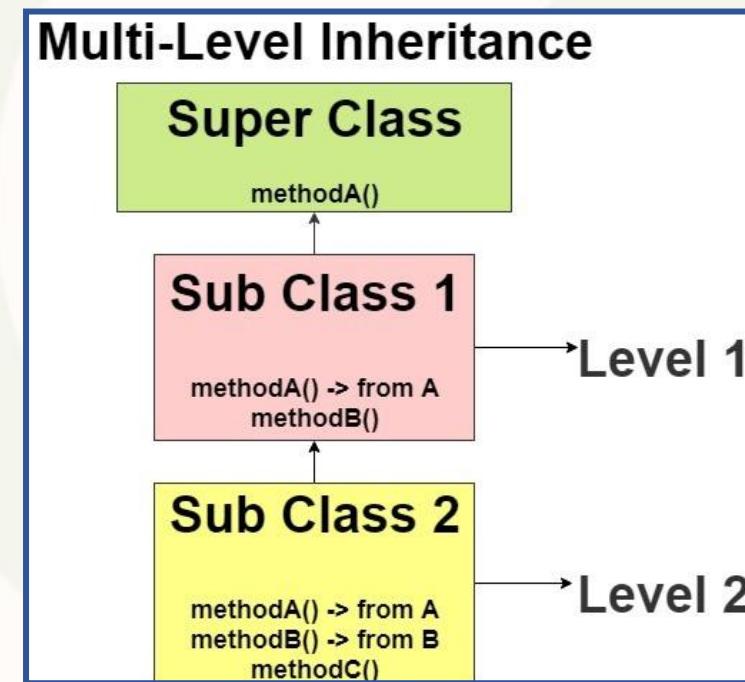


Bir cocugun ailesi bir tane olur

# 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).

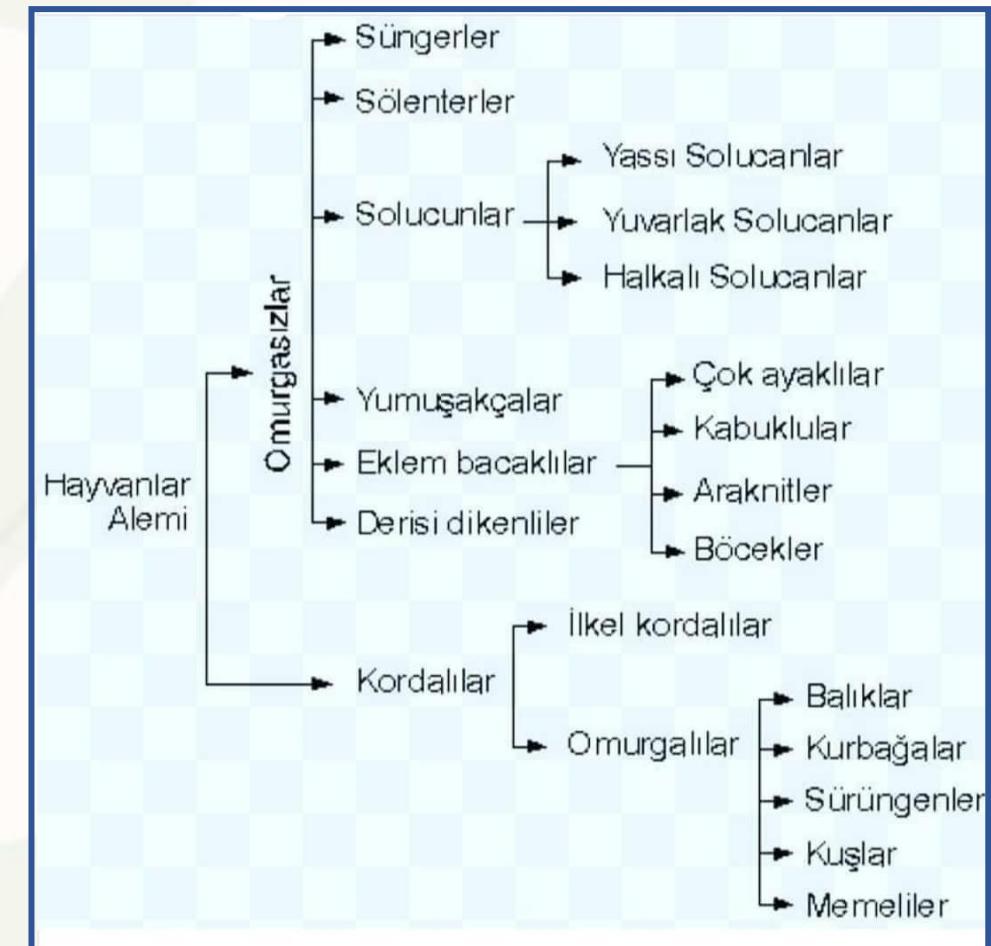
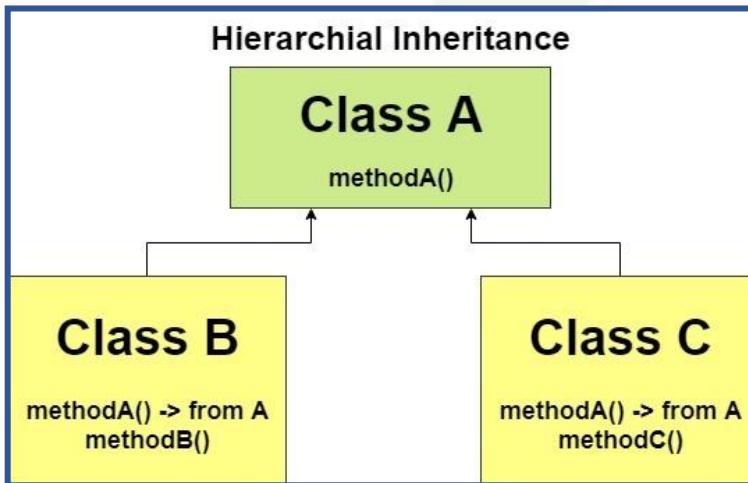


İnsanlardaki soy ağacı gibi, child class'in parent'i ve grand parent'leri olabilir.

# Inheritance Türleri

## Hierarchical Inheritance

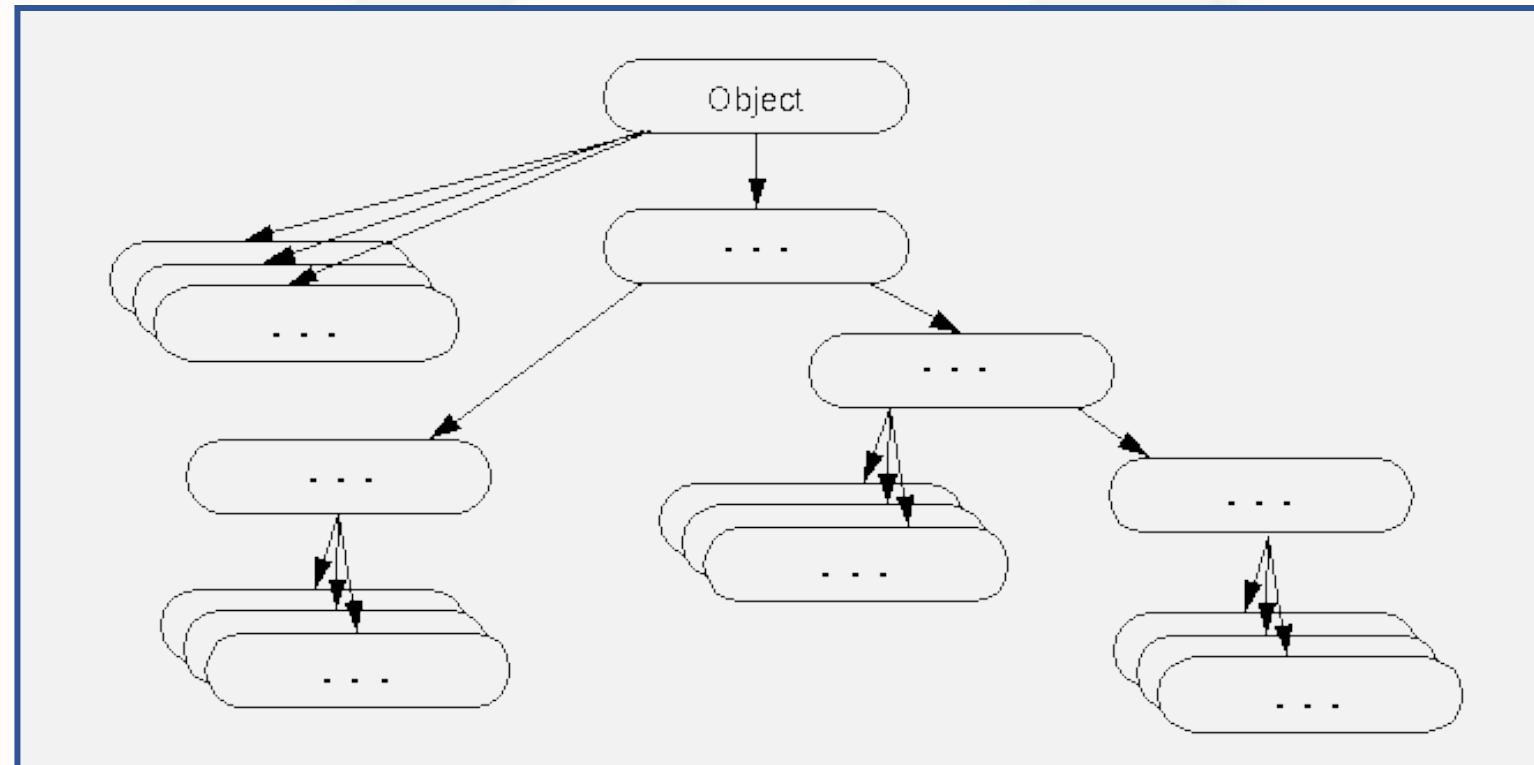
Birden fazla class aynı class'i parent olarak kullanabilir.



# Inheritance Türleri

**Java'da, butun class'lar Object Class'dan inherit ederler.**

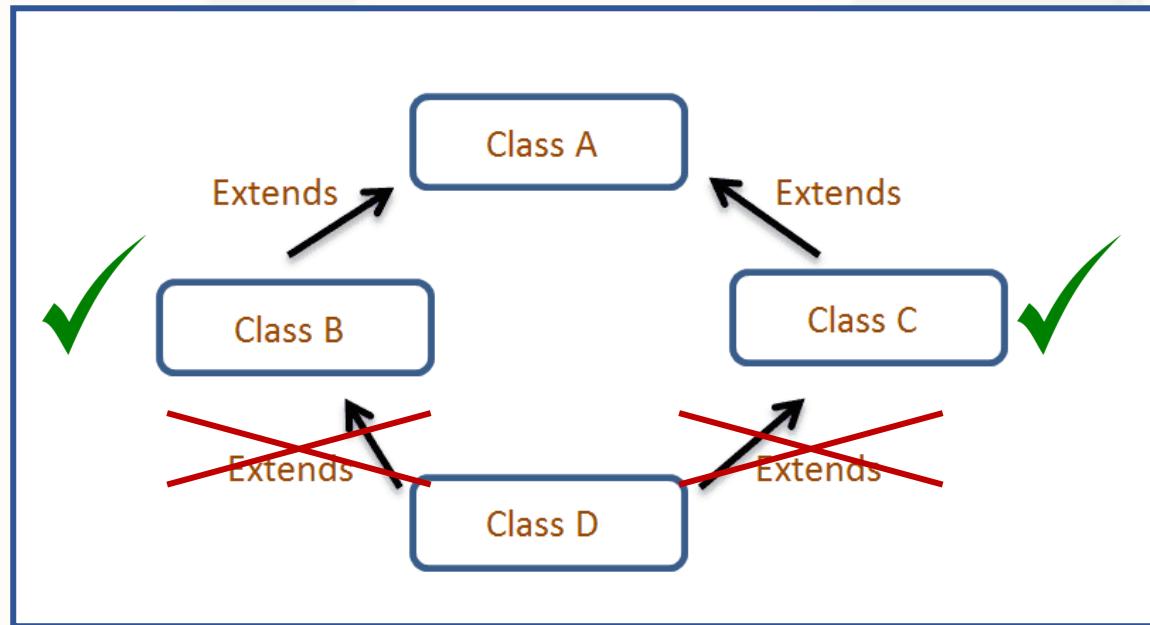
**Object Class** bütün class'ların parent'ıdır ve **Object Class** parent'i olmayan tek class'dır.



# Inheritance

## Multiple Inheritance

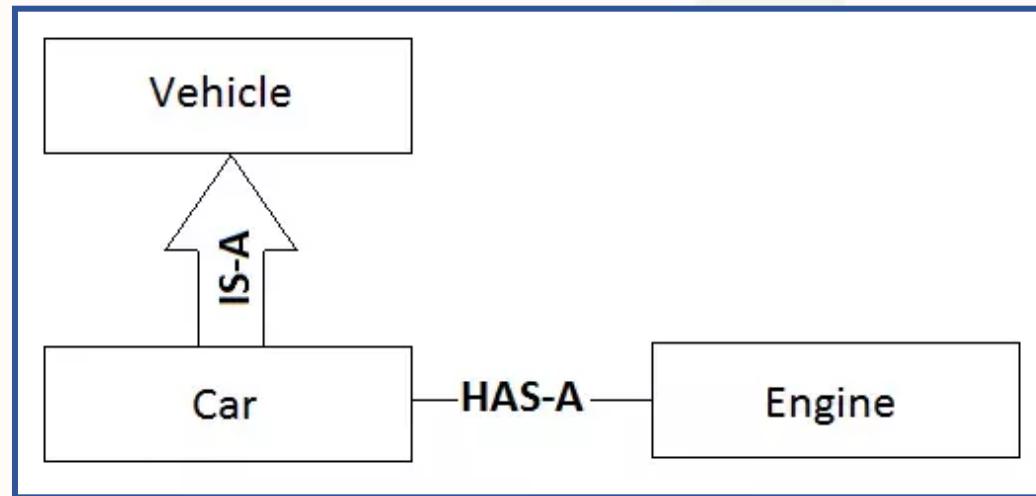
Bir class'in birden fazla class parent olarak kabul etmesi demektir,  
ancak Java multiple inheritance **KABUL ETMEZ**



## 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. Cunku 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.

Örnek 1:

```
public class Muhasebe extends Personel{  
    Muhasebe(){  
        System.out.println("Muhasebe constructor 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        Isci isci1=new Isci();  
11    }  
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{  
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 }  
16 }
```

# Inheritance'da Constructor Çağırma

- 2) super(); parent class'dan constructor çağırma için, this(); içinde olunan class'da başka bir constructor çağırma 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");  
    }  
}
```

extends

```
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?

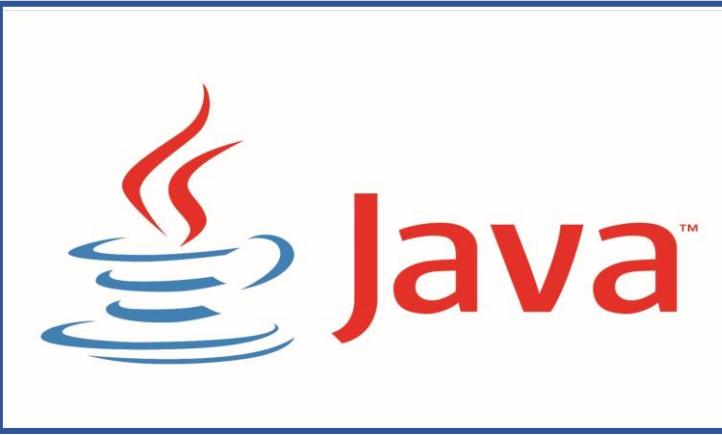
```
6 class Derived {  
7     public Derived(String temp) {  
8         System.out.println("Derived class " + temp);  
9     }  
10  
11    public class Test01 extends Derived {  
12        public Test01 (String temp) {  
13            System.out.println("Test class " + temp);  
14        }  
15    }  
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



9 ARALIK 2021  
DERS 37

## Inheritance

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Inheritance : miras veya kalitim demektir
- 2) Java parent ve child class'lar arasında isleyisi inheritance kuralları ile belirlenmiştir.
- 3) Inheritance'in temel ozellikleri
  - Child Class'lar parent class'daki protected veya public olarak tanımlanan tüm class üyelerini direk kullanabilirler

Bunu Java nasıl yapıyor ?
    - \* Child class'larda var olan her constructor'un ilk satırında super( ) olur
    - \* Eğer biz kendimiz super( ) veya super(parameter) yazarsak, java görünmez olarak kendisinin yazdığı super( )'ı siler
    - \* Bu sayede Java child class'da oluşturulan her bir obje'nin parent class'daki karşılıklarını oluşturmuş olur.
    - \* super ( ) kullanımı constructor'da gordugumuz this( ) benzer, ikisi de constructor'in ilk satırında olmak zorundadır, dolayısıyla ikisi birden çalışmaz
    - \* eğer yazdığımız veya java'nın oluşturduğu super( ) keyword ile çağrıdığımız constructor parent class'da yoksa CTE olur, kod çalışmaz
  - Objeler oluşturdukları class'a göre üst class'lardaki özelliklerini inherit ederler
  - Child class'lar bazı özelliklerini kendilerine göre adapte edebilirler

## Inheritance'da Class Uyelerini Çağırma

- “super.” keyword parent class’dan variable çağırma için kullanılır. “this.” keyword içinde bulunulan 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’ın 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 Uyelerini 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 Uyelerini 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

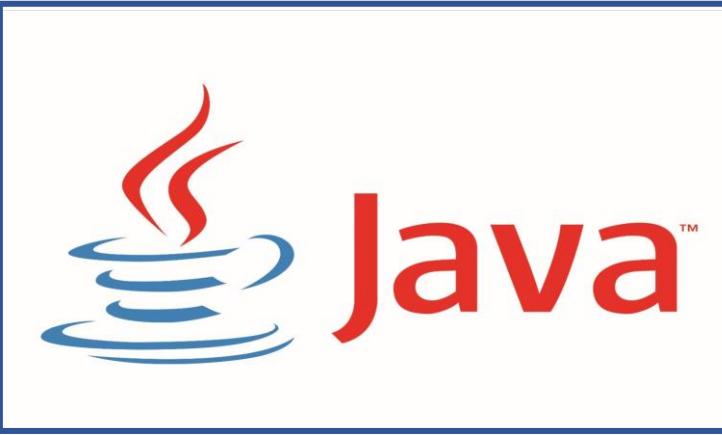
```
public static void main(String[] args) {  
    UstaBasi ub1 = new UstaBasi();  
    ub1.sorumluOlduguBirim="tamirhane"; // Ustabasidan  
    ub1.bolum="Tamirhane"; // Isciden  
    ub1.isim="Mehmet"; // Personelden  
  
    Isci ub2=new UstaBasi();  
    ub2.bolum="Atolye"; //Isciden  
    ub2.statu="Iisci"; //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.



10 ARALIK 2021  
DERS 38

Inheritance  
Overriding

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Inheritance : miras veya kalitim demektir
- 2) Java parent ve child class'lar arasında isleyisi inheritance kuralları ile belirlenmiştir.
- 3) Inheritance'in temel ozellikleri
  - Child Class'lar parent class'daki protected veya public olarak tanımlanan tüm class üyelerini direkt kullanabilirler
  - Obje'ler oluşturdukları class'a göre üst class'lardaki özelliklerini inherit ederler
    - \* Bazen child class'da oluşturduğumuz bir objenin parent class'daki genel özelliklere sahip olmasını isteriz
    - \* Bu durumda child class'da obje oluştururken constructor'u child class'dan, data turunu ise ortak özelliklerini almak istediğimiz parent class'dan seçeriz
    - \* Data turu ve constructor farklı class'lardan olduğunda objemiz data turu ve onun parent class'larındaki variable'lara ulaşabilir ancak data turundan aşağıda listelenenlere ulaşamaz
  - Child class'lar bazı özelliklerini kendilerine göre adapte edebilirler

## Tekrar Soruları

1- Inheritance'in avantajları nelerdir ?

- A) Reusability B) Maintenance C) Less Code

2- Bir Class'a Parent Class oluşturmak için Syntax nedir?

public class ChildClass{  
 extends ParentClass{}

3- Hangi access modifier'lar inherit edilebilir ?

public ve protected olanlar her yerden, default olanlar aynı paketten inherit edilebilir.

4- super() ile this()'in farkı nedir?

super() parent class'dan, this() ise içinde bulunan class'dan constructor çağırma için kullanılır

5- super() ile super.'nin farkı nedir?

super() parent class'dan constructor, super. ise variable veya method çağırma için kullanılır

6- this() ile this.'nin farkı nedir?

this() constructor, this. ise class variable veya method'u çağırma için kullanılır

## Tekrar Soruları

7- super ile this.'nin farkı nedir?

**super** parent class'dan variable veya method çağırma için kullanılır, this ise içinde bulunan class'da class level variable veya method'ları çağırma için kullanılır.

**this** ile parent class'dan da variable veya method çağrılabılır ancak aynı isimde bir variable/method hem içinde bulunan class'da hem de parent class'da olursa this parent class'da olanı değil içinde bulunan class'dakini çağırır.

Emin olmak için parent class için super kullanırız.

8- super() ve this() bulundukları constructor'da ilk sırada olmalıdır. **True** ~~False~~

9- super() ve this() bir constructor'da sadece 1 kere kullanılabilir. **True** ~~False~~

10- super() ve this() birlikte aynı constructor'da kullanılabilir. **True** ~~False~~

# Overriding

Aynı isimde farklı iki method oluşturmanın iki yolu vardır

**1- Method Signature'ini değiştirmek aynı isimde farklı iki method yapmak**

**Overloading**

**2- Method Signature'ini değiştirmeden iki method'dan sadece birinin çalışmasını sağlamak**

**Overriding**

**NOT 1:** Method Signature'ini değiştirmezsek 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 neden kullanılır ?

Overriding kullanarak, child class'in parent class'daki methodu **kendine uyarlayarak (implement)** kullanmasını sağlamış oluruz.

Overriding yapıldığında parent class'daki methoda **Overridden Method**, child class'daki methoda **Overriding Method** denir.

Eclipse menu'den Source sekmesinde bulunan **Override/Implement methods** seçenekleriyle otomatik olarak overriding method'u oluşturabiliriz. Bu şekilde yapılan işlemde Java @Override annotation'i kullanır.

@Override kullanmak zorunda değiliz, istersek silebiliriz. Ancak kodun anlaşılabilir ve okunabilir olması için değil, overridden method'da değişiklik yapıldığında Java'nın rapor etmesi için kullanılması tercih edilir.

# Overiding

## Overriding Methods

```
public class Isci extends Personel{  
  
    public static void main(String[] args) {  
  
        Isci isci=new Isci();  
        isci.isim="Mehmet";  
        isci.soyisim="Bulut";  
        isci.statu="isci";  
        System.out.println(isci.isim + " "+isci.soyisim+" "+  
        isci.statu+ " "+ isci.maasHesapla());  
  
    }  
  
    public int maasHesapla() {  
        return (30*8*20);  
    }  
  
    public void calismaSaati() {  
        System.out.println("Isciler gunluk 8 saat calisir");  
    }  
}
```

Overridden Methods

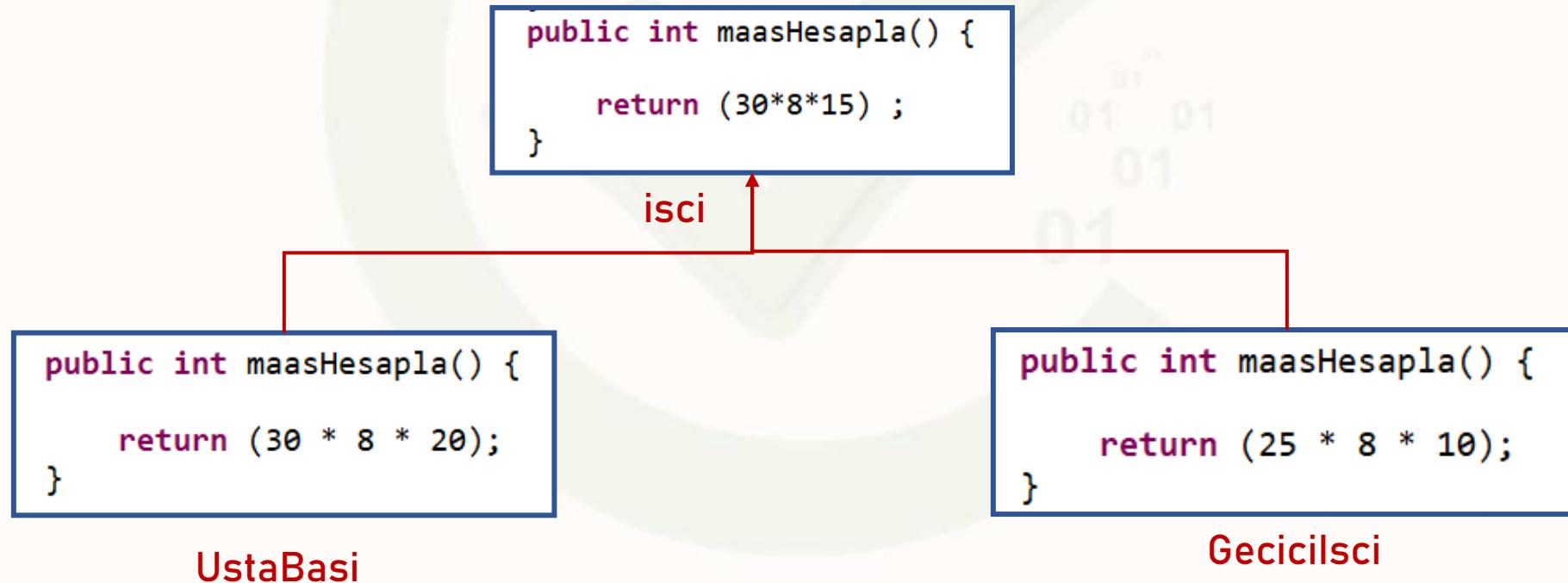
```
public class UstaBasi extends Isci {  
  
    public static void main(String[] args) {  
  
        UstaBasi ub1 = new UstaBasi();  
        ub1.isim = "Seher";  
        ub1.soyisim = "Boss";  
        ub1.statu = "Usta Basi";  
        System.out.println(ub1.isim + " "+ub1.soyisim+" "+  
        ub1.statu+ " "+ ub1.maasHesapla());  
        ub1.calismaSaati();  
    }  
  
    public int maasHesapla() {  
        return (30 * 8 * 20);  
    }  
  
    public void calismaSaati() {  
        System.out.println("Ustabasi is bitene kadar calisir");  
    }  
}
```

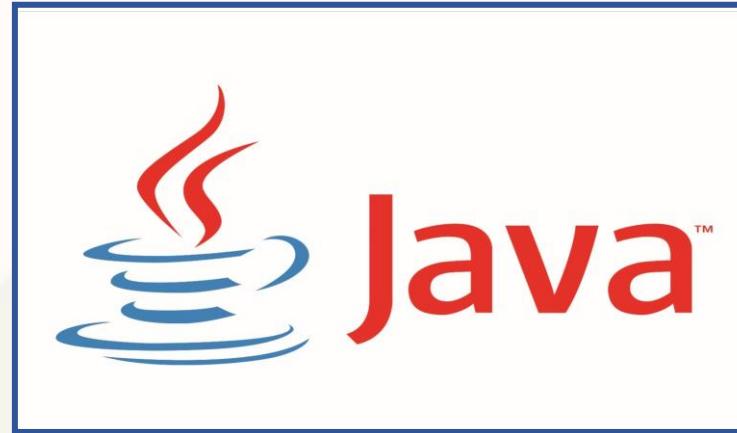
```
public class GeciciIsci extends Isci {  
  
    public static void main(String[] args) {  
  
        GeciciIsci gi1 = new GeciciIsci();  
        gi1.isim = "Faruk";  
        gi1.soyisim = "Yanik";  
        gi1.statu = "GeciciIsci";  
        System.out.println(gi1.isim + " "+gi1.soyisim+" "+  
        gi1.statu+ " "+ gi1.maasHesapla());  
        gi1.calismaSaati();  
  
    }  
  
    public int maasHesapla() {  
        return (25 * 8 * 10);  
    }  
  
    public void calismaSaati() {  
        System.out.println("Gecici isciler haftalik 25 saat calisir");  
    }  
}
```

# 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.





11 ARALIK 2021  
DERS 39

Overriding Kurallari  
Polymorphism

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Inheritance : miras veya kalitim demektir
  - 2) Java parent ve child class'lar arasında isleyisi inheritance kuralları ile belirlenmiştir.
  - 3) Inheritance'in temel özelliklerি
    - Child Class'lar parent class'daki protected veya public olarak tanımlanan tüm class üyeleri direk kullanabilirler
    - Objeler oluşturdukları class'a göre üst class'lardaki özellikleri inherit ederler
    - Child class'lar bazı özellikleri kendilerine göre adapte edebilirler
      - \* eğer child class'da parent class'dan farklı ve size özel bir özelliğini vurgulamak isterseniz parent class'daki method'u override ederiz
      - \* override edilen method (overridden) ile override eden method (overriding)'un signature'ları aynı olmalıdır. Cunku, biz iki method'dan sadece bir tanesinin çalışmasını isteriz
      - \* Data turu ile constructor farklı class'lardan olduğunda bir variable değeri istendiğinde data turunun olduğu class'dan başlanır ve üst class'lara bakılır, ilk bulunan variable kullanılır.
- Method'larda ise yine Data turunun olduğu class'dan başlanıp Yukarı doğru gidilir, ANCAK eğer bulunan method overridden ise bulduğumuz overridden method değil overriding olan method'u çalıştırırız.

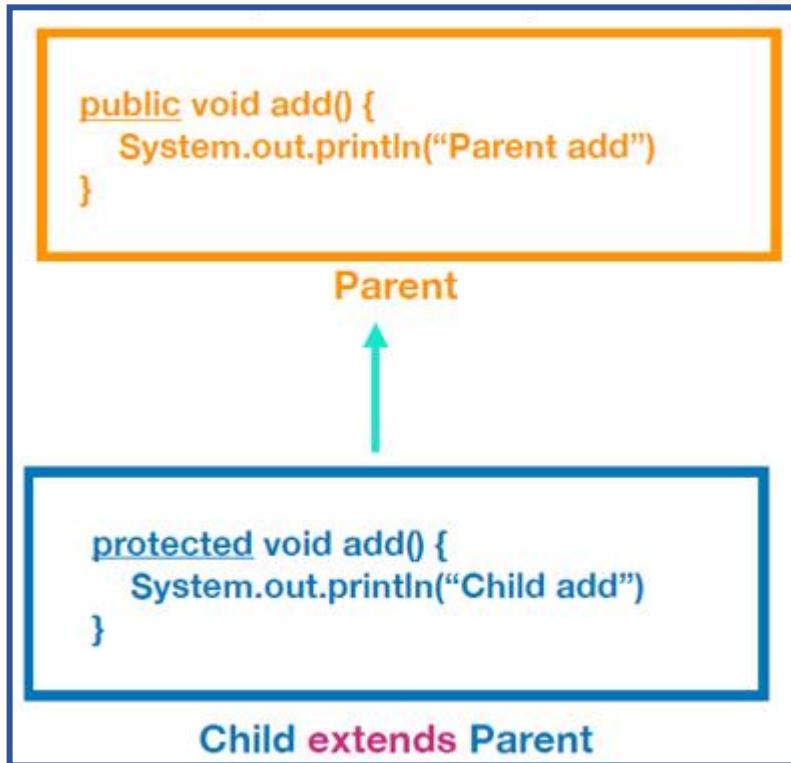
# Overriding Kurallari

- 1) Method Signature'i (*isim ve parametreler*) aynı 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

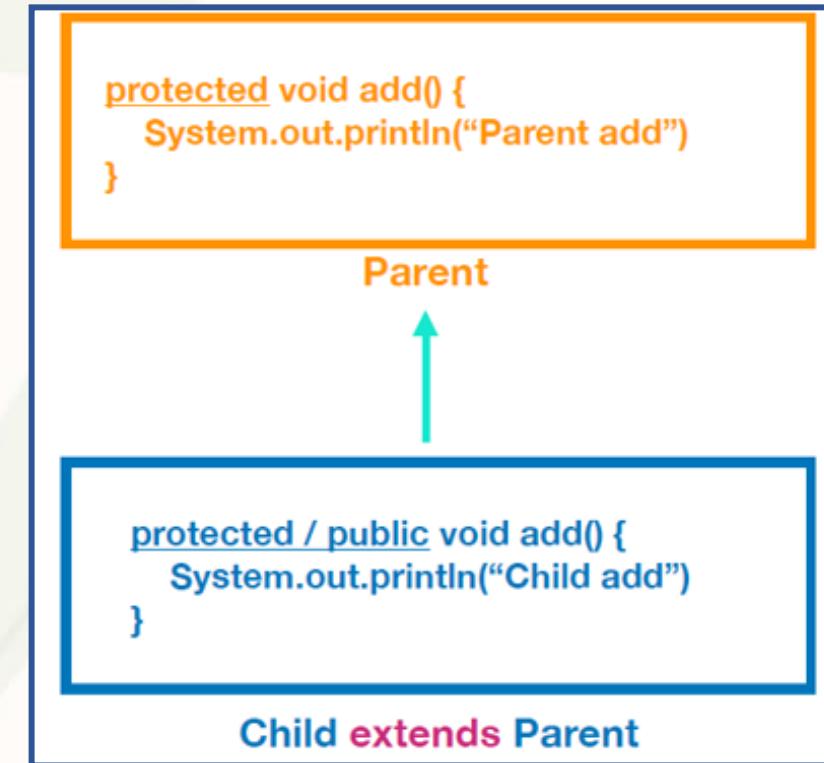
5 ve 6 sonra açıklanacak

- 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) Eğer abstract olmayan bir class **abstract class**'a extend ediyorsa veya **bir interface**'i implement ediyorsa abstract method'ların tamamı override edilmelidir

# Overriding Kurallari ? Access Modifier



Yanlis  
Child Parent'i sinirlayamaz

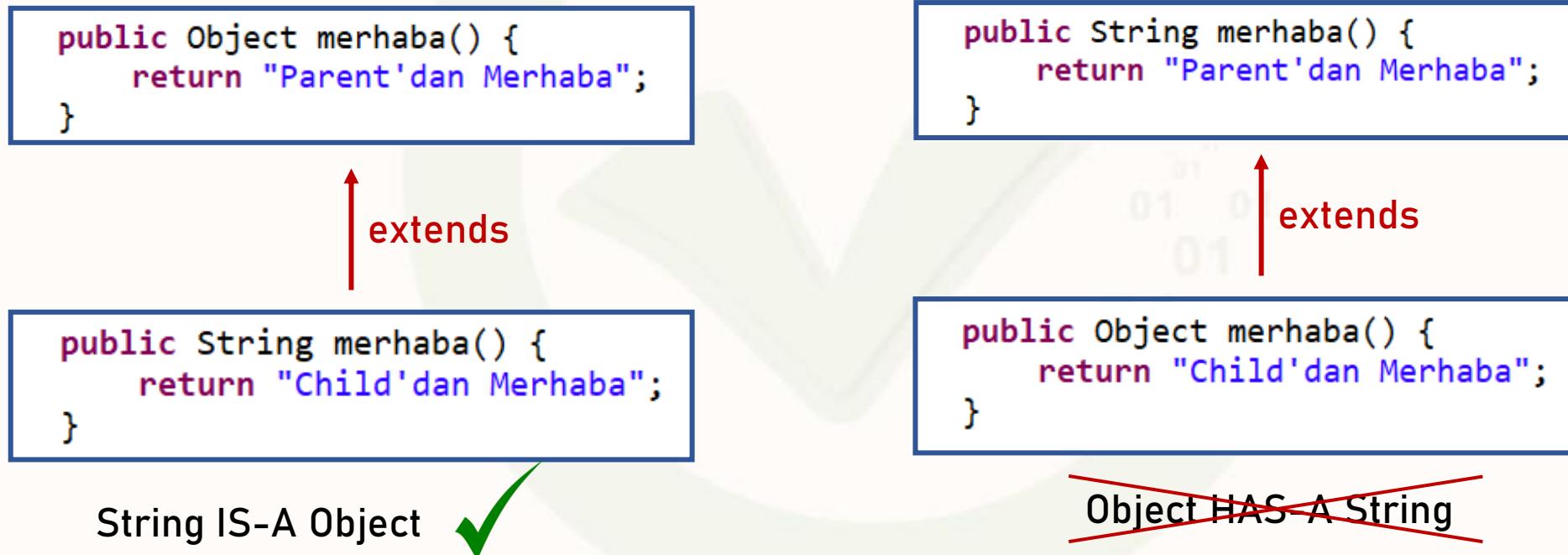


Dogru  
Child'in access modifier'i Parent ile  
ayni veya daha genis olmalıdır

NOT : Private method'lar override edilemez..

# Overriding Kurallari ? Return Type

- 1) Overriding method return type'i overridden method'un return type'i ile aynı olabilir
- 2) Eğer ikisi aynı değilse parent class(overridden)'daki return type ile child class(overriding)'daki return type arasında IS-A RELATION olmalıdır. (**covariant return types**)



- 3) Primitive data türlerinde böyle bir ilişki olmadığı için return turu aynı olmalıdır.

# Overriding Kurallari

Overridden 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 nasıl giderilebilir ? Program düzeltildip 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 nasıl giderilebilir ? Program düzeltildip 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) Aşağıdaki programdaki CTE nasıl giderilebilir? Program düzelttilip çalıştırıldığında konsolda ne yazdırır?

```
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 nasıl giderilebilir ? Program düzeltildip 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) Asagidaki program calistirildiginda konsolda ne yazdirir?

```
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?

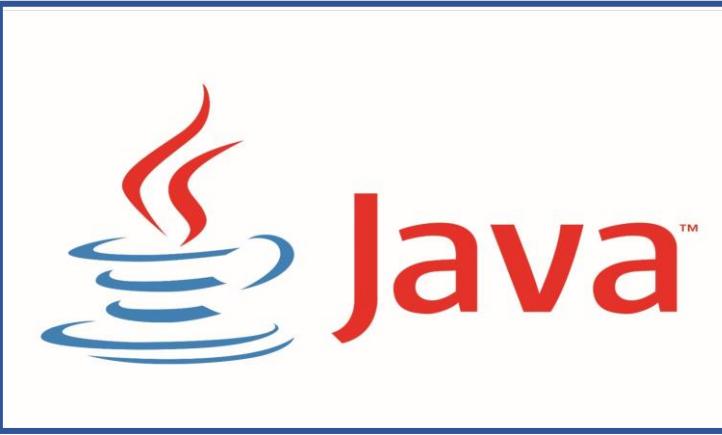
```
public class Product {
    public static void main(String[] args) {
        M m = new M();    M n = new N();
        M o = new O();    O oo = new O();
        m.product(3);    n.product(3);
        oo.product(3);
    }
}
class M {
    int product(int i) {
        int result = i * i;
        System.out.print("{" + 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



13 ARALIK 2021  
DERS 40

Exception

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# Exception



Her sey olmasi gerektigi gibi..  
No exception



Sorun var ama halledilebilir..  
No Exception



Sorun var ve halledilemez  
Throw Exception

Java'da bir program calistirildiginda, farkli sorunlar olusabilir.

- Programcilarin yazdigı kodlarda hata olabilir
- Kullanicidan istenen degerlerde uygun olmayan deger girilebilir
- Internet baglantisinin kesilmesi gibi ongorulemeyen hatalar olabilir

**NOT :** Bir program calistirildiginda, Java cozemedigi bir sorunla karsilastiginda calismayı durdurur (**stops execution**) ve “**throws an exception**”

# Exception

Java karsilastigi sorunu ve sorunla karsilastigi kod satirini bize rapor eder

```
5 public class Go {  
6  
7     public static void main(String[] args) {  
8         System.out.println("");  
9         int a=10;  
10        int b= 0;  
11  
12        System.out.println(a/b);  
13    }  
14 }  
15 }
```

```
Exception in thread "main" java.lang.ArithmetiException / by zero  
at _00_example.Go.main(Go.java:12)
```

Exception turu

Sorunun oldugu satir

Exception'in kaynagi

# Exception

Sorunu cozmek icin try - catch block kullaniriz.

Try blogu tek basina calismaz.

Try blogundan sonra mutlaka catch block(lari) veya finally block olmalidir.

```
public class Go {  
    public static void main(String[] args) {  
        int a=10;  
        int b= 0;  
  
        try {  
            System.out.println(a/b);  
        } catch (ArithmetricException e){  
            System.out.println("Bolme isleminde payda sifir olamaz");  
        }  
    }  
}
```

Sorun cikmazsa yapacagi islem  
(Bizim asil yapmak istedigimiz islem)

beklenenExceptionTuru

Beklenen exception turu  
gerceklendiginde calisacak kodlar

# Exception

Catch block'da kullanılan “e” nin görevi

- Catch block'da yazdigimiz Exception ismi class adı (Data turu) , “e” ise variable ismidir.
- e. yazinca ilgili exception class'indan kullanabilecegimiz method'lari gorebiliriz.

```
catch (FileNotFoundException e) {  
    System.out.println("Dosya okunamiyor" + e.getMessage());  
}
```

Exception'in kaynagini gosterir

```
catch (FileNotFoundException e) {  
    e.printStackTrace();  
    System.out.println("Dosya okunamiyor" );  
}
```

Detayli Exception'in raporu gosterir

# Exception

## File Input/Output Exceptions

```
1
2
3
4
5 public class Go {
6
7     public static void main(String[] args) {
8         FileInputStream fis=new FileInputStream("\\src\\_00_example\\file");
9     }
10 }
```

Java'dan bir dosya okumasini veya bir dosyaya yazmasini istedigimizde, Java olasi problemleri ongorur ve bizden cozum ister.

Buradaki CTE, kodumuzda bir hata oldugu icin degil yazdigimiz kod calistiginda olusabilecek olasi okuma hatalarinda ne yapilacagina karar vermek icindir.

# Exception

Muhtemel sorunlar birden fazla ise;

- 1) İcice try-catch blokları kullanılabilir.

```
public class Go {

    public static void main(String[] args) {
        FileInputStream fis = null;
        //You may use nested try-catch block
        try {
            fis = new FileInputStream("C:\\\\Users\\\\lenovo\\\\eclipse-workspac
                int k = 0;

                try {
                    while((k = fis.read()) != -1) {

                        System.out.print((char)k);
                    }
                } catch (IOException e) {
                    System.out.println("Dosya okunamıyor");
                }
                } catch (FileNotFoundException e) {
                    System.out.println("Dosya silinmiş veya dosya yolu hatalı");
                }
            }
        }
```

# Exception

2) Tek try- multiple catch kullanilabilir.

```
public class Go {  
  
    public static void main(String[] args) {  
        FileInputStream fis = null;  
  
        try {  
            fis = new FileInputStream("C:\\\\Users\\\\lenovo\\\\eclipse-workspace\\\\exception\\\\src\\\\main\\\\java\\\\com\\\\javatutorial\\\\Exception\\\\Go.java");  
  
            int k = 0;  
  
            while((k = fis.read()) != -1) {  
                System.out.print((char)k);  
            }  
        } catch (FileNotFoundException e) {  
            System.out.println("Dosya okunamiyor");  
        }  
        catch (IOException e) {  
            System.out.println("Dosya silinmis veya dosya yolu hatali");  
        }  
    }  
}
```

Birden fazla catch block kullanilacaksa yazilacak exception'larin sirasi onemlidir.

Birbiri ile parent-child iliskisi olan exception'lar ise once child olan yazilmalidir. Aksi durumda child exception kullanilmaz olur.



# Exception

3) Eğer tüm exception'ları içeren bir exception varsa sadece onu yazabiliriz.

```
public class Go {  
  
    public static void main(String[] args) {  
        FileInputStream fis = null;  
  
        try {  
            fis = new FileInputStream("C:\\\\Users\\\\lenovo\\\\eclipse-workspace\\\\Java\\\\exception\\\\src\\\\file.txt");  
  
            int k = 0;  
  
            while((k = fis.read()) != -1) {  
                System.out.print((char)k);  
  
            }  
        }  
        catch (IOException e) {  
            System.out.println("Dosya okuma problemi var. "  
                + "Dosya silinmis veya path hatali olabilir");  
        }  
    }  
}
```

# Exception Types

## 1) Compile Time (**Checked**) Exceptions

Kod yazildiginda Java'nin ongordugu olasi sorunlardir. Java olasi bir problem gordugunde kirmizi cizgi ile bizi uyarir. (Not: Her kirmizi cizgi exception degildir.)

(FileNotFoundException, IOException)

## 2) RunTime (**Unchecked**) Exceptions

Kod calistirildiginda ortaya cikan exception'lardir.

(ArithmaticException)

# Exception

## 2) NullPointerException

- null objesini uygun olmayan bir komutta kullanırsanız Java NullPointerException verir.
- NullPointerException run time exception'dır.

```
public class Gogo {  
  
    public static void main(String[] args) {  
        String str="";  
        System.out.println(str.length()); // 0  
  
        str=null;  
        System.out.println(str.length()); // RTE  
    }  
  
}
```

```
Exception in thread "main" java.lang.NullPointerException  
at _00_example.Gogo.main(Gogo.java:10)
```

# Exception

## 3) ArrayIndexOutOfBoundsException

- Array veya List'de olmayan bir index için işlem yapmak isterseniz Java ArrayIndexOutOfBoundsException verir.
- ArrayIndexOutOfBoundsException run time exception'dır.

```
public static void main(String[] args) {  
    int arr[] = {1,2,3};  
    System.out.println(arr[0]); // 1  
  
    System.out.println(arr[4]); // Exception  
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4  
at _00_example.Gogo.main(Gogo.java:10)
```

# Exception

## 4) ClassCastException

- Bir datayı casting yapamayacak bir dataya çevirmek istediğinizde ClassCastException verir.
- ClassCastException run time exception'dır.

```
public static void main(String[] args) {  
    Integer sayi= 10;  
    System.out.println(sayi); // 10  
  
    String str= sayi; //CTE  
  
    String str2= (String)sayi; // CTE  
  
    Object sayi2=40;  
    String str3=(String)sayi2; // Exception  
  
}
```

```
Exception in thread "main" java.lang.ClassCastException: java.lang.Integer cannot be cast to java.lang.String  
at _00_example.Gogo.main(Gogo.java:14)
```

# Exception

## 5) NumberFormatException

- Sayı olmayan bir String'i sayıya cevirmeye çalışırsanız Java NumberFormatException verir.
- NumberFormatException run time exception'dır.

```
public static void main(String[] args) {  
    String str= "123456";  
  
    int sayi =Integer.parseInt(str);  
    System.out.println(sayi+10); // 123466  
  
    str="123a4";  
    sayi =Integer.parseInt(str);  
    System.out.println(sayi+10); // Exception  
}
```

```
Exception in thread "main" java.lang.NumberFormatException: For input string: "123a4"  
at java.lang.NumberFormatException.forInputString(Unknown Source)  
at java.lang.Integer.parseInt(Unknown Source)  
at java.lang.Integer.parseInt(Unknown Source)  
at _00_example.Gogo.main(Gogo.java:12)
```

# Exception

## Soru 1)

Kullanicidan carpma yapmak icin bir String isteyin. Kullanicinin girdigi String sadece sayilardan olusuyorsa sayiyi 2 ile carpip sonucu yazdirin.

Kullanici sayilardan olusmayan bir String girerse “Girdiginiz String sayıya cevrilemez” yazdirin.

## Soru 2)

String str[], Urun isimlerini tuttugumuz bir Array olsun. Kullanicidan istedigi urunun sirasini isteyin ve istedigi urunu yazdirin.

Kullanici Array'de olan urun sayisindan buyuk bir sira no girerse “Girdiginiz sira urun sayisindan buyuk” yazdirin.

# Exception

## 6) illegalArgumentException

**Soru:** Kullanicidan yasini girmesini isteyin. Kodunuzu kullanici sifirdan kucuk bir sayi girerse Exception verecek sekilde yazin.

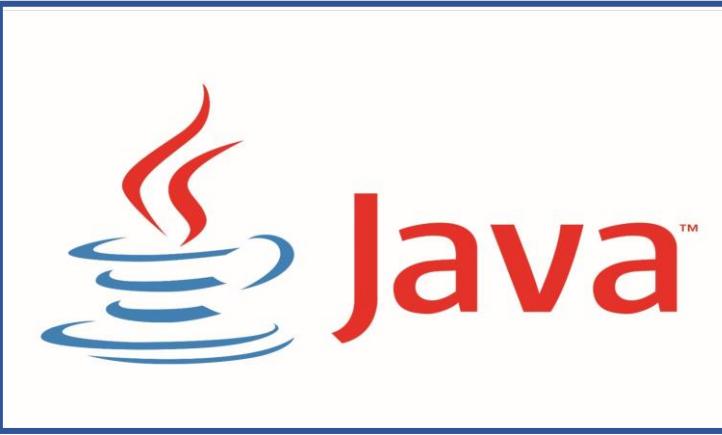
```
public class Example {  
  
    public static void main(String[] args) {  
  
        Scanner scan = new Scanner(System.in);  
        System.out.println("Konsolda gormek icin yasinizi girin");  
        int yas = scan.nextInt();  
  
        try {  
            if(yas<0) {  
                throw new IllegalArgumentException();  
            }  
        }catch(IllegalArgumentException e) {  
            System.out.println(e);  
            System.out.println("Yas icin negatif deger giremezsiniz...");  
        }  
        System.out.println(yas);  
        scan.close();  
    }  
}
```

```
-20  
java.lang.IllegalArgumentException  
Yas icin negatif deger giremezsiniz...  
-20
```

# Exception

## throws

- throws keyword “checked exceptions” icin kullanilir.
- throws keyword, exception handle yapilmak istenmiyorsa kullanilir. (Exception olusunca program calismasi durur)
- throws keyword'den sonra, aralarina virgul konularak, birden fazla exception yazilabilir
- throws keyword method body icinde kullanilamaz, kullanilacaksa method isminin oldugu satirda yazilmalidir.
- throws keyword'den sonra birden fazla exception kullanilacaksa ve yazilan exception'lar arasında parent child iliskisi varsa , child exception yazilabilir ama tavsiye edilmez. Cunku parent exception tum durumlari kapsayacaktir. (Hedef farkli durumlar icin aciklama yazip handle etmek olmadiginden, bir exception'in calismasi yeterlidir)



14 ARALIK 2021  
DERS 41

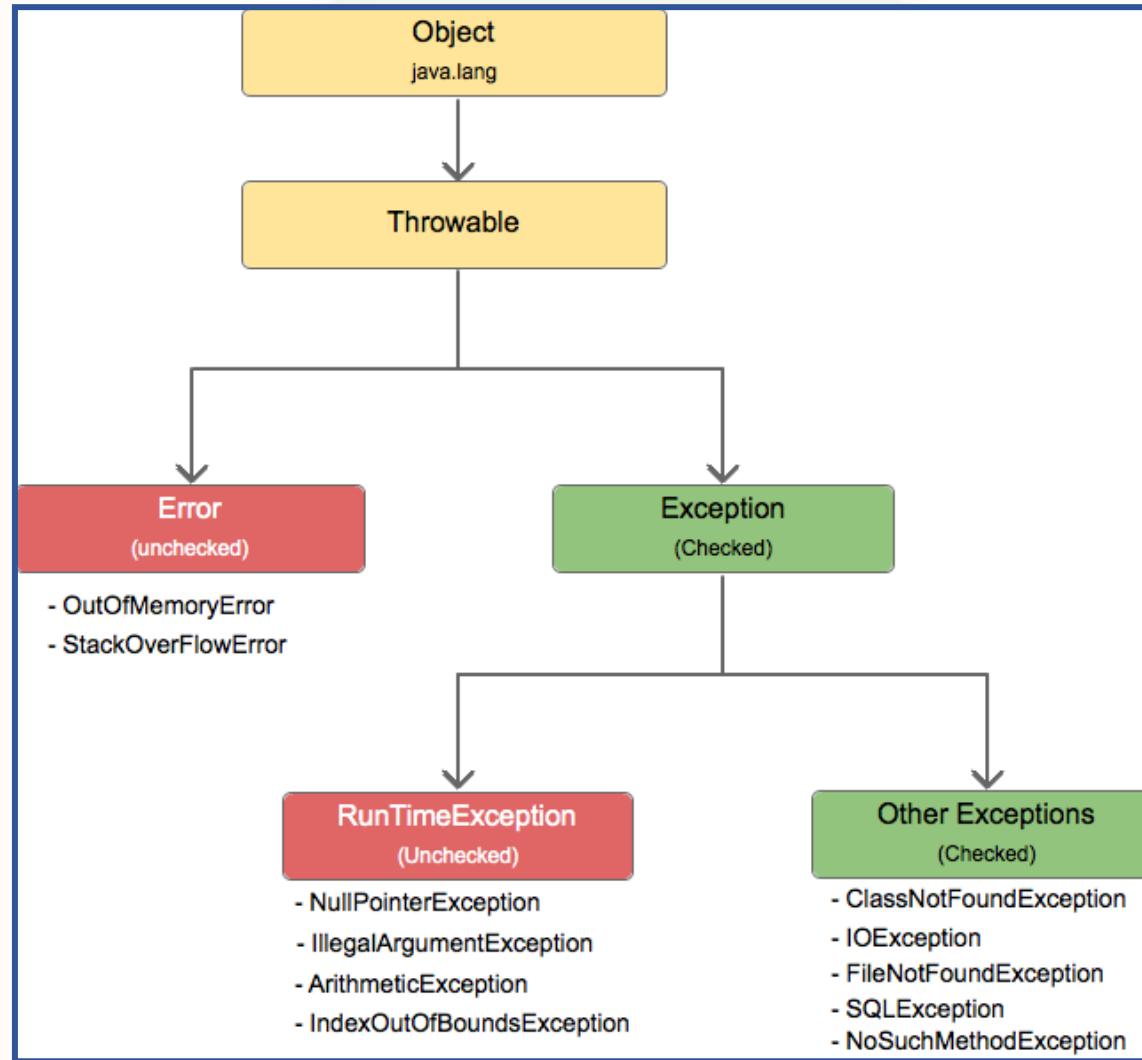
Exception  
Errors  
Garbage Collector

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Exception : Java cozemedigi bir sonucla karsilastiginda exception class'l devreye girer. Exception class'i bize karsilasilan sorunun cinsini, bu sorunun nereden kaynaklandigini ve sebebini verir.
- 2) Java exception firlattiginda kodun calismasini da durdurur (stops execution)
- 3) Java exception olusturacak bir kod icin bizden karar vermemizi ister
  - Eger hicbirsey yapmazsak beklendi gibi kodumuzun calismasini durdurur.
  - Kodumuzun calismaya devam etmesini istiyorsak try catch ile exception olusturmasi muhtemel kodlarimizi try catch blogu icinde yazmaliyiz
  - \* Try bloguna calismasini istedigimiz kodlar,
  - \* catch satirindaki (parantez icine) karsilasmamız muhtemel olan exception class ismini ve bir obje ismi (e),
  - \* catch bloguna ise exception ile karsilasildiginda yapilmasini istedigimiz kodlari yazariz.
- 4) Yazdigimiz objeden sonra . koyarsak exception turu veya kaynagi ile ilgili Java'nin verdigi hatalari consolda yazdirabiliriz. Boylece hem hata kodlarini gorup, hem de kodumuzun calismaya devam etmesini saglayabiliriz
- 5) Exception temel olarak checked ve unchecked olarak ikiye ayrilir. Checked exp. Kod yazilirken muhtemel hatalari javanin group sizden cozum istedigi exception'lardir
- 6) Birden fazla exception'i handle etmek istedigimizde siralamaya dikkat etemiz gerekir.

# Exception



# Exception throw vs throws

THROW	THROWS
<p>1- throw keyword kontrollu olarak bir exception throw etmek icin kullanilir</p> <p>2- throw keyword ile sadece bir exception throw edilebilir</p> <p>3- throw keyword method icinde kullanilir</p> <p>4- throw keyword yazdiktan sonra variable yazilir <b>(Orn:</b> new IllegalArgumentException(); )</p>	<p>1- throws keyword bir veya daha fazla exception'i deklare etmek icin kullanilir</p> <p>2- throws keyword bir veya daha fazla exception throw edilebilir</p> <p>3- throws keyword method signature ile kullanilir</p> <p>4- throws keyword yazdiktan sonra exception class ismi yazilir <b>(Orn:</b> FileNotFoundException )</p>

# Exception Finally Block

- Finally block try-catch blogu ile kullanilir.
- Finally block, her durumda calisir
- Finally block cloud veya database ile connection'i bitirme veya uzerinde calisilan dosyayı kapatma gibi islemler icin kullanilir.

```
public static void main(String[] args) {  
  
    int sayi1=10;  
    int sayi2=0;  
    try {  
        System.out.println(sayi1/sayi2);  
    } catch (ArithmetricException e) {  
        System.out.println("Kodda hata var... ");  
        e.printStackTrace();  
    } finally {  
        System.out.println("connection'i durdur...");  
    }  
}
```

- Finally block catch blogu olmadan sadece try ile de kullanilabilir.
- Bu durumda catch blogu olmadigindan Java throws exception ardindan finally ile istedigimiz islemi yapar

# Exception Finally Block

Asagidaki kod calistirildiginda konsolda ne yazdirir?

```
String s = "";
try {
    s += "t";
} catch (Exception e) {
    s += "c";
} finally{
    s += "f";
}
s += "a";
System.out.print(s);
```

# Exception

Asagidaki kod calistirildiginda konsolda ne yazdirir?

```
public static void main(String[] args) {  
    System.out.println(exceptions());  
}  
  
@SuppressWarnings("finally")  
public static String exceptions() {  
    String result="";  
    String v=null;  
  
    try {  
        try {  
            result=result+"a";  
            v.length();  
            result=result+"b";  
        } catch(NullPointerException e) {  
            result=result+"c";  
        } finally {  
            result=result+"d";  
            throw new Exception();  
        }  
    } catch (Exception e) {  
        result=result+"e";  
    }  
    return result;  
}
```

## Exception Tekrar Soruları

- 1) try blogu mutlaka catch blogu ile kullanilmalidir. ~~True / False~~
- 2) finally blogu mutlaka calisir. ~~True / False~~
- 3) Bir try blogu ile birden fazla catch blogu calistirilabilir. ~~True / False~~
- 4) Birden fazla catch blok varsa, child olan once yazilmalidir. ~~True / False~~
- 5) FileNotFoundException nedir?

Programimizda bir dosyayı okumaya çalıştığımızda, dosya bulunamazsa oluşur.  
IOException'in subclass'ıdır.

- 6) IOException nedir?

Programimizda bir file'a input/output yapılıyorsa ve program çalışırken bir problem çıkarsa oluşur.  
Checked exception'dır ve kod yazılırken mutlaka handle edilmelidir.

# Exception

## Create Custom Checked Exception

```
public class InvalidEmailIdCheckedException extends Exception {  
  
    private static final long serialVersionUID = 1L;  
  
    public InvalidEmailIdCheckedException(String message) {  
        super(message);  
    }  
  
}
```

- 1) Class isminin sonunda "Exception" kullanılır. Bu mecburi degildir ama genel isim verme konsepti boyledir.
- 2) Bir "checked exception" olusturacaksak, class'imizi "Exception" class'ina child class yapmaliyiz.
- 3) "String" parametresi olan bir constructor olusturun ve ilk satirina super(); ekleyin.

# Exception

## Create Custom Checked Exception

Soru : Yeni bir class olusturalim, icinde mailDogrula(String eMail) olsun. Email adresi @gmail.com veya @hotmail.com icermiyorsa **InvalidEmailIdCheckedException** versin.

```
public class Test {

    public static void main(String[] args) throws InvalidEmailIdCheckedException {
        mailDogrula("ab@gmail1.com");
    }

    public static void mailDogrula(String eMail) throws InvalidEmailIdCheckedException {
        if (eMail.contains("@hotmail.com") || eMail.contains("@gmail.com")) {
            System.out.println(eMail);
        } else {
            throw new InvalidEmailIdCheckedException("Email adresi uygun degil");
        }
    }
}
```

```
Exception in thread "main" _00_example.InvalidEmailIdCheckedException: Email adresi uygun degil
at _00_example.Test.mailDogrula(Test.java:16)
at _00_example.Test.main(Test.java:9)
```

# Exception

## Create Custom UnCheckedException

```
public class InvalidEmailIdUnCheckedException extends RuntimeException {  
    private static final long serialVersionUID = 1L;  
  
    public InvalidEmailIdUnCheckedException(String message) {  
        super(message);  
    }  
}
```

- 1) Class isminin sonunda "Exception" kullanılır. Bu mecburi degildir ama genel isim verme konsepti boyledir.
- 2) Bir "checked exception" olusturacaksak, class'imizi "RuntimeException" class'ina child class yapmaliyiz.
- 3) "String" parametresi olan bir constructor olusturun ve ilk satirina super(); ekleyin.

# Exception

## Create Custom UnCheckedException

**Soru:** Yeni bir class olusturalim, icinde mailDogrula(String eMail) olsun. Email adresi @gmail.com veya @hotmail.com icermiyorsa InvalidEmailIdUnCheckedException versin.

```
public class Test {  
  
    public static void main(String[] args) {  
        mailDogrula("ab@gmail1.com");  
    }  
  
    public static void mailDogrula(String eMail) {  
        if (eMail.contains("@hotmail.com") || eMail.contains("@gmail.com")) {  
            System.out.println(eMail);  
        } else {  
            throw new InvalidEmailIdUnCheckedException("Email adresi uygun degil");  
        }  
    }  
}
```

```
Exception in thread "main" _00_example.InvalidEmailIdUnCheckedException: Email adresi uygun degil  
at _00_example.Test.mailDogrula(Test.java:16)  
at _00_example.Test.main(Test.java:9)
```

# Exception

## Soru 1

Which of the following pairs fill in the blanks to make this code compile? (Choose all that apply)

```
7: public void ohNo() _____ Exception {  
8:     _____ Exception();  
9: }
```

- A. On line 7, fill in throw
- B. On line 7, fill in throws
- C. On line 8, fill in throw
- D. On line 8, fill in throw new
- E. On line 8, fill in throws
- F. On line 8, fill in throws new

**Cevap :** B, D. In a method declaration, the keyword throws is used. To actually throw an exception, the keyword throw is used and a new exception is created.

# Exception

## Soru 2

Which exception will the following throw?

```
Object obj = new Integer(3);
String str = (String) obj;
System.out.println(str);
```

- A. ArrayIndexOutOfBoundsException
- B. ClassCastException
- C. IllegalArgumentException
- D. NumberFormatException
- E. None of the above.

**Cevap :** B. The second line tries to cast an Integer to a String. Since String does not extend Integer, this is not allowed and a ClassCastException is thrown.

# Exception

## Soru 3

What will happen if you add the statement `System.out.println(5 / 0);` to a working `main()` method?

- A. It will not compile.
- B. It will not run.
- C. It will run and throw an `ArithmaticException`.
- D. It will run and throw an `IllegalArgumentException`.
- E. None of the above.

- 3) C. The compiler tests the operation for a valid type but not a valid result, so the code will still compile and run. At runtime, evaluation of the parameter takes place before passing it to the `print()` method, so an `ArithmaticException` object is raised.

# Exception

## Soru 4

Which of the following can be inserted in the blank to make the code compile? (Choose all that apply)

```
public static void main(String[] args) {  
    try {  
        System.out.println("work real hard");  
    } catch (_____ e) {  
    } catch (RuntimeException e) {  
    }  
}
```

- A. Exception
- B. IOException
- C. IllegalArgumentException
- D. RuntimeException

4) C, E. Option C is allowed because it is a more specific type than RuntimeException. Option E is allowed because it isn't in the same inheritance tree as RuntimeException. It's not a good idea to catch either of these. Option B is not allowed because the method called inside the try block doesn't declare an IOException to be thrown. The compiler realizes that IOException would be an unreachable catch block. Option D is not allowed because the same exception can't be specified in two different catch blocks. Finally, option A is not allowed because it's more general than RuntimeException and would make that block unreachable.

# Exception

## Soru 5

What is the output of the following snippet, assuming a and b are both 0?

```
3:     try {  
4:         return a / b;  
5:     } catch (RuntimeException e) {  
6:         return -1;  
7:     } catch (ArithmetricException e) {  
8:         return 0;  
9:     } finally {  
10:        System.out.print("done");  
11:    }
```

- A. -1
- B. 0
- C. done-1
- D. done0
- E. The code does not compile.
- F. An uncaught exception is thrown.

- 5) E. The order of catch blocks is important because they're checked in the order they appear after the try block. Because ArithmetricException is a child class of RuntimeException, the catch block on line 7 is unreachable. (If an ArithmetricException is thrown in try try block, it will be caught on line 5.) Line 7 generates a compiler error because it is unreachable code.

## Soru 6

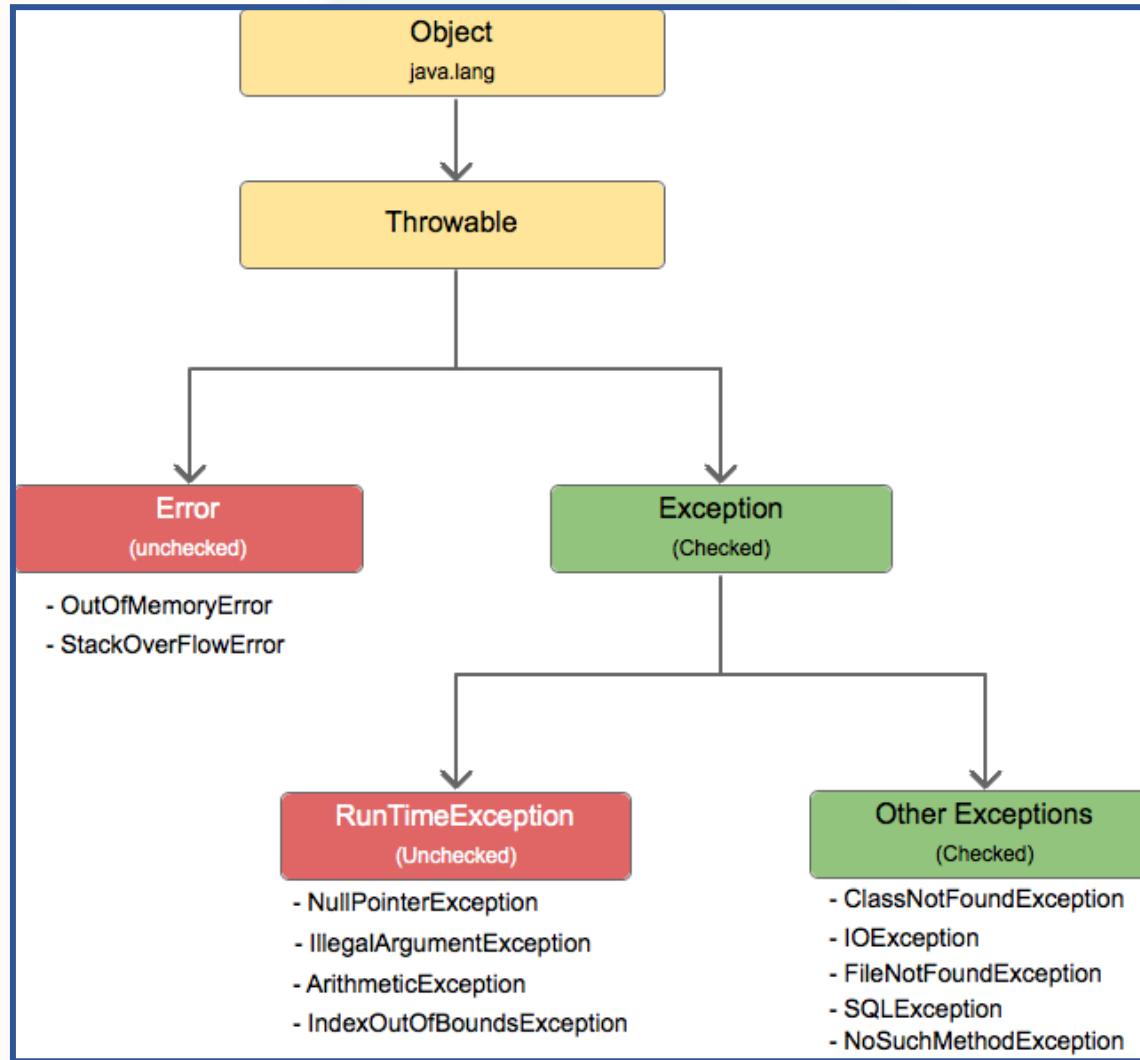
What is the output of the following program?

```
1: public class Dog {  
2:     public String name;  
3:     public void parseName() {  
4:         System.out.print("1");  
5:         try {  
6:             System.out.print("2");  
7:             int x = Integer.parseInt(name);  
8:             System.out.print("3");  
9:         } catch (NumberFormatException e) {  
10:             System.out.print("4");  
11:         }  
12:     }  
13:     public static void main(String[] args) {  
14:         Dog leroy = new Dog();  
15:         leroy.name = "Leroy";  
16:         leroy.parseName();  
17:         System.out.print("5");  
18:     } }  
  
A. 12  
B. 1234  
C. 1235  
D. 124  
E. 1245  
F. The code does not compile.  
G. An uncaught exception is thrown.
```

## Exception

6) E. The `parseName` method is invoked within `main()` on a new `Dog` object. Line 4 prints 1. The try block executes and 2 is printed. Line 7 throws a `NumberFormatException`, so line 8 doesn't execute. The exception is caught on line 9, and line 10 prints 4. Because the exception is handled, execution resumes normally. `parseName` runs to completion, and line 17 executes, printing 5. That's the end of the program, so the output is 1245.

# Errors



# Errors

- Error “throwable” class'inin bir alt class'ıdır. Errors, sistem kaynaklarının eksikliği nedeniyle ortaya çıkan kritik koşullardır ve yazacagımız kodlarla handle edilemez.
- Java error'un oluşumu hakkında herhangi bir bilgiye sahip olmadığı için hatalar her zaman unchecked'dirler.
- Errors, her zaman runtime'da ortaya çıkar.
- Error'un ortaya çıkışının sonucu, programın olağandışı bir şekilde sonlandırılmasıdır.

# Errors

## Karşılaştırma Tablosu

Karşılaştırma için temel	Hata	İstisna
Temel	Hata, sistem kaynaklarının eksikliğinden kaynaklanır.	Kod nedeniyle bir istisna ortaya çıkıyor.
Kurtarma	Bir hata düzeltilemez.	Bir istisna kurtarılabilir.
Anahtar kelimeler	Bir hatayı program koduyla çözmenin bir yolu yoktur.	İstisnalar, "try", "catch" ve "throw" üç anahtar sözcüğü kullanılarak ele alınır.
sonuçlar	Hata tespit edildiğinde program anormal şekilde sonlandırılır.	Bir istisna tespit edildiğinde, karşılık gelen "atma" ve "yakalama" anahtar sözcükleri tarafından atılır ve yakalanır.
Türleri	Hatalar denetlenmeyen tür olarak sınıflandırıldı.	İstisnalar kontrol edilmiş veya kontrol edilmemiş tip olarak sınıflandırılır.
paket	Java'da hatalar "java.lang.Error" paketi olarak tanımlanmıştır.	Java'da, bir istisna "java.lang.Exception" içinde tanımlanmıştır.
Örnek	OutOfMemory, StackOverFlow.	İşaretli İstisnalar: NoSuchMethod, ClassNotFoundException. İşaretlenmemiş İstisnalar: NullPointerException, IndexOutOfBoundsException.

# Errors

- 1) Error'lar handle edilemezler.
- 2) Programın anormal bir şekilde sonlanmasına sebep olurlar.
- 3) Error'lar unchecked'dir ve genellikle run time'da olusurlar.
- 4) Error'lara örnek; Out of Memory Error, Stack over flow error veya System Crash Error

```
public static void main(String[] args) {  
    for(int i=0; i<3; i--) {  
        System.out.print(i + " ");  
    }  
}
```

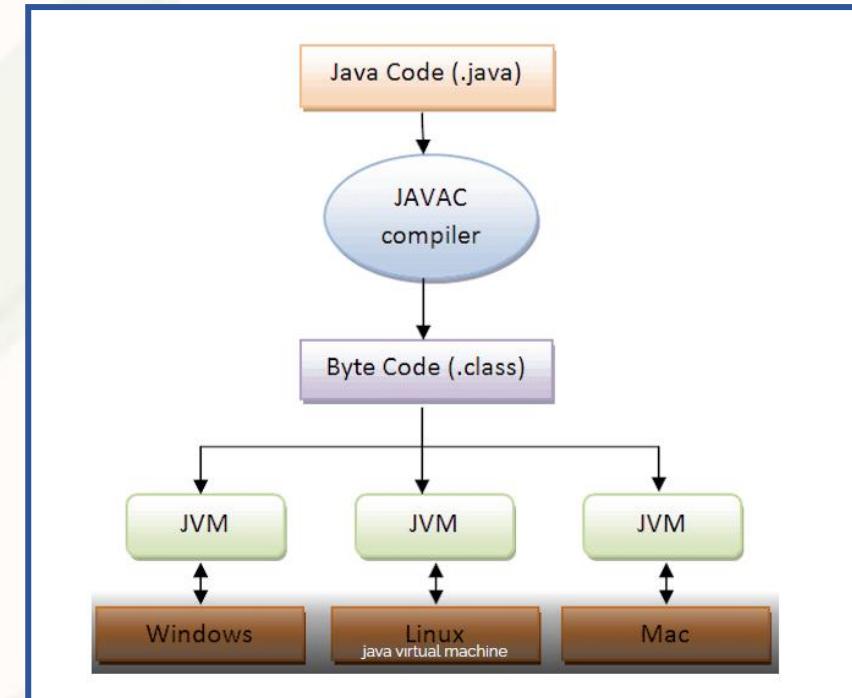
# Java Platform Bagimsiz Calisir

“write once run everywhere” sloganini gerçeklestiren yapidir.

Jvm, Java programlama dilinde yazdigimiz kodların üzerinde çalıştiği programdır.

Hangi platformda yazmis olursak olalim Java Compiler kodlarimizi byte kodlara cevirir, burada JVM devreye girerek kodu daha alt seviyede makine diline dönüştürmektir.

Her işletim sisteminde jvm kendine özgür.Her işletim sistemi için o işletim sistemine özgü Jvm'i yüklemelisiniz.



# Garbage Collector



Garbage



Finalized Garbage



Object Destroyed

Garbage Collector heap alanında çalışan ve heap alanında new operatörü vb. metodlarla oluşturulan ve referansı olmayan nesneleri heap alanında temizleyen mekanizmanın adıdır. Garbage Collector işlemini 3 adımda tamamlar. Bunlar:

**İşaretle:** Kullanılan ve kullanılmayan nesneler işaretlenir.

**Silme:** Referansı olmayan nesneleri heap alanında temizler.

**Sıkıştırma ve düzenleme:** Silme işlemine ek olarak daha büyük nesnelere boş alan oluşturmak için kalan nesneleri bir araya getirir.

# Garbage Collector



Garbage



Finalized Garbage



Object Destroyed

- finalize() method Garbage Collector tarafından imha edilmesi gereken datalar imha edilmeden önce çalıştırılır.
- Garbage collector sadece finalized yapılmış objeleri toplar ve yokeder.
- Biz kod yazarken finalize() method'unu çağırıksak da finalize() method'unun ne zaman çalışacağına ve ne yapacağına JVM karar verir

## **final – finally – finalize()**

Interview Question : "final", "finally " ve "finalize " nedir?

- **final keyword**, **finally kod blogu**, **finalize ise method'dur**

**final keyword**

**Final Variable**

Degeri degistirilmeyecek (constant) variable'lar Icin kullanilir,  
mutlaka deger atanmalidir, isimleri buyuk harfle yazilir(optional)

**Final Methods**

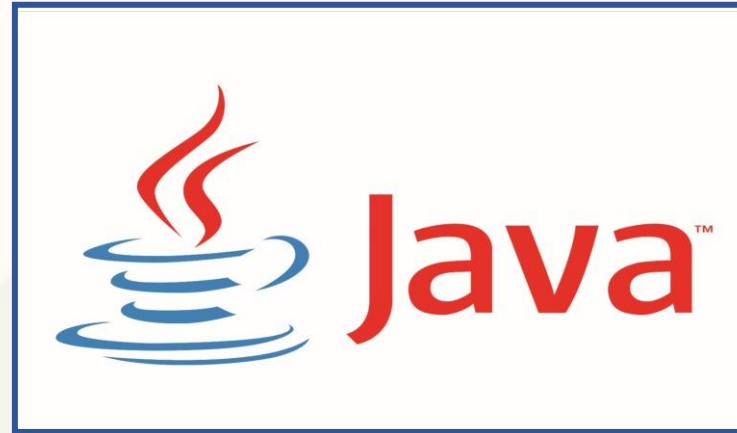
Override edilemeyecek method

**Final Classes**

Inherit edilemeyecek class

**finally kod blogu** : try veya try-catch bloklari ile kullanilir. Try-catch'in sonucu ne olursa olsun calisir. Genellikle database veya cloud ile connection'in sonlandirilmasi, calisilan file'in kapanmasi gibi islemler yapar.

**finalize method'u** : garbage collector kullanilmayan objeleri destroy etmeden once kullanilir



15 ARALIK 2021  
DERS 42

## Abstract Classes

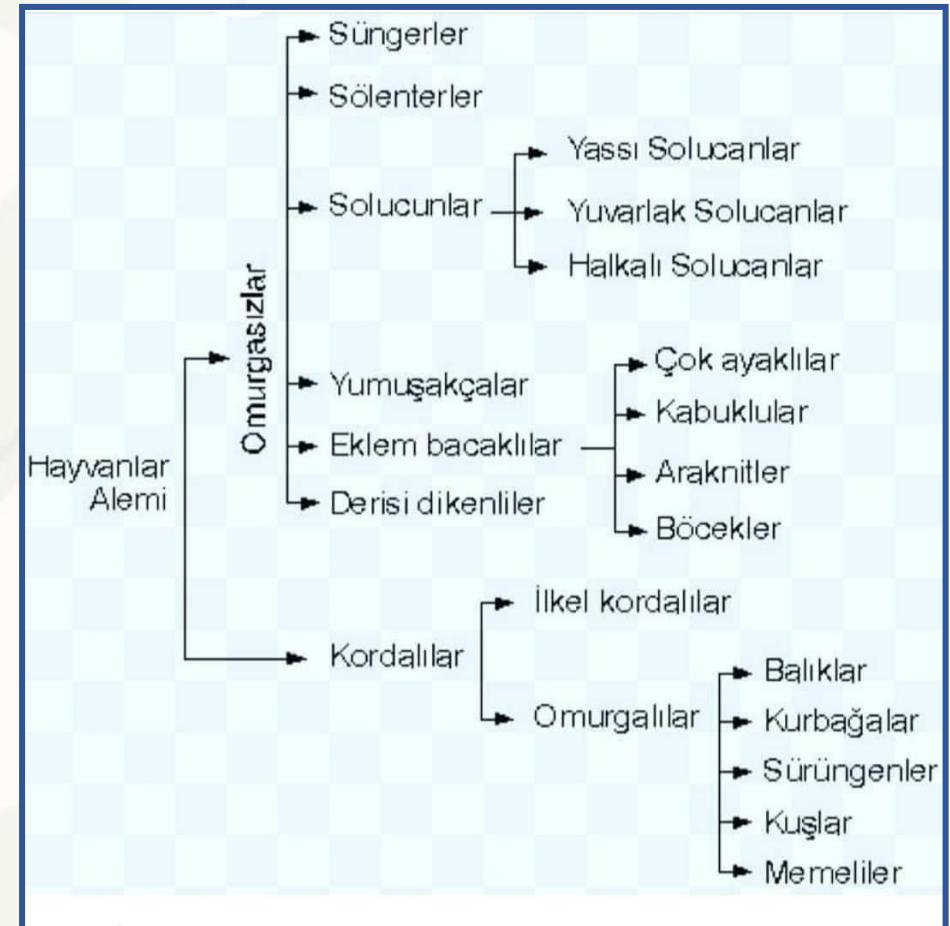
Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# Abstract Classes

Abstract class, genellikle ortak özellikleri olan nesneleri tek bir çatı altında toplamak için kullanılır.

Tüm child class'larda olmasini istedigimiz dinamik özellikler (methods) abstract class'da **abstract method** olarak oluştururuz.

Abstract olmayan (**concrete**) tüm child class'lar abstract method'lari override etmek zorundadir. Boylece tüm child class'lar **ayni dinamik ozelliklere** (methods) sahip olurlar.



# Abstract Classes

Abstract class nasıl olşturulur ?

Abstract class oluşturmak için class keyword'inden önce **abstract** keyword'u yazılmalıdır.

```
public abstract class Personel {  
}
```

Abstract method nasıl olşturulur ?

Abstract method oluşturmak için **abstract** keyword'u yazılmalıdır.

Abstract method'un body'si olmaz (No implementation), body oluşturursanız CTE olusur.

Abstract method private, final veya static olarak tanımlanamaz.

```
public abstract void maasHesapla();  
  
public abstract void mesaiBilgisi() {  
}
```

# Abstract Classes

NOT : Bir abstract class, abstract veya concrete method'lara sahip olabilir

```
public abstract class Personel {  
  
    public String isim;  
    public int maas;  
  
    public abstract void maasHesapla();  
    public abstract void mesaiBilgisi();  
  
    public void ozelSigorta() {  
        System.out.println("Bu personel ozel sigorta kapsamindadir");  
    }  
}
```

Concrete class icinde Abstract method OLUSTURULAMAZ

```
public class IdariPersonel extends Personel{  
  
    public static void main(String[] args) {  
  
    }  
  
    public abstract void vardiya();
```

# Abstract Classes

```
public abstract class Personel {  
  
    public String isim;  
    public int maas;  
  
    public abstract void maasHesapla();  
    public abstract void mesaiBilgisi();  
  
    public void ozelSigorta() {  
        System.out.println("Bu personel ozel sigorta kapsamindadir");  
    }  
}
```

```
public class Isci extends Personel{  
  
    public static void main(String[] args) {  
  
    }  
  
    @Override  
    public void maasHesapla() {  
        System.out.println("maas : "+ 30*8*15);  
    }  
    @Override  
    public void mesaiBilgisi() {  
        System.out.println("Isciler 8 saat mesai yapar");  
    }  
}
```

```
public class IdariPersonel extends Personel{  
  
    public static void main(String[] args) {  
  
    }  
}
```

```
public abstract class YanHizmetler extends Personel{  
  
    public static void main(String[] args) {  
  
        YanHizmetler obj1 = new YanHizmetler();  
    }  
}
```

# Abstract Classes

## Kural 1)

Concrete bir child class, parent'i olan abstract class'lardaki tum abstract method'lari implement etmelidir, diger turlu CTE olusur.

```
public abstract class Personel {  
    public String isim;  
    public int maas;  
  
    public abstract void maasHesapla();  
    public abstract void mesaiBilgisi();  
  
    public void ozelSigorta() {  
        System.out.println("Bu personel ozel sigorta kapsamindadir");  
    }  
}
```

## Kural 2)

Abstract method'lari implement etmek icin oncelikle overriding yapilmalidir.

```
public class Isci extends Personel{  
  
    public static void main(String[] args) {  
        Isci isci1=new Isci();  
        isci1.isim="Mehmet";  
        System.out.println(isci1.isim);  
        isci1.ozelSigorta();  
        isci1.maasHesapla();  
    }  
  
    @Override  
    public void maasHesapla() {  
        System.out.println("maas : "+ 30*8*15);  
    }  
    @Override  
    public void mesaiBilgisi() {  
        System.out.println("Isciler 8 saat mesai yapar");  
    }  
}
```

## Kural 3)

Abstract class, abstract olmayan (concrete) method'lara da sahip olabilir. Concrete method'lar implement edilmek ZORUNDA DEGILDIR. Parent-child iliskisi ile kullanilabilirler veya istenirse override edilebilirler

# Abstract Classes

## Kural 4)

Abstract bir child class, parent'i olan abstract class'lardaki method'lari implement etmek ZORUNDA DEGILDIR. Parent-child iliskisi ile tum method'lara ulasabilir.

```
public abstract class Personel {  
  
    public String isim;  
    public int maas;  
  
    public abstract void maasHesapla();  
    public abstract void mesaiBilgisi();  
  
    public void ozelSigorta() {  
        System.out.println("Bu personel ozel sigorta kapsamindadir");  
    }  
}
```

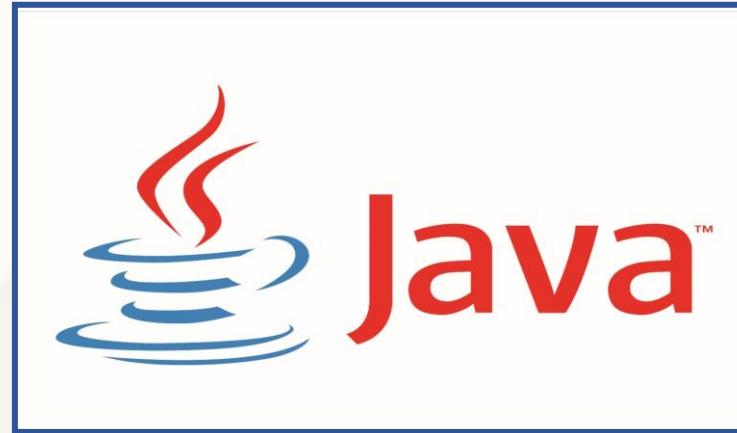
## Kural 5)

Abstract class'lar somutlastirilamaz (can not be instantiated) yani abstract class'larda obje olusturulamaz.

```
public abstract class YanHizmetler extends Personel{  
  
    public static void main(String[] args) {  
  
        YanHizmetler obj1 = new YanHizmetler();  
    }  
}
```

## Kural 6)

Abstract class'lar private veya final olarak tanimlanamaz



16 ARALIK 2021  
DERS 43

Interfaces

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

### 1) Abstraction : Sanallasirma,

Eger bir class obje uretmek icin degil de sadece kendisine child olacak class'larin neler yapmasi gerektigini belirlemek icin aciliyorsa buna abstraction denir

- Abstraction konusunda Java bize 2 alternatif sunar

A- Interface ile tam abstraction

B- Abstract class ile kismi olarak abstraction yapabiliriz

### 2) Abstract class'da abstract ve concrete method'lar bulunabillr, ama concrete blr class'da abstract method olmaz

### 3) Abstract method yapmak icin basina abstract keyword yazilir ve body yazilmaz.

### 4) Abstract method'lar concrete child class'lardan override EDILMEK ZORUNDADIR.

Eger abstract bir class'in child class'inda parent class'daki abstract method'larin tamamini override etmezsek Java elass isminin altini cizer ve bize "create unimplemented methods" der

### 5) final, static ve private method'lar abstract method yapitamaz

### 6) Abstract ctass'lardan obje otusturulamaz

# Abstract Classes

Asagidaki kod calistirilirsa compile time error olur mu ?

```
public abstract class Animal {  
    public abstract String getName();  
}
```

```
public abstract class BigCat extends Animal {  
    public abstract void roar();  
}
```

```
public class Lion extends BigCat {  
    public String getName() {  
        return "Lion";  
    }  
    public void roar() {  
        System.out.println("The Lion lets out a loud ROAR!");  
    }  
}
```

# Abstract Classes

Asagidaki kod calistirilirsa compile time error olur mu ?

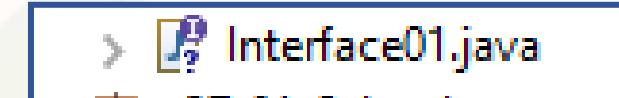
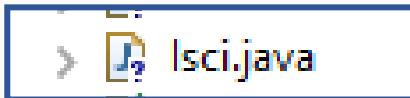
```
public abstract class Animal {  
    public abstract String getName();  
}
```

```
public abstract class BigCat extends Animal {  
    public String getName() {  
  
        return "BigCat";  
    }  
    public abstract void roar();  
}
```

```
public class Lion extends BigCat {  
    public void roar() {  
        System.out.println("The Lion lets out a loud ROAR!");  
    }  
}
```

# Interfaces

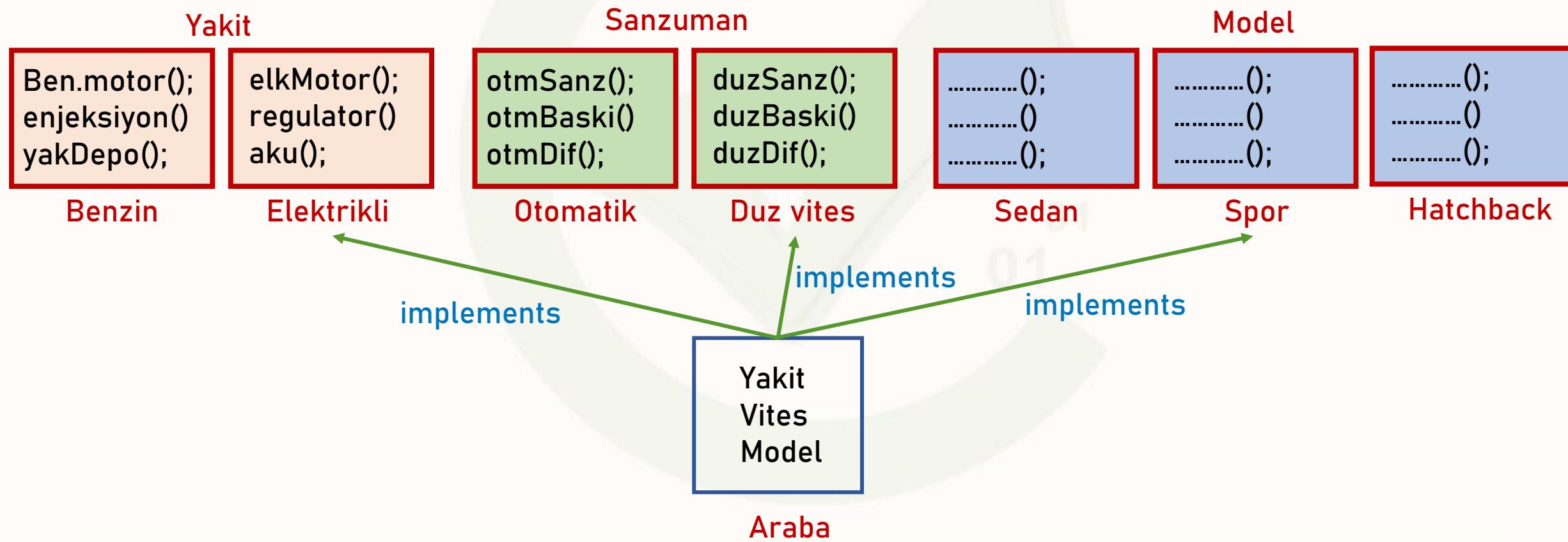
- Interface bir class degildir. Interface interface'dir



- Interface içinde sadece kendisinden türeyen sınıfların içini doldurmak zorunda olduğu, body'si olmayan method'ların oluşturduğu bir yapıdır.
- Kısacası kendisini inherit eden class'lar için, yerine getirmeleri gereken metodları belirten "**tüm alanları doldurulmak zorunda olan bir sablon**" gibidir.
- Interface'ler somutlastırılamaz (can not be instantiated) yani Interface'de obje oluşturulamaz.

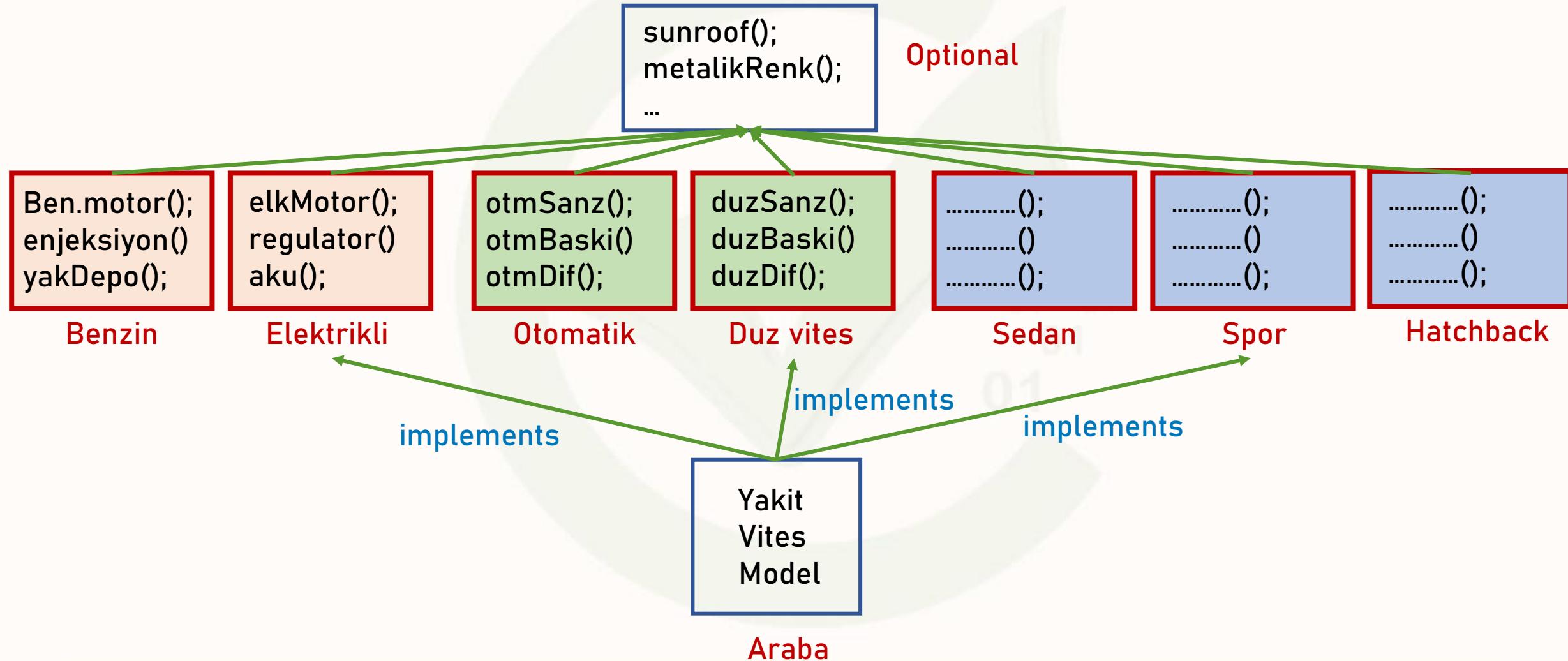
# Interfaces

- Interface bir cesit to do list'dir. Concrete class'lari interface'deki tum metodlari implement etmek zorunda birakir. **Nasil yapilacagina degil ne yapilacagina odaklanir**
- Bir class birden fazla class'a extend edilemez ama birden fazla interface'e implement edilebilir.



# Abstract Classes

Optional bazi method'lat koymak istersek, en uste concrete bir parent koyariz.



# Abstract Classes vs Interface

## Abstract classes

- 1) Class'dir
- 2) abstract ve concrete method'lar konabilir
- 3) Kismi olarak abstraction saglar.
- 4) Birden fazla abstract class'a direk child olunamaz. Coklu inheritance'i desteklemez.
- 5) Hiz acisindan avantajlidir
- 6) Icindeki tüm nesnelerin "public" olması zorunlu degildir
- 7) soyut olmayan metodlar static olarak tanımlanabilir.
- 8) Abstract class constructor'a sahiptir

## Interface

- 1) Class degildir.
- 2) sadece abstract method'lar konabilir.
- 3) Tam abstraction (soyutluk) saglar
- 4) Bir class'dan istediginiz kadar interface'i inherit edebilirsiniz. Coklu inheritance'a uygundur.
- 5) Hiz acisindan abstract class'a gore yavastir.
- 6) Icindeki tüm nesnelerin "public" olması gerekir
- 7) metodlar static olamaz
- 8) Interface constructor'a sahip degildir

# Interfaces

```
public interface Interface01 {  
  
    void vites();  
    public void aku();  
    abstract void teker();  
    public abstract void motor();  
}
```

- Interface'e **sadece abstract public method'lar** konabilir.  
Yandaki farkli yazimlar interface icin ayni seylerdir
- Return type'lar farkli olabilir

- Interface icindeki variable'lar mutlaka **public, static , ve final** olmalıdır. private veya protected variable'lar compile time error verir.

Yandaki farkli yazimlar interface icin ayni seylerdir

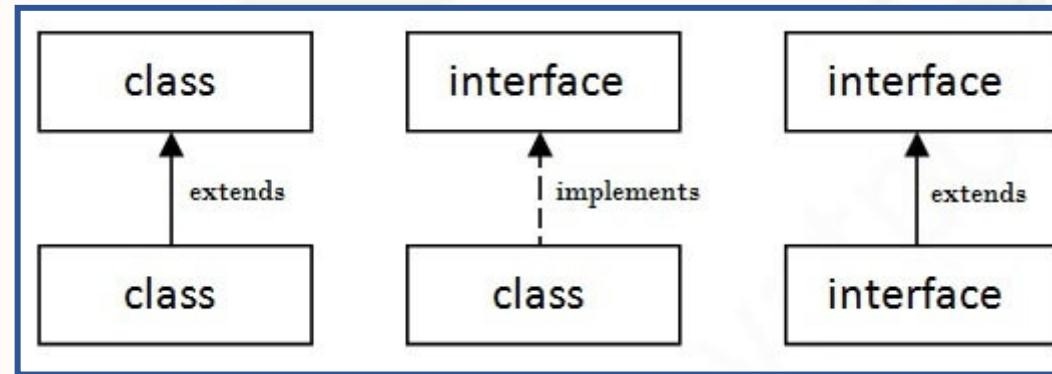
```
public interface Interface01 {  
  
    int SAYI1=10;  
    public int SAYI2=20;  
    public static int SAYI3=30;  
    public static final int SAYI4=40;  
}
```

- Interface icindeki variable'lari mutlaka **initialize** etmek zorundasiniz, aksi takdirde Compile Time Error alirsiniz.

*int a = 12;* gibi yapmalisiniz.

## Interface'ler Icin Inheritance Kurallari

- 1) Interface'lerde inheritance yapmak icin **implements** keyword'u kullanilir.



- 2) Bir class birden fazla Interface'e **implements** ile baglanabilir

```
public class Arabam implements ElektrikMotor, OtomatikVites, Sedan{  
}
```

## Interface'ler Icin Inheritance Kurallari

3) Birden fazla Interface'i **implements** ile inherit ettigimizde aynı isimde variable veya method'larla karşılasabiliriz.

Bu durumda Java ne yapacagini net olarak bilmek isteyeceginden istedigimiz variable ismini interface ismi ile birlikte yazariz.

```
public class Arabam implements ElektrikMotor, OtomatikVites, Sedan {  
  
    public static void main(String[] args) {  
        System.out.println("Kasa Model no : " + Sedan.MODELNO);  
        System.out.println("Vites Model no : " + OtomatikVites.MODELNO);  
        System.out.println("Motor Model no : " + ElektrikMotor.MODELNO);  
    }  
}
```

Method'lar mecburen override yapılacağı için hangi interface'den alındığının hiçbir önemi yoktur.

```
@Override  
public void yakitTuru() {  
  
}
```

## Interface'ler Icin Inheritance Kurallari

4) Ayni isme fakat farkli return type'a sahip methodlari olan Interface'leri ayni class'dan inherit edemeyiz .

```
public interface ElektrikMotor {  
    int MODELNO=10;  
  
    void yakitTuru();  
}
```

```
public interface OtomatikVites {  
    int MODELNO=20;  
  
    String yakitTuru();  
}
```

implements

implements

```
3 public class Arabam implements ElektrikMotor, OtomatikVites, Sedan {  
4  
5    public static void main(String[] args) {  
6        }  
7    }  
8  
9 }
```

# Interface'lerde Body'si Olan Method Yazilabilir mi ?

Java8'e kadar interface icinde "body'si olan method" olusturulamiyordu. Java8 ve uzeri versiyonlarda bu opsiyonu ekledi.

A) method'un basina "**default**" keyword'u kullanabilirsiniz.

```
public interface I05 {  
  
    default int add(int a, int b) {  
        return a+b;  
    }  
}
```

Burada kullandigimiz "default" access modifier degil, method turudur. (asagida gorulecegi üzere method'un access modifier'I public olarak belirtmisen, Java default kelimesini kullanmamiza izin veriyor)

```
public interface I05 {  
  
    public default int add(int a, int b) {  
        return a+b;  
    }  
}
```

# Interface'lerde Body'si Olan Method Yazilabilir mi ?

B) return type'dan once “**static**” keyword'u kullanabilirsiniz.

```
public interface I05 {  
    public static int add(int a, int b) {  
        return a+b;  
    }  
}
```

Burada kullandigimiz “static” daha once kullandigimiz static gibi degildir. Asagida gorulecegi gibi ayni paketteki baska class'dan clasismi.methodismi() yazarak kullanilamiyor.

```
I05.add(10,20);
```

**NOT :** default veya static keyword'u ile olusturulan method'lar override yapilmak zorunda degildir.

# Interface'lerde Body'si Olan Method Yazilabilir mi ?

**Soru :** Interface'de kullanabildigimiz "static method" ve "default method" larin farki nedir?.

**Cevap :** Default keyword'u ile olusturulan method'lari obje kullanarak cagirabiliriz ama static olanlari inherit ettigimiz interface'den ismi ile cagirabiliriz.

```
public interface I05 {  
    public static int add(int a, int b)  
        return a+b;  
}
```

```
public class Runner implements I05{  
  
    public static void main(String[] args) {  
  
        I05.add(10, 20);  
  
    }  
}
```

```
public interface I05 {  
  
    public default int topla(int a, int b) {  
        return a+b;  
    }  
}
```

```
public class Runner implements I05{  
  
    public static void main(String[] args) {  
  
        Runner rnr=new Runner();  
        System.out.println(rnr.topla(20, 40));  
    }  
}
```

## Abstract Class - Interface Tekrar Soruları

- 1) Abstract class kullanilarak obje olusturamayiz. ~~True/False~~
- 2) Interface kullanilarak obje olusturabiliriz. ~~True/False~~
- 3) `public abstract int sumOfTwo(int n1, int n2) { }` syntax olarak dogru mudur ? ~~True/False~~
- 4) `public class Animal { public abstract void sound() }` syntax olarak dogru mudur ? ~~True/False~~
- 5) `public abstract class Animal {  
 public abstract void eat();  
 public void sound(){ } }` syntax olarak dogru mudur ? ~~True/False~~
- 6) Abstract class final olarak tanimlanamaz. ~~True/False~~
- 7) `public abstract class Animal {  
 private abstract void sound(); }` syntax olarak dogru mudur ? ~~True/False~~
- 8) Java birden fazla class'a extends ile inherit etmenize izin vermez. ~~True/False~~
- 9) Java birden fazla interface'e implements ile inherit etmenize izin verir. ~~True/False~~
- 10) Bir interface abstract veya concrete method'lara sahip olabilir. ~~True/False~~

# Interfaces

## Soru 1)

Which of the following statements can be inserted in the blank line so that the code will compile successfully? (Choose all that apply)

```
public interface CanHop {}  
public class Frog implements CanHop {  
    public static void main(String[] args) {  
        _____ frog = new TurtleFrog();  
    }  
}  
  
public class BrazilianHornedFrog extends Frog {}  
public class TurtleFrog extends Frog {}
```

- A. Frog
- B. TurtleFrog
- C. BrazilianHornedFrog
- D. CanHop
- E. Object
- F. Long

A, B, D, E. The blank can be filled with any class or interface that is a supertype of TurtleFrog. Option A is a superclass of TurtleFrog, and option B is the same class, so both are correct. BrazilianHornedFrog is not a superclass of TurtleFrog, so option C is incorrect. TurtleFrog inherits the CanHope interface, so option D is correct. All classes inherit Object, so option E is correct. Finally, Long is an unrelated class that is not a superclass of TurtleFrog, and is therefore incorrect.

# Interfaces

## Soru 2)

Choose the correct statement about the following code:

```
1: interface HasExoskeleton {  
2:     abstract int getNumberOfSections();  
3: }  
4: abstract class Insect implements HasExoskeleton {  
5:     abstract int getNumberOfLegs();  
6: }  
7: public class Beetle extends Insect {  
8:     int getNumberOfLegs() { return 6; }  
9: }
```

- A. It compiles and runs without issue.
- B. The code will not compile because of line 2.
- C. The code will not compile because of line 4.
- D. The code will not compile because of line 7.
- E. It compiles but throws an exception at runtime.

D. The code fails to compile because `Beetle`, the first concrete subclass, doesn't implement `getNumberOfSections()`, which is inherited as an abstract method; therefore, option D is correct. Option B is incorrect because there is nothing wrong with this interface method definition. Option C is incorrect because an abstract class is not required to implement any abstract methods, including those inherited from an interface. Option E is incorrect because the code fails at compilation-time.

# Interfaces

## Soru 3 )

Choose the correct statement about the following code:

```
1: public interface Herbivore {  
2:     int amount = 10;  
3:     public static void eatGrass();  
4:     public int chew() {  
5:         return 13;  
6:     }  
7: }
```

- A. It compiles and runs without issue.
- B. The code will not compile because of line 2.
- C. The code will not compile because of line 3.
- D. The code will not compile because of line 4.
- E. The code will not compile because of lines 2 and 3.
- F. The code will not compile because of lines 3 and 4.

F. The interface variable amount is correctly declared, with public and static being assumed and automatically inserted by the compiler, so option B is incorrect. The method declaration for eatGrass() on line 3 is incorrect because the method has been marked as static but no method body has been provided. The method declaration for chew() on line 4 is also incorrect, since an interface method that provides a body must be marked as default or static explicitly. Therefore, option F is the correct answer since this code contains two compile-time errors.

# Interfaces

## Soru 4 )

Choose the correct statement about the following code:

```
1: public interface CanFly {  
2:     void fly();  
3: }  
4: interface HasWings {  
5:     public abstract Object getWindSpan();  
6: }  
7: abstract class Falcon implements CanFly, HasWings {  
8: }
```

- A. It compiles without issue.
- B. The code will not compile because of line 2.
- C. The code will not compile because of line 4.
- D. The code will not compile because of line 5.
- E. The code will not compile because of lines 2 and 5.
- F. The code will not compile because the class Falcon doesn't implement the interface methods.

A. Although the definition of methods on lines 2 and 5 vary, both will be converted to `public abstract` by the compiler. Line 4 is fine, because an interface can have `public` or `default` access. Finally, the class `Falcon` doesn't need to implement the interface methods because it is marked as `abstract`. Therefore, the code will compile without issue.

# Interfaces

Soru 5 )

Which statements are true for both abstract classes and interfaces? (Choose all that apply)

- A. All methods within them are assumed to be abstract.
- B. Both can contain public static final variables.
- C. Both can be extended using the extend keyword.
- D. Both can contain default methods.
- E. Both can contain static methods.
- F. Neither can be instantiated directly.

B, C, E, F. Option A is wrong, because an abstract class may contain concrete methods. Since Java 8, interfaces may also contain concrete methods in form of static or default methods. Although all variables in interfaces are assumed to be public static final, abstract classes may contain them as well, so option B is correct. Both abstract classes and interfaces can be extended with the extends keyword, so option C is correct. Only interfaces can contain default methods, so option D is incorrect. Both abstract classes and interfaces can contain static methods, so option E is correct. Both structures require a concrete subclass to be instantiated, so option F is correct.

# Interfaces

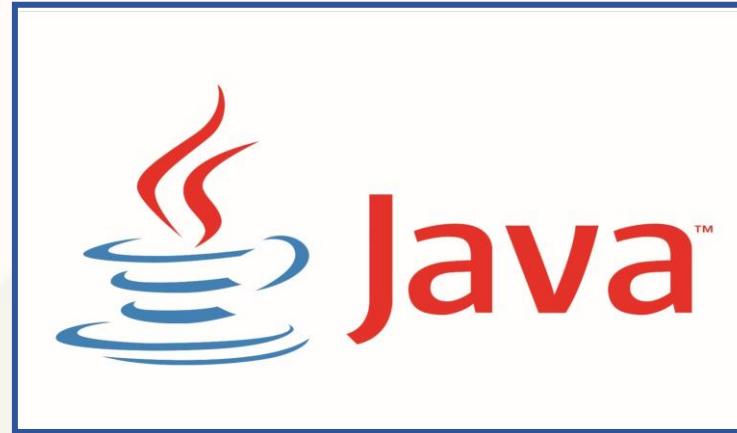
Soru 6 )

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

```
1: interface HasVocalCords {  
2:     public abstract void makeSound();  
3: }  
4: public interface CanBark extends HasVocalCords {  
5:     public void bark();  
6: }
```

- A. The CanBark interface doesn't compile.
- B. A class that implements HasVocalCords must override the makeSound() method.
- C. A class that implements CanBark inherits both the makeSound() and bark() methods.
- D. A class that implements CanBark only inherits the bark() method.
- E. An interface cannot extend another interface.

C. The code compiles without issue, so option A is wrong. Option B is incorrect, since an abstract class could implement HasVocalCords without the need to override the makeSound() method. Option C is correct; any class that implements CanBark automatically inherits its methods, as well as any inherited methods defined in the parent interface. Because option C is correct, it follows that option D is incorrect. Finally, an interface can extend multiple interfaces, so option E is incorrect.



17 ARALIK 2021  
DERS 45

Iterators  
Collections

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Interface : bildigimiz class yapisindan farklidir, interface bir class degildir. Interface'lerin constructor'lari yoktur ve obje olusturulamaz
  - 2) Java da full abstraction yapmak icin interface kullaniriz, interface icerisinde concrete method bulunmaz ve abstract class'lara gore yavastir.
  - 3) Interface Child class'larin bulundurmak zorunda oldukları method'lari belirleyen bir to-do list gibidir
  - 4) Concrete class'lardan interface'i inherit etmek icin extends keyword yerine implements keyword kullanilir
  - 5) Eger bir interface'den baska bir interface'i inherit edeceksek extends keyword kullaniriz
  - 6) Concrete class'lar birden fazla concrete class'i extends ile inherit edemezler ancak implements ile bir class birden fazla interface'l inherit edebilir
  - 7) Interface icerisindeki tum variable'lar abstract,public ve final'dir, final oldukları icin de olustururken deger atamak zorunludur.
  - 8) Bir interface icerisinde body'si olan method olusturulabilir mi ?
    - Java interface icerisinde body'si olan method olusturabilmemiz icin 2 alternatif gelistirmistir
      - A- default keyword kullanilabilir ( burada kullanılan default access modifier olan degildir)
      - B- static keyword kullanilabilir
- aralarındaki fark default keyword ile olusturulan method'a diger class'lardan obje ile ulasabilirken, static keyword ile olusturulana Interface ismi ile ulasabiliriz.

## Socrative Quiz

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

Sure : 13 Dakika

# Iterators

- Java iterator, collection elemanlarımızın arasında gezinmemize ve elemanları degistirmemize yarar.
- Collections'da her element index yapisini desteklemez, index olmadan tum elementlere ulasmak icin for-each loop kullanabiliriz ancak for-each loop ile elementleri kalici olarak degistirme veya silme imkani olmadigi icin Iterator kullanimini tercih ederiz.

Method Summary	
All Methods	Instance Methods
Modifier and Type	Method and Description
default void	<code>forEachRemaining(Consumer&lt;? super E&gt; action)</code> Performs the given action for each remaining element until all elements have been processed or the action throws an exception.
boolean	<code>hasNext()</code> Returns true if the iteration has more elements.
E	<code>next()</code> Returns the next element in the iteration.
default void	<code>remove()</code> Removes from the underlying collection the last element returned by this iterator (optional operation).

# Iterators

## Iterator method'lari (turkce aciklama ile)

Sr.No.	Metod ve Açıklama
1	<b>boolean hasNext( )</b> Hala dizide eleman varsa true döner değilse false döner
2	<b>Object next( )</b> Bir sonraki elamanı döndürür
3	<b>void remove( )</b> O anki elemani silmeye yarar.

**Soru 1) Bir listedeki tum elementleri iterator kullanarak silin.**

```
public static void main(String[] args) {  
  
    List<String> list = new ArrayList<>();  
  
    list.add("A");  
    list.add("B");  
    list.add("C");  
  
    System.out.println(list);  
  
    Iterator<String> li1=list.iterator();  
    while(li1.hasNext()) {  
        li1.next();  
        li1.remove();  
    }  
    System.out.println(list);  
}
```

# ListIterators

**NOT :** ListIterator daha fazla method'a sahip oldugu icin daha cok kullanilir.

ListIterator, Iterator interface'in child'idir.

Method and Description
<code>add(E e)</code> Inserts the specified element into the list (optional operation).
<code>hasNext()</code> Returns true if this list iterator has more elements when traversing the list in the forward direction.
<code>hasPrevious()</code> Returns true if this list iterator has more elements when traversing the list in the reverse direction.
<code>next()</code> Returns the next element in the list and advances the cursor position.
<code>nextIndex()</code> Returns the index of the element that would be returned by a subsequent call to <code>next()</code> .
<code>previous()</code> Returns the previous element in the list and moves the cursor position backwards.
<code>previousIndex()</code> Returns the index of the element that would be returned by a subsequent call to <code>previous()</code> .
<code>remove()</code> Removes from the list the last element that was returned by <code>next()</code> or <code>previous()</code> (optional operation).
<code>set(E e)</code> Replaces the last element returned by <code>next()</code> or <code>previous()</code> with the specified element (optional operation).

**Iterator ve ListIterator arasındaki en büyük fark,**

Iterator'ın koleksiyondaki öğeleri yalnızca ileri yönde hareket ettirebilmesidir ListIterator ise bir koleksiyondaki öğeleri hem ileri hem de geri yönde hareket ettirebilir .

# ListIterators

Soru 2) String bir listedeki tüm elementlerin sonuna X ekleyin

For-Each Loop ile

```
List<String> list = new ArrayList<>();  
  
list.add("A");  
list.add("B");  
list.add("C");  
  
System.out.println(list);  
  
for (String each : list) {  
    each=each+"X";  
}  
  
System.out.println(list);
```

ListIterator ile

```
public static void main(String[] args) {  
  
    List<String> list = new ArrayList<>();  
  
    list.add("A");  
    list.add("B");  
    list.add("C");  
  
    System.out.println(list);  
  
    ListIterator<String> li1=list.listIterator();  
    while(li1.hasNext()) {  
        String str=li1.next();  
        li1.set(str+"X");  
    }  
    System.out.println(list);  
}
```

## Iterators

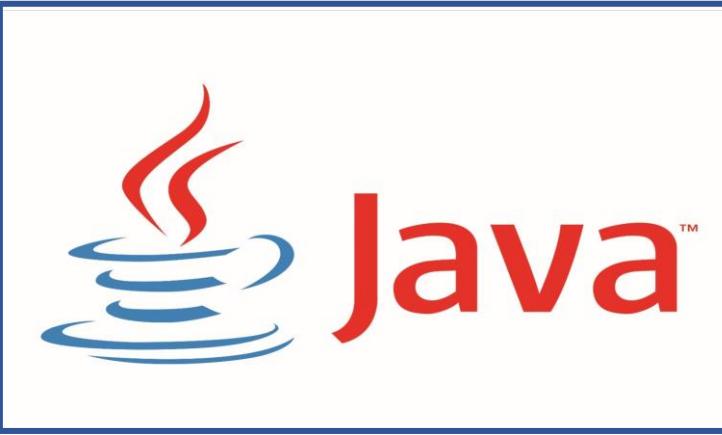
**Soru 3)** Bir listedeki istenilen sayı aralığında olmayan elementleri silen bir program yazınız ...  
(2. liste oluşturmadan, gecerli liste üzerinde işlem yapınız)

Orn : [2,13,56,23,45,14,40] istenilen aralık 20 ile 40 arası (sinirlar dahil)  
output: [23,40]

**Soru 4)** Bir listedeki elementleri iterator kullanarak sondan başa doğru yazdırın

**Soru 5) (iterator ile index kullanımına örnek)** Bir listedeki ilk n elemanı iterator kullanarak 5 artırın.

Orn :   list : [2,13,56,23,45,14,40]  
         artırılması istenilen eleman sayısı : 3  
         output: [7,18,61,23,45,14,40]



17 ARALIK 2021  
DERS 45

Collections  
Linked List

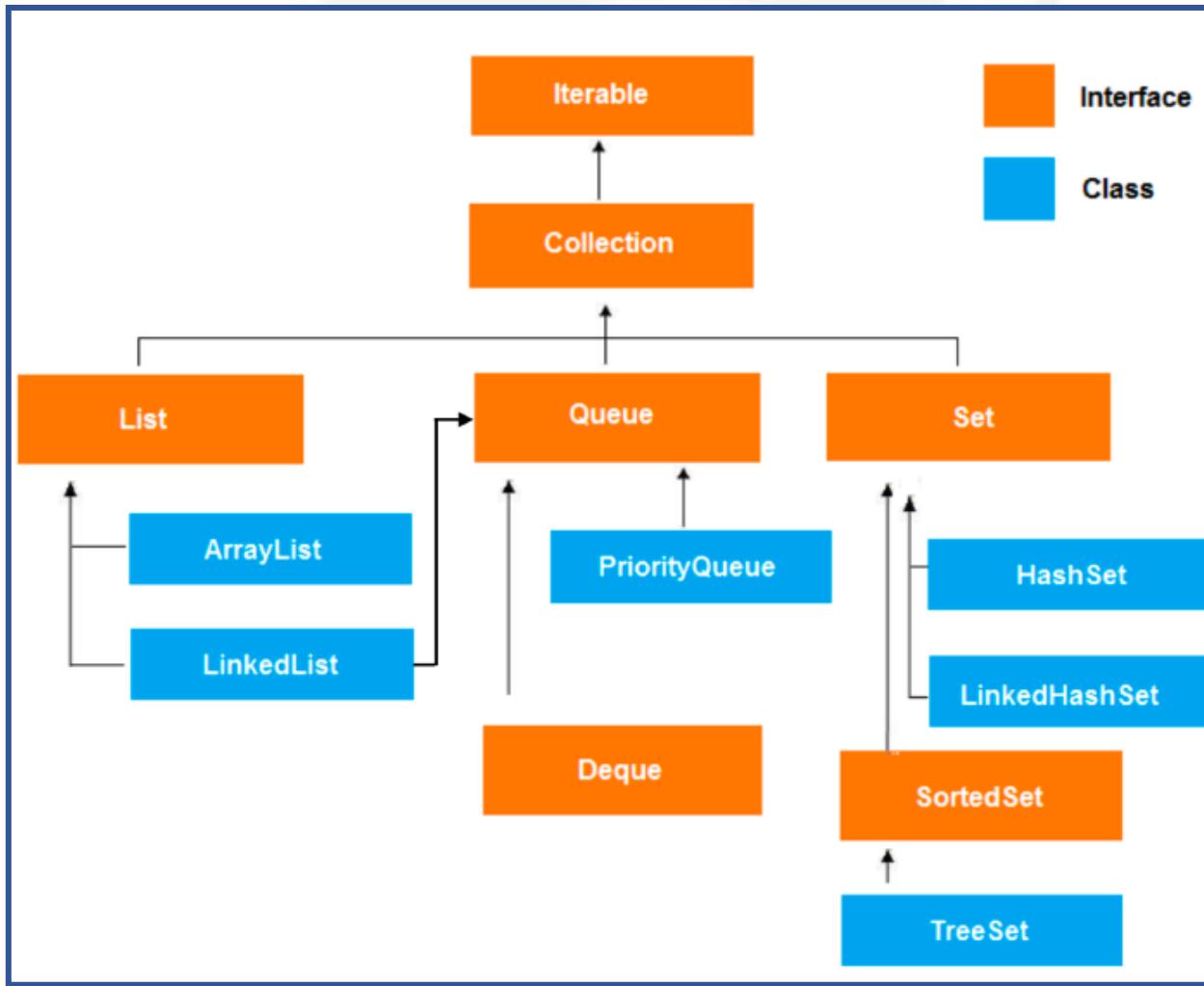
Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Iterator : Collection'larda elementler arasında gezinmemizi saglar, normalde index kullanan yapilar icin index ve loop ile elemanlar üzerinde gezinebilir ve degisiklikler yapabiliriz, ancak collection'da her yapi index kullanmadigindan iterator kullanmamız gereklidir.
- 2) Iterator interface'dir dolayisiyla kullanacagımız collection class'ında iterator method'lari override edilmistir. Biz collectionsmi.iteratorMethod() diyerek kullanacagımız collection class'ında bulunan ilgili iterator method'unu cagirmış oluruz.
- 3) Iterator interface'inde sadece 3 method  
  
hasNext( ) : iterator'in bulundugu konumdan hemen sonra eleman var mı ? diye sorar ve true veya false doner  
  
next( ) : iterator'i bir sonraki konumuna hareket ettirir ve üzerinden gectigi elemani bize dondurur  
  
remove( ) : son olarak üzerinden gecilen elemani siler.
- 4) ListIterator da Iterator interface'inin child interface'dir. ListIterator parent'ı olan Iterator'a göre daha fazla method'a sahiptir.  
  
set ( ) method'u üzerinde olduğumuz collection elementini kalıcı olarak degistirmemizi saglar  
Iterator'da hep ileri doğru hareket ve bununla ilgili method'lar olmasına karşılık ListIterator'da next ile birlikte previous method'lari da vardır.  
  
Dolayisiyla ListIterator kullanarak ileri ve geri hareket edebiliriz.
- 5) Iterator ile calisirken dikkat etmemiz gereken en onemli konu biz ileri veya geri demedikce iterator son bıraktigimiz yerde bekler.

# Collections

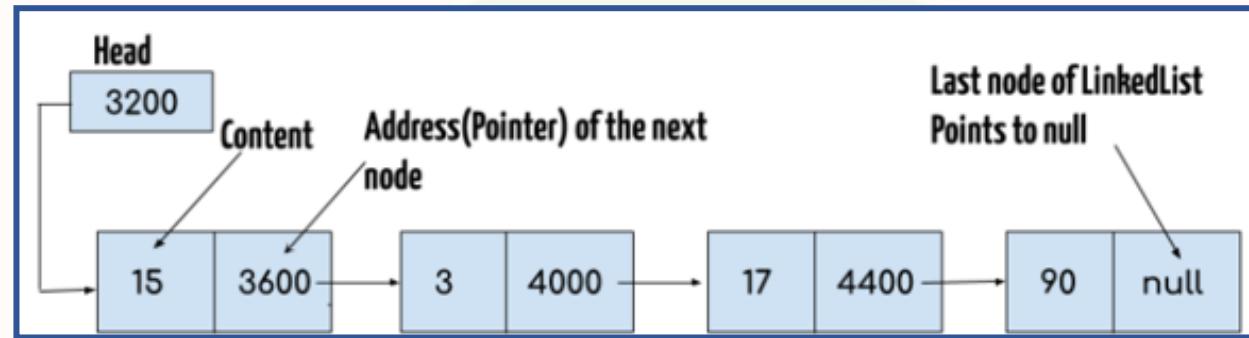
**Collections** : nesnelerden oluşan bir topluluğu bir arada tutan yapılardır.



5 kelime'ye dikkat

- 1) Set (Kume)
- 2) Queue (Sıra) 
- 3) Linked (Baglı)
- 4) Tree (DogalSıralı)
- 5) Hash 

# Collections / Linked List



## Linked List (Class)

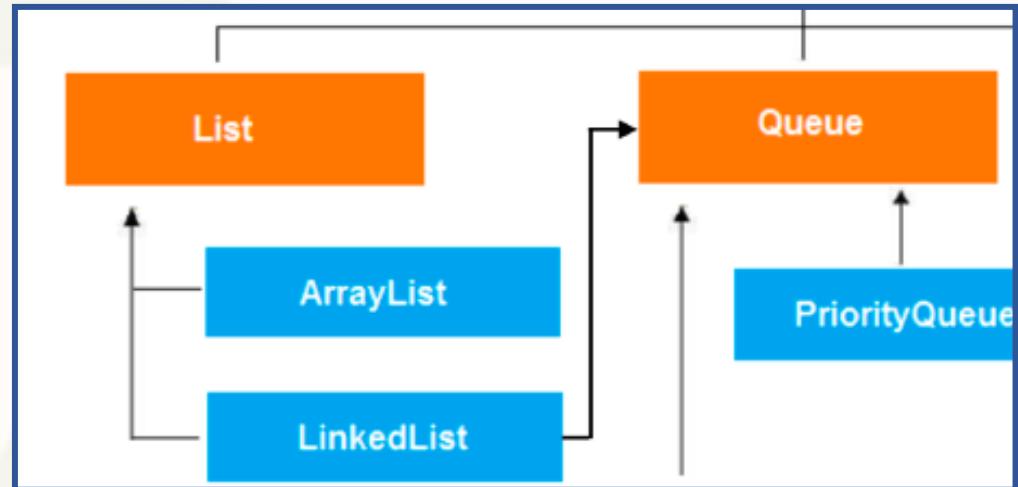
- 1) İlk eleman her zaman head'dir ve head'de data yoktur, sadece address vardır.
- 2) Son eleman(tail) null'i point eder.
- 3) Her elemanın içinde **data** ve **address** kismı olmak üzere iki kisim vardır.
- 4) Tüm elemanlar **pointer'lar** / **address'ler** kullanılarak birbirine bağlanır.
- 5) Her eleman **node** olarak adlandırılır.
- 6) Array'lerden daha **dynamic**'dırler **insert(ekleme)** ve **delete(silme)** işlemlerinde başarılıdır.
- 7) Pointer yapısından dolayı bir elemana ulaşmada yararlıdır.

# Collections / Linked List

**Linked List**, 2 interface'in child class'ıdır. Obje olustururken data turu olarak istedigimiz parent interface'i secebilir ve o interface'deki ozellikleri kullanabiliriz.

Data turu olarak **LinkedList** sectigimizde de ayni method'lari kullanabiliriz, cunku **LinkedList** concrete bir class'dir ve parent'i olan interface'lerdeki tum method'lari override etmek zorundadir.

\*\*\* **LinkedList** 2 interface'i implement ettiği için her ikisinin ustun ozelliklerini kullanabilir.



```
public static void main(String[] args) {  
  
    LinkedList<Integer> l11 = new LinkedList<>();  
  
    l11.add(10); // LinkedList class'indan add methodu  
    l11.element(); // LinkedList class'indan element methodu  
    l11.remove(1); // LinkedList class'indan remove methodu  
  
    Queue<Integer> l12 = new LinkedList<>();  
    l12.add(13); // Queue class'indan add methodu  
    l12.element(); // Queue class'indan element methodu  
  
    List<Integer> l13 = new LinkedList<>();  
    l13.add(15); // List class'indan add methodu  
    l13.remove(0); // List class'indan remove methodu  
}
```

## LinkedList Method'lari

1) **add();** LinkedList'in sonuna  
istenen elemani ekler

2) **add(1,"A");** istenen index'e istenen  
elemani ekler

3) **addAll(coll);** istenen collection'in  
tum elemanlarini ekler

4) **addAll(2, coll);** istenen collection'in tum  
elemanlarini istenen index'e ekler

## LinkedList Method'lari

5) **addFirst();** istenen elemani, ilk eleman olarak ekler

6) **addLast();** istenen elemani, son eleman olarak ekler

7) **remove();** ilk elemani siler

8) **removeFirst();** ilk elemani siler (daha hizlidir)

9) **remove(index);** istenen indexdeki elemani siler ve silinen elemani dondurur

## LinkedList Method'lari

10) `remove(eleman);` istenen elemani siler sildi ise true,  
bulamadi ise false dondurur

11) `removeFirstOccurrence("str");` istenen elemanın,ilkini  
siler

12) `removeLast();` son elemani siler

13) `removeAll(list);` istenen listedeki tüm elemanları  
siler

## LinkedList Method'lari

14) **contains(eleman);** istenen eleman listede var ise true,  
yoksa false dondurur

15) **containsAll(liste);** istenen listenin tumu aranan listede  
var ise true, yoksa false dondurur

16) **get(index);** istenen indexdeki elemani getirir

## LinkedList Method'lari

**Soru :** Node'lari "Ali", "Veli", "Can" ve "Ayse" olan bir LinkedList olusturun.

Kullanicidan bir isim alin. Bu isim LinkedList'de varsa silin ve kullaniciya "Bu isim LinkedList'de vardi ve silindi" diye mesaj verin.

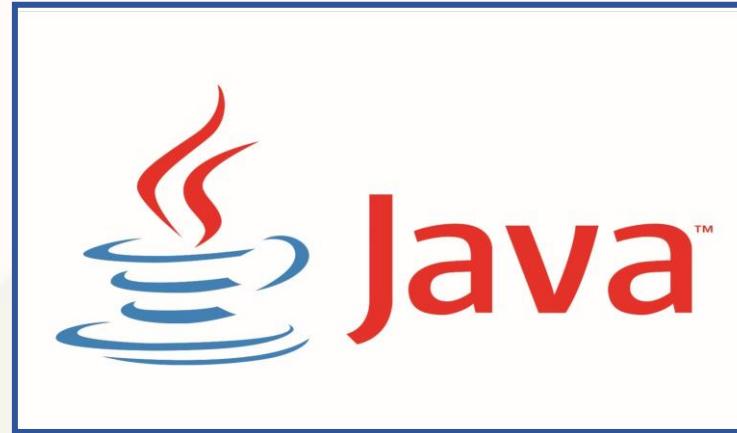
Bu isim LinkedList'de yoksa "Bu isim LinkedList'de yok bu yuzden silinemedi" diye mesaj verin.

```
public static void main(String[] args) {
    LinkedList<String> l11 = new LinkedList<>();
    l11.add("Ali");
    l11.add("Veli");
    l11.add("Can");
    l11.add("Ayse");

    Scanner scan = new Scanner(System.in);
    System.out.println("Bir isim giriniz");
    String isim = scan.nextLine();

    System.out.println(l11);

    if(l11.remove(isim)) {
        System.out.println("Bu isim LinkedList'de vardi ve silindi");
        System.out.println(l11);
    }else {
        System.out.println("Bu isim LinkedList'de yok bu yuzden silinemedi");
        System.out.println(l11);
    }
    scan.close();
}
```



20 ARALIK 2021  
DERS 46

Collections  
Sets - Queue - Deque

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

## Onceki Dersten Aklimizda Kalanlar

- 1) Collections : Icinde birden fazla element bulundurabilen java objeleridir
- 2) Collections interface'i altında bir çok interface ve class mevcuttur.
- 3) Collection ozellikleri acisindan 3 temel Interface'e ayrılır

List :

Queue : (Kuyruk) ekleme ve cıkarma iki uctan olabilir, farklı collection ogeleri için baş ve son kavramları değişiklik gösterir.

Set: (Kume) kume'de ise sıra önemli değildir, önemli olan herhangi bir elementin o kumede var olup olmadığıdır. (yanınlı tupu)

- 4) Bunun disinda collection icin yapıyi anlamamizi saglayan anahtar kelimeler vardır

Linked : Elementlerin birbirini takip ettiği yapılardır. Ekleme ve çıkarmada hızlıdır ancak linked yapısından dolayı elemanlar arasında hareket etmek yavaşır.

Hash : Hash mekanizması elementleri bulmamızı kolaylaştırır matematiksel bir formuldür.

Tree : Natural sıralı demektir. Eklendiği veya çıkarıldığı her bir element yapıdaki sıralamayı değiştireceği için yavaşır

- 5) Linked List : İki interface'in child class'ıdır. List ve Deque → Queue

Dolayısıyla iki interface'in tüm method'lari override etmiştir ve biz Linked List ile istedigimiz interface'den method'lar kullanabiliriz. İstersek List özelliklerini, istersek de Queue ve Deque özelliklerini kullanabiliriz  
Hatta istersek sadece List veya sadece Queue özelliklerini kullanmak için data turunu o interface'den seçebiliriz

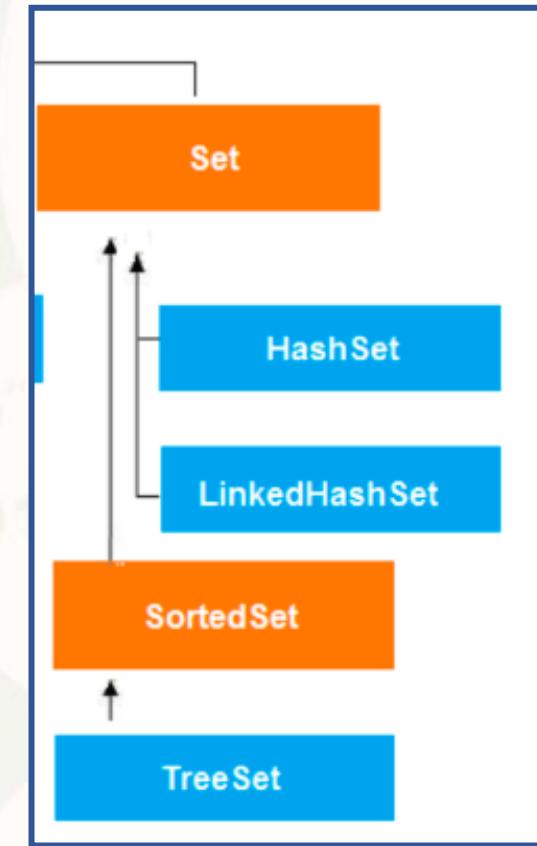
# Collections / Sets

**Set (interface)** , matematikteki kume mantigiyla calisir, her element unique'dir.

Java elementleri unique yapmak icin HASH ALGORITMASI kullanir.

Set, direkt kullanilamaz cunku interface'dir ve obje olusturulamaz. 3 Child class'indan bizim icin onemli olan ozellige gore istedigimizi kullanabiliriz.

Collections'in bir ozelligi de farkli data turundan elementleri ekleyebilmenizdir. Bunun icin esitligin sol tarafindaki  $\leftrightarrow$  (data turu) kaldırılabilir veya data turu olarak Object yazılabilir. Ancak bu tavsiye edilmez cunku Java'nin çok fazla Casting yapması gereklidir.



# Collections

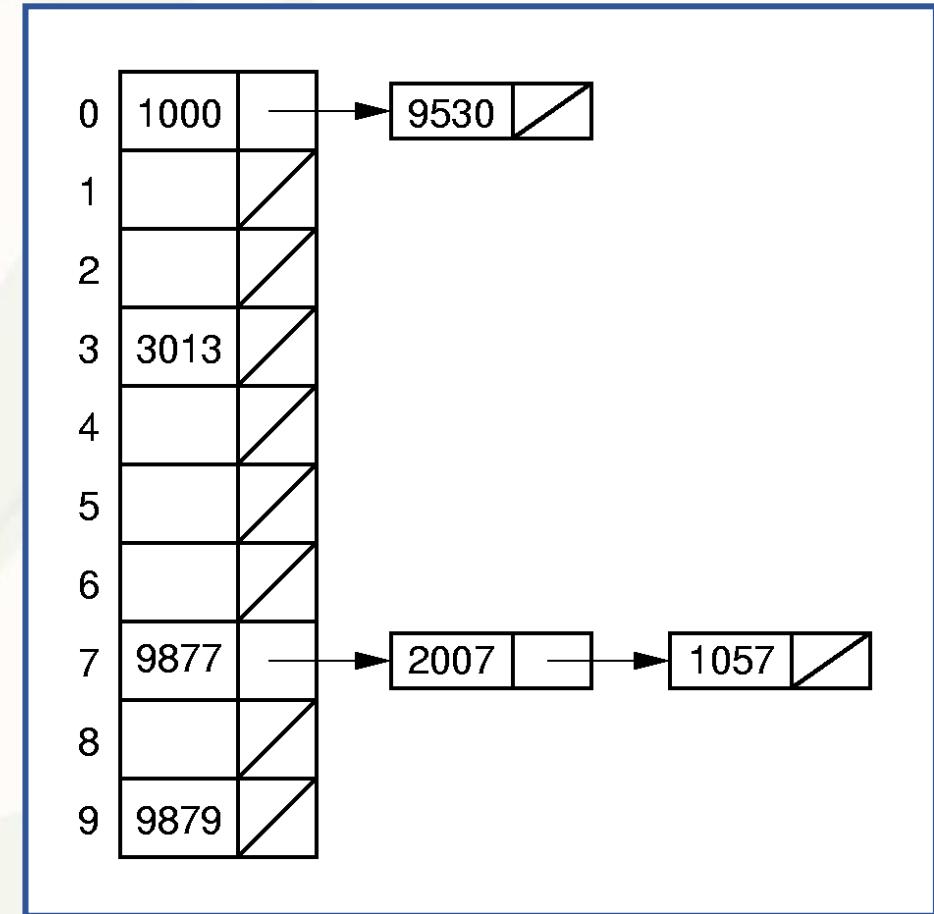
**Hashing**, farklı büyüklükteki girdilerden sabit büyüklükte bir çıktıya dönüştürme sürecine verilen isimdir.

Bu işlem, hash fonksiyonları olarak bilinen matematiksel formüllerin kullanımıyla yapılır.

Universitelerdeki öğrenci numaraları gibi bir öğrenci ismi sorulduğunda numarasını bulursanız onunla ilgili tüm bilgilere ulaşabilirsiniz.

Farklı hash fonksiyonları farklı büyüklüklerde çıktı yaratır fakat her bir hashing algoritması için olası çıktı büyüğü her zaman sabittir.

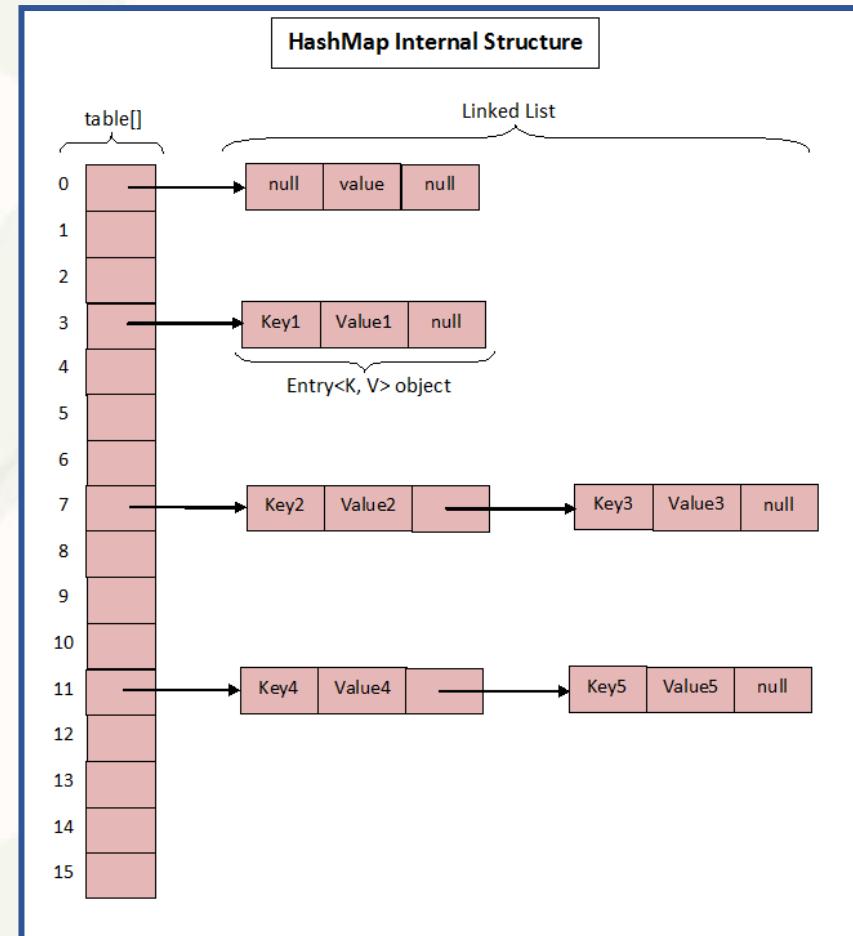
Bir collection'in hash değerini öğrenmek için **hashCode()** method'u kullanılır.



# Collections

## Hashing Nasil Calisir ?

- Bir HashCollection olusturuldugunda Java 16 **bucket** olusturur ve elementleri bu bucket'lara yerlestirmeye baslar.
- Olusturulan bucket'larin %75'i doldugunda Java 16 bucket daha olusturur. Buna **Load Factor** denir.
- Java kullandigimiz key'i kullanarak hash kod uretir. Eger uretilen hash kod daha once uretilen bir hash kod ile ayni ise buna **Hash Collision** denir
- Hash Collision gerceklestiginde cozum icin 2 yol vardır. A) LinkedList kullanmak B) Formulle belirlenen yeni bir hash kod uretmek



## Collections / HashSets

**HashSet**, elemanları için herhangi bir sıralama yapmaz. Elemanları yazdırığınızda veya çağırığınızda herhangi bir sıralama ile gelebilirler.

**HashSet**, duplication'a izin vermez. Eğer bir elemani tekrar HashSet'e eklemek isterseniz eski olan silinip, yeni olan üzerine yazılır.

**HashSet**, null değere izin verir. Bununla birlikte birden fazla null değerini bir HashSet'e eklemek isterseniz sadece bir tane null değeri olur.

Index	
0	
1	
-	
-	
-	
11	defabc
12	
13	
14	cdefab
-	
-	
-	
-	
23	bcdefa
-	
-	
-	
38	abcdef
-	
-	

## Collections / Sets

**Soru:** Verilen bir arraydeki tekrarlı elemanları silip, sadece unique değerlerden oluşan bir liste haline getiren bir program yazınız.

```
public static void main(String[] args) {  
  
    int arr[] = {2, 5, 3, 4, 2, 1, 5, 4, 6, 3, 2, 1, 5, 4};  
    Set<Integer> hs1 = new HashSet<>();  
  
    for (Integer each : arr) {  
        hs1.add(each);  
    }  
  
    System.out.println(hs1);  
}
```



```
[1, 2, 3, 4, 5, 6]
```

## Set Method'lari

1) `add();` Set'e eleman ekler

2) `addAll(coll);` istenen collection'in  
tum elemanlarini ekler

3) `clear();` Tum elemanlari siler

4) `contains(eleman);` istenen eleman sett'e varsa  
true, yoksa false dondurur

5) `containsAll(coll);` istenen coll'in tumu aranan  
sette var ise true, yoksa false  
dondurur

## Set Method'lari

6) **equals(set2);** istenen set'le tum elemanlar ayni ise true, yoksa false dondurur

7) **isEmpty();** Sette hic eleman yoksa true, varsa false dondurur

8) **remove(eleman);** istenen eleman bulursa siler ve true dondurur, bulamazsa false dondurur

9) **removeAll(coll);** coll'nin tum elemanlarini bulursa siler ve true dondurur, bulamazsa false dondurur

10) **size();** set'in eleman sayisini verir

# Collections / Sets

## LinkedHashSet:

- 1) Tekrarli eleman kabul etmezler
- 2) Elemanlari ekleme sirasina(insertion order) gore dizerler.
- 3) Ekleme ve remove islemlerinde hizlidirlar.
- 4) LinkedHashSet, HashSet'den yavastir.

```
public static void main(String[] args) {  
  
    Set<String> lhs1 = new LinkedHashSet<>();  
    lhs1.add("Ali");  
    lhs1.add("Canan");  
    lhs1.add("Veli");  
    lhs1.add("Remziye");  
    System.out.println(lhs1); // [Ali, Canan, Veli, Remziye]  
}
```

## Collections / Sets Method'lari

11) **retainAll(coll);** coll'nin elemanlarinin disindaki tum elemanlari siler, silme islemi yapti ise true, yoksa false dondurur

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

    Set<String> lhs1 = new TreeSet<>();
    lhs1.add("Ali");
    lhs1.add("Canan");
    lhs1.add("Veli");
    lhs1.add("Remziye");
    System.out.println(lhs1); // [Ali, Canan, Remziye, Veli]

    Set<String> lhs2 = new TreeSet<>();
    lhs2.add("Ali");
    lhs2.add("Canan");

    System.out.println(lhs1.retainAll(lhs2)); // true
    System.out.println(lhs1); // [Ali, Canan]

}
```

# Collections / Sets

## TreeSet:

- 1) TreeSet tekrarlı eleman kabul etmez, cunku Set'tir.
- 2) Elemanları natural Order'a(String ise alfabetik, sayı ise küçükten büyük) göre dizer.
- 3) TreeSet, setlerin en yavaşıdır. Bu yuzden az kullanılır

```
public static void main(String[] args) {  
  
    Set<String> lhs1 = new TreeSet<>();  
    lhs1.add("Ali");  
    lhs1.add("Canan");  
    lhs1.add("Veli");  
    lhs1.add("Remziye");  
    System.out.println(lhs1); // [Ali, Canan, Remziye, Veli]  
}
```

## Collections / Sets

**Soru 1 :** Bir TreeSet ve HashSet'e random 100 sayı ekleyin, işlem sürelerini kıyaslayın

**Soru 2 :** İlk soruya 3.bir işlem ekleyelim, set'i HashSet olarak oluşturup elemanları ekleyelim ve sonra TreeSet'e çevirip yazdırıyalım

Long time=System.currentTimeMillis() method'unu kullanın

# Collections / Queue

**Queue** interface'dir dolayisiyla constructor'i yoktur.

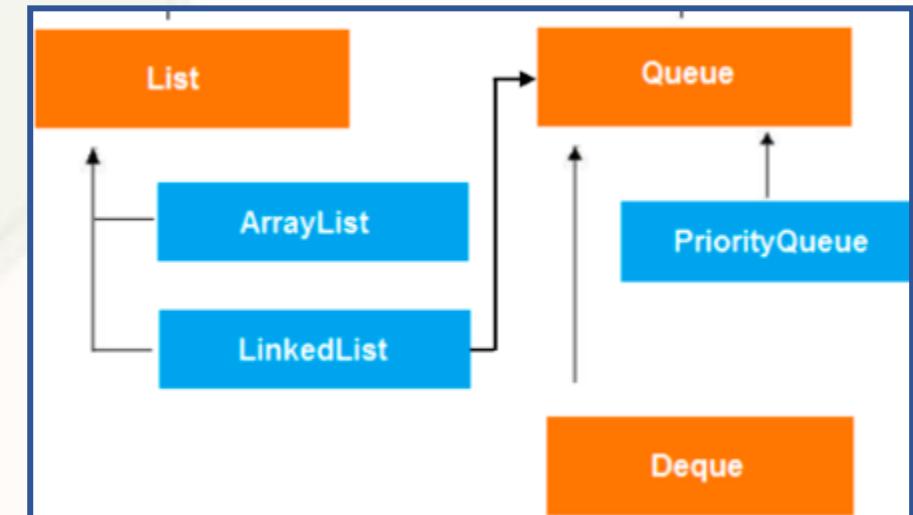
Queue olusturmak icin child class'l olan **LinkedList** veya **PriorityQue** kullanilabilir.

1 ) PriorityQueue constructor'i kullanarak Queue uretirseniz,

Java kendisi bir "priority"(oncelik) kurali uretir ve urettigi bu kurala gore elemanlari dizer.

Istersek biz kendi "priority"(oncelik) kuralimizi uretip elemanlari bu kurala gor dizebiliriz.

2) LinkedList constructor'i kullanarak Queue uretirseniz elemanlari insertion sirasina gore ekler



**NOT :** Queue icin ayirici ozellik : Elemanlar en sona eklenir ve en bastan silinir.

Bu sisteme **FIFO(First In First Out)** denir. Eczaneler, Yemekhaneler bu sistemi kullanir..

## Collections / Queue Method'lari

1) **peek()**; ilk elemani silmeden bize retrun eder.

2) **poll()**; ilk elemani queue'dan siler ve bize return eder

3) **offer()**; eleman eklemek icin kullanilir

NOT : remove() ve poll() ilk elemani siler ve return eder. Ama collection'da eleman yoksa remove() methodu Exception atar poll( ) methodu Exception atmaz null return eder.

# Collections / Deque

## Deque

### Double Ended Queue

Queue'larda FIFO gecerli, Deque'lerde hem FIFO hem de LIFO(Last In First Out) gecerlidir.

Deque bir interface'dir dolayisiyla constructor'i yoktur. LinkedList constructor'u kullanilarak deque olusturulabilir.

Deques do not accept Null as an element.

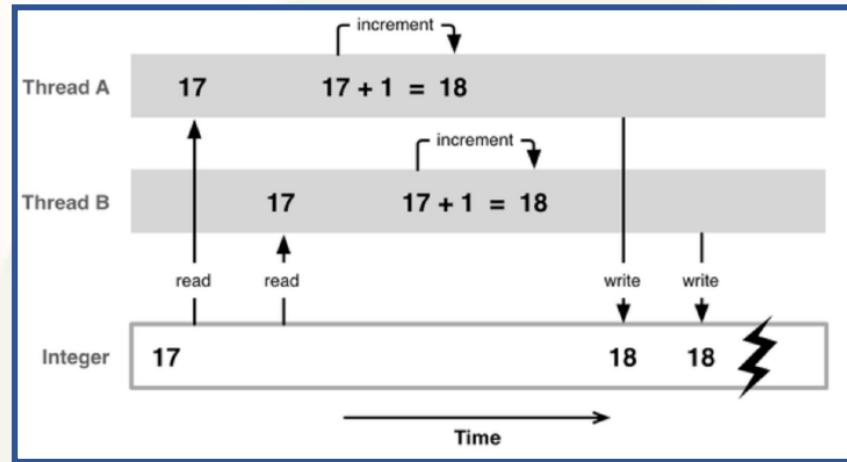
Deque'de ilk ve son eleman onemli oldugu icin ilk ve son elemana ozel bircok method vardir.

getFirst() - getLast()

peekFirst() - peekLast()

pollFirst() - pollLast()

# Multi Thread



Multithreading, CPU'nun maksimum kullanımı için bir programın iki veya daha fazla bölümünün aynı anda yürütülmesine izin veren bir Java özelliğidir.

Böyle bir programın her bir parçası bir iş parçacığı (**thread**) olarak adlandırılır.

Threads iki mekanizma kullanılarak oluşturulabilir:

- 1 ) Thread class'ına extend edilerek
- 2 ) Çalıştırılabilir Interface'in implement edilmesi ile

# Synchronizing

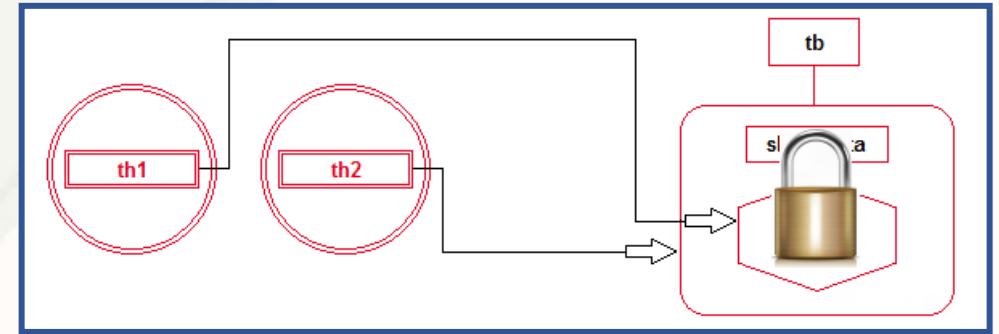
Multi-thread programlar genellikle birden çok thread'in aynı kaynaklara erişmeye çalıştığı ve sonunda hatalı ve öngörülemeyen sonuçlar ürettiği bir duruma gelebilir.

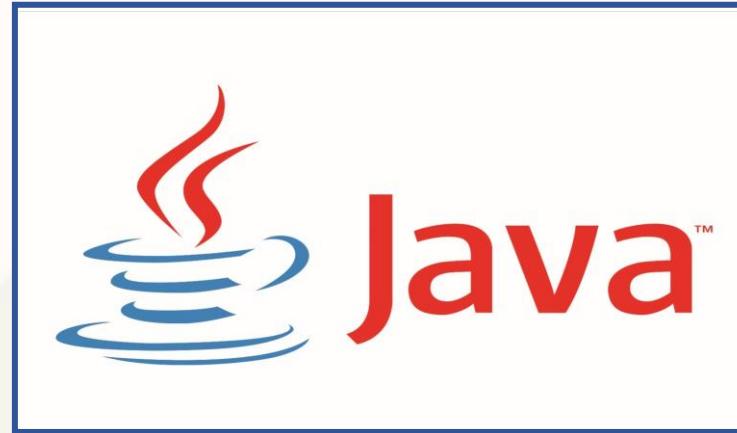
Bu nedenle, belirli bir zaman aralığında kaynağı yalnızca bir thread'in erişebileceğinden emin olunması gereklidir.

Java, senkronize bloklar kullanarak thread oluşturmayı ve threads'in görevlerini senkronize bir şekilde yapmasını sağlar.

Java'da senkronize bloklar, **synchronized** keyword ile işaretlenir.

Java'da senkronize edilmiş blok mekanizması aynı obje üzerinde tek zaman diliminde tek thread çalışmasını sağlar.. Blokda bir thread çalışmaya başlayınca, diğer tüm thread'ler ilk thread'in işlemi bitene kadar bekletilir.





21 ARALIK 2021  
DERS 47

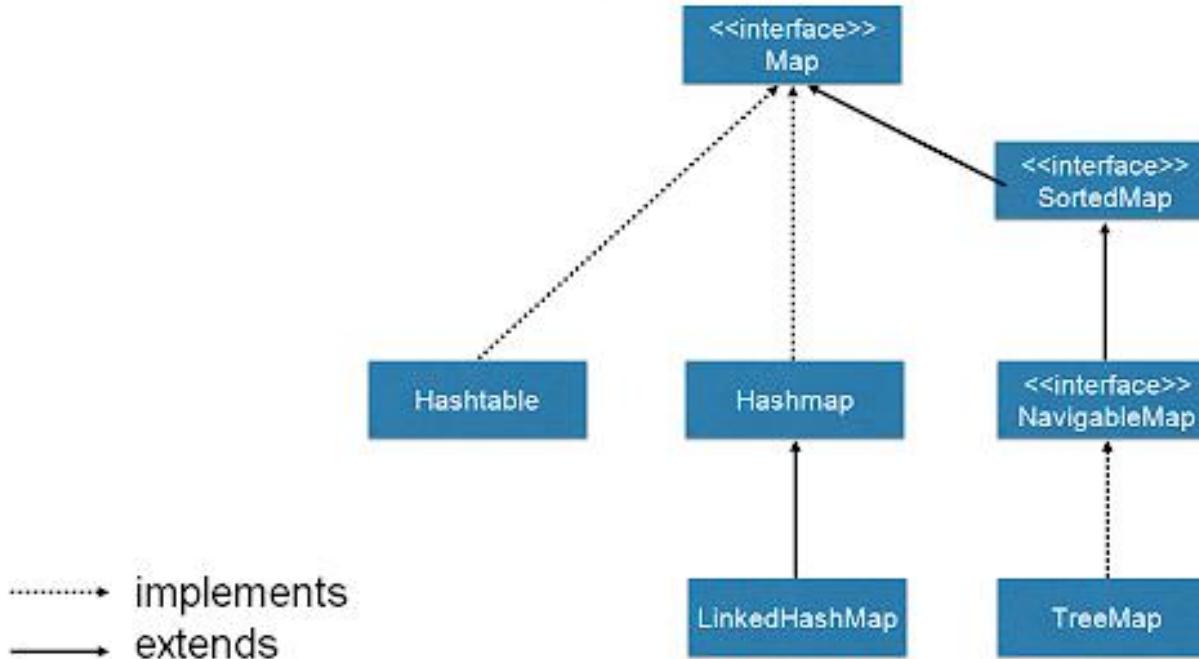
Maps

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# Maps

Maps key – value pairs kullanır. ( anahtar –deger(ler) ). Key'ler unique olmalıdır.

## Map Interface



# Synchronizing & Maps

- 1) HashMap synchronized degildir. Thread-safe degildir
- 2) HashTable synchronized'dir. Thread-safe'dir ve thread'ler tarafından ortak kullanabilir
- 3) TreeMap synchronized degildir. Thread-safe degildir



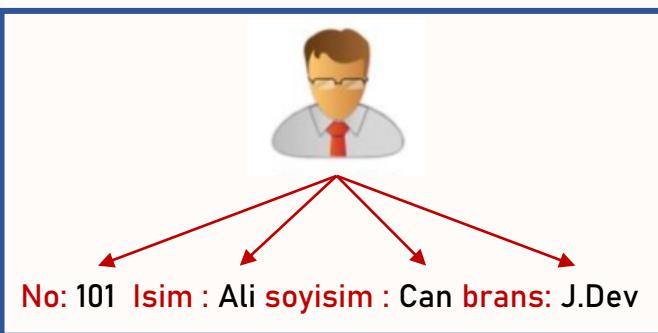
# Maps

Reel projelerde kullanılan database yapısına en uygun Java objesidir.

Maps **key – value pairs** kullanır ( anahtar –değer(ler) ).

Key'ler unique olmalıdır.

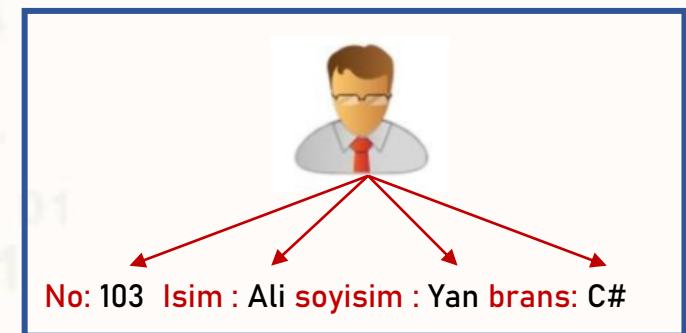
Map ile aynı özelliklere sahip birden fazla objeyi ve özelliklerini store edebilirsiniz.



Ogrenci 1



Ogrenci 2



Ogrenci 3

```
public static void main(String[] args) {  
  
    HashMap<Integer, String> objeMap= new HashMap<>();  
    objeMap.put(101, "Ali, Can, Java");  
    objeMap.put(102, "Veli, Yan, Java");  
    objeMap.put(103, "Ali, Yan, C#");  
  
    System.out.println(objeMap);  
}
```

{101=Ali, Can, Java, 102=Veli, Yan, Java, 103=Ali, Yan, C#}

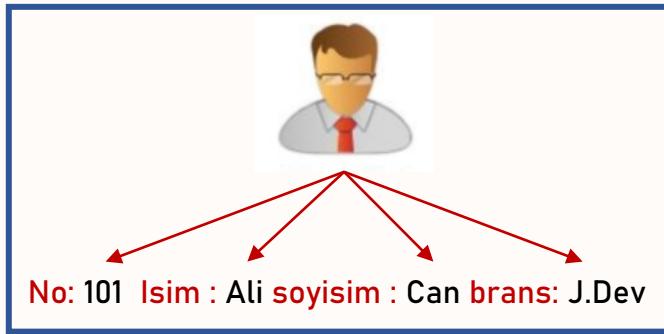
Ogrenci 1

Ogrenci 2

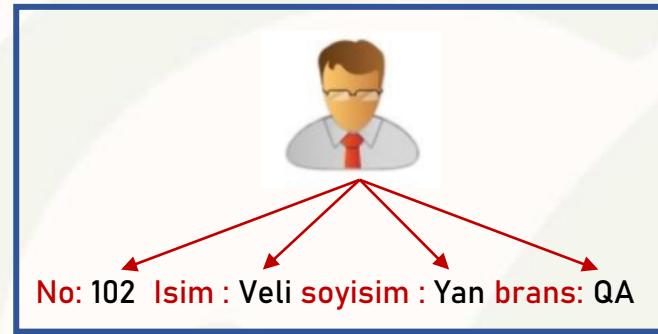
Ogrenci 3

# Maps

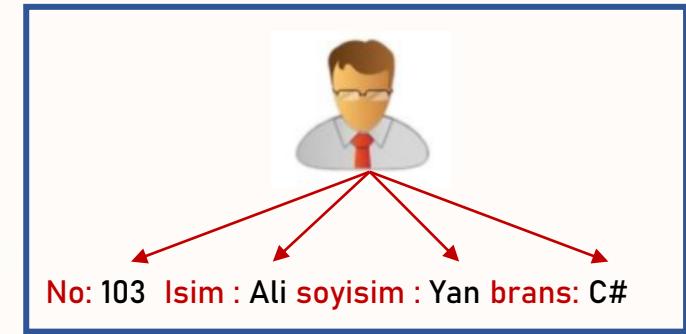
Map ile key ve value bilgilerine ayri ayri ulasabilir, istedigimiz degisiklikleri ayri ayri yapabiliriz



Ogrenci 1



Ogrenci 2



Ogrenci 3

```
System.out.println(objeMap.keySet());
```

[101, 102, 103]

keyset() method'u Set olarak key degerlerini verir.

```
System.out.println(objeMap.values());
```

[Ali, Can, Java, Veli, Yan, Java, Ali, Yan, C#]

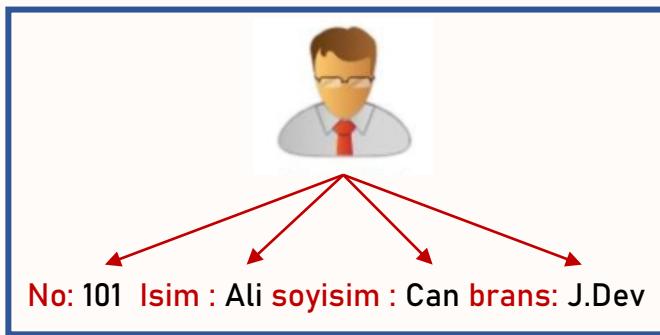
Ogrenci 1

Ogrenci 2

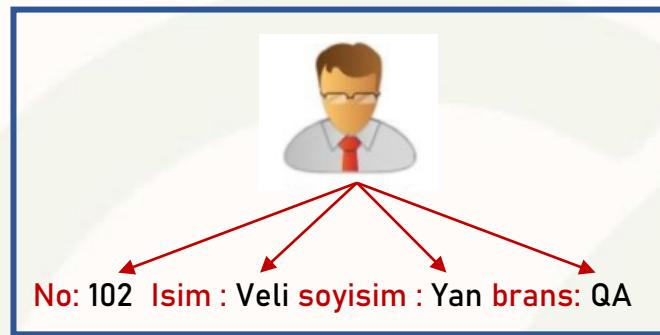
Ogrenci 3

values() method'u Collection olarak "value"lari verir. Collections'dan istedigimiz bir variable'a degerleri ekleyebilir ve kullanabiliriz.

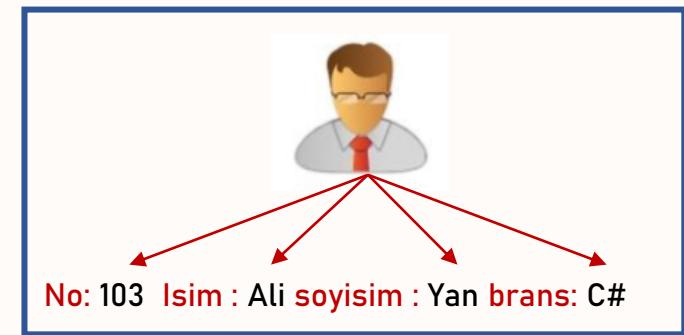
## Maps ( Nested Maps )



Ogrenci 1



Ogrenci 2



Ogrenci 3

Key	Value	
sinifMap= {	101={Isim : Ali, soyisim : Can, brans: J.Dev },	→ Ogrenci 1
	102={Isim : Veli, soyisim : Yan, brans: QA },	→ Ogrenci 2
	103={Isim : Ali, soyisim : Yan, brans: C# }	→ Ogrenci 3
}		

Key	Val.	Key	Val.	Key	Val.	
{Isim : Ali, soyisim : Can, brans: J.Dev }						→ Ogrenci 1

# Maps

Map ile bir objeye ait farklı bilgileri kategorilerine göre ayırip depolayabiliriz



Java Bank



Musteri :  
Ali Can

Tc No:  
Isim :  
Soyisim :  
Telefon:

Kisisel  
bilgiler

Hesap No:  
Kur :  
Acilis Tar. :  
Bakiye:

TL  
Hesap

Hesap No:  
Kur :  
Acilis Tar. :  
Bakiye:

Euro  
Hesap

Hesap No:  
Kur :  
Acilis Tar. :  
Bakiye:

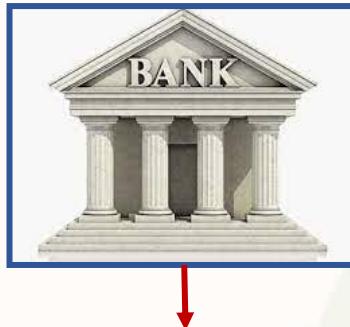
Usd  
Hesap

Kart No:  
Limit :  
Verilis Tar. :  
Kul.Limit :

Kredi  
Karti

# Maps ( Nested Maps )

Java Bank



Map Bank = { M.No1={Musteri Map1}, M.No2={Musteri Map2}, ... }

Musteri :  
Ali Can



Musteri Map1 = {KisiselBilgiler={Kis.Bil.Map} , TlHesap={TlHesapMap}...}

TlHesapMap = {HesapNo="12345" , Kur="TL", AcTar="1.1.2021", Bakiye="0" }

Tc No:  
Isim :  
Soyisim :  
Telefon:

Kisisel  
bilgiler

Hesap No:  
Kur :  
Acilis Tar. :  
Bakiye:

TL  
Hesap

Hesap No:  
Kur :  
Acilis Tar. :  
Bakiye:

Euro  
Hesap

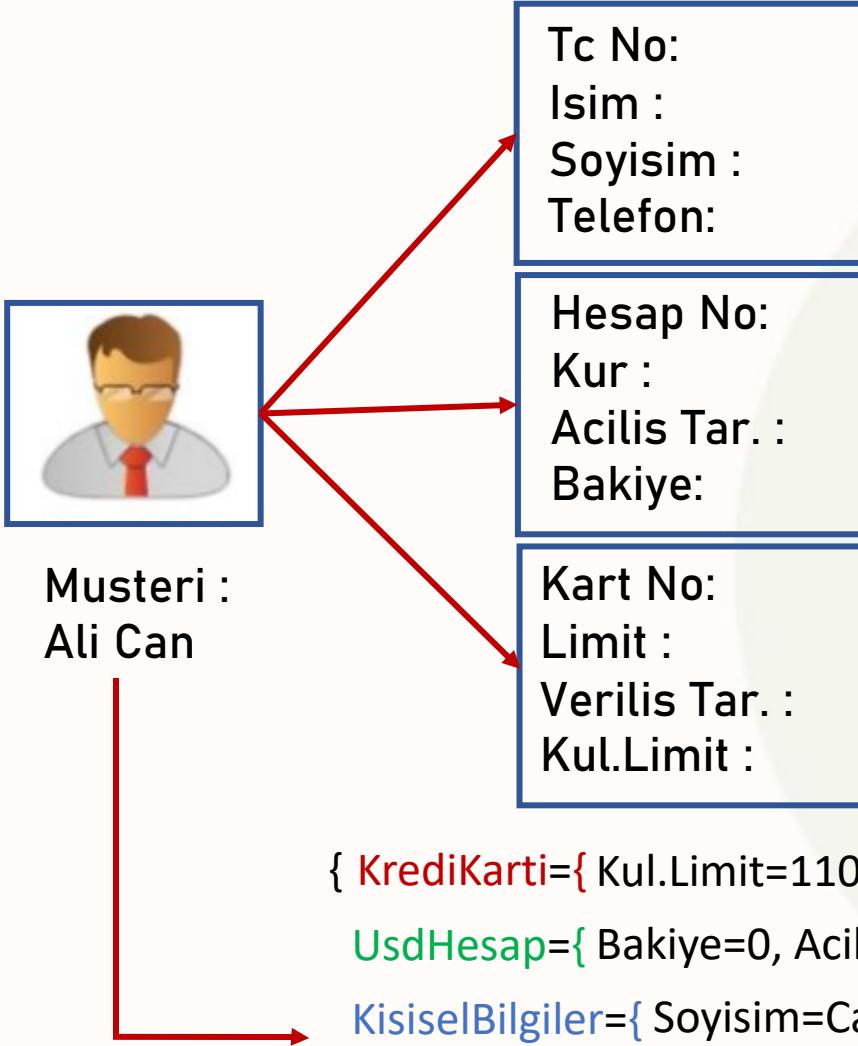
Hesap No:  
Kur :  
Acilis Tar. :  
Bakiye:

Usd  
Hesap

Kart No:  
Limit :  
Verilis Tar. :  
Kul.Limit :

Kredi  
Karti

# Maps



## Kisisel bilgiler

{Soyisim=Can, TcNo=12345678901, Isim=Ali, Telefon=5553456789}

## Hesaplar

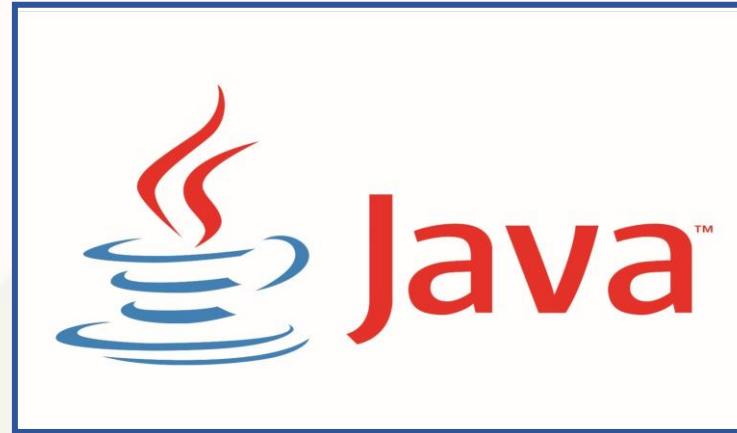
{Bakiye=0, Acilis Tarihi=12.12.2021, Kur=TL, HesapNo=1234-1}

{Bakiye=0, Acilis Tarihi=12.12.2021, Kur=Eur, HesapNo=1234-2}

{Bakiye=0, Acilis Tarihi=12.12.2021, Kur=Usd, HesapNo=1234-3}

## Kredi Karti

{Kul.Limit=11000, Limit=20000, KartNo=1234567890123456,  
Verilis Tarihi=12.12.2021}



22 ARALIK 2021  
DERS 48

Maps

Mehmet BULUTLUOZ  
Elk.Elektronik Yuk.Muh.

# Maps

**Soru 1)** Verilen bir String'deki harfleri ve harflerin kaçar kez kullanıldığını return eden bir method yazınız

Ornek : Input : Hellooo output : H=1, e=1, l=2, o=3

```
public static void main(String[] args) {
    System.out.println(harfSayisiBul("Hellooo")); // {e=1, H=1, l=2, o=3}

}
public static HashMap<String, Integer> harfSayisiBul(String str) {

    HashMap<String, Integer> map = new HashMap<>();
    String arr[] = str.split("");
    System.out.println(Arrays.toString(arr));

    for (String w : arr) {

        if (!map.containsKey(w)) {

            map.put(w, 1);
        } else {
            map.put(w, map.get(w) + 1);
        }
    }
    return map;
}
```

# Maps

Soru 3 ) Verilen map'te istenen programlama dilini bilen kisileri list olarak donduren bir method yaziniz.

map → { 101=Ali, Can, java, 102=Veli, Yan, java, 103=Ali, Yan, C#}

Istenen dil → java

Sonuc → [Ali, Veli]

```
Map<Integer, String> map1 = new HashMap<>();
map1.put(101, "Ali, Can, java");
map1.put(102, "Veli, Yan, java");
map1.put(103, "Ali, Yan, C#");

String istenenDil="JAVA";

List<String> isimList = javaBilenler(map1,istenenDil);
System.out.println(isimList);
}

private static List<String> javaBilenler(Map<Integer, String> map1, String istenenDil) {

    List<String> isimListesi=new ArrayList<>();

    for (String each : map1.values()) {

        String arr[] = each.split(", ");

        if(arr[2].equalsIgnoreCase(istenenDil)) {
            isimListesi.add(arr[0]);
        }
    }
    return isimListesi;
}
```

# Maps

1) **containsKey(key);** istenen key degeri Map'de varsa true, yoksa false doner .

2) **containsValue(value);** istenen key degeri Map'de varsa true, yoksa false doner .

3) **entrySet();** Map'deki entry'leri bir Set olarak verir.

**Entry :** Map'de her bir elemani olusturan key-value ikilisidir

4) **equals(map);** Map'deki tum elemanlari karsilastirir. Hepsi ayni ise true farkli olan varsa false dondurur

# Maps

5) **get(key);** istenen key degeri Map'de varsa o key'e ait value'yu, map'de yoksa null doner.

6) **getOrDefault(key,defaultDeger);** istenen key degeri Map'de varsa o key'e ait value'yu, key map'te yoksa default degeri doner.

7) **putAll(map);** verilen map'deki tum elemanlari bizim map'imize ekler, tekrarlanan eleman varsa uzerine yazar

8) **compute(key, (key,value)->yeniDeger);** verilen map'deki istenen key degerine sahip elemanın value'sunu günceller key map'te yoksa ekler

# Maps

9) **ComputeIfPresent(key, (key,value)-> yeniDeger);** istenen key degeri Map'de varsa o key'e ait value'yu günceller, map'de yoksa birsey yapmaz

10) **ComputeIfAbsent(key, k -> yeniDeger);** istenen key degeri map'de yoksa o key'i ve value'yu ekler, map'de varsa birsey yapmaz

11) **putIfAbsent(key, value);** verilen key map'de yoksa ekler.

12) **size();** map'teki entry sayisini verir

# Maps

**Soru 2 )** Verilen bir listedeki tekrar etmeyen elmanlari veren bir method yaziniz

Ornek : Input : Hellooo output : [H, e]

```
public static List<String> getNonRepeatedChars(String str){

    List<String> list = new ArrayList<>();
    HashMap<String, Integer> map = new HashMap<>();
    String arr[] = str.split("");

    for(String w : arr) {
        map.computeIfPresent(w, (key, value)-> value+1);
        map.computeIfAbsent(w, k->1);
    }
    System.out.println(map);

    for(Entry<String, Integer> w : map.entrySet()) {

        if(w.getValue()==1) {
            list.add(w.getKey());
        }
    }
    return list;
}
```

# Maps

**Soru 4 )** Bir csv file'i okuyup icerigi map'e ceviren bir method yaziniz.

csv → isim,Ali Id,101 Adres,Ankara

map → { isim=Ali, Id=101, Adres=Ankara}

```
String dosyaYolu="C:\\\\Users\\\\lenovo\\\\Desktop\\\\NewCsv.csv";
Map<String, String> map1 = csvdenMapYap(dosyaYolu);
System.out.println("main method map olarak " + map1);
}

private static Map<String, String> csvdenMapYap(String dosyaYolu) {
    List<String> satirListesi = new ArrayList<>();
    Map<String, String> mapCsv=new HashMap<>();

    try {
        BufferedReader br1= new BufferedReader(new FileReader(dosyaYolu));
        String line=br1.readLine();

        while(line != null){
            satirListesi.add(line);
            line=br1.readLine();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    System.out.println("Liste olarak : " + satirListesi);
    for (String each : satirListesi) {
        String[] str= each.split(",");
        mapCsv.put(str[0], str[1]);
    }
    return mapCsv;
}
```

## HashMaps VS HashTable

- HashMap, key olarak sadece 1 tane null, value olarak ise istedigimiz kadar null'a izin verir, HashTable ise null'in kullanilmasina izin vermez
- HashMap thread-safe ve synchronized degildir (dolayisiyla hizlidir). HashTable ise thread-safe ve synchronized'dir(dolayisiyla yavastir).

	Synchronized	Thread Safe	Null Keys And Null Values	Performance	Extends	Legacy
HashMap	No	No	Only one null key and multiple null values	Fast	AbstractMap	No
HashTable	Yes	Yes	No	Slow	Dictionary	Yes

## Maps / TreeMap

- TreeMap, elemanlari natural order'a gore siralar. Siralama icin key'i dikkate alir
- HashMap thread-safe ve synchronized degildir
- Yavastir

1) **ceilingEntry(key);** key map'te varsa entry'yi dondurur, key map'de yoksa olmasi gereken yerden sonraki ilk entry'yi dondurur . En buyuk keyden daha buyuk deger girilirse null doner

2) **descendingKeySet();** key'leri descending order'la dondurur

3) **firstEntry();** ilk Entry'i dondurur

## Maps / TreeMap

4) **floorEntry(key);** girdigimiz key map'te yoksa, key'i girdigimiz sayidan kucuk olan en yakin Entry'yi dondurur

5) **headMap(key);** girdigimiz key exclusive olmak uzere onceki Entry'leri bir map olarak verir

6) **headMap(key,true);** girdigimiz key inclusive olmak uzere onceki Entry'leri bir map olarak verir

7) **tailMap(key);** girdigimiz key inclusive olmak uzere sonraki Entry'leri bir map olarak verir

8) **tailMap(key,false);** girdigimiz key exclusive olmak uzere sonraki Entry'leri bir map olarak verir

The End

EN İYİ KOD ...  
CALISAN KOD'DUR.

ONDAN DA İYİSİ...  
CALISAN VE ANLASILIR KOD YAZMAKTIR.



OMUR BOYU YUZUNUZDEKİ GULUMSEME EKSIK OLMASIN...