
Java
MEGEP
Erkan Kocaman
MUSTAFAKEMALPAŞA
İBRAHİM ÖNAL FEN LİSESİ

1.DEĞİŞKENLER VE SABİTLER

1.1.Değişkenler

Değişkenler bir programlama dilinde verilerin depolanma alanlarını temsil eder. Tanımlanan her değişkene bellek bölgesinden bir alan ayrılır. Bu bellek bölgesine okuma ve yazma işlemleri ise değişken ismi üzerinden sağlanır. Genel olarak değişkenler aşağıdaki şekilde tanımlanır.

<veri tipi><değişken adı>;

Örneğin;
int i;

Yukarıdaki örnekte bir int (sayı) veri tipinde bir değişken tanımlanmıştır. Böylelikle bellek bölgesinde bir veri saklamak ve ileri de kullanmak üzere 4 byte'lık bir alan açmış bulunuyoruz. Program içinde bu bellek bölgesine erişmek için değişken ismi olan i ifadesini<kullanıyoruz.

Örneğin;

int i; //i adında bellekte 4 byte'lık bir bölge aç.
i = 5; //i adının temsil ettiği bellek bölgesine 5 değerini yaz.

- Bir değişkene değer atama işlemi tanımlarken yapılabilir.
- Bir değişkene değer atama işlemi yukardaki örnekte olduğu gibi program içinde herhangi bir satırda yapılabilir.
- Bir veri tipi altında birden fazla isimle farklı değişkenler tanımlanabilir.

Örnek 1.1:

```
boolean dogruMu=true;
double yuzde=98.32,ortalama=35.32;
char karakter='A';
```

Bazı programlama dillerinde yerel bir değişken tanımlıyorsak bu değişkenikullanmadan önce bir değer ataması yapmak zorundayız.

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

   int sayi;
   //sayi = 5;
       System.out.println(sayi);
    }
}
```

Yukarıdaki kod blokunda sayi değişkenine değer atamadığınız için hata verecektir.(Error:(7, 28) java: variable sayi might not have been initialized). Çünkü sayi değişkeni yerel bir değişkendir. Eğer//sayi = 5; açıklama satırındaki // işaretlerini siler ve bu satırı koda dâhil ederseniz hatanın ortadan kalktığını görürsünüz.

Örnek 1.1-2:

```
package tr.meb.fenlisesi;

public class Main {
   public static void main(String[] args) {
      { //Birinci blok
         int a = 10;
      }
      {//İkinci blok
         int a = 20;
      }
      System.out.println(a);
      // a değişkeni bu blokta tanımlanmadığı
      // için program hata verecektir.
   }
}
```

Yukarıdaki örnekte birinci ve ikinci blokta tanımlanan "a" isimli değişkenler sadece kendi bloklarında geçerlidir. Main bloğunda kullanılmaya çalışıldığı zaman program hata verecektir.

1.2. Değişkenleri İsimlendirme Kuralları

• Değişkenlerin isimleri alfabede bulunan karakterlerle veya _(alt çizgi) ile başlamalıdır. Ama ilk harf hariç diğer karakterler sayı olabilir.

• Bazı programlama dilleri büyük ve küçük harf duyarlıdır. Yani Sayi, sayi ve SAYI hepsi ayrı değişken olarak algınalır.

• Değişken isimleri birden fazla kelime olduğu zaman; kelimelerin arasına boşluk konmaz. Bu tür değişkenleri ya kelimeleri birleştirerek veya kelimeler arasına _(alt çizgi) karakteri koyararak isimlendiririz.

• Değişkenlerin isimleri !, ?, {,Unicode karakterlerinden oluşan string] gibi karakterler içeremez.

• Programlama dili için tanımlanmış anahtar kelimelerini de değişken isimleri olarak kullanamayız.

Standart yazım şekilleri:

Camel notasyonunda isim küçük harfle başlar, eğer değişken isminde birden fazlakelime geçiyor ise isimdeki diğer kelimeler büyük harfle başlar.

Camel Notasyonu:

maas;

maasMiktari;

massMiktariAciklama;

Pascal Notasyonunda kelime büyük harfle başlar. Camel Notasyonunda da olduğu gibi diğer kelimelerde büyük harfle başlar.

Pascal Notasyonu:

Maas();

MaasHesapla();

Bu notasyonların kullanımı mecburi değildir. Fakat sürekli olarak bu tür bir notasyona uyarak kodlarınızı yazarsanız, kodlarınız daha anlaşılır bir hâle girer.

1.3. Veri Tipleri

Java da iki çeşit veri tipi vardır:

- Değer tipleri (value type)
- Referans tipleri(reference type)

Değişkenler bellekte bulunan verilerdir. Bir değişkeni kullandığımız zaman o değişkenin bellekte bulunduğu yerdeki bilgiyi kullanırız. Değer tipleri belleğin 'stack' bölgesinde saklanır ve veriyi direkt olarak bellek bölgesinden alırken referans tipleri bellekte 'heap' alanında saklanır. Yani referans tipleri içinde veri değil bellekteki 'heap' alanının adres bilgisini tutarlar.

int, double, float gibi veri tipleri değer tiplerine örnek gösterilebilir. Herhangi bir sınıf türü ise referans tipine örnek gösterilebilir. Değer tipleri birbirine eşitlenirken değişkenin barındırdığı değer bir diğer değişkene kopyalanır. Böylece iki farklı bağımsız değişken oluşur. Referans tipleri ise eşitleme sırasında değişkenlerin taşıdıkları veri değil 'heap' bölgesinde işaret ettikleri adres kopyalanır. Böylece eğer iki referans değişkeni birbirine eşitledi isek ve daha sonra bunlardan birinde bulunan veriyi değiştirdi ise otomatik olarak diğer referans değişkeninin değeri de değişir. Çünkü adresde bulunan veri değişince bu adresi işaret eden iki değişkende yeni veri bilgisine ulaşır.

JAVA programlama dilinin temel tipleri

Temel tip	Boyut	En küçük	En büyük	Sarmalıyıcı sınıf
boolean	-	-	-	Boolean
char	16- bit	Unicode 0	Unicode 2^16-1	Character
byte	8- bit	-128	+127	Byte
short	16- bit	-2^15	+2^15-1	Short
int	32- bit	-2^31	+2^31-1	Integer
long	64- bit	-2^63	+2 ^63-1	Long
float	32- bit	IEEE754	IEEE754	Float
double	64- bit	IEEE754	IEEE754	Double
void	-	-	-	Void

Referans tipleri

Adı	Açıklama
object	Bütün veri türlerinin türediği kök eleman
string	Unicode karakterlerinden oluşan string

Uygulama -1

Kod	Çıktı
public class Main { **public static void** main(String[] args) { **int** a=5; **int** b; b=a; System.***out***.println("**a: "**+a+" **b: "**+b); } }	a: 5 b: 5

İlk önce bellekte a değişkenine belleğin Stack bölgesinde 4 Byte'lik yer ayrılır ve hemen içerisine 5 değeri atanır. Sonrasında b değişkenine yine 4 Byte'lık yer ayrılır ve içine herhangi bir değer atanmaz. *b=a* ifadesinde, a değişkeninin değerinin sahip olduğu RAM bölgesindeki değer, b değişkeni için ayrılan ve henüz boş olan bellek bölgesinin içine değerini yani 5'i kopyalar. Böylece iki bellek alanında da aynı değer olmuş olur.

Uygulama -2

Kod	Çıktı
package tr.meb.fenlisesi; **class** Sayi{ **int i**; } **public class** Main { **public static void** main(String[] args) { Sayi s1=**new** Sayi(); Sayi s2=**new** Sayi(); System.***out***.println("**s1 nesnesi: "**+s1); System.***out***.println("**s2 nesnesi: "**+s2); s1.**i**=5; s2.**i**=10; System.***out***.println("**s1.i:"**+s1.**i**+" **s2.i:"**+s2.**i**); s1=s2; System.***out***.println("**s1.i:"**+s1.**i**+" **s2.i:"**+s2.**i**); System.***out***.println("**s1 nesnesi: "**+s1); System.***out***.println("**s2 nesnesi: "**+s2); } }	s1 nesnesi: Sayi@61bbe9ba s2 nesnesi: Sayi@610455d6 s1.i:5 s2.i:10 s1.i:10 s2.i:10 s1 nesnesi: Sayi@610455d6 s2 nesnesi: Sayi@610455d6

Referans tipinde olan Sayi sınıfımız olsun. Sayi sinifimizin içinde int değer tipinde i alanımız bulunsun. Heap bellek alanında Sayi sınıfımızdan iki tane nesne (referans) oluşturalım. İlk s1 adlı nesnesmizin referansı: Sayi@61bbe9ba iken s2 adlı nesnesimizin referans değeri Sayi@610455d6 dir. s1 adlı nesnesimiz i alanına 5 değeri atanırken s2 adlı nesnesimizin i alanına 10 değeri atanmaktadır. s1= s2 demek; s2 nesnesimizin referansını, s1 nesnesinin referansı ile aynı yapmak anlamına gelir. Sayi@610455d6 adresinde 10 değeri olduğu için hem s1 hem s2 nesnesi 10 değerini gösterir. s1'in eski gösterdiği Sayi nesnesine (Sayi@61bbe9ba) artık ulaşılmaz ve böylece çöp toplayıcısı tarafından bellekten silinir.

1.4. Sabitler

Program boyunca sabit kalacak veriler için kullanılan tanımlamalardır. Bir sabit tanımlamak için final anahtar kelimesini kullanırız. Ayrıca sınıflarda yine bu anahtar kelime ile türetmeyi engellemiş oluruz.Bir global alana, final ve statik özellikler belirtirseniz, bu global alanımız, bu sınıfa ait olan tüm nesneler için tek olur ve değeri sonradan değiştirilemez.

- Sabitler tanımlanırken ilk değer ataması yapılmak zorundadır.
- Program boyunca sabit değeri değiştirelemez.

Örnek 1.4-1:

```
final int X_SABIT_DEGER = 34 ;
final static int Y_SABIT_DEGER = 35 ;
```

Kullanımı: final <veri tipi><değişken adı>=değer;

```
final int X1_SABIT_DEGER = 34 ;//Doğru
final int X2_SABIT_DEGER;      //Hatalı
```

Yukarıdaki tanımlanan sabitlerden birincisi doğru ikincisi ise yanlış bir tanımlamadır.Çünkü sabitler tanımlanırken ilk değer ataması yapılmak zorundadır.

Sabit kullanım hata mesajları :

Error: cannot assign a value to final variable X1_SABIT_DEGER:
Tanımlanan sabit değişkenin değeri değiştirilmeye çalışdığı zamanoluaşacak hata mesajıdır. Bu yüzden sabit değişken tanımlanırken atanan değer program boyunca sabit kalmalıdır.

Örnek 1.4-2:

```
final double PI=3.14 ;
PI=PI*2;//hata Error:(13, 9) java: cannot assign a value to final variable PI
```

Error: variable PI not initialized in the default constructor:

Örnek 1.4-2:

```
final static double PI ;// Burada pi sabitine değer atanmadığı için hata mesajı alırsınız.
```

1.5. Atama İşlemi

= operatörü: Genel atama işlemlerinde kullanılır. Eşitliğin sağındaki değer eşitliğin

solundaki değişkene atanır.

Örnek 1.5-1:

int x, y=5; // 5 değerini y değişkenine atamak için = operatörü kullanılmıştır.

x = y + 2; // y değişken değeri ile 2 sayısı toplanarak x değişkenine atamak için = operatörü kullanılmıştır.

Uygulama 3

Kod	Çıktı
package tr.meb.fenlisesi; **public class** Atama { **public static void** main(String[] args) { **int** x, y=5; x = y + 2; System.***out***.println("**x:** "+x+" **y:** "+y); } }	x: 7 y: 5

+= operatörü: Eşitliğin sağındaki değerle eşitliğin solundaki değişken değerini toplayıp tekrar eşitliğin solundaki değişkene atar.

Örnek 1.5-2:

int x=0, y=0, z=0;

x += 5;//x'e 5 ekle ve x'e eşitle 2.yol x = x + 5 şeklinde de yazılabilir.

y += 7;//y'ye 7 ekle ve y'ye eşitle 2.yol y = y + 7 şeklinde de yazılabilir.

z += x;//z'ye x'i ekle ve z'ye eşitle 2.yol z = z + x şeklinde de yazılabilir.

İşlem sonucu: x=5, y=7, z=5 olur.

Not : Bir bir artırma işlemi için x+=1 (veya x=x+1) yerine x++ işlemi kullanılabilir.

Kod	Çıktı
package tr.meb.fenlisesi; **public class** Atama { **public static void** main(String[] args) { **int** x=0, y=0, z=0; x += 5; *//x=x+5* y += 7; *//y=y+7* z += x; *//z=z+x* System.***out***.println("**x:** "+x+" **y:** "+y+" **z:** "+z); } }	x: 5 y: 7 z: 5

Örnek 1.5-3:

int x=0, y=0,toplam;

x++;//x'i bir artır

y++;//y'yi bir artır

toplam = x + y;//x ve y'yi toplayarak toplam değişkenine ata.

İşlem sonucu: x=1, y=1, toplam=2 olur.

Kod	Çıktı
package tr.meb.fenlisesi; **public class** Atama { **public static void** main(String[] args) { **int** x=0, y=0,toplam; x++;*//x'i bir artır* y++;*//y'yi bir artır* *//x ve y'yi toplayarak toplam değişkenine ata.* toplam = x + y; System.***out***.println("**x :**"+x+" **y:**"+y+" **toplam:** "+toplam); } }	x :1 y:1 toplam: 2

- **++ değişkenden sonra kullanılırsa önce atama işlemi yapılır sonra artırma yapılır.**

Örnek 1.5-4:
int x=0, y=0,toplam;
x=y++;
toplam = x + y;
önce x y'ye eşitlenir, daha sonra y artırılır. İşlem sonucu: x=0, y=1, toplam=1 olur.

Kod	Çıktı
package tr.meb.fenlisesi; **public class** Atama { **public static void** main(String[] args) { **int** x=0, y=0,toplam; x=y++; toplam = x + y; System.***out***.println("**x :**"+x+" **y:**"+y+" **toplam:** "+toplam); } } 	x :0 y:1 toplam: 1

- **++ değişkenden önce kullanılırsa önce artırım yapılır daha sonra atama işlemi yapılır.**

Örnek 1.5-5:
int x=0, y=0,toplam;
x=++y;
toplam = x + y;
önce y artırılır daha sonra x y'ye eşitlenir. İşlem sonucu: x=1, y=1, toplam=2 olur.

Kod	Çıktı
package tr.meb.fenlisesi; **public class** Atama { **public static void** main(String[] args) { **int** x=0, y=0,toplam; x=++y; toplam = x + y; System.***out***.println("**x :**"+x+" **y:**"+y+" **toplam:** "+toplam); } } 	x :1 y:1 toplam: 2

Örnek 1.5-6:
int x=50, y=50, z=100;
x -= 5;//x'den 5'i çıkar ve x'e eşitle 2.yol x = x - 5 şeklinde de yazılabilir.
y -= 7;//y'den 7 yi çıkar ve y'ye eşitle 2.yol y = y - 7 şeklinde de yazılabilir.
z -= x;//z'den x'i çıkar ve z'ye eşitle 2.yol z = z - x şeklinde de yazılabilir.
İşlem sonucu: x=45 , y=43 , z=55 olur.

Kod	Çıktı
package tr.meb.fenlisesi; public class Main { public static void main(String[] args) { int x=50, y=50, z=100; x -= 5; y -= 7; z -= x; System.out.println("x :"+x+" y:"+y+" z: "+z); } }	x :45 y:43 z: 55

Örnek 1.5-7:
int x=20, y=10,fark;
x--;//x'i bir azalt
y--;//y'yi bir azalt
fark = x - y;//x ve y'yi çıkararak fark değişkenine ata.
İşlem sonucu: x=19 , y=9 , fark=10 olur.

Kod	Çıktı
package tr.meb.fenlisesi; public class Main { public static void main(String[] args) { int x=20, y=10,fark; x--;//x'i bir azalt y--;//y'yi bir azalt fark = x - y;//x ve y'yi çıkararak fark değişkenine ata. System.out.println("x :"+x+" y:"+y+" fark: "+fark); } }	x :19 y:9 fark: 10

- **-- değişkenden sonra kullanılırsa önce atama işlemi yapılır, sonra azaltma yapılır.**

Örnek 1.5-8:
int x=10, y=10,fark;
x=y--;
fark = x - y;
önce x y'ye eşitlenir, daha sonra y azalttırılır. İşlem sonucu: x=10, y=9 , fark=1 olur.

Kod	Çıktı
package tr.meb.fenlisesi; public class Main { public static void main(String[] args) { int x=10, y=10,fark; x=y--; fark = x - y; System.out.println("x :"+x+" y:"+y+" fark: "+fark); } }	x :10 y:9 fark: 1

- **-- değişkenden önce kullanılırsa önce azaltma yapılır daha sonra atama işlemi yapılır.**

Örnek 1.5-9:
int x = 10, y = 10, fark;
x = --y;
fark = x - y;
önce y artırılır daha sonra x y'ye eşitlenir. İşlem sonucu : x=9 , y=9 , fark=0 olur.

Kod	Çıktı
package tr.meb.fenlisesi; public class Main { public static void main(String[] args) { int x = 10, y = 10, fark; x = --y; fark = x - y; System.out.println("x :"+x+" y:"+y+" fark: "+fark); } }	x :9 y:9 fark: 0

- ***= operatörü: Eşitliğin sağındaki değerle eşitliğin solundaki değişken değeri çarpılıp tekrar eşitliğin solundaki değişkene atar.**

Örnek 1.5-10:
int x = 2, y = 3, z = 2;
x *= 2;//x ile 2'i çarp ve x'e eşitle 2.yol x = x * 2 şeklinde de yazılabilir.
y *= 2;//y ile 2 yi çarp ve y'ye eşitle 2.yol y = y * 2 şeklinde de yazılabilir.
z *= x;//z ile x'i çarp ve z'ye eşitle 2.yol z = z * x şeklinde de yazılabilir.
İşlem sonucu: x=4 , y=6 , z=8 olur.

Kod	Çıktı
package tr.meb.fenlisesi; public class Main { public static void main(String[] args) { int x = 2, y = 3, z = 2; x *= 2; y *= 2; z *= x; System.out.println("x :"+x+" y: "+y+" z: "+z); } }	x :4 y: 6 z: 8

- **/= operatörü: Eşitliğin solundaki değişken değerini eşitliğin sağındaki değere bölerek tekrar eşitliğin solundaki değişkene atar.**

Örnek 1.5-11:
int x = 4, y = 10, z = 64;
x /= 2;//x'i 2'ye böl ve x'e eşitle 2.yol x = x / 2 şeklinde de yazılabilir.
y /= 2;//y'yi 2'ye böl ve y'ye eşitle 2.yol y = y / 2 şeklinde de yazılabilir.
z /= x;//z'yi x'e böl ve z'ye eşitle 2.yol z = z / x şeklinde de yazılabilir.
İşlem sonucu: x=2, y=5, z=32 olur.

Kod	Çıktı
package tr.meb.fenlisesi; public class Main { public static void main(String[] args) { int x = 4, y = 10, z = 64; x /= 2; y /= 2; z /= x; System.out.println("x :"+x+" y: "+y+" z: "+z); } }	x :2 y: 5 z: 32

1.6 Çıkış İşlemleri

Bir metin ifadesini ekrana yazdırmak için aşağıdaki metot kullanılır:

System.out.println():Yazdırma işleminden sonra imleç yazdırılan ifadenin alt satırında bekler.

Örnek 1.6-3:

Kod	Çıktı
public static void main(String[] args) { System.out.println("Merhaba Dünya"); }	Merhaba Dünya

Örnek 1.6-4:

Kod	Çıktı
public static void main(String[] args) { System.out.println("Merhaba Dünya"); System.out.println("Günaydın"); }	Merhaba Dünya Günaydın

Programı çalıştırdığımızda ikinci metin bir alt satıra yazılmıştır.
Çıkış parametreleri:
\n: Bir alt satıra geçmek için kullanılır.
\r: Paragraf başı yapmak için kullanılır.

Kod	Çıktı
public static void main(String[] args) { System.out.println("\n\nMerhaba Dünya"); }	Merhaba Dünya

Ekran çıktısına baktığımızda 2 tane \n kullandığımız için yazımız 2 satır aşağıdan başlamıştır.

1.6.2. İlk Değer Atanan Değişken Değerini Ekrana Yazdırma

Değişken tanımlamayı daha önce görmüştük. Şimdi değer atadığımız değişkeni ekrana yazdırma işlemlerini göreceğiz. Değişkene ilk değeri tanımlarken veya tanımladıktan sonra atayabiliriz.

int x = 5; // ilk değeri tanımlanırken atadık.

int y;
y = 3;

// ilk derğeri tanımlandıktan sonra atadık.
Bu değişkenleri ekrana nasıl yazdırabileceğimize bakalım.

Kod	Çıktı
public static void main(String[] args) { int x = 5; // ilk değeri tanımlanırken atadık. int y; y = 3;// ilk derğeri tanımlandıktan sonra atadık. System.out.println("x = " + x); System.out.println("y = " + y); }	x = 5 y = 3

Örnek 1.6-7:

Kod	Çıktı
int x = 5; // ilk değeri tanımlanırken atadık. int y; y = 3;// ilk derğeri tanımlandıktan sonra atadık. System.out.printf("x değişkeninin değeri = %d", x);	x değişkeninin değeri = 5

%d anlamı: Onluk sayı demektir. Değişkenler virgülle ayrılarak tanımlanır.

Kod
public static void main(String[] args) { int x = 5; // ilk değeri tanımlanırken atadık. int y; y = 3;// ilk derğeri tanımlandıktan sonra atadık. System.out.printf("x değişkeninin değeri = %d y değişkeninin değeri = %d", x,y); }

Çıktı
x değişkeninin değeri = 5 y değişkeninin değeri = 3

Değişken değerlerini ekrana birlikte yazdırmanın bir diğer yoluda + işaretini kullanmaktır. + işareti ifadeleri birleştirme işlemini gerçekleştirir.

Kod
public static void main(String[] args) { int x = 5; // ilk değeri tanımlanırken atadık. int y; y = 3;// ilk derğeri tanımlandıktan sonra atadık. System.out.println("x değişkeninin değeri :"+x+" y değişkeninin değeri :"+y); }

Çıktı
x değişkeninin değeri = 5 y değişkeninin değeri = 3

Örnek 1.6

Kod	Çıktı
public static void main(String[] args) { String ad = "İbrahim";//İlk değer tanımlanırken atandı. String soyad; soyad = "Önal"; //İlk değeri tanımladıktan sonra atadık. System.out.println(ad+" "+soyad); }	İbrahim Önal

Örnek

Kod	Çıktı
public static void main(String[] args) { int x = 5; // ilk değeri tanımlanırken atadık. int y; y = 3;// ilk derğeri tanımlandıktan sonra atadık. System.out.println("x + y toplamı ="+x+y); }	x + y toplamı =53

Yapmak istediğimiz işlem aslında x ve y değişken değerlerini toplayıp ekrana yazdırmaktı. Fakat + işaretinin buradaki görevi ifadeleri birleştirmek olduğundan x ve y değişken değerlerini yan yana yazarak yanlış sonuç üretmiştir.

Örnek

Kod	Çıktı
public static void main(String[] args) { int x = 5; // ilk değeri tanımlanırken atadık. int y; y = 3;// ilk derğeri tanımlandıktan sonra atadık. System.out.println("x + y toplamı ="+(x+y)); }	x + y toplamı =8

System.out.println("x + y toplamı ="+(x+y)); satırında x + y işlemini parantez içine alınca doğru sonuç elde edilmiştir.
System.out.printf("x + y toplamı = %d",x+y); şeklinde de yazsaydık aynı sonucu elde ederdik.

1.6.3. Formatlı Çıkış İşlemleri

Tam sayı tipinde tanımlanmış değişkenler üzerinde uygulanabilecek format biçimleri aşağıdaki tablola belirtilmiştir.(https://docs.oracle.com/javase/tutorial/java/data/numberformat.html)

Karakter	**Bayrak**	**Açıklama**
d		Onluk tamsayı
f		Ondalıklı bir sayı
n		Yeni satır ekleme karakteri. Her zaman \n kullanmaktansa %n kullanın.
tB		Tarih zaman çevrimi—Yerel bölgeye göre ayın tam adı
td, te		Tarih zaman çevrimi— ay günü iki basamaklı sayı. **td** gerekli olduğu durumlarda başa 0 ekler, **te** eklemez.
ty, tY		Tarih zaman çevrimi—ty = 2 haneli yıl, tY = 4 haneli yıl.
tl		Tarih zaman çevrimi—12li saat dilimi.
tM		Tarih zaman çevrimi—minutes in 2 digits, with leading zeroes as necessary. Gerekli olduğunda 0 ile başlayan 2 haneli dakika
tp		Tarih zaman çevrimi—yerel gösterim am/pm (küçük harf).
tm		Tarih zaman çevrimi—minutes in 2 digits, with leading zeroes as necessary. Gerekli olduğunda 0 ile başlayan 2 haneli ay.
tD		Tarih zaman çevrimi— %tm%td%ty olarak tarih.
	08	Gerekli olduğunda 0 ile başlayan 8 karakter genişliğinde
	+	Includes sign, whether positive or negative. İster pozitif ister de negatif olsun işaret içerir.
	,	Yöreye özgü gruplanmış karakterleri içerir.
	-	Sola-Dayalı..
	.3	Ondalıktan sonra 3 hane.
	10.3	Sağa dayalı ve ondalık ayracın sonra 3 haneyle birlikte toplam 10 hane

Yukarıdaki özellikleri aşağıdaki örnekte inceleyebiliriz.

Kod	Çıktı
long n = 461012; System.out.format("%d%n", n); System.out.format("%08d%n", n); System.out.format("%+8d%n", n); System.out.format("%,8d%n", n); System.out.format("%+,8d%n%n", n); double pi = Math.PI;	 461012 00461012 +461012 461,012 +461,012

System.out.format("%f%n", pi); System.out.format("%.3f%n", pi); System.out.format("%10.3f%n", pi); System.out.format("%-10.3f%n", pi); System.out.format(Locale.FRANCE,"%-10.4f%n%n", pi); Calendar c = Calendar.getInstance(); System.out.format("%tB %te, %tY%n", c, c, c); System.out.format("%tl:%tM %tp%n", c, c, c); System.out.format("%tD%n", c);	3.141593 3.142 3.142 3.142 3,1416 Kasım 5, 20188:37 am 11/05/18

Örnek

Kod	Çıktı
long phoneFmt = 2123552154L; DecimalFormat phoneDecimalFmt = new DecimalFormat("0000000000"); String phoneRawString= phoneDecimalFmt.format(phoneFmt); java.text.MessageFormat phoneMsgFmt=new java.text.MessageFormat("({0}) {1} {2}"); String[] phoneNumArr={phoneRawString.substring(0, 3), phoneRawString.substring(3,6),phoneRawString.substring(6)}; System.out.println(phoneMsgFmt.format(phoneNumArr));	(212) 355 2154

1.7 Giriş İşlemleri

1.7.1. Klavyeden Değişkene Değer Atama :

nesne.nextLine(): Scanner sınıfından bir nesne oluşturuyoruz. Oluşturduğumuz nesnenin nextLine() fonksiyonu ile String türünde değer alabiliriz. Bu metot ile kullanıcının klavyeden bir değer girmesini sağlar ve bu değeri metin(string) bir ifade olarak geri döndürür .

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { // java Scanner kullanımı ve kullanıcıdan veri alma Scanner scan=new Scanner(System.in); System.out.println("Veri giriniz:"); String veri=scan.nextLine(); System.out.println("Girmiş olduğunuz veri :"+veri); } }	Veri giriniz: fen lisesi Girmiş olduğunuz veri :fen lisesi

Örnek

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { int a,b; Scanner scan=new Scanner(System.in); System.out.println("1. Sayıyı Giriniz:"); a=scan.nextInt(); System.out.println("2. Sayıyı Giriniz:"); b=scan.nextInt(); System.out.printf("%d + %d = %d", a,b,(a+b)); } }	1. Sayıyı Giriniz: 3 2. Sayıyı Giriniz: 3 3 + 3 = 6

1.8 Giriş-Çıkış İşlemleri Hata Mesajları

Hazırlayacağımız programın en önemli özelliklerinden biri de stabil çalışması olmalıdır. Stabil çalışması programımızın hatalara karşı ne kadar hazırlıklı ve kullanıcıya verdiği geri dönütle eş değerdir. Programımızın çalışması sırasında oluşabilecek hatalar genellikle kullanıcı girişlerinden kaynaklanır. Bu yüzden kullanıcı girişlerini kontol altına alarak çalışma zamanında oluşabilecek hataları en aza indirmek ise biz programcıların görevidir. Önce hatalar oluştuğunda programın nasıl sonlandığını görelim daha sonra bunun için bir çözüm arayalım.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { int a; Scanner scan=new Scanner(System.in); System.out.println("Bir sayı giriniz"); a=scan.nextInt(); } }	Bir sayı giriniz deneme Exception in thread "main" java.util.InputMismatchException at java.util.Scanner.throwFor(Scanner.java:909) at java.util.Scanner.next(Scanner.java:1530) at java.util.Scanner.nextInt(Scanner.java:2160) at java.util.Scanner.nextInt(Scanner.java:2119) at com.company.Main.main(Main.java:12)

Programda istenilen sayı yerine ‘string’ bir ifade girdiğimiz zaman ekran çıktısı yukarıdaki çıktıda gibi olacaktır. ‘String’ ifade ‘int’ tipine çevrilmede zorlanılacağından program duracaktır. Giriş dizesinin doğru olmadığına dair bir hata verecektir.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { int a,b; Scanner scan=new Scanner(System.in); System.out.println("Birinci sayıyı giriniz"); a=scan.nextInt(); System.out.println("İkinci sayıyı giriniz"); b=scan.nextInt(); System.out.printf("%d / %d = %d", a,b,(a/b)); } }	Birinci sayıyı giriniz 6 İkinci sayıyı giriniz 0 Exception in thread "main" java.lang.ArithmeticException: / by zero at com.company.Main.main(Main.java:15)

Programımızı çalıştıralım ve yukarıdaki gibi değerleri girelim. Girilen sayılardan biri sıfır olduğu zaman programımız sıfıra bölme hatasıyla karşılaşacağından dolayı duracaktır . Yukardaki kodlarda da görüldüğü gibi program esnasında kullanıcılar tarafından yapılan hatalı girişler programın hatayla karşılaşmasına ve durmasına sebep olmaktadır. Biz bu hataları program içinde nasıl yakalarız ve kullanıcıya hata hakkında nasıl mesaj veririz buna bakalım.
İstisnaları program esnasında yakalamak ve kullanıcıya hata mesajını vermek için try{} catch{} finally{} bloklarını kullanıyoruz.

try{} bloku: İstisnanın çıkması muhtemel kodların yazıldığı bloktur.

catch{} bloku: Oluşan istisnanın yakalandığı ve kullanıcıya sunulduğu bloktur.

finally{} bloku: Try bloku içinde hata olsa da olmasa da çalışmasını istediğimiz kodların yazıldığı bloktur. Finally bloğu genellikle bazı kaynakları serbest bırakmak için kullanılır. Kullanılması isteğe bağlı bir bloktur. En sık kullanıldığı yerler açık olan veri tabanı bağlantılarının program kırılsa da kırılmasa da kapatılması durumlarıdır.

Programımızı çalıştırıp aşağıdaki gibi değerleri girdiğimiz zaman herhangi bir hatayla karşılaşılmadığından istediğimiz sonucu üretecektir.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { int a,b; Scanner scan=new Scanner(System.in); System.out.println("Birinci sayıyı giriniz"); a=scan.nextInt(); System.out.println("İkinci sayıyı giriniz");	Birinci sayıyı giriniz 6 İkinci sayıyı giriniz 0 Hata Oluştu :java.lang.ArithmeticException: / by zeroİyi Günler

```
      b=scan.nextInt();
      try
      {
        System.out.printf("%d / %d = %d", a, b, a / b);
      }
      catch (Exception e)
      {
         System.out.printf("Hata Oluştu :"+ e);
      }
      finally
      {
         System.out.printf("İyi Günler");
      }
   }
}
```

Fakat programımıza yukarıdaki gibi değerleri girdiğimiz zaman hatayla karşılaşılacak ve programımız olduğu yerde durup catch{} blokuna atlayacak ve buradan çalışmaya devam edecektir. Dikkat ederseniz her iki durumda da finally bloku içine yazdığmız kodlar çalıştırılmaktadır.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { Scanner scan=new Scanner(System.in); byte a ; try { System.out.print("0-255 Arasında Bir Sayı Giriniz: "); a=scan.nextByte(); System.out.println("Doğru Değer Girdiniz"); } catch (Exception e) { System.out.println("Yanlış Değer Girdiniz"); System.out.println("Hata Oluştu :"+ e); } finally { System.out.println("İyi Günler"); } } }	----- doğru veri giriş sonucu----- 0-255 Arasında Bir Sayı Giriniz: 34 Doğru Değer Girdiniz İyi Günler ------- hatalı veri giriş sonucu------ 0-255 Arasında Bir Sayı Giriniz: 300 Yanlış Değer Girdiniz Hata Oluştu :java.util.InputMismatchException: Value out of range. Value:"300" Radix:10 İyi Günler

Eğer bizden istenildiği gibi 0-255 arası bir değer girersek programımız herhengi bir hatayla karşılaşmayacağı için try bloku içindeki tüm kodlar icra edilip finally bloğuna atlar ve çalışmasına ordan devam eder. Yukardaki ekran çıktısında görüldüğü gibi Fakat istenilen değer dışında bir değer girdiğimiz zaman program hatayla karşılaşacağından hatanın oluştuğu satırda program durdurulur

ve catch blokuna atlanır ve çalışmasına ordan devam eder. Yukardaki ekran çıktısında görüldüğü üzere hata a=scan.nextByte(); satırında oluştuğundan bir sonraki satır icra edilmeden catch blokuna atlanmış ve program buradan devam etmiştir.

1.9 Açıklama Satırları

Açıklama satırları programcıya kod içinde tanımlama metinleri yazma imkânı sağlar. Bu sayede kod parçacıklarının ne iş yaptıkları anlatılmış olur. Kodlarımız arasına açıklama satırları eklemek oldukça önemlidir. Az satırlı program kodlarında birşey ifade etmeyebilir fakat büyük programlarda kod bloklarının ne işe yaradıkları yazılarak programcının ileride karşılaşacağı problemleri kolay çözmesinde yardımcı olacaktır. Ayrıca açıklama satırları program derlenirken dosya içerisine alınmadığından oluşan dosyanın boyutunu ya da çalışmasını etkilememektedir.

```
public static void main(String[] args) {
      // Bu satır "İbrahim Önal Fen Lisesi" yazdırır
         System.out.print("İbrahim Önal Fen Lisesi ");
        /*
        * Bu satır "İbrahim Önal Fen Lisesi" yazdırır
        * Bu kısım derleyici tarafından yok sayılır.
        */   }
```

2. OPERATÖRLER

Programlama dillerinde tanımlanmış sabit ve değişkenler üzerinde işlemler yapmamızı sağlayan karakter ya da karakter topluluklarına operatör denir.
Örneğin;

```
int sayi = 2 + 3;
```

Yukardaki örnekte + ve = karakterleri birer operatördür. + karakteri 2 ve 3 sabitlerini toplama yapıyor ve = karakteri ise toplanan değeri tanımlanan değişkene atama işlemini gerçekleştiriyor.

2.1. Aritmetiksel Operatörler

Aritmetik işlemler yaparken kullandığımız operatörlerdir.

2.1.1. Dört İşlem

Operatör	Açıklama
+	Toplama İşlemi
-	Çıkarma İşlemi
*	Çarpma İşlemi
/	Bölme İşlemi

Kod	Çıktı
public class Main { public static void main(String[] args) { int x=10; int y=5; System.out.printf("%d + %d = %d %n", x,y,(x+y)); System.out.printf("%d - %d = %d %n", x,y,(x-y)); System.out.printf("%d * %d = %d %n", x,y,(x*y)); System.out.printf("%d / %d = %d %n", x,y,(x/y)); } }	10 + 5 = 15 10 - 5 = 5 10 * 5 = 50 10 / 5 = 2

Yukarıdaki kodda %n karakteri bir alt satıra geçmek için kullanılmıştır.

Kod	Çıktı
public class Main { public static void main(String[] args) { int x=10; int y=4;	10 / 4 = 2 10 / 4 = 2,500000 10 / 4 = 2,500000

```
        int intSonuc = x / y;
        float floatSonuc =(float) x / y;
        double doubleSonuc =(double) x / y;

        System.out.printf("%d / %d = %d %n", x,y,intSonuc);
        System.out.printf("%d / %d = %f %n", x,y,floatSonuc);
        System.out.printf("%d / %d = %f %n", x,y,doubleSonuc);
    }
}
```

Başka bir örnekte tip dönüştürmeyi görelim.

Kod	Çıktı
public class Main { public static void main(String[] args) { int x=10; int y=4; int sonuc; float floatSonuc; sonuc = x / y; System.out.printf("%d / %d = %d %n", x,y,sonuc); floatSonuc = (float)x / y; System.out.printf("%d / %d = %f %n", x,y,floatSonuc); floatSonuc = x / (float)y; System.out.printf("%d / %d = %f %n", x,y,floatSonuc); floatSonuc = (float)x / (float)y; System.out.printf("%d / %d = %f %n", x,y,floatSonuc); floatSonuc = (float)(x / y); System.out.printf("%d / %d = %f %n", x,y,floatSonuc); } }	10 / 4 = 2 10 / 4 = 2,500000 10 / 4 = 2,500000 10 / 4 = 2,500000 10 / 4 = 2,000000

Birinci çıktı: sonuc = x / y = 10/4 bir tam sayı bölme işlemi olduğundan bölümün tam sayı kısmı alınmıştır.
İkinci çıktı: floatSonuc = (float)x / y = (float)10 / 4 deyiminde, önce 10 sayısı float tipine dönüştürülüyor, sonra 4 sayısına bölünüyor. Bir float tipin bir tam sayıya bölümü yine float tipindendir. Dolayısıyla, (float)10 / 4 = 2.500000'tir.
Üçüncü çıktı: Bu çıktı ikinci çıktının simetriğidir. floatSonuc = x / (float)y = 10 / (float)4 deyiminde, önce 4 sayısı float tipine dönüştürülüyor, sonra 10 tam sayısı 4.0 sayısına bölünüyor. Bir tam sayı tipin bir float tipine bölümü yine float tipindendir. Dolayısıyla, 10 / (float)4 = 2.500000'tir.
Dördüncü çıktı: İkinci ve üçüncü çıktının birleşimidir. floatSonuc = (float)x / (float)y = (float)10 / (float)4 deyiminde, önce 10 ve 4 sayılarının her ikisi de float tipine donüştürülür. Sonra iki float tipin birbirine bölümü yapılır. Bu işlemin sonucu, doğal olarak bir float tipidir. Dolayısıyla, (float)10 / (float)4 = 2.500000'tir.
Beşinci çıktı: floatSonuc = (float)(x / y) = (float)(10/4) deyiminde, önce (10 / 4) bölme işlemi yapılır. Bu bir tam sayı bölme işlemi oldugu için birinci çıktıda oldugu gibi çıkan sonuç 2 dir. (float) 2 = 2.0000000 olduğundan çıktı 2'dir.

2.1.2. Mod Alma

Bir sayının başka bir sayıya bölümünden kalan sonucu alma işlemine mod alma denir. Bu işlemi yapmak için % karakteri kullanılır.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { Scanner scan=new Scanner(System.in); int x,y; System.out.print("Birinci değeri giriniz "); x=scan.nextInt(); System.out.print("İkinci değeri giriniz "); y=scan.nextInt(); System.out.printf("%d MOD %d = %d",x,y,x%y); } }	Birinci değeri giriniz 9 İkinci değeri giriniz 3 9 MOD 3 = 0

2.1.2. İlişkisel Operatörler

İlişkisel operatörler iki değerin karşılaştırılması işlemi için kullanılır. Programımızda koşul ifadelerinde kullanılarak programın akışını değiştirmemizi sağlar. Karşılaştırma sonucunda doğru(true) ve yanlış(false) olmak üzere boolean bir değer döndürür.

Örneğin, günlük hayattan bir örnek verelim.

Şema 2.1: İki değerin karşılaştırılması

Hava yağmurlu mu sorusu bizim için bir şart ve bu şart içinde havanın durumunu yağmurla karşılaştırıyoruz. Bu soruya dışarıda yağmur yağıyorsa evet(true) yağmıyorsa hayır(false) şeklinde boolean bir cevap veriyoruz. Dolayısıyla verdiğimiz cevaba göre de programımızın akışı yön değiştirmektedir.

Operatör	Açıklama
==	Eşittir
!=	Eşit değildir
>	Büyüktür
<	Küçüktür
>=	Büyük Eşittir
<=	Küçük Eşittir

== Operatörü: Aynı türdeki iki değerin birbirine eşitliğinin kontolü için kullanılan operatördür.

```
int x = 10;
int y = 4;
String str1 = "megep";
String str2 = "megep";
x == y // false
str1 == str2 // true
3 == 3 // true
3 == "3" // hatalı kullanım. int tipi ile String tipi karşılaştırılamaz.
```

!= Operatörü: Aynı türdeki iki değerin birbirine eşit olmadığının(eşit değil) kontolü için kullanılan operatördür.

```
int x = 10;
int y = 4;
String str1 = "megep";
String str2 = "megep";
x != y // true
str1 != str2 // false
3 != 3 // false
1 != 3 // true
```

> Operatörü: Bir değerin aynı türdeki başka bir değerden büyüklüğünün kontrolünün yapıldığı operatördür. Bu operatör String işlemlere uygulanmaz.

```
int x = 10;
int y = 4;
x > y // true
y > x // false
3 > 3 // false
5 > 3 // true
```

< Operatörü: Bir değerin aynı türdeki başka bir değerden küçüklüğünün kontrolünün yapıldığı operatördür. Bu operatör String işlemlere uygulanmaz.

```
int x = 10;
int y = 4;
x < y // false
y < x // true
3 < 3 // false
1 < 3 // true
```

>= Operatörü: Bir değerin aynı türdeki başka bir değerden büyük veya eşitliği kontrolünün yapıldığı operatördür. Bu operatör string işlemlere uygulanmaz.

```
int x = 10;
int y = 4;
x >= y // true
y >= x // false
3 >= 3 // true
```

<= Operatörü: Bir değerin aynı türdeki başka bir değerden küçük veya eşitliği kontrolünün yapıldığı operatördür. Bu operatör string işlemlere uygulanmaz.

```
int x = 10;
int y = 4;
x <= y // false
y <= x // true
3 <= 3 // true
```

2.3. Mantıksal Operatörler

Mantıksal operatörler birden fazla şartın olduğu durumlarda kullanılır. Birden çok boolean değeri tek bir boolean değere indirmek için kullanılır.

Operatör	**Açıklama**
&&	Ve
||	Veya
!	Değil

&& Operatörü: 'Ve' anlamındadır. Sorgulanan tüm şartlar doğru(true) olduğu zaman doğru(true), şartlardan birinin yanlış(false) olması durumunda yanlış(false) değerini döndürür.

```
int x = 10, y = 4;
String str1 = "megep";
x == y && "megep" == str1 // 1. şart = false, 2. şart = true -> sonuç = false
x == 10 && y == 4 && true == true // 1. şart = true, 2. şart = true, 3. şart = true -> sonuç = true
```

|| Operatörü: 'Veya' anlamındadır. Sorgulanan şartlardan birinin doğru(true) olması durumunda doğru(true), şartların hepsinin yanlış(false) olması durumunda yanlış(false) değerini döndürür.

```
int x = 10, y = 4;
string str1 = "megep";
x == y || "megep" == str1 // 1. şart = false, 2. şart = true -> sonuç = true
x == 4 || y == 10 || "megep" == str1 // 1. şart = false, 2. şart = false, 3. şart = true -> sonuç = true
x == 4 || y == 10 || true == false // 1. şart = false, 2. şart = false, 3. şart = false -> sonuç = false
```

! Operatörü: 'Değil' anlamındadır. ! işareti değeri tersine çevirir.

```
(!true) // sonuç = false
(!false) // sonuç = true
```

2.4. İşlem Önceliği

İşlem öncelik sırası aşağıdaki tabloda en yüksekten en düşüğe doğru sıralanmıştır.

En Yüksek
()
!
*, /, %
+, -
<, >
= =, !=
&&
||
En Düşük

Yapılan işlemde yukarıdaki sıra tamamlandıktan sonra eğer aynı tür işlemler kaldıysa işlem soldan sağa doğru yapılır.

Örnek 2.4-1: 3+5*2 işleminin sonucu nedir?

Yukardaki işlemde önce 3 ile 5'i toplar ve sonucu 2 ile çarparsanız sonuç yanlış çıkar. İşlem önceliğine göre önce 2 ile 5'i çarpıp çıkan sonuçla 3'ü topladığımız zaman sonuç doğru çıkar.

3+5*2 = 8*2 = 16 // yanlış cevap
3+5*2 = 3 + 10 = 13 // doğru cevap

Örnek 2.4-3: (5+2)*4-6/2 işleminin sonucu nedir?

(5+2)*4-6/2 = 7*4-6/2 = 28-6/2 = 28-3 = 25 // doğru cevap

1. KARAR KONTROL DEYİMLERİ

Program yazarken bazı noktalarda belirli koşullar altında gerçekleşmesini istenilen durumlar olabilir. Bu bölümde anlatılan if-else ve switch deyimleri ile bu tür kapsamlı programlar geliştirilebilir.

Genel anlamda programlama dilinde kullanılan koşul yapıları iki çeşittir. Bunlar;

- if-else deyimi
- switch deyimi

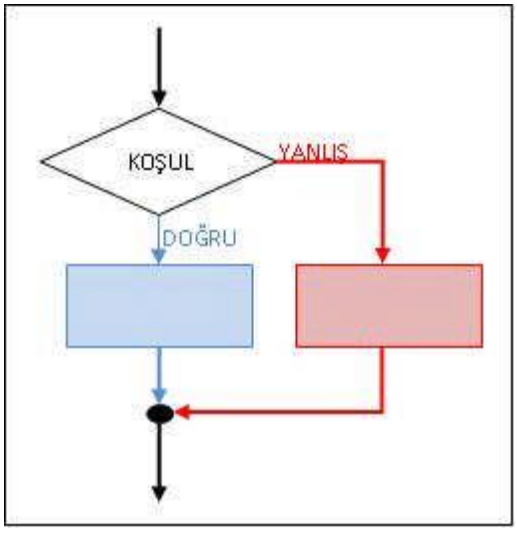
dir.

1.1. If-Else Deyimi

If deyimi bir programın akışını kontrol etmek için kullanılır. Belirli bir şarta göre yapılması istenilen işlemler, If-Else deyimi kullanılarak gerçekleştirilir. If-Else deyiminin kullanımı ve akış diyagramları ile gösterimi ise şu şekildedir.

Kullanımı:

```
if (koşul)
{
Koşul doğruysa yapılacak işlemler;
}
else
{
Koşul yanlışsa yapılacak işlemler;
}
```



Yukarıdaki diyagramdan da görüleceği üzere, programın akışı If deyiminin olduğu satıra geldiğinde parantezler içerisindeki KOŞUL ifadesi çalıştırılır. Bu koşul ifadesi true (Doğru) yada false (Yanlış) olmak üzere bir değer üretmektedir.

Şayet koşulumuz doğruysa (true) programımızın akışı mavi renkle gösterilen doğruysa kısmından devam edecek ve kırmızıyla gösterilen yanlışsa kısmına uğramayacaktır. Eğer koşulumuz yanlışsa (false) bu sefer programımız yanlışsa kısmından kırmızıyla belirtilen yoldan devam edecektir.

Not 1: Eğer programımızın akışında sadece koşulun doğru olmasına bağlı işlem yapılması isteniyor, koşulun yanlış olduğu durumlarda işlem yapılması istenmiyorsa Else bloğu program içerisinde hiç kullanılmaz.

Kullanımı:

```
if(koşul)
{
Koşul doğruysa yapılacak işlemler;
}
```

Not 2: Eğer If veya Else'den sonra sadece bir komut yazılacak ise küme parantezleri ({}) kullanılmayabilir.

Kullanımı:

```
if(koşul)
Koşul doğruysa yapılacak işlemler;
else
Koşul yanlışsa yapılacak işlemler;
```

Örnek 1-1: Klavyeden yaşı girilen kişinin ehliyet alıp alamayacağını belirten programı yazınız. Bu örneğimizde sayıların karşılaştırılmasını inceleyelim.

Kod	**Çıktı**
import java.util.Scanner; public class Main { public static void main(String[] args) { // write your code here Scanner oku=new Scanner(System.in); System.out.print("Yaşınızı giriniz: "); int yas=oku.nextInt(); if (yas < 18) System.out.println("Yaşınız 18'den küçük olduğu için ehliyet alamazsınız"); else System.out.println("Ehliyet alabilecek yaştasınız."); } }	Yaşınızı giriniz: 20 Ehliyet alabilecek yaştasınız. Yaşınızı giriniz: 16 Yaşınız 18'den küçük olduğu için ehliyet alamazsınız

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler	**Ekran Çıktısı**
16	
25	
18	
17	

Eşitlik bakımından değişkenleri karşılaştırmak için = = operatörünü kullandığına, özellikle dikkat edin. Bu amaç için = operatörünü kullanmayınız. Tek bir = operatörü, değişkenleri atamak için kullanılır.

Örnek 1-2: "Ünlü şairimiz Mehmet Akif'in soyadı nedir?" sorusunu kullanıcıya soran cevabını isteyen programı yazınız.

Bu örneğimizde metinsel ifadelerin karşılaştırılmasını inceleyelim.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { // write your code here Scanner oku=new Scanner(System.in); System.out.print("Ünlü şairimiz Mehmet Akif'in soyadı nedir?\nCevabınız :"); String cevap= oku.next(); if(cevap.equals("Ersoy")) { System.out.println("Tebrikler bu sorumuza doğru cevap verdiniz..."); } else { System.out.println("Malesef yanlış cevap"); } } }	Ünlü şairimiz Mehmet Akif'in soyadı nedir? Cevabınız :Ersoy Tebrikler bu sorumuza doğru cevap verdiniz... Ünlü şairimiz Mehmet Akif'in soyadı nedir? Cevabınız :ersoy Malesef yanlış cevap (Küçük büyük harf duyarlı)

Not 3: Bazı programlama dilleri büyük/küçük harf duyarlı bir dil olduğu için "Ersoy", "ersoy" veya "ERSOY" cevaplarından yalnızca "Ersoy" cevabını kabul edecektir.

If koşul deyimlerde zaman zaman birden fazla koşula bağlı bir takım işlemler yapmamız gerekebilir.

Kullanımı:

VE (&&) bağlacı ile

```
if((koşul1) && (koşul2))
{
koşul1 ve koşul2 doğruysa yapılacak işlemler;
}
else
{
koşullardan en az birisi veya her ikisi de yanlış ise
yapılacak işlemler;
}
```

VEYA (||) bağlacı ile

```
if((koşul1) || (koşul2))
{
koşul1 veya koşul2'den en az birisi veya her ikisi de doğruysa
yapılacak işlemler;
}
else
{
koşullardan her ikisi de yanlış ise yapılacak işlemler;
}
```

Örnek 1-3: Klavyeden girilen sayının hem 3'e hem de 5'e kalansız bölünüp bölünemediğini ekrana yazan programı yazınız.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { // write your code here Scanner oku=new Scanner(System.in); System.out.print("Bir sayı giriniz:"); int sayi = oku.nextInt(); if((sayi % 3 == 0) && (sayi % 5 ==0)) System.out.printf("%d sayısı hem 3'e hem de 5'e kalansız bölünebilir",sayi); else System.out.printf("%d sayısı hem 3'e hem de 5'e kalansız bölünemez", sayi); } }	Bir sayı giriniz:15 15 sayısı hem 3'e hem de 5'e kalansız bölünebilir Bir sayı giriniz:10 10 sayısı hem 3'e hem de 5'e kalansız bölünemez

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler	Ekran Çıktısı
150	
38	
64	
90	

Örnek 1-4: Klavyeden girilen cinsiyet ve yaş bilgilerine göre, kişinin askere gidip gidemeyeceğini yazan programı yazınız.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { // write your code here Scanner oku = new Scanner(System.in); String cinsiyet; int yas; System.out.print("Lütfen cinsiyetinizi giriniz (E/K):"); cinsiyet=oku.next(); System.out.print("Lütfen yaşınızı giriniz:"); yas =oku.nextInt(); System.out.println(cinsiyet); if ((cinsiyet.equals("E") || cinsiyet.equals("e")) && (yas >= 20))	

```
        {
            System.out.println("Askere Gidebilir");
        }
        else
        {
            System.out.println("Askere Gidemez");
        }
    }
}
```

Girilecek Değerler		Ekran Çıktısı
Cinsiyset	**Yaş**	
E	18	
K	21	
E	23	
K	19	

1.2. İç-İçe If İfadesi

Birden fazla koşula ihtiyaç duyulan durumlarda iç-içe If ifadeleri kullanılırlar. Bir if koşuluna kaç tane else if ekleyebileceğiniz konusunda hiçbir sınır yoktur. İç-içe If ifadelerinin kullanımı ise şu şekildedir.

Kullanımı:

```
if(koşul1)
{
koşul1 doğruysa yapılacak işlemler;
}
else if(koşul2)
{
koşul1 yanlışsa ve koşul2 doğruysa yapılacak işlemler;
}
else
{
her iki koşul da yanlışsa yapılacak işlemler;
}
```

Dilerseniz iç-içe If ifadelerini birkaç örnekle açıklamaya çalışalım.

Örnek 1-5: Klavyeden girilen iki sayıyı karşılaştıran programı yazınız.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { // write your code here Scanner oku = new Scanner(System.in); int sayi1, sayi2; System.out.print("1. sayıyı giriniz: "); sayi1 = oku.nextInt(); System.out.print("2. sayıyı giriniz: "); sayi2 = oku.nextInt(); if(sayi1>sayi2)	1. sayıyı giriniz: 15 2. sayıyı giriniz: 10 15 sayısı 10 sayısından büyüktür. 1. sayıyı giriniz: 10 2. sayıyı giriniz: 15 15 sayısı 10 sayısından büyüktür. 1. sayıyı giriniz: 10 2. sayıyı giriniz: 10 10 sayısı ile 10 sayısı birbirine eşittir.

```
            System.out.printf("%d sayısı %d sayısından büyük-
tür.",sayi1,sayi2);
        else if(sayi1<sayi2)
            System.out.printf("%d sayısı %d sayısından büyük-
tür.", sayi2,sayi1);
        else
            System.out.printf("%d  sayısı ile %d sayısı birbirine
eşittir.", sayi1, sayi2);

    }
}
```

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler		**Ekran Çıktısı**
sayi1	**sayi2**	
12	17	
43	43	
98	21	
-106	106	
-66	-16	

Örnek 1-6: Klavyeden girilen puanın 5'lik sistemdeki not karşılığını yazan programı yazınız.

Kod	**Çıktı**
import java.util.Scanner; public class Main { public static void main(String[] args) { // write your code here Scanner oku = new Scanner(System.in); System.out.print("Puanınızı giriniz (0-100):"); int puan= oku.nextInt(); if (puan >= 0 && puan < 25) System.out.print("Puanınızın 5'lik sistemdeki karşılığı 0'dır"); else if (puan >= 25 && puan < 45) System.out.print("Puanınızın 5'lik sistemdeki karşılığı 1'dir"); else if (puan >= 45 && puan < 55) System.out.print("Puanınızın 5'lik sistemdeki karşılığı 2'dir"); else if (puan >= 55 && puan < 70) System.out.print("Puanınızın 5'lik sistemdeki karşılığı 3'tür"); else if (puan >= 70 && puan < 85) System.out.print("Puanınızın 5'lik sistemdeki karşılığı 4'tür"); else if (puan >= 85 && puan <= 100) System.out.print("Puanınızın 5'lik sistemdeki karşılığı 5'tir"); else System.out.print("Hatalı puan girdiniz. Puanınız 0 ile 100 arasında olmalıdır."); } }	Puanınızı giriniz (0-100):55 Puanınızın 5'lik sistemdeki karşılığı 3'tür

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

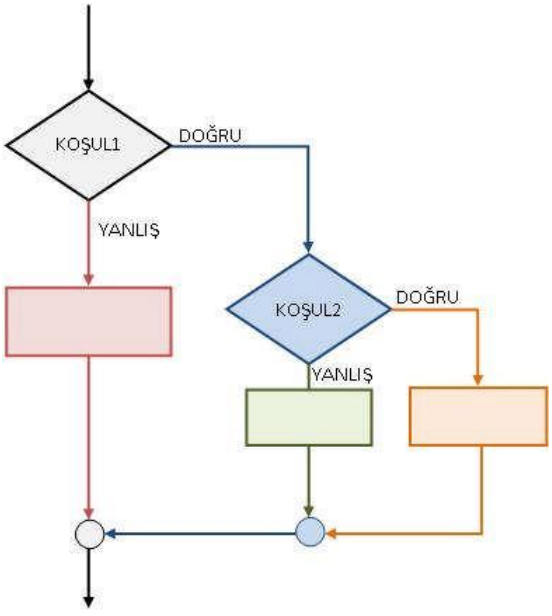
Girilecek Değerler	Ekran Çıktısı
86	
69	
43	
77	
14	
52	

İç-içe If ifadelerinin bir başka kullanımı da şu şekildedir.

Örneğin bir koşulun sağlanması durumunda başka koşullara göre işlem yapılması istenilen durumlarda yine iç-içe If ifadeleri kullanılırlar. Bu durumdaki iç-içe If ifadelerinin kullanımları ve akış diyagramlarıyla gösterimi şu şekildedir;

Kullanımı:

```
if(koşul1)
{
if(koşul2)
{
koşul2 doğruysa yapılacak işlemler;
}
else
{
koşul2 yanlışsa yapılacak işlemler;
}
}
else
{
koşul1 yanlışsa yapılacak işlemler;
}
```



Örnek 1-7: Daha önceden belirlenen kullanıcı adı ve şifreyi kontrol eden programı yazınız.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { // write your code here Scanner oku = new Scanner(System.in); String kullaniciAdi, sifre; System.out.print("Lütfen kullanıcı adınızı giriniz:"); kullaniciAdi = oku.next(); if (kullaniciAdi.equals("Admin") || kullaniciAdi.equals("ADMİN") || kullaniciAdi.equals("admin")) { System.out.print("Lütfen şifrenizi giriniz:"); sifre = oku.next(); if (sifre.equals("123rty")) System.out.print("Tebrikler Kullanıcı ve Şifreniz Doğru"); else System.out.print("Şifrenizi Hatalı Girdiniz"); } else System.out.print("Kullanıcı Adınızı Hatalı Girdiniz"); } }	

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler		Ekran Çıktısı
Kullanıcı Adı	**Şifre**	
Admin	123RTY	
Yönetici	123rty	
Admin	123rty	
admin	123rty	
ADMİN	123rty	

!!! Uyarı: Yukarıdaki örnekte şifre "123rty" şeklinde verilmiştir.

Örnek 1-8: Basit bir hesap makinesi yapımı.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { // write your code here Scanner oku = new Scanner(System.in); Byte secim; double sayi1,sayi2,sonuc; System.out.println("1.TOPLAMA"); System.out.println("2.ÇIKARMA"); System.out.println("3.ÇARPMA");	

```
System.out.println("4.BÖLME");
System.out.println("-------");
System.out.print("İşlem tipinizi seçiniz (1-4):");
secim = oku.nextByte();
 System.out.println(secim);
if (secim == 1)
{

   System.out.println("**********************************");
   System.out.println("* Seçilen işlem TOPLAMA işlemi *");
   System.out.println("**********************************");
   System.out.print("1.Sayıyı giriniz:");
   sayi1 = oku.nextDouble();
   System.out.print("2.Sayıyı giriniz:");
   sayi2 = oku.nextDouble();
   sonuc = sayi1 + sayi2;
   System.out.printf("Sonuç = %f", sonuc);
}
else if (secim == 2)
{

   System.out.println("**********************************");
   System.out.println("* Seçilen işlem ÇIKARMA işlemi *");
   System.out.println("**********************************");
   System.out.print("1.Sayıyı giriniz:");
   sayi1 = oku.nextDouble();
   System.out.print("2.Sayıyı giriniz:");
   sayi2 = oku.nextDouble();
   sonuc = sayi1 - sayi2;
    System.out.printf("Sonuç = %f", sonuc);
}
else if (secim == 3)
{

   System.out.println("**********************************");
   System.out.println("* Seçilen işlem ÇARPMA işlemi *");
   System.out.println("**********************************");
   System.out.print("1.Sayıyı giriniz:");
   sayi1 = oku.nextDouble();
   System.out.print("2.Sayıyı giriniz:");
   sayi2 = oku.nextDouble();
   sonuc = sayi1 * sayi2;
    System.out.printf("Sonuç = %f", sonuc);
}
else if (secim == 4)
{

   System.out.println("**********************************");
   System.out.println("* Seçilen işlem ÇIKARMA işlemi *");
   System.out.println("**********************************");
   System.out.print("1.Sayıyı giriniz:");
   sayi1 = oku.nextDouble();
```

```
        System.out.print("2.Sayıyı giriniz:");
        sayi2 = oku.nextDouble();
        if (sayi2 != 0)
        {
           sonuc = sayi1 / sayi2;
            System.out.printf("Sonuç = %f", sonuc);
        }
        else
           System.out.print("!!! SIFIRA BÖLME HATASI !!!");
     }
   }
}
```

1.3. Switch-Case Deyimi

Switch-Case deyimi de If-Else deyimleri gibi karar kontrol mekanizmalarında kullanılmaktadır. Switch-Case deyimi genellikle karmaşık if-else bloklarının yerine, daha okunabilir oldukları için tercih edilmektedir. Switch-Case ile yapabileceğimiz karşılaştırmaları if-else ile de yapabiliriz.

Switch-Case yapısı şu şekilde çalışır; bir deyimin değeri, sabitlerden oluşan bir listede peş peşe test edilir. Deyimin değeri sabitlerden birisiyle eşleşince, bu eşleşmeyle ilgili işlemler gerçekleştirilir.

Switch-Case ifadesinin genel formu şu şekildedir;

Kullanımı:

```
switch(ifade)
{
case sabit1:
Yapılacak işlemler;
break;
case sabit2:
Yapılacak işlemler;
break;
case sabit3:
Yapılacak işlemler;
break;
.
.
.
default:
Yapılacak işlemler;
break;
      }
```

Switch-Case yapısının çalışmasına bir göz atalım;

- Önce switch parantezleri içerisindeki ifade hesaplanır.
- Programın akışı, hesaplanan ifade ile aynı case sabitinin bulunduğu satıra gelir.
- Eğer hesaplanan ifade, mevcut case sabitlerinden herhangi birisi ile eşleşmiyorsa **default** anahtar sözcüğünün bulunduğu yere gelir ve program buradan devam eder.

Eğer aşağıdaki örnekteki gibi break komutu kullanılmazsa, "***Control cannot fall through from one case label ('case 1:') to another***" yani "***Bir case etiketinden ('case1:')başka bir case etiketine geçilemez***" hatasını alırız.

```
switch(ifade)
{
```

```
case 1:
Yapılacak işlemler;
case 2:
Yapılacak işlemler;
break;
case 3:
Yapılacak işlemler;
break;
default:
Yapılacak işlemler;
break;
      }
```

Switch-case yapısında case durumların sırasının sorun olmamaktadır. Default durumunu bile ilk sıraya koyabilirsiniz. Sonuç olarak, iki durum aynı olamayacağı için ilgili case yapısına gelindiğinde o satırın çalışması sağlanacaktır.

```
switch(ifade)
{
default:
Yapılacak işlemler;
break;
case 3:
Yapılacak işlemler;
break;
case 1:
Yapılacak işlemler;
break;
case 2:
Yapılacak işlemler;
break;
      }
```

Switch-Case Yapısı İle İlgili Önemli Kurallar:

- Case anahtar sözcüğünün yanındaki ifadeler sabit olmak zorundadırlar. Bu ifadeler içerisinde değişken bulunamaz.
- Case ifadeleri herhangi bir tam sayı sabiti, karakter veya string sabiti olabilir.
- Default durumunu istediğimiz yere yazabiliriz. Aynı şekilde case ifadelerini de istediğimiz sırada yazabiliriz.
- Bir switch bloğunda iki veya daha fazla sayıda aynı değere sahip case ifadesi bulunamaz.
- Bir switch bloğunda default case olmak zorunda değildir.

Örnek 1-9: Klavyeden girilen 1-12 arasındaki sayı değerine göre o sıradaki ayın ismini veren programı yazınız

Kod

```
import java.util.Scanner;

public class Main {

   public static void main(String[] args) {
      // write your code here
      Scanner oku = new Scanner(System.in);
      Byte ay;
      System.out.print("1-12 arasında bir sayı giriniz:");
      ay=oku.nextByte();
      switch (ay){
```

```
        case 1:
           System.out.printf("%d.ay OCAK ayıdır.", ay);
           break;
        case 2:
           System.out.printf("%d.ay ŞUBAT ayıdır.", ay);
           break;
        case 3:
           System.out.printf("%d.ay MART ayıdır.", ay);
           break;
        case 4:
           System.out.printf("%d.ay NİSAN ayıdır.", ay);
           break;
        case 5:
           System.out.printf("%d.ay MAYIS ayıdır.", ay);
           break;
        case 6:
           System.out.printf("%d.ay HAZİRAN ayıdır.", ay);
           break;
        case 7:
           System.out.printf("%d.ay TEMMUZ ayıdır.", ay);
           break;
        case 8:
           System.out.printf("%d.ay AĞUSTOS ayıdır.", ay);
           break;
        case 9:
           System.out.printf("%d.ay EYLÜL ayıdır.", ay);
           break;
        case 10:
           System.out.printf("%d.ay EKİM ayıdır.", ay);
           break;
        case 11:
           System.out.printf("%d.ay KASIM ayıdır.", ay);
           break;
        case 12:
           System.out.printf("%d.ay ARALIK ayıdır.", ay);
           break;
           default:
              System.out.println("Girmiş olduğunuz değer 1-12 arasında değildir."); break;
     }

  }
}
```

Örnek 1-10: Klavyeden girilen değer ile seçimi yapılan şeklin alanını veya çevresini bulan programı yazınız

Kod	Çıktı

```
import java.util.Scanner;

public class Main {

   public static void main(String[] args) {
      // write your code here
      Scanner oku = new Scanner(System.in);
      String sekil,secim;
      int kenar1, kenar2;
      System.out.println("1.KARE---------->(kare)");
      System.out.println("2.DİKDÖRTGEN--->(dikdörtgen)");
      System.out.println("---------------");
      System.out.print("Lütfen şeklin ismini yazınız:");
      sekil = oku.next();

      switch (sekil)
      {
         case "kare":
            System.out.print("Karenin bir kenar uzunluğunu giriniz:");
            kenar1 = oku.nextInt();

            System.out.println(" # ALAN---------->(alan)");
            System.out.println(" # ÇEVRE-------->(çevre)");
            System.out.println("------------------------");
            System.out.print("Lütfen seçiminizi yazınız:");
            secim = oku.next();
            switch (secim)
            {
               case "alan":
                  System.out.printf("Karenin alanı=%d", kenar1 * kenar1);
                  break;
               case "çevre":
                  System.out.printf("Karenin çevresi=%d", kenar1 * 4);
                  break;
               default:
                  System.out.print("Geçerli bir seçim yapmadınız...");
                  break;
            } break;
         case "dikdörtgen":
            System.out.println(" # ALAN---------->(alan)");
            System.out.println(" # ÇEVRE-------->(çevre)");
            System.out.println("------------------------");
            System.out.print("Lütfen seçiminizi yazınız:");
            secim = oku.next();

            switch (secim)
            {
               case "alan":
                  System.out.print("Dikdörtgenin bir kenar uzunluğunu giriniz:");
```

```
                kenar1 = oku.nextInt();
                System.out.print("Dikdörtgenin diğer kenar uzunluğunu giriniz:");
                kenar2 = oku.nextInt();
                System.out.printf("Dikdörtgenin alanı=%d", kenar1 * kenar2);
                break;
            case "çevre":
                System.out.print("Dikdörtgenin bir kenar uzunluğunu giriniz:");
                kenar1 = oku.nextInt();
                System.out.print("Dikdörtgenin diğer kenar uzunluğunu giriniz:");
                kenar2 = oku.nextInt();
                System.out.printf("Dikdörtgenin çevresi=%d", (kenar1+kenar2) * 2);
                break;
            default:
                System.out.print("Geçerli bir seçim yapmadınız...");
                break;
          }
          break;
       default:
          System.out.println("Geçerli bir seçim yapmadınız...");
          break;
    }
  }
}
```

Örnek 1-11: Bir önceki konuda Örnek-1-8'de IF-ELSE ile yaptığımız basit hesap makinesi programını Switch-Case ile biraz değiştirerek tekrar yapalım.

Kod

```
import java.util.Scanner;

public class Main {

   public static void main(String[] args) {
      // write your code here
      Scanner oku = new Scanner(System.in);
      String secim;
      double sonuc;
      int sayi1, sayi2;
      System.out.println("1.TOPLAMA--->T");
      System.out.println("2.ÇIKARMA--->C");
      System.out.println("3.ÇARPMA --->R");
      System.out.println("4.BÖLME --->B ");
      System.out.println("---------------");
      System.out.println();
      System.out.print("İşlem tipinizi seçiniz (T-C-R-B):");
      secim = oku.next();

      switch(secim){
         case "T":
         System.out.println("*********************************");
         System.out.println("* Seçilen işlem TOPLAMA işlemi *");
         System.out.println("*********************************");
         System.out.print("1.Sayıyı giriniz:");
```

```
        sayi1 = oku.nextInt();
        System.out.print("2.Sayıyı giriniz:");
        sayi2 = oku.nextInt();
        sonuc = sayi1 + sayi2;
        System.out.printf("Sonuç= %f", sonuc);
        break;
        case "C":
        System.out.println("**********************************");
        System.out.println("* Seçilen işlem ÇIKARMA işlemi *");
        System.out.println("**********************************");
        System.out.print("1.Sayıyı giriniz:");
        sayi1 = oku.nextInt();
        System.out.print("2.Sayıyı giriniz:");
        sayi2 = oku.nextInt();
        sonuc = sayi1 - sayi2;
        System.out.printf("Sonuç=%f", sonuc);
        break;
        case "R":
        System.out.println("**********************************");
        System.out.println("* Seçilen işlem ÇARPMA işlemi *");
        System.out.println("**********************************");
        System.out.print("1.Sayıyı giriniz:");
        sayi1 = oku.nextInt();
        System.out.print("2.Sayıyı giriniz:");
        sayi2 = oku.nextInt();
        sonuc = sayi1 * sayi2;
        System.out.printf("Sonuç=%f", sonuc);
        break;
        case "B":
        System.out.println("**********************************");
        System.out.println("* Seçilen işlem ÇIKARMA işlemi *");
        System.out.println("**********************************");
        System.out.print("1.Sayıyı giriniz:");
        sayi1 = oku.nextInt();
        System.out.print("2.Sayıyı giriniz:");
        sayi2 = oku.nextInt();
        switch (sayi2)
        {
           default:
              sonuc = sayi1 / sayi2;
              System.out.printf("Sonuç=%f", sonuc);  break;
           case 0:
              System.out.println("!!! SIFIRA BÖLME HATASI !!!"); break;
        }
        break;
        default:
           System.out.println("T-C-R-B 'den farklı bir değer girdiniz...");
           break;
     }
  } }
```

Not: Yukarıdaki bazı satırlar alta kaymıştır, programı yazarken kayan satırların tek satırda olmasına dikkat ediniz.

2. DÖNGÜ DEYİMLERİ

Döngüler bir program içerisinde belirli işlerin defalarca yapılmasını sağlayan komut bloklarıdır. Sonsuz döngüler yapılabildiği gibi belirli kriterler sağlanana kadar devam eden döngüler de yapılabilir. 4 tip döngü vardır. Bunlar:

- for döngüleri
- while döngüleri
- do while döngüleri
- foreach döngüleri' dir.

2.1. Döngü Çeşitleri

2.1.1. For Döngüsü

Belirlenen başlangıç değerinden itibaren belirtilen koşul sağlanana kadar içine yazıldığı kod parçasını ardı ardına çalıştıran bir döngü çeşididir.

For döngüsünün kullanımı şu şekildedir;

Kullanımı:

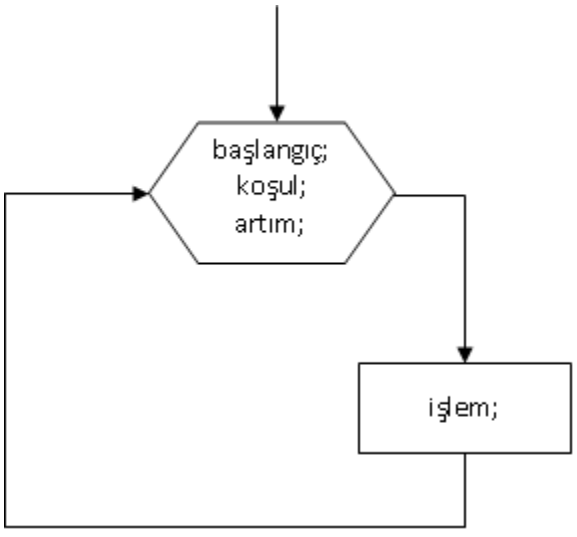
```
for(başlangıç;koşul;artım)
{
yapılacak işler;
}
```

Başlangıç, döngü kontrol değişkeni olarak da ifade edilebilir. Döngü içerisinde bir sayaç görevini görür.

Koşul, döngünün ne kadar çalışacağını denetleyen mekanizmadır. Koşul sağlanıyorken döngü çalışmaya devam eder. Koşul sağlanmadığında ise döngü durur. Koşulda genellikle başlangıç değerinin durumu denetlenir.

Artım, başlangıç değerinin döngünün her adımda artma ya da azaltma miktarını belirler. Eğer başlangıç değeri hiç değişmez ise sonsuz döngü oluşur.

Akış diyagramlarıyla for döngüsünün gösterimi de şu şekildedir.



Şimdi basit bir örnekle for döngüsünün çalışmasını inceleyelim.

Örnek 2-1: 1'den 10'a kadar olan sayıları ekrana yazdırınız.

```
for (int i = 1; i <=10 ; i++) {
        System.out.println(i);
    }
```

Yukarıdaki kodu incelediğimizde;

- Döngü kontrol değişkenimiz olan i'ye 1 değerini atayarak başlangıç değerimizi,
- Döngümüzün ne zamana kadar döneceğini belirlediğimiz koşulumuzu i<=10 ifadesini,
- i++ ile de i değerimizi döngümüzün her dönüşünde 1 arttıracağımızı belirliyoruz.

Döngü her seferinde koşul kısmını kontrol eder ve buradaki koşul `false`(yanlış) olana kadar küme parantezleri ({ }) ile sınırlandırılan kod bloğunu çalıştırmaya devam edecektir.

Kod parçamızı çalıştırdığımızda aşağıdaki gibi bir ekran çıktısı alabiliriz

```
1
2
3
4
5
6
7
8
9
10
```

For terimiyle döngü kurarken başlangıç değerimiz herhangi bir tam sayı olabileceği gibi char türünde bir değişkende olabilir.

```
char i;
    for ( i = 'd'; i <='r' ; i++) {
        System.out.println(i);
    }
```

```
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
```

For döngüsüyle sonsuz bir döngü oluşturulmak istenirse şu şekilde kodlanması gerekir;

```
for(;;)
{
//.....
}
```

Uyarı: Bu şekilde bir sonsuz döngüyü bilgisayarınızda çalıştırdığınız zaman uygulamanız sonsuza kadar devam eder. For döngüleri ileriye doğru sayabildiği gibi geriye dönük sayma işlemlerinde de kullanılırlar.

Örnek 2-3: 10'dan 0'a geriye doğru sayan ve sayıları ekrana yazdıran programı yazdırınız.

Kod	Çıktı
public class Main { public static void main(String[] args) { int i; for (i=10;i>=0;i--) System.out.println(i); } }	10 9 8 7 6 5 4 3 2 1 0

For döngüsü içerisinde birden fazla döngü kontrol değişken kullanma şansına da sahibiz.

Kod	Çıktı
int i,j; for (i = 0, j = 10; i <= j; i++, j--) System.out.println("i'nin "+i+" değeri için j="+j); }	i'nin 0 değeri için j=10 i'nin 1 değeri için j=9 i'nin 2 değeri için j=8 i'nin 3 değeri için j=7 i'nin 4 değeri için j=6 i'nin 5 değeri için j=5

Örnek 2-5: 0'dan klavyeden girilen sayıya kadar olan sayıların toplamını ekrana yazdıran programı yazınız.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { int bitis,i,toplam; System.out.print("Bir sayı giriniz:"); Scanner oku=new Scanner(System.in); bitis=oku.nextInt(); toplam = 0; for (i = 0; i <= bitis; i++) { toplam = toplam + i;	Bir sayı giriniz:20 Toplam = 210

```
    }
    System.out.printf("Toplam = %d", toplam );
}}
```

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler	Ekran Çıktısı
15	
50	

Şimdiye kadar gördüğümüz örneklerde for döngüsünü hep tek başına kullandık. Aynı koşul kontrol mekanizmalarında olduğu gibi döngüler de iç-içe kullanılabilirler. İç-içe kullanılacak döngü sayılarında herhangi bir kısıtlama söz konusu değildir. İstediğimiz kadar sayıda döngüyü iç-içe kullanabiliriz

Sıradaki örneklerimizde de iç içe for döngüsü nasıl kullanılır buna göz atalım.

Kod	Çıktı
public class Main { public static void main(String[] args) { String yildiz = ""; for (int i = 1; i <= 5; i++) { for (int k = 0; k <= 5-i;k++) yildiz = yildiz + "*"; System.out.println(yildiz); yildiz = ""; } } }	***** **** *** ** *

Örnek 2-7: 1'den 10'a kadar olan sayılar için çarpım tablosunu ekrana yazdıran programı yazınız.

Kod	Çıktı
public static void main(String[] args) { int i,k; for (i = 1; i <=10; i++) { System.out.println("-"+i+" ve Katları-"); System.out.println("----------"); for (k = 1; k <= 10; k++) { int carpim = i * k; System.out.printf("%d x %d = %d \n", i, k, carpim); } System.out.println("----------"); } }	-1 ve Katları- ---------- 1 x 1 = 1 1 x 2 = 2 1 x 3 = 3 1 x 4 = 4 1 x 5 = 5 1 x 6 = 6 1 x 7 = 7 1 x 8 = 8 1 x 9 = 9 1 x 10 = 10 ---------- -2 ve Katları- ---------- 2 x 1 = 2 2 x 2 = 4 2 x 3 = 6 2 x 4 = 8 2 x 5 = 10 2 x 6 = 12 2 x 7 = 14 2 x 8 = 16 2 x 9 = 18 2 x 10 = 20 ---------- -3 ve Katları- ---------- 3 x 1 = 3 3 x 2 = 6 3 x 3 = 9 3 x 4 = 12 3 x 5 = 15 3 x 6 = 18 3 x 7 = 21 3 x 8 = 24 3 x 9 = 27 3 x 10 = 30 ---------- -4 ve Katları- ---------- 4 x 1 = 4 4 x 2 = 8 4 x 3 = 12 4 x 4 = 16 4 x 5 = 20 4 x 6 = 24 4 x 7 = 28 4 x 8 = 32 4 x 9 = 36 4 x 10 = 40 ---------- -5 ve Katları- ---------- 5 x 1 = 5 5 x 2 = 10 5 x 3 = 15 5 x 4 = 20 5 x 5 = 25 5 x 6 = 30 5 x 7 = 35 5 x 8 = 40 5 x 9 = 45 5 x 10 = 50 ---------- -6 ve Katları- ---------- 6 x 1 = 6 6 x 2 = 12 6 x 3 = 18 6 x 4 = 24 6 x 5 = 30 6 x 6 = 36 6 x 7 = 42 6 x 8 = 48 6 x 9 = 54 6 x 10 = 60 ---------- -7 ve Katları- ---------- 7 x 1 = 7 7 x 2 = 14 7 x 3 = 21 7 x 4 = 28 7 x 5 = 35 7 x 6 = 42 7 x 7 = 49 7 x 8 = 56 7 x 9 = 63 7 x 10 = 70 ---------- -8 ve Katları- ---------- 8 x 1 = 8 8 x 2 = 16 8 x 3 = 24 8 x 4 = 32 8 x 5 = 40 8 x 6 = 48 8 x 7 = 56 8 x 8 = 64 8 x 9 = 72 8 x 10 = 80 ---------- -9 ve Katları- ---------- 9 x 1 = 9 9 x 2 = 18 9 x 3 = 27 9 x 4 = 36 9 x 5 = 45 9 x 6 = 54 9 x 7 = 63 9 x 8 = 72 9 x 9 = 81 9 x 10 = 90 ---------- -10 ve Katları- ---------- 10 x 1 = 10 10 x 2 = 20 10 x 3 = 30 10 x 4 = 40 10 x 5 = 50 10 x 6 = 60 10 x 7 = 70 10 x 8 = 80 10 x 9 = 90 10 x 10 = 100 ----------

2.1.2. While Döngüsü

While döngüsü bir koşul sağlanıyorken dönmeye devam eder. Koşul yanlış (false) sonucunu verdiği zaman ise sonlandırılır.

Genel yazım şekli şöyledir.

- **Kullanımı:**

```
while(koşul)
{
yapılacak işler;
}
```

Örnek 2-8: 0'dan 20'ye kadar olan çift sayıları ekrana yazdırınız.

Kod	Çıktı
public class Main { public static void main(String[] args) { int i=0; while (i <= 20) { System.out.println(i); i = i + 2; } } }	0 2 4 6 8 10 12 14 16 18 20

Örnek 2-9: Bilgisayara rastgele ürettirdiğimiz bir sayıyı 5 hakta tahmin etmeye çalışan bir bilgisayar programı yazınız.

Kod	Çıktı
import java.util.Random; import java.util.Scanner; public class Main { public static void main(String[] args) { int hak = 5; Random rnd = new Random(); Scanner oku=new Scanner(System.in); int tutulan = rnd.nextInt(50)+1; int sayi=0; while (hak>0) { System.out.print("Bir sayı giriniz: "); sayi =oku.nextInt(); hak = hak - 1; if (sayi == tutulan) { System.out.println("Tebrikler sayıyı doğru tahmin ettiniz"); break; } else	Bir sayı giriniz: 20 Yukarı Kalan tahmin hakkınız:4 Bir sayı giriniz: 30 Aşağı Kalan tahmin hakkınız:3 Bir sayı giriniz: 25 Tebrikler sayıyı doğru tahmin ettiniz

{ if (sayi > tutulan) System.out.println("Aşağı"); else System.out.println("Yukarı"); } System.out.printf("Kalan tahmin hakkınız:%d \n", hak); } if(hak==0) System.out.printf("Tahmin hakkınız bitti. Sayımız:%d",tutulan); } }	

Yukarıdaki programda karşımıza çıkan Random komutu bize belirtilen bir aralıkta rastgele sayı üretmemizi sağlayan bir komuttur. Programımızda bizler 1-50 arasında bir sayı üretmesini sağladık.

Bir diğer dikkat etmemiz gereken komutumuz da break komutudur. Aynı bir önceki öğrenme faaliyetinde gördüğümüz select-case yapısındaki gibi sonlandırma işine yarayan break komutunun While döngüleriyle birlikte kullanımı oldukça yaygındır. Döngülerden istenilen koşulun sağlanmasını beklemeden çıkmak için kullanılır.

2.1.3. Do…While Döngüsü

For ve while döngülerinde döngü bloklarının koşul sağlanmadığı takdirde hiç çalıştırılmama ihtimali vardır. Ancak döngünün en az bir kere çalıştırılması istenilen durumlarda do-while döngüleri kullanılırlar.

Do-While döngülerinde koşul döngü içerisindeki işlemler bir kez gerçekleştirildikten sonra kontrol edilir. Koşul doğru olduğu müddetçe de döngü içerisindeki işlemler tekrarlanmayı sürdürür.

Genel yazım şekli şöyledir.

- **Kullanımı:**

```
do
{
yapılacak işler;
}
while(koşul);
```

Örnek 2-10: 1'den 20'ye kadar olan tek sayıları ekrana yazdırınız.

Kod	Çıktı
public static void main(String[] args) { int i=1; do{ System.out.println(i); i = i + 2; } while (i < 20); }	1 3 5 7 9 11 13 15 17 19

2.1.4. Foreach Döngüsü

foreach, dizi (Array) ve koleksiyon (collection) tabanlı nesnelerin elemanları üzerinden ilerleyen bir döngüdür.

Genel kullanım şekli şöyledir;

- **Kullanımı:**

```
for(tip değişken : koleksiyon)
{
yapılacak işler;
}
```

- `Tip:` buradaki tip koleksiyonun veri tipi ile aynı veya uyumlu olmak zorundadır.
- `Değişken:` foreach döngüsü içerisinde koleksiyonda bulunan sıradaki elemanı temsil eder.
- `Koleksiyon:` ArrayList ya da dizi gibi aynı tip verileri barındıran koleksiyon.

Uyarı: Bir sonraki öğrenme faaliyeti olan Diziler konusunda foreach döngüsünün kullanımına yönelik daha fazla örnek gösterilecektir. Sadece ön bilgi amacıyla aşağıdaki örneği inceleyiniz.

Örnek 2-11: Gunler isimli dizi içerisindeki elemanları ekrana yazdıran programı yazınız.

Kod	Çıktı
public class Main { public static void main(String[] args) { String[] gunler=new String[7]; gunler[0]="Pazartesi"; gunler[1]="Salı"; gunler[2]="Çarşamba"; gunler[3]="Perşembe"; gunler[4]="Cuma"; gunler[5]="Cumartesi"; gunler[6]="Pazar"; for (String gun:gunler) { System.out.println(gun); } } }	Pazartesi Salı Çarşamba Perşembe Cuma Cumartesi Pazar

Burada, foreach, dizi boyunca her seferinde bir elemanı adımlar. Dizinin her bir elemanının değerini gun adındaki string değişkenine aktarır ve daha sonra döngüyü başlatır.

2.2. Jump (Dallanma – Atlama) Komutları

Programın akışı esnasında başka satırlara atlama işlemi gerçekleştiren bir takım anahtar sözcükler vardır.
Bunlar;

- break,
- continue,
- goto,
- return anahtar sözcükleridir.

2.2.1. Break Anahtar Sözcüğü

Break anahtar sözcüğü döngülerden çıkmak için kullanılır. Döngülerde, break anahtar sözcüğüne rastlandığı anda döngüden çıkılır ve program döngü bloğundan sonraki ilk deyimle akışına devam eder.

Break anahtar sözcüğü döngü bloklarının ya da switch bloklarının dışında kullanılamazlar.

Örnek 2-12: 'A' harfinden başlayıp 'Z'ye kadar devam eden bir döngü de 'K' harfine gelindiğinde döngüden çıkan programın kodunu yazınız.

Kod	Çıktı
public static void main(String[] args) { for (char i = 'A'; i <= 'Z'; i++) { if (i == 'K') break; System.out.println(i); System.out.println("Döngüye devam ediliyor..."); } System.out.println("Döngüden çıkıldı..."); }	A Döngüye devam ediliyor... B Döngüye devam ediliyor... C Döngüye devam ediliyor... D Döngüye devam ediliyor... E Döngüye devam ediliyor... F Döngüye devam ediliyor... G Döngüye devam ediliyor... H Döngüye devam ediliyor... I Döngüye devam ediliyor... J Döngüye devam ediliyor... Döngüden çıkıldı...

Yukarıdaki örnek incelendiğinde döngümüz 'A' harfinden başlayarak teker teker harfleri yazmaktayken 'K' harfine geldikten sonra `break` komutuyla karşılaşır. Bu komutu gören program o anda içerisinde bulunduğu döngüyü sonlandırır ve programın akışına kaldığı yerden devam eder

Örnek 2-13: 0'dan 100'e kadar sayılardan asal olanları ekrana yazdıran programın kodunu yazınız.

Kod	Çıktı
public static void main(String[] args) { for(int i=1; i < 100; i++){ boolean asalMi = true; //Sayının asal olup olmadığı kontrol ediliyor for(int j=2; j < i; j++) { if(i % j == 0){ asalMi = false; break; } } // asal olan sayılar ekrana yazdırılıyor	1 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

if(asalMi) System.out.println(i + " "); } }	

2.2.2. Continue Anahtar Sözcüğü

Continue ifadesi, break ifadesine benzerdir ve bir for, foreach, while ya da do...while döngüsü içinde de kullanılabilir. Ancak, döngünün dışına çıkmak yerine mevcut döngüden çıkarak bir sonraki döngüye geçişi sağlayacaktır.

Şimdiki örneğimizde continue anahtar sözcüğünün kullanımını inceleyelim.

Örnek 2-14: Continue anahtar sözcüğünün kullanımı.

Kod	Çıktı
public static void main(String[] args) { int i = 1; int k = 1; while (i < 10) { System.out.printf("i:%d iken k:%d \n",i,k); i++; continue; k++; } }	Error:(16, 13) java: unreachable statement

Yukarıdaki kod parçasını incelediğimizde döngümüzün koşulu, `i`'nin 10'dan küçük olan değerleri sağlanması durumunda `TRUE` değerini almasıdır.

`i` ve `k` değişkenlerimizin değerleri döngümüz içerisinde `i++` ve `k++` ifadeleriyle arttırılmaktadır. Lakin programımız k değişkenine ulaşamadığı için hiç derlenmez.

2.2.3. Goto Anahtar Sözcüğü gibi Etiket Kullanma

Goto anahtar sözcüğü, koşulsuz atlama komutudur. Programın akışı esnasında goto anahtar sözcüğüyle karşılaşıldığı anda program goto ile belirlenen etiketin bulunduğu satıra atlama işlemi gerçekleştirir.

Goto anahtar sözcüğünün kullanımı nesne yönelimli programlama tekniğinde pek uygun görülmese de bazı durumlarda (örneğin switch deyiminde case ifadeleri arasında dolaşma) gerekebilir.

Uyarı: Goto anahtar sözcüğü ile bir döngü ve koşul bloğu içerisine dallanma işlemi gerçekleştirilemez.

Sıradaki örneğimizde goto anahtar sözcüğünün kullanımını inceleyelim;

Örnek 2-15: Goto anahtar sözcüğünün kullanımı.

Kod	Çıktı
outer: for (int i = 0; i < 10; i++) { for (int j = 0; j < 10; j++) { if (j == 1) break outer; System.out.println(" value of j = " + j); } } //	value of j = 0

2.2.4. Return Anahtar Sözcüğü

Return anahtar sözcüğü, metotlardan geriye bir değer döndürmek için kullanılır. Metotlarla ilgili ayrıntılı bilgiyi ve return anahtar sözcüğünün kullanımını bir sonraki Alt Programlar Modülü içerisinde inceleyeceğiz.

3. DİZİLER

Değişkenleri öğrenirken gördük ki her değişkene sadece bir değer atayabiliriz. Bazı durumlarda aynı tipteki değişkenleri bir arada tutma ihtiyacı duyabiliriz. Javada bize aynı tipteki değişkenleri tek bir adla saklayabileceğimiz dizileri (Array) sunmaktadır.

Dizi (array), ortak isimle anılan aynı tipteki veriler topluluğudur.

Diziler bir programlama dilindeki en önemli veri yapılarından biridir. Bu modül içerisinde dizi oluşturma, diziye değer girme, diziyi yazdırma, dizilerde arama, dizilerde sıralama, dinamik diziler yapmayı öğreneceğiz.

Bir dizi, aynı tipe ait bir miktar eleman içeren bir veri yapısıdır. Dizileri hep bir arada yer alan değişkenler listesi gibi düşünebiliriz. Örneğin 5 tane sebze ismini tek bir liste içerisinde tutmak istersek bir dizi kullanabiliriz.

3.1. Dizi Oluşturma

Bir dizi, boş parantezler ve bir değişken ismi tarafından takip edilen dizi içindeki elemanların tipini tanımlayarak bildirilir;

- **Tanımlanması:**

1.Yol:

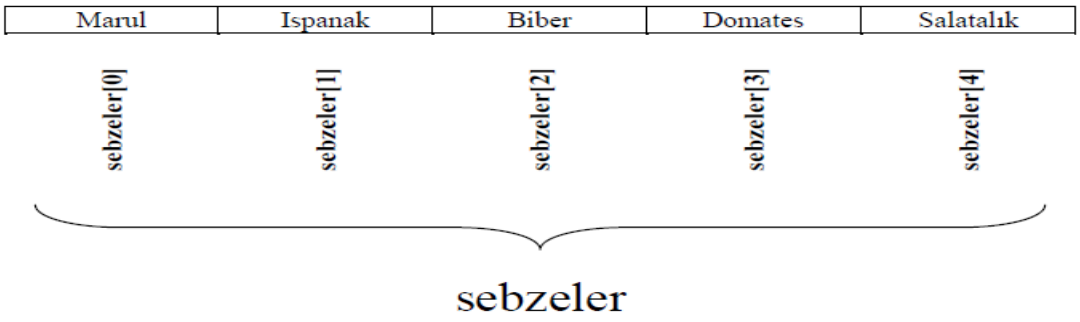
```
tip[] dizi-ismi=new tip[eleman-sayısı];
```

Burada tip, dizinin temel veri türünü belirlemek için kullanılır. Temel veri türü, dizi içerisinde saklanacak olan verinin türünü belirler. Tip ifadesinden hemen sonra köşeli parantezler ([ ]) geldiğine dikkat edin. Köşeli parantezler, burada tek boyutlu bir dizinin tanımlandığını belirtirler. Eleman-Sayısı ile de dizinin içerisine ne kadar eleman tutulacağını belirtilir.

Örneğin, metinsel ifade (string) türde elemanlarını içeren bir dizi aşağıdaki gibi bildirilir;

```
String[] sebzeler=new String[5];
```

Yukarıda tanımlaması gerçekleştirilen *sebzeler* isimli dizi içerisinde *5* adet S*tring* türünde veri tutabiliriz. Oluşturmuş olduğumuz dizi kavramsal olarak şu şekilde görünür;



Resim 0-1. sebzeler Dizisinin Kavramsal Görünüşü

Görmüş olduğumuz şekilde dizi tanımlaması gerçekleştirilebileceği gibi, aşağıdaki gibi de dizi tanımlama işlemi gerçekleştirebilir.

2.Yol:

```
int[] notlar;
rakamlar=new int[10];
```

Eğer tam sayı (integer) türünde eleman içeren bir dizi tanımlamak istiyorsak yukarıdaki yolu da izleyebiliriz.

Yukarıdaki tanımlamada da *rakamlar* isimli dizi içerisinde *10* adet *int* türünde veri tutabiliriz.

Dizi tanımlama işlemlerinde üçüncü bir yol da dizinin ilk değerlerinin küme parantezleri ({}) içerisinde dizinin istenilen boyutu kadar başlangıçta belirtilmesiyle tanımlanmasıdır;

3.Yol:

```
int[] notlar={65,76,85};
```

Bu tanımlama yöntemiyle tek boyutlu *3* elemandan oluşan *int* türünde bir dizi tanımlamış olduk.

Dikkat ederseniz yukarıdaki dizinin tanımlanması esnasında herhangi bir boyut (eleman sayısı) belirtilmedi. Bu durumlarda ilk anda kaç adet eleman girişi yapıldıysa dizinin boyutu da o kadar olur.

İstenirse aşağıdaki gibi dizinin boyutu belirtilerek de tanımlama gerçekleştirilebilir;

```
int[] notlar=new int[3]{65,76,85};
```

3.2. Diziye Değer Girme

Bir dizi tanımlandıktan sonra sıra o diziye değer girmeye gelir. Bir diziye değer girişleri tanımlama esnasında yapılabildiği gibi, programın akışı esnasında da gerçekleştirilebilir.

Dizi tanımlandıktan sonra, dizinin her bir elemanı için indeks değerleriyle elemana erişerek değer ataması şu şekildedir;

```
int[] notlar=new int[3];
notlar[0]=65;
notlar[1]=76;
notlar[2]=85;
```

Bir diğer yöntemde bir önceki konuda gördüğümüz, dizi oluşturulurken değer girmeyi tekrar hatırlarsak;

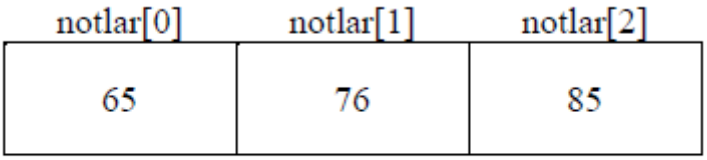
```
int[] notlar=new int[3]{65,76,85};
```

veya

```
int[] notlar= {65,76,85};
```

şeklinde tanımlama esnasında değer girişi yapabiliriz.

Yukarıda her iki örnekte de verilen notlar dizisinin kavramsal gösterimi şu şekildedir;



Resim 0-2. notlar Dizinin Kavramsal Görünüşü

Char (karakter) türündeki bir dizinin ilk kullanımına hazırlanması da şu şekillerde gerçekleştirilir;

```
char[] harfler = new char[]{'r','T','h','Y'};
```

veya

```
char[] harfler=new char[4];
harfler[0] = 'r';
harfler[1] = 'T';
harfler[2] = 'h';
harfler[3] = 'Y';
```

String (metinsel) türdeki bir dizinin ilk kullanımına hazırlanması da şu şekillerde gerçekleştirilir;

```
String[] sebzeler = new String[] { "Marul", "Ispanak", "Biber",
"Domates", "Salatalık" };
```

veya

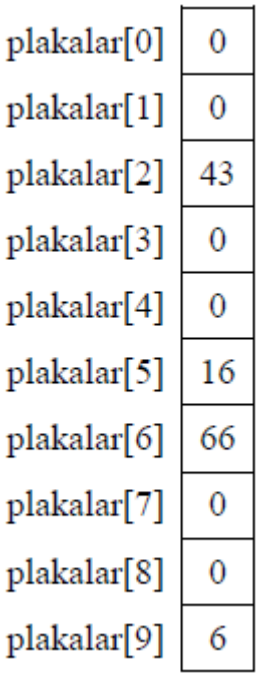
```
String[] sebzeler=new String[5];
sebzeler[0] = "Marul";
sebzeler[1] = "Ispanak";
sebzeler[2] = "Biber";
sebzeler[3] = "Domates";
sebzeler[4] = "Salatalık";
```

Bir dizi içerisindeki elemanlara tek tek dizi indeksi yardımıyla erişilebilir. Dizi indeksi (array index), bir elemanın dizi içerisindeki konumunu ifade eder. Genellikle programlama dillerinde dizilerin ilk elemanının indeksi sıfır (0)'dır. Örneğin 10 elemanlı bir dizi varsa, bu dizinin indeks numaraları 0-9 arasındadır.

Dizinin tüm elemanlarına değil de bir kısmına değer girişi yapmamız isteniyorsa, ilgili değerin barındırılacağı indeksine değer atama işlemi gerçekleştirilir.

```
int[] plakalar=new int[10];
plakalar[2] = 43;
plakalar[5] = 16;
plakalar[6] = 66;
plakalar[9] = 6;
```

Yukarıda tanımlanan plakalar isimli dizinin 2,5,6 ve 9 numaralı indeks konumlarına değer ataması gerçekleştirilmiş ve kavramsal görüntüsü şu şekilde oluşmuştur.



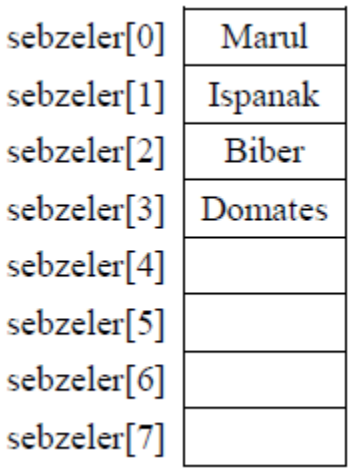
Başlangıçta eleman sayısı belli fakat değerleri daha sonra girilecekse tanımlama şu şekilde yapılabilir. Örneğin sebzeler dizisinin eleman sayısı 5 değil de 8 olsun, başlangıçta da 4 adet değer girilecek olsun;

```
String[] sebzeler = new String[] { "Marul", "Ispanak", "Biber",
"Domates","" ,"","","","" };
```

veya

```
String[] sebzeler = new String[] { "Marul", "Ispanak", "Biber",
"Domates",null, null, null, null, null };
```

Yukarıda tanımlaması gerçekleştirilen sebzeler isimli dizinin kavramsal görüntüsü de şu şekildedir;



Dikkat ederseniz int türündeki dizilerde boş kalan dizi hücrelerine sıfır(0) değeri, String türündeki dizilerde de boş kalan hücrelere boş (null-“”) değer yüklenmektedir.

Örnek 3-1: Rakamlar isimli dizi içerisine 0-9 arası rakamları tersten bir döngü yardımıyla yükleyiniz.

Kod	Çıktı
public static void main(String[] args) { int[] rakamlar = new int[10]; int i; for (i = 0; i <= 9; i++) rakamlar[i] = 9 - i; }	

Dizilerle çalışırken dikkat etmemiz gereken noktalardan en önemlisi dizi sınırlarına sadık kalmaktır. Eğer 10 elemanlı bir dizi tanımlamışsak ve bu dizi tanımlanırken belirlenen eleman sayısından fazla sayıda eleman değeri atamaya çalışırsak hata alırız.

Örnek 3-2: 10 elemanlı ciftSayilar isimli bir dizi tanımlayınız. İçerisine 0-25 arasındaki çift sayıları ekleyen kodu yazınız.

Kod	Çıktı
int[] ciftSayilar = new int[10]; int i,j; j = 0; for (i = 0; i <= 25; i = i + 2) { ciftSayilar[j] = i; j++; }	Exception in thread "main" ja- va.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 10 at com.company.Main.main(Main.java:15)

Örnek3-2'de verilen kodlar incelendiğinde ciftSayilar isimli dizinin başlangıçta 10 elemanlı bir dizi olarak tanımlandığı görülmektedir. Ancak 0-25 arasındaki çift sayılar bu diziye bir for döngüsü yardımıyla (i değişkenin değeri olarak) eklenirken, dizinin 10. elemanı eklendikten sonra (j değişkeninin değerinin 10 olmasından sonra) döngünün devam etmesi sebebiyle 11. eleman eklemeye çalışılınca "Dizi Sınırlarının Dışı" hata mesajı ile karşılaşırız.

Dizi eleman sayısının üst limitini aşmak gibi alt sınır değerinin altına girilmeye çalışılması da hata mesajı almamıza sebebiyet verir.

Kod	Çıktı
int[] ciftSayilar = new int[10]; int i,j; j = 5; for (i = 0; i <= 25; i = i + 2) { ciftSayilar[j] = i; j--; }	Exception in thread "main" ja- va.lang.ArrayIndexOutOfBoundsException: Index -1 out of bounds for length 10 at com.company.Main.main(Main.java:15)

3.3. Diziyi Yazdırma

Bir dizinin elemanlarına indeks numaraları vasıtasıyla erişebileceğimizi daha önce de bahsetmiştik. Erişilen bu elemanlarla ilgili işlemlerden birisi de ekrana yazdırma işlemidir. Erişilen değerlerinin ekrana yazdırılması işlemi şu şekilde gerçekleştirilir;

```
int[] plakalar=new int[10];
plakalar[2] = 43;
plakalar[5] = 16;
plakalar[6] = 66;
plakalar[9] = 6;
```

Aşağıdaki işlemler yukarıda tanımlanmış olan diziye göre gerçekleştirilmektedir.

```
System.out.println(plakalar[2]);
System.out.println(plakalar[5]);
System.out.println(plakalar[6]);
System.out.println(plakalar[9]);
```

Yukarıdaki kod parçası çalıştırıldığında ekrana

```
43
16
66
6
```

değerleri yazılır.

Bir dizi içerisindeki tüm değerleri ekrana yazdırmak istiyorsak döngü kullanmak gayet mantıklı olacaktır. Örneğin 200 elemanlı bir dizinin tüm elemanlarını ekrana yazdırmak istersek alt alta 200 satır kod yazmamız mümkün değildir.

Örnek 3-4: Plakalar isimli dizi içerisinde bulunan bütün elemanları ekrana yazdıran programın kodunu yazınız.

Kod	Çıktı
public static void main(String[] args) { int[] plakalar = new int[10]; int sayac = 0; plakalar[2] = 43; plakalar[5] = 16; plakalar[6] = 66; plakalar[9] = 6; for (int note : plakalar) { System.out.println("plakalar["+sayac+"] :"+note); sayac++; } }	plakalar[0] :0 plakalar[1] :0 plakalar[2] :43 plakalar[3] :0 plakalar[4] :0 plakalar[5] :16 plakalar[6] :66 plakalar[7] :0 plakalar[8] :0 plakalar[9] :6

Örnek 3-5: Örnek3-4'teki plakalar dizisini bir de for döngüsüyle ekrana yazalım.

Kod	Çıktı
int[] plakalar = new int[10]; int sayac = 0; plakalar[2] = 43; plakalar[5] = 16; plakalar[6] = 66; plakalar[9] = 6; for(sayac=0;sayac<10;sayac++) { System.out.println("plakalar["+sayac+"] :"+plakalar[sayac]); }	plakalar[0] :0 plakalar[1] :0 plakalar[2] :43 plakalar[3] :0 plakalar[4] :0 plakalar[5] :16 plakalar[6] :66 plakalar[7] :0 plakalar[8] :0 plakalar[9] :6

Yukarıdaki kod parçası çalıştırıldığı zaman karşımıza bir önceki ekran görüntüsünün aynısı karşımıza çıkar.

Ancak, burada dikkat etmemiz gereken husus for döngüsünün bitiş değerini dizimizin eleman sayısını bildiğimiz için buna göre belirledik.

3.4. Bazı Dizi Özellikleri ve Metotları

Diziler, Array sınıfı ile temsil edilir. Tüm diziler Array sınıfında tanımlı özellikleri ve metotları kullanırlar. Bu metotlardan ve özelliklerden en sık kullanılanları şunlardır;

- length,
- remove
- reverse

3.4.1. length

Dizinin saklayabileceği toplam eleman sayısını veren ve int türünde bir değer veren özelliktir.

- **Kullanımı:**

```
dizi-adi.length;
```

Örnek 3-6: ciftSayilar isimli dizinin içerisinde kaç adet eleman olduğunu ekrana yazan programın kodunu yazınız.

Kod	Çıktı
int[] ciftSayilar = new int[10]; int elemanSayisi = **ciftSayilar.length;** System.out.printf("ciftSayilar dizi içerisinde toplam %d eleman bulunmaktadır.",elemanSayisi);	ciftSayilar dizi içerisinde toplam 10 eleman bulunmaktadır.

3.4.2. remove

Parametre olarak verilen dizinin, belirtilen indeks aralığındaki elemanları temizler.

Kod	Çıktı
public static void main(String[] args) { ArrayList<Integer> arrayList = new ArrayList<Integer>(); arrayList.add(43); arrayList.add(16); arrayList.add(66); arrayList.add(26); System.out.println("Temizlenmeden önce dizinin elemanları"); System.out.println(arrayList); **arrayList.remove(1);** **arrayList.remove(2);** System.out.println("Temizlendikten sonra dizinin elemanları"); System.out.println(arrayList); }	Temizlenmeden önce dizinin elemanları [43, 16, 66, 26] Temizlendikten sonra dizinin elemanları [43, 66]

Örnek 3-8: Clear metodunu bir de string bir dizi üzerinde deneyip sonuçlarını inceleyelim.

Kod	Çıktı
import java.util.ArrayList; public class Main { public static void main(String[] args) { ArrayList<String> arrayList = new ArrayList<String>(); arrayList.add(new String("rtyucel")); arrayList.add(new String("moymul")); arrayList.add(new String("tavşanlı")); arrayList.add(new String("kütahya")); System.out.println("Temizlenmeden önce dizinin elemanları"); System.out.println(arrayList); **arrayList.remove(2);** **arrayList.remove(2);** System.out.println("Temizlendikten sonra dizinin elemanları"); System.out.println(arrayList); } }	Temizlenmeden önce dizinin elemanları [rtyucel, moymul, tavşanlı, kütahya] Temizlendikten sonra dizinin elemanları [rtyucel, moymul]

Yukarıdaki örnekten String olarak girilen verilerden baştan 2. index numarasına sahip olan "tavşanlı" verisi silinmekte, sonrasında ise "kütahya" verisi silinmeden önce 3. index numarasına sahip iken "tavşanlı" verisi silinince 2. index numarasına düşmekte ve tekrar index 2 numarasını yazarsak sondan iki tane veri silinmiş olacaktır.

3.4.3. Collections.reverse(list)

Örnek 3-10: Alfabe adlı dizi içerisine girilen A-Z'ye harfleri tersten ekrana yazdıran programın kodunu yazınız.

Kod	Çıktı
import java.util.*; public class Main { public static void main(String[] args) { List<Character> alfabe=new ArrayList<>(); int i=0; System.out.println("A'dan Z'ye İngiliz Alfabesi"); System.out.println("--------------------------"); for (char harf = 'A'; harf <= 'Z'; harf++) { alfabe.add(harf);	A'dan Z'ye İngiliz Alfabesi -------------------------- A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Z'den A'ya İngiliz Alfabesi -------------------------- Z Y X W V U T S R Q P O N M L K J I H G F E D C B A

```
            System.out.print(alfabe.get(i) + " ");
            i++;
        }
        Collections.reverse(alfabe);
        System.out.println("\n");
        System.out.println("Z'den A'ya İngiliz Alfabesi");
        System.out.println("--------------------------");
        for (i = 0; i < 26; i++)
            System.out.print(alfabe.get(i) + " ");
        }
    }
```

3.4.4. sort(dizi)

Parametre olarak verilen dizinin elemanlarını küçükten büyüğe sıralar. Eğer dizi numerik ise rakamların büyüklüğüne göre, yazı tiplerinde ise baş harflerine göre sıralanır.

reverse'ün tersidir.

- **Kullanımı:**

```
Arrays.sort(dizi);
```

Örnek 3-11: Klavyeden girilen 5 sayıyı küçükten büyüğe sıralayan programı yazınız.

Kod	Çıktı
public static void main(String[] args) { int[] sayilar = new int[5]; Scanner oku=new Scanner(System.in); int i=0; for (i = 0; i < 5; i++) { System.out.print(i + 1 + ". Sayıyı Giriniz :"); sayilar[i] =oku.nextInt(); } Arrays.sort(sayilar); System.out.println(""); System.out.println("Sıralanmış Halleri: "); for (int sayi:sayilar) { System.out.print(sayi+" "); } }	1. Sayıyı Giriniz :5 2. Sayıyı Giriniz :98 3. Sayıyı Giriniz :-9 4. Sayıyı Giriniz :41 5. Sayıyı Giriniz :69 Sıralanmış Halleri: -9 5 41 69 98

Bu örnekte aynı zamanda foreach döngüsünün kullanımını da görmüş olduk. Bundan sonra dizilerle ilgili örneklerimizde foreach döngüsünü sıkça kullanacağız.

Yukarıdaki kod parçası çalıştırıldığı zaman yukarıdaki gibi bir ekran çıktısıyla karşılaşırız;

3.4.5. binarySearch(Dizi,arananDeger)

İlk parametrede verilen dizide, ikinci parametrede verilen değeri arar. Aranan değer dizide bulunursa bulunan elemanın indeks değeri, bulunamazsa - değer döndürür.

Kod	Çıktı
public static void main(String[] args) { String[] iller = new String[] { "Ankara", "İstanbul", "Kütahya", "İzmir", "Yozgat" }; String aranan = "Kütahya"; int i=0; for (String il:iller) { System.out.printf("iller[%d]: %s \n" ,i, il); i++; } int index=Arrays.binarySearch(iller,aranan); System.out.printf("İller dizisi içerisinde Kütahya'nın\n" + "indeksi: "+index); }	iller[0]: Ankara iller[1]: İstanbul iller[2]: Kütahya iller[3]: İzmir iller[4]: Yozgat İller dizisi içerisinde Kütah- ya'nın indeksi: 2

Uyarı: Eğer `aranan="KÜTAHYA"` veya `aranan="kütahya"` yazarsanız `Array.IndexOf(iller,aranan)` ifadesi geriye -4 değerini döndürür

Örnek 3-13: 1-49 arasında 6 adet rastgele sayı üreten bir Sayısal Loto Programı hazırlayınız.

Kod	Çıktı
public static void main(String[] args) { *// Boyutu 6 olan int array'i tanımlayın.* **int[] sayilar = new int[6];** *//Random tipinden bir değişken oluşturun.* **Random r = new Random();** *//int tipinden bir değişken oluşturun ve ilk değerini 0 olarak atayın.* **int counter = 0;** *//Bir while döngüsü tanımlayın ve koşul olarak counter<6 olarak belirtin.* **while (counter < 6)** **{** */* int tipinden bir değişken oluşturun ve değerini Random değişkenin 1 ile 50 arası ürettiği tamsayıya eşitleyin. */* **int sayi = r.nextInt(50)+1;** */*Sayi adlı değişkenin değerinin sayilar adlı dizide var olup olmadığını Array.IndexOf metodu ile kontrol edin. */* **if (Arrays.binarySearch(sayilar, sayi) < 0)** **{** */*Eğer sayi adlı değişkenin değeri sayilar dizisinde yoksa dizinin counter numaralı öğesine sayi değerini eşitleyin. */* **sayilar[counter] = sayi;** *//counter'ı 1 arttırın.* **counter++;**	Bu haftasın şanslı sayıları: 3 7 16 17 24 36

```
            }
        }
        Arrays.sort(sayilar);
        //Sayilar dizisini ekrana yazdırın
        System.out.print("Bu haftasın şanslı sayıları: ");
        for (int i:sayilar) {
           System.out.print(i+" ");
        }

    }
```

Uyarı: Random() komutu sayesinde program her çalıştırıldığında farklı sayılar üreteceğinden yukarıdaki gibi sayıların sizin programınızda da çıkma olasılığı oldukça düşüktür.

3.5. Dinamik Diziler

Şimdiye kadar gördüğümüz klasik dizilerin programlama tekniklerine getirdikleri kolaylıkların dışında birtakım kısıtlamaları da vardır. Bu kısıtlamaların en başında da dizilerin boyutları gelmektedir. Bir dizinin boyutu, dizi tanımlanırken belirlenir ve programın akışı esnasında genişletilip-daraltılamazdı.

Bir diğer kısıtlama da; örneğin, programın başlangıcında 250 elemanlı bir dizi tanımladık ve bunun yalnızca 120'sini kullandık, geriye kalan 130 elemanlık bellek alanı ise boşu boşuna bellekte yer kaplamış olur.

İşte dizilerde sıkça karşılaşılan bu kısıtlamalar ArrayList sınıfı ile çözümlenir. ArrayList, büyüklüğü, dinamik olarak artıp azalabilen nesne referanslarından oluşan değişken uzunlukta biz dizidir. Bu veri yapısı Java sınıf kütüphanesinin koleksiyon framworkun bir parçasıdır ve java.util paketi altında bulunur.

ArrayList yapısının, bu dinamik boyut dışında bizlere sunduğu bir diğer avantaj da bir dizi içerisinde saklanacak olan verilerin tür sınırlamasını ortadan kaldırmasıdır. Örneğin bir dizi içerisinde hem int türünden veriler, hem string türünden veriler, hem char türünden veriler hem de bool türünden veriler saklamak mümkündür.

ArrayList ile dinamik bir dizi şu şekilde tanımlanır;

- **Tanımlanması:**

```
ArrayList diziAdi=new ArrayList();
```

ArrayList'leri örneklerimizde kullanmadan önce sizlere ArrayList'ler ile sıkça kullandığımız bazı metotlardan ve özelliklerden bahsetmemizde fayda olacaktır.

3.5.1. size Özelliği:

ArrayList içerisinde bulunan eleman sayısını int türünde veren özelliktir.

- **Kullanımı:**

```
int elemanSayisi=diziAdi.size();
```

3.5.2. add Metodu:

Bir nesneyi ArrayList'in sonuna eklemeye yarar.

- **Kullanımı:**

```
ArrayList liste=new ArrayList();
liste.add(123); //int türünde değer ekleme
liste.add("Tevfik"); //string türünde değer ekleme
liste.add('H'); //char türünde değer ekleme
liste.add(true); //bool türünde değer ekleme
```

liste.add(3.14d); //double türünde değer ekleme
liste.add(3.666f); //float türünde değer ekleme

Örnek 3-14: 0-50 arasında 3'e kalansız bölünebilen sayıları ArrayList içerisine ekleyen programın kodunu yazınız.

Kod	Çıktı	
public static void main(String[] args) { ArrayList liste=new ArrayList(); for (int sayi = 0; sayi < 100; sayi++) { if (sayi % 3 == 0) { liste.add(sayi); System.out.println(sayi+" listeye eklendi."); } } }	0 listeye eklendi. 3 listeye eklendi. 6 listeye eklendi. 9 listeye eklendi. 12 listeye eklendi. 15 listeye eklendi. 18 listeye eklendi. 21 listeye eklendi. 24 listeye eklendi. 27 listeye eklendi. 30 listeye eklendi. 33 listeye eklendi. 36 listeye eklendi. 39 listeye eklendi. 42 listeye eklendi. 45 listeye eklendi. 48 listeye eklendi.	51 listeye eklendi. 54 listeye eklendi. 57 listeye eklendi. 60 listeye eklendi. 63 listeye eklendi. 66 listeye eklendi. 69 listeye eklendi. 72 listeye eklendi. 75 listeye eklendi. 78 listeye eklendi. 81 listeye eklendi. 84 listeye eklendi. 87 listeye eklendi. 90 listeye eklendi. 93 listeye eklendi. 96 listeye eklendi. 99 listeye eklendi.

3.5.4. liste.add(index,eleman) Metodu:

Parametre olarak belirtilen indeks değerine yine parametre olarak verilen nesneyi ekler. Ekleme işleminden önce o indeksteki ve o indeksten sonraki tüm değerler birer sonraki indekslere kaydırılır.

- **Kullanımı:**

```
liste.add(5,123); //5 nolu indekse 123 değerini ekler
```

Örnek 3-15: Oluşturacağınız bir ArrayList'e 5 adet nesne ekleyiniz. Daha sonra **liste.add(index,eleman)** metodunu kullanarak 10. indekse başka bir nesne eklemeye çalışınız. Aldığınız ekran çıktısını arkadaşlarınızla paylaşınız.

Kod	Çıktı
public static void main(String[] args) { ArrayList liste=new ArrayList(); liste.add(123); //int türünde değer ekleme liste.add("Tevfik"); //string türünde değer ekleme liste.add('H'); //char türünde değer ekleme liste.add(true); //bool türünde değer ekleme liste.add(3.14d); //double türünde değer ekleme liste.add(10,3.666f); }	Exception in thread "main" java.lang.IndexOutOfBoundsException: Index: 10, Size: 5 at java.base/java.util.ArrayList.rangeCheckForAdd(ArrayList.java:787) at java.base/java.util.ArrayList.add(ArrayList.java:512) at com.company.Main.main(Main.java:14)

Örnek 3-16: 0'dan 9'a kadar rakamları barındıran bir ArrayList'in aşağıda verilen değerleri sırasıyla 5.indeksine ekleyen kodu yazınız.

Kod	Çıktı
public static void main(String[] args) { ArrayList liste = new ArrayList(); for (int i = 0; i < 10; i++) liste.add(i); System.out.println("İndexli eleman ekleme işleminden önce liste:"); for (Object obj:liste) { System.out.println(obj); } liste.add(5, 123); liste.add(5, "Tevfik"); liste.add(5, 'H'); liste.add(5, true); liste.add(5, 3.14d); liste.add(5, 3.666f); System.out.println("İndexli eleman ekleme işleminden sonra liste:"); for (Object obj:liste) { System.out.println(obj); } }	İndexli eleman ekleme işleminden önce liste: 0 1 2 3 4 5 6 7 8 9 İndexli eleman ekleme işleminden sonra liste: 0 1 2 3 4 3.666 3.14 true H Tevfik 123 5 6 7 8 9

3.5.5. remove Metodu:

Parametre olarak verilen indeks konumundaki elemanı siler.

- **Kullanımı:**

```
liste.remove(indeks); //indeks olarak verilen konumda bulunan elemanı siler.
```

Örnek 3-18: Aşağıda verilen değerleri sırasıyla bir ArrayList'e ekledikten sonra 2. Ve 4. indeksteki elemanları silen kodu yazınız. Yani index 0'dan başlarsa 2. index 'H' olur ve 4.index ise 3.14 olur.

123	Tevfik	H
true	3.14d	3.666f

Aşağıdaki kodda 2. index silindikten sonra 4 nolu index otomatik olarak bir aşağıya kayacaktır. Böylece 3 nolu index silindikten sonra istenilen sonuca erişilecektir.

Kod	Çıktı
import java.util.*; public class Main { public static void main(String[] args) { ArrayList liste = new ArrayList(); liste.add(123); liste.add("Tevfik"); liste.add('H'); liste.add(true); liste.add(3.14d); liste.add(3.666f); System.out.println("Remove işleminden önce liste:"); for (Object obj:liste) { System.out.println(obj); } liste.remove(2); liste.remove(3); System.out.println("Remove işleminden sonra liste:"); for (Object obj:liste) { System.out.println(obj); } } }	Remove işleminden önce liste: 123 Tevfik H true 3.14 3.666 Remove işleminden sonra liste: 123 Tevfik true 3.666

3.5.7. Sort Metodu:

ArrayList içerisindeki elemanları küçükten büyüğe sıralar.

- **Kullanımı:**

```
liste.Sort();
```

Kod	Çıktı
import java.util.*; public class Main { public static void main(String[] args) { List<String> liste = new ArrayList(); Scanner oku=new Scanner(System.in); String isim; for (int i = 1; i <= 5; i++) { System.out.print(i+". ismi giriniz: "); isim = oku.next(); liste.add(isim); } System.out.println("Sıralamadan önce liste:"); for (Object obj:liste) { System.out.println(obj); } // Sıralama Collections.sort(liste, new Comparator<String>() { @Override public int compare(String isim2, String isim1) { return isim2.compareTo(isim1); } }); System.out.println("Sıralamadan sonra liste:"); for (Object obj:liste) { System.out.println(obj); } } }	1. ismi giriniz: Yücel 2. ismi giriniz: Naz 3. ismi giriniz: Tevfik 4. ismi giriniz: Hadiye 5. ismi giriniz: Remzi Sıralamadan önce liste: Yücel Naz Tevfik Hadiye Remzi Sıralamadan sonra liste: Hadiye Naz Remzi Tevfik Yücel

1. METOTLAR

Programların hazırlanması esnasında, aynı işlemi gerçekleştiren program parçalarına programın birçok yerinde ihtiyaç duyulabilir. Bu ihtiyaçlar, metotlar yazılarak giderilir. Eğer metotlar kullanılmazsa; programda aynı kodu defalarca yazmamız gerekebilir ve program kodlarının okunması zorlaşır. Aynı zamanda kaynak kodun gereksiz uzamasına sebep olur. Bunun için programın birçok yerinde ihtiyaç duyulan ve aynı işlemleri yapan program parçaları metotlar olarak hazırlanırlar.

1.1. Metot Kavramı

Programların herhangi bir yerinde kullanılmak için belirli bir işi yerine getirmek amacıyla tasarlanmış alt programlara metot denilir. Metotlar belirli bir işi yapması için geliştirilirler.

Bir sefere mahsus yazılan bu kod parçaları programın akışı içerisinde defalarca çağrılarak kullanılabilirler.

Metotların amacı; programın yapısal olmasını sağlamak ve birbiriyle ilgili komutları veya programın bir bölümünü istenen isim altında toplamaktır. Bu şekilde programın okunması kolaylaşmakta ve yapısal bir görünüm kazanmaktadır.

Bir metot, bir veya daha fazla ifade içerebilir. İyi yazılmış bir programda, her metot yalnızca tek bir görev yürütür.

Metotlar tek başına çalışabilen yapılar değillerdir. Ancak ana program içerisinden çağrılarak çalıştırılırlar.

1.2. Metot Tanımlama

Her metodun bir ismi vardır ve program içerisinde metot çağrılırken bu isim kullanılır.

Bir metodun iş yapabilmesi için kendi çağıran metottan aldığı bilgilere parametre, kendisini çağıran fonksiyona döndürdüğü değere de metot geri dönüş değeri (return value) denir.

Metotlar genellikle şu şekilde tanımlanırlar;

Tanımlanması:

```
erişim dönüş-tipi isim(parametre-listesi)
{
//
metodun gövdesi;
}
```

Erişim: Bu metoda, programın diğer bölümlerinin nasıl erişebileceğini belirleyen bir erişim niteleyicisidir. Bunun kullanımı isteğe bağlıdır. Eğer herhangi bir erişim belirteci kullanılmazsa varsayılan olarak sınıfa özel (**private**) olarak belirlenir. Private olarak kullanıldığında yalnızca metodun yazıldığı sınıf içerisinden çağrılabilmesini öngörür. Eğer programın içerisinde bulunan diğer kodlar içerisinden de bu metot çağrılabilsin isteniyorsa, erişim belirteci **public** olarak belirtilmelidir. Nesne yönelimli programlama dillerinde metotlar, tanımlandıkları sınıf adı ile birlikte çağrılırken eğer metot, programın ana metodu (Main()) içerisinden çağrılacaksa **static** olarak tanımlanır ve sınıf adını yazmaya gerek kalmadan çağrılır.

Dönüş-tipi: Bu metodun çalıştırıldıktan sonra programda çağrıldığı noktaya döndürdüğü verinin tipini belirlediğimiz kısımdır. Eğer metot bir değer döndürmeyecekse dönüş-tipi **void** olarak belirtilmelidir.

İsim: Metodunun isminin belirtildiği kısımdır. Metodumuza isim verirken yapacağı iş ile alakalı bir isim vermek hem metodun ne işe yaradığıyla ilgili bize bilgi verecektir, hem de bizden başka aynı programı kodlayacak kimselere yol gösterecektir. Metoda isim verirken aynı değişken isimleri tanımlarken kullandığımız kuralları yine göz önünde bulundurmalıyız. Geri dönüş tiplerinin veya parametre-listesinin farklı olması durumunda aynı isme sahip birden fazla metot olabilir.

Parametre-Listesi: Virgül (,) ile ayrılmış tip ve tanımlayıcı çiftlerden oluşan bir listedir. Parametreler, metot çağrıldığında, metodun kullanması için gönderilen bilgilerdir. Eğer metot hiç parametre kullanmayacaksa parametre listesi de boş olur.

Örneklerle metot tanımlamalarını inceleyelim;

Örnek 0-1: Geri dönüş değeri ve parametre-listesi boş olan, ekrana "Merhaba Dünya" yazdıran metodu tanımlayıp program içerisinde kullanımına bir örnek veriniz.

Kod	Açıklama
static void MerhabaDunyaYazdir() { System.out.println("Merhaba Dünya"); } public static void main(String[] args) { MerhabaDunyaYazdir(); }	Merhaba Dünya (Parametresiz ve Geri Dönüş Türü Void Olan Metot Tanımlama)

Örnek 0-2: Klavyeden girilen bir tam sayının karesini bulan metodu ve bu metodun program içerisinde kullanımını gösteren programın kodunu yazınız.

Kod	Açıklama
import java.util.Scanner; public class Main { static int KareAl(int sayi) { int karesi = sayi * sayi; return karesi; } public static void main(String[] args) { Scanner scan=new Scanner(System.in); System.out.print("Bir sayı giriniz: "); int s1,sonuc; s1=scan.nextInt(); sonuc = KareAl(s1); System.out.printf("%d sayısının karesi: %d",s1,sonuc); } }	Bir sayı giriniz: 5 5 sayısının karesi: 25 (Geri Dönüş Değeri ve Parametre-Listesi Olan Metot Tanımlama)

Örnek 0-3: Klavyeden girilen bir mesajı ekrana 10 defa yazdıran metodun kodunu yazınız.

Kod	Açıklama
import java.util.Scanner; public class Main { static void MesajYaz(String msj) { for (int i = 1; i <= 10; i++) System.out.println(msj); } public static void main(String[] args) { Scanner scan=new Scanner(System.in); String mesaj; System.out.print("Mesajınızı giriniz: "); mesaj=scan.nextLine(); MesajYaz(mesaj); } }	Mesajınızı giriniz: Bir daha yaramazlık yapmayacağım Bir daha yaramazlık yapmayacağım Bir daha yaramazlık yapmayacağım Bir daha yaramazlık yapmayacağım Bir daha yaramazlık yapmayacağım Bir daha yaramazlık yapmayacağım Bir daha yaramazlık yapmayacağım Bir daha yaramazlık yapmayacağım Bir daha yaramazlık yapmayacağım Bir daha yaramazlık yapmayacağım Bir daha yaramazlık yapmayacağım (Geri Dönüş Türü Void olan Parametreli Metot Uygulaması)

Yukarıdaki kodları incelediğimiz zaman; klavyeden girilen yazı mesaj isimli değişken içerisine aktarılmakta ve daha sonra MesajYaz() isimli metota gönderilmektedir. MesajYaz() metodu ise kendisine parametre olarak verilen string türdeki mesajı ekrana 10 defa yazmaktadır.

1.3. Metotlarda Parametre Kullanımı

Parametrenin tanımını ve kullanımını daha önce metotların tanımlanması sırasında parametre listelerini oluştururken gördük.

Parametre-listeleri, tek bir türde verileri içeren bir liste olabileceği gibi, farklı türlerde de veriler içerebilen listelerdir.

Parametreler veri türünde olabileceği gibi nesneler de parametre olarak bir metoda gönderilebilirler.

Her bir parametre aralarına virgül kullanılarak birbirinden ayrılırlar. Aynı veri türüne sahip parametrelerin her biri için değişken isimlerinden önce ayrı ayrı veri türleri de yazılmak zorundadır.

Çeşitli veri türlerini parametre olarak metotlarımıza nasıl gönderdiğimizi örneklerle inceleyelim;

Örnek 0-4: Klavyeden girilerek parametre olarak gönderilen bir sayının, asal sayı olup olmadığını kontrol eden, eğer sayı asal ise true, değilse false değeri döndüren metodu yazınız.

Kod	Açıklama
import java.util.Scanner; public class Main { static boolean AsalMi(int s) { boolean durum=false; for (int i = 2; i < s / 2 + 1; i++) { if (s % 2 == 0) durum=false; else durum=true; } return durum; } public static void main(String[] args) { Scanner scan=new Scanner(System.in); int sayi = 0; boolean drm; System.out.print("Bir sayı giriniz: "); sayi=scan.nextInt(); drm=AsalMi(sayi); if (drm == true) System.out.printf("%d sayısı asaldır.",sayi); else System.out.printf("%d sayısı asal değildir.",sayi); } }	

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler	Ekran Çıktısı
16	
43	
66	
89	

Örnek 0-5: Parametre olarak gönderilen kullanıcı adı ve şifreyi kontrol eden, önceden belirlenmiş olan bir kullanıcı adı ve şifreyle karşılaştıran metodun kodlarını yazınız.

Kod

```
import java.util.Scanner;

public class Main {

   static void KullaniciKontrol(String kAdi, String psw)
   {

   if ((kAdi.equals("Admin")) || (kAdi.equals("ADMİN")) || (kAdi.equals("admin")))
      {
         if (psw.equals("123rty"))
            System.out.print("Tebrikler Kullanıcı ve Şifreniz Doğru");
         else
            System.out.print("Şifrenizi Hatalı Girdiniz");
      }
      else
      {
         System.out.print("Kullanıcı adınız hatalı.");
      }
   }
  public static void main(String[] args)
   {
      String kullaniciAdi, sifre;
      Scanner scan=new Scanner(System.in);
      System.out.print("Lütfen kullanıcı adınızı giriniz: ");
      kullaniciAdi = scan.nextLine();
      System.out.print("Lütfen şifrenizi giriniz: ");
      sifre = scan.nextLine();
      KullaniciKontrol(kullaniciAdi, sifre);
   } }
```

Yukarıdaki uygulamayı aşağıda verilen değerler için tek tek deneyiniz ve ekran çıktılarını yanlarındaki boş kısma yazınız.

Girilecek Değerler		**Ekran Çıktısı**
Kullanıcı Adı	**Şifre**	
Admin	123RTY	
Yönetici	123rty	
Admin	123rty	
admin	123rty	
ADMİN	123rty	

Örnek 0-6: Klavyeden girilen değerler arasında rastgele sayı üreten ve bu değerleri 10 elemanlı bir dizi içerisine atayan SayiUret( ) isimli bir metot yazınız. Dizinin elemanlarını ekrana yazdıran DiziYazdir( ) isimli bir metot daha yazarak elemanları ekrana yazdırınız. Daha sonra bu dizi içerisindeki en büyük sayı değerini bulan EnBuyuk( ) isimli, en küçük değeri bulan EnKucuk( ) isimli iki metot daha yazınız. EnBuyuk ve EnKucuk metotlarından dönen sayıları ekrana yazdıran programın kodlarını yazınız.

Bu kısımda rastgele sayılar üretilip parametre olarak gönderilen dizi isimli diziye değerler aktarılıyor ve dizi ana programa geri döndürülüyor;

Kod

```
static int[] SayiUret(int bas, int bit,int[] dizi)
   {
    int tutulan = 0;
      Random rnd=new Random();

      for (int i = 0; i < 10; i++)
      {

         tutulan = rnd.nextInt(bit)+bas;
         if (tutulan>bit) tutulan-=bas;
         dizi[i] = tutulan;
      }
      return dizi;
   }
```

Bu kısımda parametre olarak gönderilen dizi içerisindeki değerler ekrana yazdırılıyor;

Kod

```
static void DiziYazdir(int[] dizi1)
   {
      System.out.println("----------------");
      System.out.println("Tutulan sayılar:");

      for (int sayi:dizi1) {
         System.out.println(sayi);
      }
      System.out.println("----------------");
   }
```

Bu kısımda parametre olarak gönderilen dizi içerisindeki en büyük değer bulunup ana programa geri döndürülüyor;

Kod

```
static int EnBuyuk(int[] dizi2)
   {
      int ebs=0;//en küçük değer
      for (int s:dizi2) {
         if (s > ebs) //eğer sayı ebs'den büyükse
             ebs = s; //yeni ebs, sayının değeri olur
       }
      return ebs;
   }
```

Bu kısımda parametre olarak gönderilen dizi içerisindeki en küçük değer bulunup ana programa geri döndürülüyor;

Kod

```
static int EnKucuk(int[] dizi3)
   {
      int eks = 100;//en büyük değer
      for(int x:dizi3)
      {
         if (x < eks) //eğer sayı ebs'den küçükse
            eks = x; //yeni eks, sayının değeri olur
      }
      return eks;
   }
```

Ana programımız;

Kod

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

      int[] sayilar=new int[10];
      int baslangic, bitis;
      Scanner oku=new Scanner(System.in);
      System.out.print("Başlangıç değerini giriniz: ");
      baslangic = oku.nextInt();
      while (true){
         System.out.print("Bitiş değerini giriniz: ");
         bitis = oku.nextInt();
         if (bitis <= baslangic)
         {
            System.out.printf("Bitiş değeri başlangıçtan (%d) küçük ya da eşit olamaz tekrar deneyiniz.%n",baslangic);

         }else {
            break;
         }
      }
      //Rastgele sayılar üretilip diziye aktarılıyor
      sayilar = SayiUret(baslangic, bitis, sayilar);
      //Dizi ekrana yazdırılıyor
      DiziYazdir(sayilar);
      //En büyük değer bulunuyor
      int mak = EnBuyuk(sayilar);
      //En küçük değer bulunuyor
      int min = EnKucuk(sayilar);
      //Sonuçlar ekrana yazdırılıyor
      System.out.print("En büyük sayı: " + mak+"\n");
      System.out.print("En küçük sayı: " + min);

   }
```

1.4. Metotlarla ilgili Önemli Özellikler

Metotlarla ilgili bilinmesi gereken bazı önemli özellikler şunlardır;

• Metotlara isim verilirken aynı değişkenlere isim verirken uyduğumuz kurallara uymamız gerekir. main( ) ismi programımızın çalışmasını başlatan ana metodun ismi olduğu için bu ismi metot ismi olarak veremeyiz.
• Aynı isime sahip farklı geri dönüş tiplerine veya farklı parametre-listesine sahip metotlar oluşturabiliriz.

Kod

```
static int Topla(int sayi1, int sayi2, int sayi3)
   {
      int toplam;
      toplam = sayi1 + sayi2 + sayi3;
      return toplam;
   }
   static int Topla(int sayi1, int sayi2)
   {
      int toplam;
      toplam = sayi1 + sayi2;
      return toplam;
   }
   static void Topla(int sayi1)
   {
      System.out.printf("Parametresiz metodun sonucu= %d", sayi1);
   }

   public static void main(String[] args)
   {

      int sonuc, s1, s2, s3;
      s1 = 43;
      s2 = 16;
      s3 = 66;
      sonuc = Topla(s1, s2, s3);
      System.out.printf("3 parametreli metodun sonucu= %d %n",sonuc);
      sonuc = Topla(s1, s2);
      System.out.printf("2 parametreli metodun sonucu= %d %n",sonuc);
      Topla(s1);
    }
```

Bu yöntem pek de tavsiye edilen bir yöntem değildir. Bu şekilde aynı isme sahip farklı metotlar oluştururken çok dikkatli olmalıyız.

• Metotlar çağrılırken, başlangıçta belirlenen parametre sayısından ne az ne de çok sayıda parametre girmeliyiz. Eğer metodumuz 2 parametre ile işlem yapıyorsa, biz bu metoda 1 veya 3 adet parametre gönderemeyiz. Aksi takdirde hata mesajı alırız.

Kod

```
static int Topla(int sayi1, int sayi2, int sayi3)
   {
      int toplam;
      toplam = sayi1 + sayi2 + sayi3;
      return toplam;
   }


   public static void main(String[] args)
   {

      int sonuc, s1, s2, s3;
      s1 = 43;
      s2 = 16;
      s3 = 66;
      sonuc = Topla(s1, s2, s3);
      sonuc=Topla(s1); //hata
      System.out.printf("Sonuç %d",sonuc);

   }
```

Yukarıdaki kodları incelediğimizde; Topla isimli metoda ait parametre listesinde iki adet parametre alabileceği tanımlanmış. Ancak ilk koyu renkli satırdan da göreceğimiz üzere metot çağrılırken 3 parametre gönderildiğinde veya bir sonraki satırdaki gibi tek parametre gönderildiğinde hata mesajı alırız.

- Metotların geri dönüş değerleri vardır. Geri dönüş değeri olmayacak olan metotlarda geri dönüş tipi void olarak belirtilir ve return anahtar kelimesinin bu türdeki metotlarda kullanımına izin verilmez.

Kod

```
static void Topla(int sayi1, int sayi2)
   {
      int toplam;
      toplam = sayi1 + sayi2;
      return toplam; //hata
   }
```

Yukarıdaki kodları incelediğimizde; geri dönüş tipi belirtilmeyen (void olarak tanımlanan) metottan return anahtar kelimesi kullanılarak geriye bir değer döndürülmeye çalışıldığında hata mesajı alırız.

- Geri dönüş türü void olan metotlar, herhangi bir değişken içerisine atanamazlar.

Aşağıdaki kodları incelediğimizde; geri dönüş değeri bulunmayan Topla isimli metot, koyu renkli satırdan da görüleceği üzere - sonuc isimli değişkene atama işlemi yapılmaya çalışılırsa hata mesajı alırız.

Kod

```
static void Topla(int sayi1, int sayi2)
   {
      int toplam;
      toplam = sayi1 + sayi2;
      System.out.println(toplam);
   }
   public static void main(String[] args)
   {
      int sonuc, s1, s2, s3;
      s1 = 43;
      s2 = 16;
      s3 = 66;
      sonuc=Topla(s1,s2); //hata
      System.out.printf("Sonuç %d",sonuc);
   }
```

• Metotların geri dönüş değerleri herhangi bir veri türünde olabilir. Metot içerisindeki bir değer **return** anahtar sözcüğüyle metodun çağrıldığı yere geri döndürülür. Burada metodun geri dönüş tipine uyumlu bir değişken içerisine atanmalıdır. Aksi takdirde tür uyumsuz- luğundan dolayı hata mesajı alırız.

Kod

```
static float Topla(int sayi1, int sayi2)
   {
      float toplam;
      toplam = sayi1 + sayi2;
      return toplam;
   }

   public static void main(String[] args)
   {

      int sonuc, s1, s2, s3;
      s1 = 43;
      s2 = 16;
      s3 = 66;

      sonuc=Topla(s1,s2);//hata
      System.out.printf("Sonuç %d",sonuc);

   }
```

Yukarıdaki kodları incelediğimizde; int türünde tanımlanmış olan sonuc değişkeni içerisine float türünde tanımlanmış bir metodun geri dönüşü atanmaya çalışılırsa hata mesajı alırız.

• Bir metot parametre almadan da tanımlanabilir. Bu şekilde tanımlanan bir metoda parametre gönderilmez. Parametre-listesi parantezleri boş bırakılır.

Kod

```
static void yazdir()
{
   System.out.print("Merhaba Dünya");
}

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

- Metotlar tanımlanırken oluşturulan parametre-listesindeki tüm parametreler virgül (,) ile birbirinden ayrılmalıdır. Tek bir tür yazıp virgülle değişken isimlerini ayıramayız.

Kod

```
static float Topla(int sayi1,sayi2)
{
   float toplam;
   toplam = sayi1 + sayi2;
   return toplam;
}
```

Yukarıdaki gibi bir parametre listesi tanımlamaya çalışırsak hata mesajı alırız. Her bir parametreyi virgülle ayırarak tek tek tanımlamamız gerekir.

Kod

```
static float Topla(int sayi1, int sayi2)
{
float toplam;
toplam = sayi1 + sayi2;
return toplam;
}
```

- Parametre-listesinde tanımlanan değişkenlerin isimleri, metot içerisinde tanımlanacak başka bir değişkende tekrar kullanılamaz.

Kod

```
static float Topla(int sayi1, int sayi2)
{
   float toplam;
   int sayi1,sayi2;//hata
   toplam = sayi1 + sayi2;
   return toplam;
}
```

Yukarıdaki gibi bir parametre listesinde tanımlanmış sayi1 ve sayi2 değişken isimlerini, metot içerisinde tekrardan her ne veri türünde olursa olsun yeniden kullanamayız.

- Bir metot içerisinde başka bir metot tanımlanamaz. Ancak başka bir metot çağrılabilir.

Kod

```
static void yazdir()
   {
      static void MerhabaDunyaYazdir()
      {
         System.out.print("Merhaba Dünya");
      }
   }
```

Yukarıdaki kullanım hatalı bir kullanımdır. Metot içerisinde metot tanımlaması yapılamaz. Ancak aşağıdaki örnek gibi metot içerisinden başka bir metot çağrılabilir.

Kod

```
static void MerhabaDunyaYazdir()
   {
      System.out.println("Merhaba Dünya");
   }
   static void yazdir()
   {
      MerhabaDunyaYazdir();
   }
```

Yazdir isimli metot içerisinden MerhabaDunyaYazdir isimli metodu yukarıdaki gibi çağırabiliriz.

- Metotların içerisinde tanımlanan tüm değişkenler metot dışında kullanılamazlar, geçersiz olurlar.

Kod

```
static float Topla(int sayi1, int sayi2)
   {
      int toplam;
      toplam = sayi1 + sayi2;
      return toplam;
   }
  public static void main(String[] args)
   {
      int s1, s2;
      s1 = 43;
      s2 = 16;
      toplam = Topla(s1, s2); //hata
      System.out.printf("Sonuç %d",toplam); //hata

   }
```

Yukarıdaki kodlar incelendiğinde; Topla isimli metot içerisinde tanımlanan float türündeki toplam isimli değişkene metot dışında tekrar erişilmek istenirse hata mesajı alırız.

1.5. Özyineli (Rekürsif-Recursive) Metotlar

Bir metodun kendi kendini çağırmasına yinelenme (recursion), kendi kendini çağıran metotlara da yinelenen veya özyineli (recursive) metotlar denir.

Bir metodun kendi kendini çağırması, zaman zaman da olsa program yazarken ihtiyaç duyulan bir olaydır. Yinelenen metotlar tasarlanırken çok dikkatli olunmalıdır. Aksi takdirde sonsuz bir döngü içerisine girilebilir. Bu döngünün bir şekilde sonlandırılması gerekmektedir.

Yinelenen metotlara basit bir örnekle giriş yapalım;

Örnek 0-7: 1'den klavyeden girilen sayıya (n) kadar olan sayıların toplamını hesaplayan programın kodunu yazınız.

Kod

```
static int Topla(int n)
   {
      if (n == 0)
         return 0;
      return n + Topla(n - 1);
   }

   public static void main(String[] args)
   {
      int sayi=5;
      int sonuc = Topla(sayi);
      System.out.printf("1'den %d sayısına kadar olan sayıların toplamı = %d",sayi,sonuc);
   }
```

Hemen hemen bütün bilgisayar programlama kitaplarında yinelenen metotlarla ilgili klasik olarak bir sayının faktöriyelini hesaplayan programlara yer verilir. Bizlerde bu geleneği bozmayalım ve bir önceki örnekteki mantıkla klavyeden girilen bir sayının faktöriyelini hesaplayan özyineli bir metot hazırlayalım.

Örnek 0-8: Klavyeden girilen sayının faktöriyelini hesaplayan programın kodunu yazınız.

Kod

```
static double Faktoriyel(int n)
   {
      if (n == 0)
         return 1;
      return n * Faktoriyel(n - 1);
   }
   public static void main(String[] args)
   {
      Scanner scan=new Scanner(System.in);
      System.out.print("bir sayı giriniz:");
      int sayi=scan.nextInt();
      if (sayi>0){
         double sonuc= Faktoriyel(sayi);
         System.out.printf("%d! = %f ",sayi,sonuc);
      }else{
         System.out.println("Lütfen pozitif bir sayı girin");
      }

   }
```

1.6. Main Metodu

Metotlar modülümüzün başından bu yana örneklerimizde hep main( ) isminde bir metot içerisinde ana programlarımızın yazımını gerçekleştirdik. Peki nedir bu main( ) metodu?

Aslında main( ) metodu şimdiye kadar yazdığımız veya kullandığımız metotlardan pek de farkı olmayan bir metot türüdür. Tek ve en önemli farkı main( ) metodunun ana programın başlamasını sağlayan nokta olmasıdır. İşte bu yüzden main( ) metodu diğer metotlara göre daha özeldir.

2. HAZIR METOTLAR

Programlama dili kütüphaneleri içerisinde önceden tanımlanmış ve programcıların işlerini kolaylaştıran bir takım hazır metotlar vardır.

Bu modülümüz içerisinde metinsel (String), matematiksel, tarih ve zaman işlemlerinde sıkça kullanacağımız metotlarını inceleyeceğiz.

2.1. Metinsel (String) Metotları

Programlama dili içerisindeki String sınıfı altında bulunan ve metinsel (String) ifadelerle ilgili bir takım işlemleri daha kolay yapabilmek için bir takım hazır metotlar vardır.

2.1.1. compareTo( )

Parametre olarak verilen iki string ifadeyi karşılaştırır ve geriye int türünde bir veri döndürür. Eğer dönüş değeri sıfır (0) ise iki metin birbirine eşittir. Aksi takdirde parametre olarak verilen metinleri ilk harflerinden itibaren tek tek karşılaştırır ve farklılığın olduğu ilk harflerin alfabedeki sıralarına göre - veya + sayı değerlerini döndürür.

if str1 > str2, pozitif sayı döndürür

if str1 < str2, negatif sayı döndürür

if str1 == str2, sıfır döndürür

Burada metinlerin büyüktür/küçüktür karşılaştırmaları, harflerin alfabetik sırasıyla ilgilidir.

Örnek 0-1: Klavyeden girilen iki metnin karşılaştırılmasını yapan programın kodlarını yazınız.

Kod

```
import java.util.*;

public class Main {

      public static void main(String[] args) {

         Scanner oku=new Scanner(System.in);
         System.out.print("1. metni giriniz :");
         String metin1=oku.next();
         System.out.print("2. metni giriniz :");
         String metin2=oku.next();
         int donusDegeri=metin1.compareTo(metin2);

         if (donusDegeri<0){
            System.out.printf(" %s kelimesi ve %s kelimesinden küçüktür.\n Geri dönüş değeri: %d",metin1,metin2,donusDegeri);
         }
         else if (donusDegeri>0){
            System.out.printf(" %s kelimesi ve %s kelimesinden büyüktür.\n Geri dönüş değeri: %d",metin1,metin2,donusDegeri);

         }
```

```
        else {
            System.out.printf(" %s ve %s kelimeleri birinin aynıdır.\n Geri dönüş değeri: %d",metin1,metin2,donusDegeri);
        }


    }
}
```

Yukarıdaki programı çalıştırdıktan sonra aşağıdaki tabloda verilen değerleri tek tek deneyerek String.compareTo( ) metoduyla geri dönen değerleri, tabloda boş bırakılan alanlara yazınız. Karakterlerin ascii kodlarına göre karşılaştırma yapar. A (65) ve a (97) karşılaştırıldığında fark -32 olacaktır.

Girilecek Değerler		**Ekran Çıktısı**
metin1	**metin2**	
Tavşanlı	tavşanlı	
Tavşanlı	Tavsanlı	
Tavşanlı	Tavşanlı	
Tavşanlı	Davşanlı	
Tavşanlı	Tavışanlı	

2.1.2. concat( )

Parametre olarak verilen nesneleri string türünde birbirine peşisıra ekler ve geriye string türünde bir değer döndüren String metodudur.

Örnek 0-2: Klavyeden girilen iki metni birleştiren programın kodlarını yazınız.

Kod

```
import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner oku=new Scanner(System.in);
System.out.print("1. metni giriniz :");
String metin1=oku.next();
System.out.print("2. metni giriniz :");
String metin2=oku.next();

String birlestirilen = metin1.concat(metin2);
System.out.printf(" %s ve %s kelimelerinin birleştirilmiş  hali : %s", metin1, metin2, birleştirilen );

    }
}
```

Yukarıdaki kodlar çalıştırıldığında ve metin1 olarak "Linyit", metin2 olarak da "spor" kelimeleri girildiğinde birleştirilen kelime "Linyitspor" olarak karşımıza çıkacaktır.

String.concat( ) ile farklı türlerdeki verileri de birleştirme şansımız vardır. Aşağıdaki örnekte hem metin, hem sayı, hem de boolean türündeki verileri birleştirip, geriye String türünde bir veri elde etme işlemini inceleyeceğiz.

Örnek 0-3: Metin, sayı ve boolean türünde verilerin birleştirilmesini sağlayan programın kodlarını yazınız.

Kod	Açıklama
public class Main { public static void main(String[] args) { String metin = "Kütahya"; int sayi = 43; boolean durum = false; String birlestirilen = metin**.concat**(String.valueOf(sayi))**.concat**(String.valueOf(durum)); System.out.printf("%s ve %d %b'nin birleştirilmiş hali: %s ", metin, sayi,durum,birlestirilen); } }	Kütahya ve 43 false'nin birleştirilmiş hali: Kütahya43false

2.1.3. copyValueOf()

Parametre olarak verilen String türündeki metnin bir kopyasını almaya yarayan String metodudur.

Kod	Açıklama
public class Main { public static void main(String[] args) { String metin="Tavşanlı"; String kopyaMetin=String**.copyValueOf**(metin.toCharArray()); System.out.printf("%s metninin kopyası :%s", metin,kopyaMetin); } }	Tavşanlı metninin kopyası :Tavşanlı

2.1.4. Format( )

Programlama esnasında bazı ifadeleri belirli bir biçim içerisinde yazmamız istenirse String.format( ) metodu kullanılır.

Bu metot geriye **String** türünde bir veri döndürür.

- Örneğin **metinsel ifadelerin** belirli bir biçim içerisinde ekranda yazılmasını istiyorsak;

Örnek 0-6: Sıra No, Adınız, Soyadınız şeklinde başlıkları olan ve içeriği dolu olan bir tablo hazırlayıp, ekrana yazdırınız.

Kod	Açıklama
public class Main { public static void main(String[] args) { System.out.println("------------------------------"); System.out.println("Sıra No | Adınız | Soyad"); System.out.println("------------------------------"); System.out.println(String.format("%7s | %-8s| %10s", 1,"Remzi", "ERTÜRK")); System.out.println(String.format("%7s | %-8s| %10s", 2,"Tevfik", "ULUÇ")); System.out.println(String.format("%7s | %-8s| %10s", 3,"Yücel", "CAN")); System.out.println("------------------------------"); } }	------------------------------ Sıra No | Adınız | Soyad ------------------------------ 1 | Remzi | ERTÜRK 2 | Tevfik | ULUÇ 3 | Yücel | CAN ------------------------------

· Örneğin **int türündeki sayısal ifadelerin** belirli bir biçim içerisinde ekranda yazılmasını istiyorsak;

System.out.println(String.format("%05d",15));//00015

ifadesiyle 15 sayısı ekrana başına 3 adet 0 eklenerek toplamda 5 karakter olarak yazılır.

System.out.println(String.format("%06d",-15));// -00015

ifadesiyle -15 sayısı ekrana başına 3 adet 0 eklenerek toplamda 5 karakter olarak yazılır.

System.out.println(String.format("%5d",15));// 15

ifadesiyle 15 sayısı ekrana başına 3 adet boşluk eklenerek toplamda 5 karakterlik bir alana sağa hizalı olarak yazılır.

Örneğin **tarih/saat türündeki** ifadelerin belirli bir biçim içerisinde ekranda yazılmasını istiyorsak;

Örnek 0-7: 25.02.2019 10:07:15 zamanına ait değerlerin gösterimleri şu şekilde sağlanır;

Kod
import java.text.DateFormat; import java.text.SimpleDateFormat; import java.util.Date; public class Main { public static void main(String[] args) { DateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy HH:mm:ss"); Date date = new Date(); System.out.println("Tarih :"+dateFormat.format(date));

```
        DateFormat dateFormatYil = new SimpleDateFormat("y yy yyy yyyy");
        System.out.println("Yıl Gösterimleri:\t"+dateFormatYil.format(date));

        DateFormat dateFormatAy = new SimpleDateFormat("M MM MMM MMMM");
        System.out.println("Ay Gösterimleri:\t"+dateFormatAy.format(date));

        DateFormat dateFormatGun = new SimpleDateFormat("d dd ddd dddd");
        System.out.println("Gün Gösterimleri:\t"+dateFormatGun.format(date));

        DateFormat dateFormatSaat = new SimpleDateFormat("h hh H HH");
        System.out.println("Saat Gösterimleri:\t"+dateFormatSaat.format(date));

        DateFormat dateFormatDakika = new SimpleDateFormat("m mm");
        System.out.println("Dakika Gösterimleri:\t"+dateFormatDakika.format(date));

        DateFormat dateFormatSaniye = new SimpleDateFormat("s ss");
        System.out.println("Saniye Gösterimleri:\t"+dateFormatSaniye.format(date));

        DateFormat dateFormatZamanDilimi = new SimpleDateFormat("z zz zzz zzzz");
        System.out.println("Zaman Dilimi Gösterimleri:\t"+dateFormatZamanDilimi.format(date));

    }
}
```

Açıklama

```
Tarih :25.02.2019 10:07:15
Yıl Gösterimleri:       2019 19 2019 2019
Ay Gösterimleri:        2 02 Şub Şubat
Gün Gösterimleri:       25 25 025 0025
Saat Gösterimleri:      10 10 10 10
Dakika Gösterimleri:    7 07
Saniye Gösterimleri:    15 15
Zaman Dilimi Gösterimleri:      EET EET EET Eastern European Time
```

2.1.5. isBlank() , isEmpty()

isEmpty fonksiyonu boşluk dahi kabul etmezken isBlank fonksiyonu içersine boşluk kabul eder.

Kod	Açıklama
public static void main(String[] args) { String metin=""; if (metin.isEmpty()){ System.out.println("[Empty] boş"); }if (metin.isBlank()){ System.out.println("[Blank] boş"); } }	[Empty] boş [Blank] boş

Kod	Açıklama
String metin=" "; if (metin.isEmpty()){ System.out.println("[Empty] boş"); }if (metin.isBlank()){ System.out.println("[Blank] boş"); }	[Blank] boş

2.1.6. contains( )

Birlikte çağrıldığı metinsel ifade içerisinde parametre olarak verilen char türündeki karakteri veya yine parametre olarak verilen string türündeki metinsel ifadeyi arar ve geriye boolean türünde bir değer döndürür. Metinsel ifade içerisinde string arama;

Kod	Açıklama
public static void main(String[] args) { String metin="İbrahim Önal Fen Lisesi"; String aranan= "Fen"; Boolean donusDegeri=metin.**contains**(aranan); System.out.println(donusDegeri); }	true

Örnek 0-11: Klavyeden girilen metin içerisinde rakamsal ifade olup olmadığını kontrol eden bir metot yazıp, kullanıcıyı uyaran programın kodlarını yazınız.

Kod
import java.util.Scanner; public class Main { public static void main(String[] args) { Scanner oku=new Scanner(System.in); System.out.print("Kelime giriniz: "); String ifade = oku.next(); boolean sonuc = rakamVarMi(ifade); if (sonuc) System.out.println("Girmiş olduğunuz metin rakamsal ifadeler içeriyor"); else System.out.println("Girmiş olduğunuz metin rakamsal ifadeler içermiyor"); } public static boolean rakamVarMi(String metin) { if (metin.contains("0")) return true; else if (metin.contains("1")) return true; else if (metin.contains("2"))

```
            return true;
        else if (metin.contains("3"))
            return true;
        else if (metin.contains("4"))
            return true;
        else if (metin.contains("5"))
            return true;
        else if (metin.contains("6"))
            return true;
        else if (metin.contains("7"))
            return true;
        else if (metin.contains("8"))
            return true;
        else if (metin.contains("9"))
            return true;
        else
            return false;
         }


}
```

Yukarıdaki uygulamayı rakamsal ifade içeren ve içermeyen çeşitli kelime girişleriyle deneyip pekiştiriniz.

Örnek 0-12: Klavyeden girilen metin içerisinde, yine klavyeden girilen bir metni arayan programın kodlarını yazınız.

Kod

```
import java.util.Scanner;

public class Main {

        public static void main(String[] args) {
            Scanner oku=new Scanner(System.in);
            System.out.print("Metin giriniz: ");
            String ifade = oku.nextLine();
            System.out.print("Aranan metni giriniz: ");
            String aranan = oku.next();
            if (ifade.contains(aranan))
                System.out.printf("%s ifadesi içerisinde %s kelimesi mevcuttur.", ifade, aranan);
            else
                System.out.printf("%s ifadesi içerisinde %s kelimesi yoktur.", ifade, aranan);

        }

}
```

Birden fazla kelimeden oluşan cümle girmek için **nextLine()** metodu kullanılırken bir tek kelime için de **next()** metodu kullanılır.

2.1.7. System.arraycopy()

Bu metot kaynakBaslangicIndexi (int türünde), hedefDizisi (char dizisi türünde), , kaynak-Dizisi (char dizisi türünde), hedefBaslangicIndexi (int türünde) ve miktar (int türünde) olmak üzere 5 parametre alır.

Kullanımı

System.arraycopy(array, 0, copiedArray, 0, 3);

array: Kaynak dizi,
0: Kaynak dizinin hangi index numarasından itibaren kopyalanacağı,
copiedArray: Hedef dizi,
0: Hedef dizinin hangi index numarasından itibaren kopyalanacağı,
3:Kopyalancak elemanların sayısı

Örnek 0-13: Klavyeden girilen 10 harfli bir metnin içeriğini 2.karakterinden başlayarak 5 karakterini hedefDizi isimli char türündeki 10 elemanlı bir dizinin 3.elemanından başlayarak kopyalayınız. hedefDizi isimli dizinin içeriğini kopyalamadan önce ve sonra ekrana yazdırarak değişimi gözlemleyiniz.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { Scanner oku=new Scanner(System.in); System.out.print("Metin giriniz: "); String metin = oku.next(); int i = 0; char[] kaynakDizi=metin.toCharArray(); char[] hedefDizi={'a','b','c','d','e','f','g','h','i','j'}; System.out.println("Dizinin kopyalamadan önceki içeriği"); for (char harf:hedefDizi) { System.out.printf("copiedArray[%d]: %c %n",i,harf); i++; } System.arraycopy(kaynakDizi,1,hedefDizi,2,5); i = 0; System.out.println("Dizinin kopyalamadan sonraki içeriği"); for (char harf:hedefDizi) { System.out.printf("copiedArray[%d]: %c %n",i,harf); i++; } } }	Metin giriniz: 1234567890 Dizinin kopyalamadan önceki içeriği copiedArray[0]: a copiedArray[1]: b copiedArray[2]: c copiedArray[3]: d copiedArray[4]: e copiedArray[5]: f copiedArray[6]: g copiedArray[7]: h copiedArray[8]: i copiedArray[9]: j Dizinin kopyalamadan sonraki içeriği copiedArray[0]: a copiedArray[1]: b copiedArray[2]: 2 copiedArray[3]: 3 copiedArray[4]: 4 copiedArray[5]: 5 copiedArray[6]: 6 copiedArray[7]: h copiedArray[8]: i copiedArray[9]: j

2.1.8. `endsWith()`

Birlikte çağrıldığı metinsel ifade parametre olarak verilen String türündeki ifade ile bitip bitmediğini kontrol eden metottur. Geriye bool türünde bir değer döndürür. Eğer metin parametre olarak verilen ifade ile bitiyorsa geriye **true** değerini döndürür. Eğer metin parametre olarak verilen ifade ile bitmiyorsa geriye **false** değerini döndürür.

- **Kullanımı:**

```
metin.endsWith(ifade);
```

Örnek 0-14: Klavyeden girilen metin sesli harf ile bitip bitmediğini kontrol eden ve geriye bool türünde bir değer döndüren metot tanımlayınız.

Kod	Çıktı
import java.util.Scanner; public class Main { public static void main(String[] args) { Scanner oku=new Scanner(System.in); System.out.print("Metin giriniz: "); String ifade= oku.nextLine(); if (bitisSesliMi(ifade)) System.out.println("Klavyeden girilen metin sesli harf ile bitiyor"); else System.out.println("Klavyeden girilen metin sesli harf ile bitmiyor"); } static boolean bitisSesliMi(String metin) { if (metin.endsWith("a")) return true; else if (metin.endsWith("e")) return true; else if (metin.endsWith("ı")) return true; else if (metin.endsWith("i")) return true; else if (metin.endsWith("o")) return true; else if (metin.endsWith("ö")) return true; else if (metin.endsWith("u")) return true; else if (metin.endsWith("ü")) return true; else return false; } }	Metin giriniz: İbrahim Önal Klavyeden girilen metin sesli harf ile bitmiyor

2.1.9. indexOf( )

Bu metodun birden fazla kullanım şekli vardır.

2.1.9.1. IndexOf(char)

Birlikte çağırıldığı metinsel ifade içerisinde parametre olarak verilen karakteri arar ve geriye bu karakterin metin içerisinde **ilk** bulunduğu karakter sırasını döndürür. Metnin ilk karakterinin indeks numarasının sıfır (0) olduğunu unutmayınız.

Eğer aranan karakter kelime içerisinde bulunamazsa geriye -1 değeri döndürür. Bu metot büyük/küçük harf duyarlı olduğu için aranan karakterin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.indexOf(char);
```

Örnek 0-15: IndexOf(char) metodunun kullanımı;

Kod
String metin = "Bilişim teknolojileri"; System.out.println(metin.indexOf('T')); // -1 System.out.println(metin.indexOf('e')); // 9 System.out.println(metin.indexOf('i')); // 1 System.out.println(metin.indexOf('z')); // -1

2.1.9.2. indexOf(string)

Birlikte çağırıldığı metinsel ifade içerisinde parametre olarak verilen String ifadeyi arar ve geriye bu ifadenin, metin içerisinde ilk bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.indexOf(string);
```

Örnek 0-16: indexOf(string) metodunun kullanımı;

Kod
String metin = "Bilişim teknolojileri"; System.out.println(metin.indexOf("bilişim")); // -1 System.out.println(metin.indexOf("Bilişim")); // 0 System.out.println(metin.indexOf("loji")); // 13 System.out.println(metin.indexOf("Bilisim")); //-1

2.1.9.3. indexOf(char deger,int baslangic)

Birlikte çağırıldığı metinsel ifade içerisinde, parametre olarak verilen karakteri, yine parametre olarak verilen başlangıç indeksinden başlayarak arar ve geriye bu ifadenin, metin içerisinde başlangıç indeksinden sonra ilk bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.indexOf(char deger,int baslangic);
```

Örnek 0-17: indexOf(char deger,int baslangic) metodunun kullanımı;

Kod
String metin = "Bilişim teknolojileri"; System.out.println(metin.indexOf('T',3)); // -1 System.out.println(metin.indexOf('e',10)); // 18 System.out.println(metin.indexOf('i',4)); // 5 System.out.println(metin.indexOf('o',18)); // -1

2.9.1.4. indexOf(String deger,int baslangic)

Birlikte çağırıldığı metinsel ifade içerisinde, parametre olarak verilen metinsel ifadeyi, yine parametre olarak verilen başlangıç indeksinden başlayarak arar ve geriye bu ifadenin, metin içerisinde başlangıç indeksinden sonra ilk bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.indexOf(String deger,int baslangic);
```

Örnek 0-18: indexOf(String deger, int baslangic) metodunun kullanımı

Kod
String metin = "Bilişim teknolojileri"; System.out.println(metin.indexOf("bilişim",0)); // -1 System.out.println(metin.indexOf("Bilişim",1)); // -1 System.out.println(metin.indexOf("loji",3)); // 13 System.out.println(metin.indexOf("il",2)); // 16

2.1.10. StringBuilder(metin).insert(int baslangic, String deger)

Parametre olarak verilen **int** türündeki başlangıç indeksinden başlayarak, yine parametre olarak verilen metinsel ifadeyi, çağrıldığı metnin içerisine eklemeye yarayan metottur. Geriye **string** türünde metinsel bir ifade döndürür.

- **Kullanımı:**

```
String yeniMetin = new StringBuilder(metin).insert(int baslangic,
String eklenecek).toString();
```

Örnek 0-19: StringBuilder(metin).insert(int baslangic, String deger) metodunun kullanımı;

Kod
String metin = "Elektrik teknolojileri"; String yeniMetin = new StringBuilder(metin).insert(9, "ve Elektronik ").toString(); System.out.println(yeniMetin);

Çıktı
Elektrik ve Elektronik teknolojileri

2.1.11. lastIndexOf( )

Bu metodun da indexOf metotdu gibi birden fazla kullanım şekli vardır.

2.1.11.1. LastIndexOf (char)

Birlikte çağırıldığı metinsel ifade içerisinde parametre olarak verilen karakteri arar ve geriye bu karakterin metin içerisinde **son** bulunduğu karakter sırasını döndürür. Metnin ilk karakterinin indeks numarasının sıfır (0) olduğunu unutmayınız.

Eğer aranan karakter kelime içerisinde bulunamazsa geriye -1 değeri döndürür. Bu metot büyük/küçük harf duyarlı olduğu için aranan karakterin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.lastIndexOf(char);
```

Örnek 0-20: lastIndexOf(char) metodunun kullanımı;

Kod
String metin = "Bilişim teknolojileri"; System.out.println(metin.lastIndexOf('T')); // -1 System.out.println(metin.lastIndexOf('e')); // 18 System.out.println(metin.lastIndexOf('i')); // 20 System.out.println(metin.lastIndexOf('z')); // -1

2.1.11.2. lastIndexOf (String)

Birlikte çağırıldığı metinsel ifade içerisinde parametre olarak verilen String ifadeyi arar ve geriye bu ifadenin, metin içerisinde son bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.lastIndexOf(String);
```

Örnek 0-21: lastIndexOf(String) metodunun kullanımı;

Kod
String metin = "Bilişim teknolojileri"; System.out.println(metin.lastIndexOf("bilişim")); // -1 System.out.println(metin.lastIndexOf("Bilişim")); // 0 System.out.println(metin.lastIndexOf("il")); // 16 System.out.println(metin.lastIndexOf("Bilisim")); // -1

2.1.11.3. lastIndexOf (char deger,int baslangic)

Birlikte çağırıldığı metinsel ifade içerisinde, parametre olarak verilen karakteri, yine parametre olarak verilen başlangıç indeksinden başlayarak arar ve geriye bu ifadenin, metin içerisinde başlangıç indeksinden sonra son bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

- **Kullanımı:**

```
int indeks=metin.LastIndexOf(char deger,int baslangic);
```

Kod
String metin = "Bilişim teknolojileri"; System.out.println(metin.lastIndexOf('T',3)); // -1 System.out.println(metin.lastIndexOf('e',10)); // 9 System.out.println(metin.lastIndexOf('i',4)); // 3 System.out.println(metin.lastIndexOf('o',18)); // 14

2.1.11.4. lastIndexOf(String deger,int baslangic)

Birlikte çağırıldığı metinsel ifade içerisinde, parametre olarak verilen metinsel ifadeyi, yine parametre olarak verilen başlangıç indeksinden başlayarak arar ve geriye bu ifadenin, metin içerisinde başlangıç indeksinden sonra son bulunduğu karakter sırasını döndürür.

Eğer aranan ifade metin içerisinde bulunamazsa geriye -1 değeri döndürür.

Bu metot büyük/küçük harf duyarlı olduğu için aranan ifadenin büyük/küçük olma durumlarına dikkat ediniz.

· **Kullanımı:**

```
int indeks=metin.lastIndexOf(String deger,int baslangic);
```

Örnek 0-23: lastIndexOf(String deger,int baslangic) metodunun kullanımı;

Kod
String metin = "Bilişim teknolojileri"; System.out.println(metin.lastIndexOf("bilişim",0)); // -1 System.out.println(metin.lastIndexOf("Bilişim",1)); // 0 System.out.println(metin.lastIndexOf("im",15)); // 5 System.out.println(metin.lastIndexOf("il",2)); // 1

2.1.12.1. replace (char eski, char yeni)

Birlikte çağrıldığı metin içerisinde, ilk parametredeki karakterleri, ikinci parametredeki karakter değerleriyle değiştiren metottur. Geriye değiştirme işleminin gerçekleştirildiği String türünde bir ifade döndürür.

· **Kullanımı:**

```
String yeniMetin=metin.replace(char eski,char yeni);
```

Örnek 0-30: replace(int deger,int adet) metodunun kullanımı;

Kod
String metin = "Bilişim Teknolojileri"; System.out.println("Değişimden önce: "+metin); System.out.println("Değiştirilen: "+metin.replace('i', '#'));

Çıktı
Değişimden önce: Bilişim Teknolojileri Değiştirilen: B#l#ş#m Teknoloj#ler#

2.1.12.2. replace (String eski, String yeni)

Replace metodunun bir önceki kullanımından farkı parametre olarak bu kez char türünde karakterler yeni string türünde metinsel ifade almasıdır.

· **Kullanımı:**

```
String yeniMetin=metin.replace(string eski, string yeni);
```

Örnek 0-31: `replace(string eski, string yeni)` metodunun kullanımı;

Kod
String metin = "Bilişim Teknolojileri"; System.out.println("Değişimden önce: "+metin); System.out.println("Değiştirilen: "+metin.replace("Bilişim","Metal"));

Çıktı
Değişimden önce: Bilişim Teknolojileri Değiştirilen: Metal Teknolojileri

2.1.13. split ( )

Split( ) metodu, çağrıldığı metni istenilen karakterden itibaren parçalara bölmek için kullanılan bir metottur. Eğer istenilen karakter mevcut metin ifadesi içerisinde yer alıyorsa, Split( ) metodu metni karakterlerden öncesi ve sonrası şeklinde parçalara ayırır ve bu parçaları String türünde bir dizi içerisinde saklar. Geriye de bu String[] türündeki diziyi döndürür.

- **Kullanımı:**

```
String[] dizi=metin.Split(char karakter);
```

Örnek 0-32: split(char karakter) metodunun kullanımı;

Kod
String metin = "Bilişim Teknolojileri"; System.out.println("Boşluktan bölünürse"); System.out.println("--------------------"); for (String harf:metin.split(" ")) { System.out.println(harf); } System.out.println(" "); System.out.println("i harflerinden bölünürse"); System.out.println("--------------------"); for (String harf:metin.split("i")) { System.out.println(harf); }

Çıktı
Boşluktan bölünürse -------------------- Bilişim Teknolojileri i harflerinden bölünürse -------------------- B l ş m Teknoloj ler

2.1.14. startsWith ( )

Birlikte çağrıldığı metinsel ifade parametre olarak verilen String türündeki ifade ile başlayıp başlamadığını kontrol eden metottur. Geriye bool türünde bir değer döndürür. Eğer metin parametre olarak verilen ifade ile başlıyorsa geriye **true** değerini döndürür. Eğer metin parametre olarak verilen ifade ile başlamıyorsa geriye **false** değerini döndürür.

- **Kullanımı**:

```
metin.startsWith(ifade);
```

Örnek 0-33: Klavyeden girilen kullanıcı adının rakamla başlayıp başlamadığını kontrol eden, eğer kullanıcı adı rakam ile başlıyorsa uyarı mesajı veren bir metot tanımlayınız.

Kod

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

        Scanner oku=new Scanner(System.in);
        System.out.print("Kullanıcı adı belirleyiniz: ");
        String kAdi= oku.nextLine();
        if (KullaniciAdiKontrol(kAdi))
           System.out.println("Kullanıcı adı tanımınız başarılı");
        else
           System.out.println("Kullanıcı adı sayı ile başlayamaz");

   }
   static boolean KullaniciAdiKontrol(String kAdi)
   {
      if (kAdi.startsWith("1"))
         return false;
      else if (kAdi.startsWith("2"))
         return false;
      else if (kAdi.startsWith("3"))
         return false;
      else if (kAdi.startsWith("4"))
         return false;
      else if (kAdi.startsWith("5"))
         return false;
      else if (kAdi.startsWith("6"))
         return false;
      else if (kAdi.startsWith("7"))
         return false;
      else if (kAdi.startsWith("8"))
         return false;
      else if (kAdi.startsWith("9"))
         return false;
      else if (kAdi.startsWith("0"))
         return false;
      else
         return true;
   }
```

Yukarıdaki uygulamayı çeşitli kelime girişleri ile deneyerek pekiştiriniz.

2.1.15.1. substring (int indeks)

Birlikte çağrıldığı metni parametre olarak verilen indeks değerinden itibaren keser ve arta kalan metni geriye String türünde döndüren metottur.

- **Kullanımı:**

```
string yeniMetin=metin.substring(int indeks);
```

Örnek 0-34: substring(int indeks) metodunun kullanımı;

Kod
String metin = "Bilişim Teknolojileri"; System.out.println(metin.substring(3)); //işim Teknolojileri System.out.println(metin.substring(8)); //Teknolojileri System.out.println(metin.substring(14)); //ojileri

2.1.15.2. substring (int bas indeks, int son index)

Substring metodunun bu kullanımında ise ilk parametre başlangıç indeks değerini, ikinci parametre ise son indeks numarasını alır ve metnin kesileceğini belirtir.

- **Kullanımı:**

```
String yeniMetin=metin.substring(int baş_indeks,int son_indeks);
```

Örnek 0-35: substring (int basindeks, int soninedx) metodunun kullanımı;

Kod
String metin = "Bilişim Teknolojileri"; System.out.println(metin.substring(3,7)); //işim System.out.println(metin.substring(8,11)); //Tek System.out.println(metin.substring(14,15)); //o

2.1.16. toLowerCase()

Birlikte çağrıldığı metninin tüm karakterlerini küçük harfe dönüştürerek yeni bir metin geriye döndürür.

- **Kullanımı:**

```
String yeniMetin=metin.toLowerCase();
```

Örnek 0-36: toLowerCase() metodunun kullanımı;

Kod
String metin = "Bilişim Teknolojileri"; System.out.println(metin.toLowerCase()); //bilişim teknolojileri String metin2= "iBRAHİM öNAL fEN LİseSİ"; System.out.println(metin2.toLowerCase());//ibrahim önal fen lisesi

2.1.17. toUpperCase ( )

ToLower( ) metodunun tam tersi şeklinde çalışır ve birlikte çağrıldığı metninin tüm karakterlerini büyük harfe dönüştürerek yeni bir metin geriye döndürür.

- **Kullanımı:**

```
String yeniMetin=metin.toUpperCase();
```

Kod
String metin = "Bilişim Teknolojileri"; System.out.println(metin.toUpperCase()); //BİLİŞİM TEKNOLOJİLERİ String metin2= "iBRAHİM öNAL fEN LİseSİ"; System.out.println(metin2.toUpperCase());//İBRAHİM ÖNAL FEN LİSESİ

2.2. Matematiksel (Math) Metotları

Programlama dili içerisindeki Math sınıfı altında bulunan ve matematiksel bazı işlem ve fonksiyonları daha kolay yapabilmek için bir takım hazır metotlar vardır.

2.2.1. abs( )

abs( ) metodu parametre olarak verilen sayının mutlak değerini veren metottur. Parametre olarak farklı sayı türlerinde değerler alabilir ve aldığı değerin türünde bir değer geri döndürür.

- **Kullanımı:**

```
int mutlakDeger=Math.Abs(int sayi);
double mutlakDeger=Math.Abs(double sayi);
float mutlakDeger=Math.Abs(float sayi);
long mutlakDeger=Math.Abs(long sayi);
```

Örnek 0-38: | x - 2| + 2 + |2 + x| ifadesinin sonucunu x'in 0'dan 10'a kadar olan değerleri için tek tek ekrana yazdıran programın kodunu yazınız.

Kod
int sonuc=0; for (int x = 0; x <= 10; x++) { sonuc = Math.abs(x - 2) + 2 + Math.abs(2 + x); System.out.printf("x: %d için sonuç: %d %n",x,sonuc); }

Kod
x: 0 için sonuç: 6 x: 1 için sonuç: 6 x: 2 için sonuç: 6 x: 3 için sonuç: 8 x: 4 için sonuç: 10 x: 5 için sonuç: 12 x: 6 için sonuç: 14 x: 7 için sonuç: 16 x: 8 için sonuç: 18 x: 9 için sonuç: 20 x: 10 için sonuç: 22

2.2.3. ceil( )

Parametre olarak verilen **double** türündeki ondalıklı sayıdan büyük, en küçük tam sayının değerini veren metottur.

- **Kullanımı:**

```
double sonuc=Math.ceil(double a);
```

Örnek 0-40: Math.ceil( ) metodunun kullanımı

Kod
double sayi= 2.00; double sonuc=0.00; sonuc=Math.ceil(sayi); // 2 System.out.println(sonuc); sayi= 2.01; sonuc=Math.ceil(sayi); // 3 System.out.println(sonuc); sayi= 2.50; sonuc=Math.ceil(sayi); // 3 System.out.println(sonuc); sayi= 2.99; sonuc=Math.ceil(sayi); // 3 System.out.println(sonuc);

2.2.4. max( )

Parametre olarak verilen iki sayıdan büyük olanı geriye döndüren metottur. Bütün sayı türleri tarafından desteklenen bir metot çeşididir.

- **Kullanımı:**

```
int maksimum=Math.max(int sayi1,int sayi2);
long maksimum=Math.max(long sayi1,long sayi2);
double maksimum=Math.max(double sayi1,double sayi2);
float maksimum=Math.max(float sayi1,float sayi2);
```

Örnek 0-42: Math.max( ) metodu ile klavyeden girilen sayılardan büyüğünü bulma

Kod
Scanner oku=new Scanner(System.in); System.out.print("1. sayıyı giriniz:"); int sayi1 = oku.nextInt(); System.out.print("2. sayıyı giriniz:"); int sayi2 = oku.nextInt(); int maksimum = Math.max(sayi1,sayi2); System.out.printf("%d ve %d sayılarından en büyüğü %d sayısıdır.", sayi1, sayi2, maksimum);

Çıktı
1. sayıyı giriniz:1943 2. sayıyı giriniz:1905 1943 ve 1905 sayılarından en büyüğü 1943 sayısıdır.

2.2.5. min( )

Parametre olarak verilen iki sayıdan küçük olanı geriye döndüren metottur. Bütün sayı türleri tarafından desteklenen bir metot çeşidir.

- **Kullanımı:**

```
int minimum=Math.min(int sayi1,int sayi2);
double minimum=Math.min(double sayi1,double sayi2);
float minimum=Math.min(float sayi1,float sayi2);
long minimum=Math.min(long sayi1,long sayi2);
```

Örnek 0-43: Math.min( ) metodu ile klavyeden girilen sayılardan küçüğünü bulma

Kod
Scanner oku=new Scanner(System.in); System.out.print("1. sayıyı giriniz:"); int sayi1 = oku.nextInt(); System.out.print("2. sayıyı giriniz:"); int sayi2 = oku.nextInt(); int minimum = Math.min(sayi1,sayi2); System.out.printf("%d ve %d sayılarından en küçüğü %d sayısıdır.", sayi1, sayi2, minimum);

Çıktı
1. sayıyı giriniz:1943 2. sayıyı giriniz:1905 1943 ve 1905 sayılarından en küçüğü 1905 sayısıdır.

2.2.6. pow( )

Parametre olarak verilen ilk sayının, yine parametre olarak verilen ikinci sayı kadar üssünü hesaplayan metottur.

- **Kullanımı:**

```
double usluSayi=Math.pow(double x, double y);
```

$x^y = Math.Pow(x,y);$

Örnek 0-44: Math.pow( ) metodunun kullanımı

Kod
double usluSayi =Math.pow(3,3); // 27.0 double usluSayi1=Math.pow(2,16); // 65536.0 double usluSayi2=Math.pow(12,0); // 1.0 double usluSayi3=Math.pow(5,-2); // 0.04 double usluSayi4=Math.pow(-10,-2); // 0.01

2.2.7. round( )

Parametre olarak verilen sayıyı en yakın tam sayıya yuvarlayan metottur.

- **Kullanımı:**

```
long yuvarlanmis=Math.round(double sayi);
int yuvarlanmis2=Math.round(float sayi);
```

Örnek 0-45: Math.round( ) metodunun kullanımı

Kod
double yuvarlanmis=Math.round(3.14); // 3.0 double yuvarlanmis1=Math.round(3.499); // 3.0 double yuvarlanmis2=Math.round(3.5); // 4.0 double yuvarlanmis3=Math.round(3.9999); // 4.0 long yuvarlanmis4=Math.round(3.14); // 3 int yuvarlanmis5=Math.round(3.501f); // 4

2.2.8. signum( )

Parametre olarak verilen sayının işaretini verir. Sayı pozitif ise 1, negatif ise -1, sayı sıfıra eşitse de geriye 0 değerini döndüren metottur.

- **Kullanımı:**

```
float isaret=Math.signum(int sayi);
```

Örnek 0-46: Math.signum( ) metodunun kullanımı

Kod
float işaret = Math.signum(26638); // 1 float isaret1= Math.signum(-26638); // -1 float isaret2= Math.signum(0); // 0

2.2.9. sqrt( )

Parametre olarak verilen **double** türündeki sayının karekök değerini **double** türünde geriye döndüren metottur.

- **Kullanımı:**

```
double karekok=Math.sqrt(double sayi);
```

Örnek 0-47: Klavyeden 2 kenar uzunluğu girilen dik üçgenin hipotenüsünün uzunluğunu hesaplayan programın kodunu yazınız.

Kod
Scanner oku=new Scanner(System.in); System.out.print("1. kenar uzunluğunu giriniz: "); double kenar1= oku.nextDouble(); System.out.print("2. kenar uzunluğunu giriniz: "); double kenar2 = oku.nextDouble(); double kenarlarinKareToplami = Math.pow(kenar1, 2) + Math.pow(kenar2, 2); double kenar3= Math.sqrt(kenarlarinKareToplami); System.out.print("Hipotenüsün uzunluğu: "+kenar3);

Uyarı: Trigonometrik fonksiyonlarda açı değerleri radyan cinsinden verilmelidir.

Derece	0°	30°	45°	60°	90°	180°	270°	360°
Radyan	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	π	$\frac{3\pi}{2}$	2π

Dilerseniz trigonometrik metotlara geçmeden önce verilen derece cinsinden verilen açı değerini radyana dönüştüren ve radyan değeri verilen açının derece cinsinden değerini veren basit metotlar yazalım ve bundan sonraki örneklerimizde bu metottan faydalanalım.

Örnek 0-48: Derece cinsinden açı değerini radyan cinsinden açı değerine dönüştüren metodun yazımı;

Kod

```
static double radyanaDonustur(double derece)
   {
      double radyan = 0;
      double piSayisi = Math.PI;
      radyan = piSayisi * (derece / 180);
      return radyan;
   }
   public static void main(String[] args) {

      Scanner oku=new Scanner(System.in);
      System.out.print("Açı değeri :");
      double aci=oku.nextDouble();

      double radyanDegeri=radyanaDonustur(aci);
      System.out.println("Radyan değeri :"+radyanDegeri);

   }
```

Çıktı

```
Açı değeri :180
Radyan değeri :3.141592653589793

Açı değeri :90
Radyan değeri :1.5707963267948966

Açı değeri :30
Radyan değeri :0.5235987755982988
```

2.2.10. cos( )

Parametre olarak verilen radyan açı değerinin kosinüs değerini veren metottur.

- **Kullanımı:**

```
double kosinus=Math.cos(double aci);
```

2.2.11. acos( )

Parametre olarak verilen kosinüs değerinin radyan açı değerini veren metottur.

- **Kullanımı:**

```
double kosinusAcisi=Math.acos(double kosinus);
```

Örnek 0-50: 0-180 arası 15 ve katlarının kosinüs değerini ekrana yazdıran programın kodlarını yazınız.

Kod

```
  static double radyanaDonustur(double derece)
{
   double radyan = 0;
   double piSayisi = Math.PI;
   radyan = piSayisi * (derece / 180);
   return radyan;
}

public static void main(String[] args) {

   for (double aci = 0; aci <= 180; aci = aci + 15)
   {
      double radyanDegeri = radyanaDonustur(aci);
      double kosinus = Math.cos(radyanDegeri);
      System.out.printf("%f açısının kosinüs değeri:\t%.2f %n",aci, kosinus);
   }
}
```

Çıktı

```
0,000000 açısının kosinüs değeri:  1,00
15,000000 açısının kosinüs değeri:  0,97
30,000000 açısının kosinüs değeri:  0,87
45,000000 açısının kosinüs değeri:  0,71
60,000000 açısının kosinüs değeri:  0,50
75,000000 açısının kosinüs değeri:  0,26
90,000000 açısının kosinüs değeri:  0,00
105,000000 açısının kosinüs değeri: -0,26
120,000000 açısının kosinüs değeri: -0,50
135,000000 açısının kosinüs değeri: -0,71
150,000000 açısının kosinüs değeri: -0,87
165,000000 açısının kosinüs değeri: -0,97
180,000000 açısının kosinüs değeri: -1,00
```

2.2.12. sin( )

Parametre olarak verilen radyan açı değerinin sinüs değerini veren metottur.

- **Kullanımı:**

```
double sinus=Math.Sin(double aci);
```

2.2.13. asin( )

Parametre olarak verilen sinüs değerinin radyan açı değerini veren metottur.

- **Kullanımı:**

```
double sinusAcisi=Math.asin(double sinus);
```

Örnek 0-51: 0-180 arası 15 ve katlarının sinüs değerini ekrana yazdıran programın kodlarını yazınız.

Kod

```
static double radyanaDonustur(double derece)
   {
      double radyan = 0;
      double piSayisi = Math.PI;
      radyan = piSayisi * (derece / 180);
      return radyan;
   }

   public static void main(String[] args) {

      for (double aci = 0; aci <= 180; aci = aci + 15)
      {
         double radyanDegeri = radyanaDonustur(aci);
         double sinus = Math.sin(radyanDegeri);
         System.out.printf("%f açısının sinüs değeri: %.2f %n",aci, sinus);
      }
   }
```

Çıktı

```
0,000000 açısının sinüs değeri: 0,00
15,000000 açısının sinüs değeri: 0,26
30,000000 açısının sinüs değeri: 0,50
45,000000 açısının sinüs değeri: 0,71
60,000000 açısının sinüs değeri: 0,87
75,000000 açısının sinüs değeri: 0,97
90,000000 açısının sinüs değeri: 1,00
105,000000 açısının sinüs değeri: 0,97
120,000000 açısının sinüs değeri: 0,87
135,000000 açısının sinüs değeri: 0,71
150,000000 açısının sinüs değeri: 0,50
165,000000 açısının sinüs değeri: 0,26
180,000000 açısının sinüs değeri: 0,00
```

2.2.14. tan( )

Parametre olarak verilen radyan açı değerinin tanjant değerini veren metottur.

- **Kullanımı:**

```
double tanjant=Math.tan(double aci);
```

2.2.15. atan( )

Parametre olarak verilen tanjant değerinin radyan açı değerini veren metottur.

- **Kullanımı:**

```
double tanjantAcisi=Math.atan(double tanjant);
```

Örnek 0-52: 0-180 arası 15 ve katlarının tanjant değerini ekrana yazdıran programın kodlarını yazınız.

Kod

```
static double radyanaDonustur(double derece)
   {
      double radyan = 0;
      double piSayisi = Math.PI;
      radyan = piSayisi * (derece / 180);
      return radyan;
   }

   public static void main(String[] args) {



      for (double aci = 0; aci <= 180; aci = aci + 15)
      {
         double radyanDegeri = radyanaDonustur(aci);
         double tanjant = Math.tan(radyanDegeri);
         System.out.printf("%f açısının tanjant değeri: %.2f %n",aci, tanjant);
      }
   }
```

Çıktı

```
0,000000 açısının tanjant değeri: 0,00
15,000000 açısının tanjant değeri: 0,27
30,000000 açısının tanjant değeri: 0,58
45,000000 açısının tanjant değeri: 1,00
60,000000 açısının tanjant değeri: 1,73
75,000000 açısının tanjant değeri: 3,73
90,000000 açısının tanjant değeri: 16331239353195370,00
105,000000 açısının tanjant değeri: -3,73
120,000000 açısının tanjant değeri: -1,73
135,000000 açısının tanjant değeri: -1,00
150,000000 açısının tanjant değeri: -0,58
165,000000 açısının tanjant değeri: -0,27
180,000000 açısının tanjant değeri: -0,00
```

2.3. Tarih/Saat Metotları

Programlama dili içerisinde, tarih ve zamanlar ile ilgili işlemler yaparken bir takım işleri daha kolay yapabilmemiz için önceden tanımlanmış Tarih/Zaman metotlarını kullanırız

2.3.1. Instant.MIN

Bu özellik ile Tarih zaman yapısı ile kullanabileceğimiz en küçük tarih-saat bilgisine erişebiliriz.

· **Kullanımı**:

```
Instant enKucuk=Instant.MIN;
```

Bu sabit özellik çağrıldığında geriye veri türü Instant olan "-1000000000-01-01T00:00:00Z" değerini döndürür. Bu değer değiştirilemeyen "Salt Okunur" bir veridir.

2.3.2. Instant.MAX

Bu özellik ile Tarih zaman yapısı ile kullanabileceğimiz en büyük tarih-saat bilgisine erişebiliriz.

· **Kullanımı:**

```
Instant enBuyuk=Instant.MAX;
```

Bu sabit özellik çağrıldığında geriye veri türü Instant olan "+1000000000-12-31T23:59:59.999999999Z" değerini döndürür. Bu değer de aynı MinValue gibi değiştirilemeyen "Salt Okunur" bir veridir.

Örnek 0-53: Instant sınıfı ile kullanılabilen en büyük ve en küçük tarihlerin ekrana yazdırınız;

Kod
Instant enBuyuk=Instant.MAX; Instant enKucuk=Instant.MIN; System.out.println("En büyük tarih değeri: "+ enBuyuk); System.out.println("En küçük tarih değeri: "+ enKucuk);

Çıktı
En büyük tarih değeri: +1000000000-12-31T23:59:59.999999999Z En küçük tarih değeri: -1000000000-01-01T00:00:00Z

2.3.3. Date()

Bugünün tarihini geri döndüren formatlı kodu aşağıda görüyoruz.

Kod
SimpleDateFormat format = new SimpleDateFormat("dd.MM.yyyy"); String dateString = format.format(new Date()); System.out.println(dateString);

Çıktı
06.03.2019

Bugünün tarihini ve saatini geri döndüren formatlı kodu aşağıda görüyoruz.

Kod
SimpleDateFormat format = new SimpleDateFormat("dd.MM.yyyy hh:mm:ss"); String dateString = format.format(new Date()); System.out.println(dateString);

Çıktı
06.03.2019 12:06:06

Örnek 0-55: ZonedDateTime sınıfı ile kullanılabilen özelliklerin kullanımı ve ekrana yazdırılması

Kod

```
ZonedDateTime zaman = ZonedDateTime.now( ZoneId.of( "Europe/Istanbul" ) );
DateTimeFormatter tarihSaat= DateTimeFormatter.ofPattern("dd MM yyyy hh:mm:ss");
DateTimeFormatter tarih= DateTimeFormatter.ofPattern("dd MM yyyy");
System.out.println("Şuandaki zaman    :\t"+zaman.toLocalDateTime().format(tarihSaat));
System.out.println("Tarih Bilgisi    :\t"+zaman.toLocalDate().format(tarih));
System.out.println("Ay Bilgisi       :\t"+zaman.getMonthValue());
System.out.println("Gün Bilgisi      :\t"+zaman.getDayOfMonth());
System.out.println("Yıl Bilgisi      :\t"+zaman.getYear());
System.out.println("Haftanın günü    :\t"+zaman.getDayOfWeek());
System.out.println("Yılın Kaçıncı günü:\t"+zaman.getDayOfYear());
System.out.println("Saat             :\t"+zaman.getHour());
System.out.println("Dakika           :\t"+zaman.getMinute());
System.out.println("Saniye           :\t"+zaman.getSecond());
System.out.println("Salise           :\t"+zaman.getNano()  );
```

Çıktı

```
Şuandaki zaman    :    06 03 2019 12:51:48
Tarih Bilgisi     :    06 03 2019
Ay Bilgisi        :    3
Gün Bilgisi       :    6
Yıl Bilgisi       :    2019
Haftanın günü     :    WEDNESDAY
Yılın Kaçıncı günü:    65
Saat              :    12
Dakika            :    51
Saniye            :    48
Salise            :    951546700
```

Örnek 0-56: 31/12/1990 tarihinden önce doğmuş kişilerin başvuru yapabildikleri bir sınava, klavyeden doğum tarihi bilgileri girilen kişinin, sınava girip giremeyeceğini kontrol eden programın kodlarını yazınız.

Kod

```
public static LocalDate okuTarih(){
    Scanner oku=new Scanner(System.in);
    System.out.println("Doğum tarihini giriniz");
    System.out.print("Gun :");
    Byte gun=oku.nextByte();
    System.out.print("Ay :");
    Byte ay =oku.nextByte();
    System.out.print("Yıl :");
    int yil=oku.nextInt();

    return  LocalDate.of(yil,ay,gun);

  }
```

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

    LocalDate tarih1 = LocalDate.of(1990, 12, 31);
    LocalDate tarih2 = okuTarih();
    if (tarih2.isBefore(tarih1)){
       System.out.println("Tebrikler, sınava girebilirsiniz...");
    }else {
       System.out.println("Üzgünüz, sınava giremezsiniz...");
    }

}
```

2.3.4. isLeapYear( )

Bu metot ile **int** türünde parametre olarak verilen yılın, "***artık yıl***" (şubat ayının 29 günden, yılın toplam 366 günden oluştuğu) olup olmadığını kontrol eder, geriye **boolean** türünde bir değer döndürür. Eğer yıl artık yılsa geriye **true**, değilse **false** değeri döndürür.

- **Kullanımı:**

```
boolean artikYilMi= tarih.isLeapYear();
```

Kod
LocalDate tarih1 = LocalDate.of(2008,1,1);//true LocalDate tarih2 = LocalDate.of(2010,1,1);//false LocalDate tarih3 = LocalDate.of(2012,1,1);//true LocalDate tarih4 = LocalDate.of(1996,1,1);//true boolean artikYilMi=tarih1.isLeapYear();

2.3.5. SimpleDateFormat()

Bu metot ile **String** türünde parametre olarak verilen metni Date türündeki tarih ve saat bilgisine dönüştürür.

Kod
String sDate1="31/12/1998"; String sDate2 = "12 31, 1998"; SimpleDateFormat formatter1=new SimpleDateFormat("dd/MM/yyyy"); SimpleDateFormat formatter2=new SimpleDateFormat("MM dd, yyyy"); Date date1=formatter1.parse(sDate1); Date date2=formatter2.parse(sDate2); System.out.println(sDate1+"\t"+date1); System.out.println(sDate2+"\t"+date2);
Çıktı
31/12/1998 Thu Dec 31 00:00:00 EET 1998 12 31, 1998 Thu Dec 31 00:00:00 EET 1998

Örnek 0-57: Şuandaki zamandan doğum tarihinizi çıkartarak ne kadar süredir hayatta olduğunuzu hesaplayınız

Kod
LocalDate bugun=LocalDate.now(ZoneId.of("Europe/Istanbul")); LocalDate dogumTarihi=okuTarih(); Duration fark = Duration.between(dogumTarihi.atStartOfDay(), bugun.atStartOfDay()); long diffDays = fark.toDays(); long diffHours= fark.toHours(); long diffMinutes=fark.toMinutes(); long diffSeconds=fark.getSeconds(); System.out.printf("Dünya üzerinde geleli %d gün olmuş.%n",diffDays); System.out.printf("Dünya üzerinde geleli %d saat olmuş.%n",diffHours); System.out.printf("Dünya üzerinde geleli %d dakika olmuş.%n",diffMinutes); System.out.printf("Dünya üzerinde geleli %d saniye olmuş.%n",diffSeconds);
Çıktı
Doğum tarihini giriniz Gun :12 Ay :5 Yıl :1983 Dünya üzerinde geleli 13083 gün olmuş. Dünya üzerinde geleli 313992 saat olmuş. Dünya üzerinde geleli 18839520 dakika olmuş. Dünya üzerinde geleli 1130371200 saniye olmuş.

2.3.6. plusDays()

Bu metot ile LocalDate türünde türetilmiş bir nesneye, parametre olarak verilen **long** türündeki değer kadar gün eklenir ve geriye LocalDate türünde bir değer döndürür.

Örnek 0-59: Aşağıdaki örnek bugünün tarih değerine 1 değeri ekleyerek yarının tarihini bulmayı sağlar.

Kod
DateTimeFormatter formatter= DateTimeFormatter.ofPattern("dd.MM.yyyy"); LocalDate bugun=LocalDate.now(ZoneId.of("Europe/Istanbul")); LocalDate yarin=**bugun.plusDays(1);** System.out.println("Bugün :"+bugun.format(formatter)); System.out.println("Yarın :"+yarin.format(formatter));
Çıktı
Bugün :07.03.2019 Yarın :08.03.2019

2.3.7. plusMonths();

Bu metot ile LocalDate türünde türetilmiş bir nesneye, parametre olarak verilen **long** türündeki değer kadar ay eklenir ve geriye LocalDate türünde bir değer döndürür.

Örnek 0-60: Aşağıdaki örnek bugünün tarih değerine 1 değeri ekleyerek bir dahaki ayın tarihini bulmayı sağlar.

Kod
DateTimeFormatter formatter= DateTimeFormatter.ofPattern("dd.MM.yyyy"); LocalDate bugun=LocalDate.now(ZoneId.of("Europe/Istanbul")); LocalDate **gelecekAy=bugun.plusMonths(1);** System.out.println("Bugün :"+bugun.format(formatter)); System.out.println("Gelecek Ay :"+gelecekAy.format(formatter));
Çıktı
Bugün :07.03.2019 Gelecek Ay :07.04.2019

2.3.8. plusYears();

Bu metot ile LocalDate türünde türetilmiş bir nesneye, parametre olarak verilen **long** türündeki değer kadar yıl eklenir ve geriye LocalDate türünde bir değer döndürür.

Örnek 0-61: Aşağıdaki örnek bugünün tarih değerine 1 değeri ekleyerek bir sonraki yılın tarihini bulmayı sağlar.

Kod
DateTimeFormatter formatter= DateTimeFormatter.ofPattern("dd.MM.yyyy"); LocalDate bugun=LocalDate.now(ZoneId.of("Europe/Istanbul")); LocalDate gelecekYil=**bugun.plusYears(1);** System.out.println("Bugün :"+bugun.format(formatter)); System.out.println("Gelecek Yıl :"+gelecekYil.format(formatter));
Çıktı
Bugün :07.03.2019 Gelecek Yıl :07.03.2020

2.3.9. plusHours();

Bu metot ile ZonedDateTime türünde türetilmiş bir nesneye, parametre olarak verilen **long** türündeki değer kadar saat eklenir ve geriye ZonedDateTime türünde bir değer döndürür.

Örnek 0-62: Aşağıdaki örnek bugünün saat değerine 1 değeri ekleyerek bir sonraki saati bulmayı sağlar.

Kod

```
ZonedDateTime zaman = ZonedDateTime.now( ZoneId.of( "Europe/Istanbul" ) );
DateTimeFormatter saat= DateTimeFormatter.ofPattern("hh:mm:ss");
ZonedDateTime sonrakiSaat=zaman.plusHours(1);

System.out.println("Şuandaki zaman    :\t"+zaman.toLocalDateTime().format(saat));
System.out.println("Bir saat sonra    :\t"+sonrakiSaat.toLocalDateTime().format(saat));
```

Çıktı

```
Şuandaki zaman    :    11:25:17
Bir saat sonra    :        12:25:17
```

2.3.10. plusMinutes()

Bu metot ile ZonedDateTime türünde türetilmiş bir nesneye, parametre olarak verilen **long** türündeki değer kadar dakika eklenir ve geriye ZonedDateTime türünde bir değer döndürür.

Örnek 0-63: Aşağıdaki örnek bugünün saat değerine 30 değeri ekleyerek bir sonraki yarım saati bulmayı sağlar.

Kod

```
ZonedDateTime zaman = ZonedDateTime.now( ZoneId.of( "Europe/Istanbul" ) );
DateTimeFormatter saat= DateTimeFormatter.ofPattern("hh:mm:ss");
ZonedDateTime yarimSaat=zaman.plusMinutes(30);

System.out.println("Şuandaki zaman      :\t"+zaman.toLocalDateTime().format(saat));
System.out.println("Yarim saat sonra    :\t"+yarimSaat.toLocalDateTime().format(saat));
```

Çıktı

```
Şuandaki zaman      :  11:30:27
Yarim saat sonra    :  12:00:27
```

2.3.11. plusSeconds()

Bu metot ile ZonedDateTime türünde türetilmiş bir nesneye, parametre olarak verilen **long** türündeki değer kadar saniye eklenir ve geriye ZonedDateTime türünde bir değer döndürür.

Örnek 0-64: Aşağıdaki örnek bugünün saat değerine 15sn değeri ekleyerek bir sonraki 15 saniyeli zaman değerini bulmayı sağlar.

Kod

```
ZonedDateTime zaman = ZonedDateTime.now( ZoneId.of( "Europe/Istanbul" ) );
DateTimeFormatter saat= DateTimeFormatter.ofPattern("hh:mm:ss");
ZonedDateTime saniyeEkle=zaman.plusSeconds(15);

System.out.println("Şuandaki zaman      :\t"+zaman.toLocalDateTime().format(saat));
System.out.println("15 sn sonra         :\t"+saniyeEkle.toLocalDateTime().format(saat));
```

Çıktı

```
Şuandaki zaman      :  11:37:17
15 sn sonra         :      11:37:32
```

Örnek 0-65: 12/07/2008 20:12:23:33 tarihinden 5 yıl, 6 ay, 28 gün, 23 saat, 29 dakika, 33 saniye ve 43 salise sonrasının tarihi nedir?

Kod

```
DateTimeFormatter formatla= DateTimeFormatter.ofPattern("dd.MM.yyyy HH:mm:ss:nn");
LocalDate tarih=LocalDate.of(2008, 07, 12);
LocalTime saat =LocalTime.of(20,12,23,33);
ZonedDateTime ilkTarih =ZonedDateTime.of(tarih,saat, ZoneId.of( "Europe/Istanbul" ));
ZonedDateTime bitisTarihi =
ilkTarih.plusYears(5).plusMonths(6).plusDays(28).plusHours(23)
.plusMinutes(29).plusSeconds(33).plusNanos(43);
System.out.println("İlk Tarih     :"+ilkTarih.format(formatla));
System.out.println("Bitiş Tarihi :"+bitisTarihi.format(formatla));
```

Çıktı

```
İlk Tarih      :12.07.2008 20:12:23:33
Bitiş Tarihi :10.02.2014 19:41:56:76
```