JAVA 101

TEMEL KAVRAMLAR

Soru İşareti Operatörü

? işareti operatörü ile Java'da mantıksal kıyaslama yapılabilir. ? ifadesi Java'daki "if-else" yapısı yerine kullanılabilir. Tek satırda bunu yapabilmemizi sağlar. Kullanımı ise if'in içerisinde yer alacak ifadeyi soru işaretinden önce yazılır, ifadenin doğru olması durumunda yapılacak işlemler soru işareti ile iki nokta arasına yazılır. İfadenin yanlış olması durumunda yapılacaklar ise iki noktadan sonra yazılır.

Kullanımı:

```
( kontrol edilecek ifade ) ? doğru olması durumunda yapılacaklar : yanlış olması durumunda yapılacaklar
```

Örnek:

```
public class Test {

   public static void main(String args[]) {
      int a, b;
      a = 10;
      b = (a == 1) ? 20: 30;
      System.out.println( "Value of b is : " +  b );

      b = (a == 10) ? 20: 30;
      System.out.println( "Value of b is : " + b );
   }
}
```

Sonuç:

```
Value of b is : 30

Value of b is : 20
```

NESNEYE YÖNELİK PROGRAMLAMAYA GİRİŞ

Sınıf ve Nesne Kavramları

Bildiğimiz gibi her nesnenin kendine ait **nitelikleri** ve **davranışları** vardır. Nesneler birbirlerinden farklıdır ve kendi **varoluşlarına** göre davranırlar ve kendi kimliklerine sahiptirler.

Nitelik: Nitelik kavramı, bir nesnenin özellikleridir ve nesnenin mevcut durumunu tanımlar. Mesela bir ördeğin rengi ve ağırlığı o ördeğin nitelikleridir. Bir ördeğimiz rengi siyah, diğer ördeğimizin rengi beyazdır. Bu durumda her iki nesne (yani ördek) nitelikleri sebebi ile birbirilerinden bağımsızdırlar.

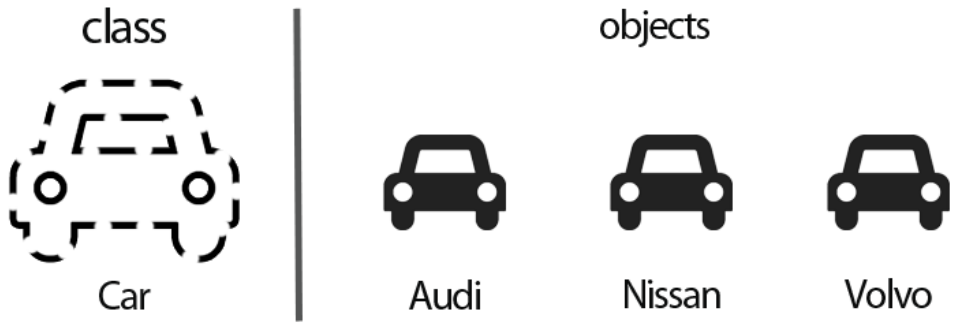
Davranış: Bir nesnenin kendine özel yaptığı eylemlerdir. Yine ördek örneğine bakarsak, bir ördek uçabiliyorken diğer bir ördek fiziksel durumu gereği uçamayabilir.

Nesneler hakkında bilmemiz gereken bir diğer husus ise, nesneler bir isimdir. Nesnelerin kendilerine ait nitelik ve davranışları vardır.

Sınıf (Class) Nedir?

NYP sınıflar ve nesneler üzerine kurulmuştur, "**Sınıflar**" bir problemi soyutlamak ve genelleştirmek için kullanılan yapılardır veya kılavuzlardır. Sınıflar, bir nesneye ait tüm özellikleri temsil eder. Bu özellikler nesnenin ne tür **nitelikleri ve davranışları** olacağını belirler.

Mesela "**Araba**" bir sınıftır. Arabalara ait nitelikler **renk, hız, vites sayısı, yakıt türü vb.** bir sürü nitelik olabilir. Ayrıca bazı arabalara özel davranışlar olabilir, park sensörü, oto pilot, hız sabitleme gibi arabaların kendilerine özel davranışları da olabilir. Bir araba üretilirken, bir yapım kılavuzuna ihtiyaç vardır. Programlama da bu kılavuzlara "**Sınıf (Class)**" denir.



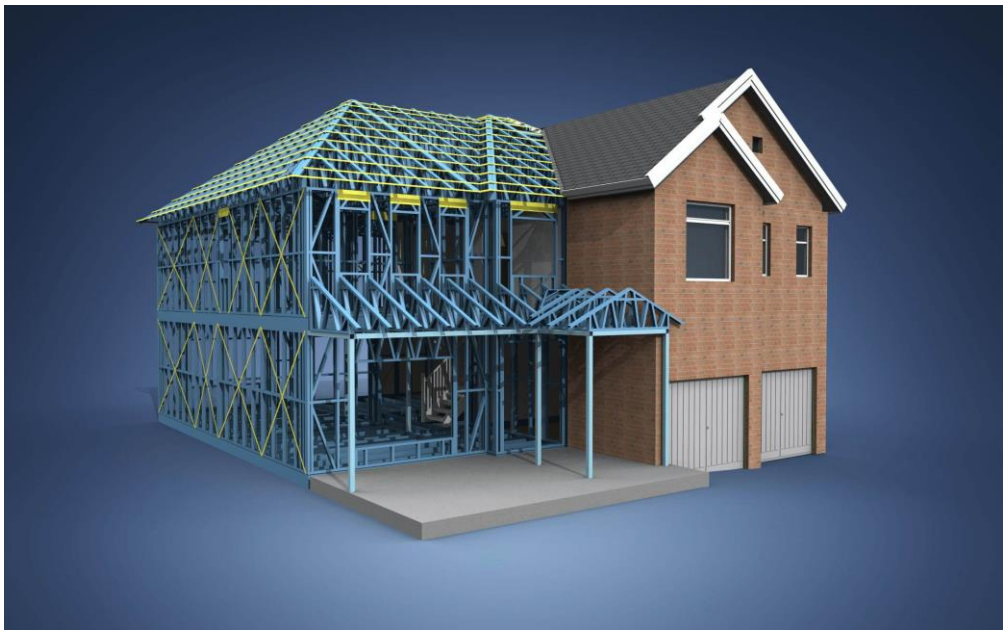
Nesne Yönelimli Programlama Nedir?

Nesne Yönelimli Programlama (Object Oriented Programming), sınıflar ve nesneler kavramına dayanan bir programlama yaklaşımıdır. Bu yaklaşımın amacı, ihtiyaç duyulan programı daha küçük parçalara bölerek, yönetilebilir ve yeniden kullanılabilir hale getirmektir. Her küçük parçanın kendine ait özelliği, verileri ve diğer küçük parçalarla nasıl iletişim kuracağı bilgileri bulunur.

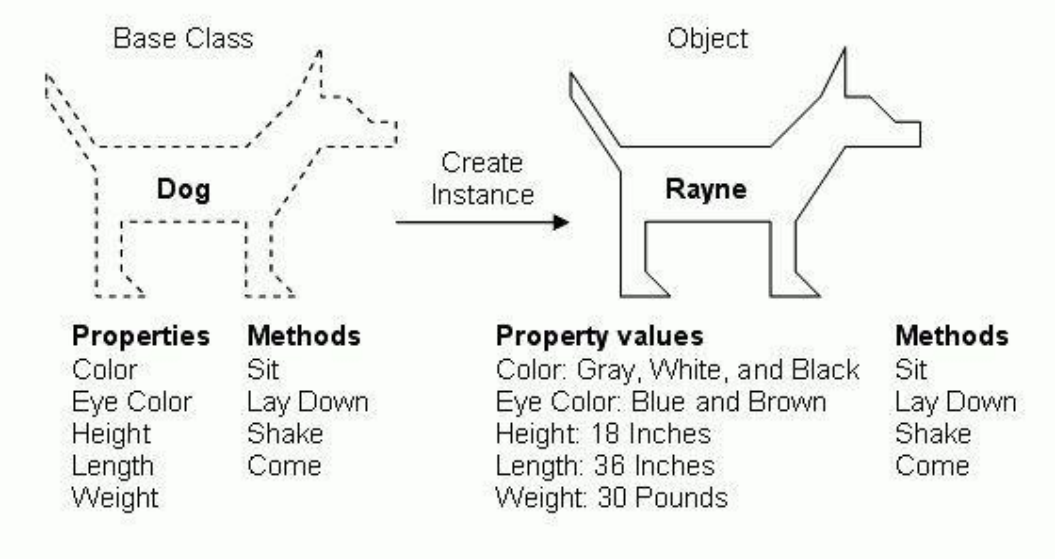
NYP'de programlar, nesnelerin birbirileriyle etkileşime geçmeleri sağlanmasıyla tasarlanır. Bizler gerçek hayattaki karmaşıklığı bir şekilde modelleyerek bunu bilgisayarın anlamasını sağlamaktayız. Modelleme, insanın problem çözmek üzere eskiden beri kullandığı bir yöntemdir. Büyükçe bir problemin tamamını zihinde canlandırıp çözmeye çalışmak yerine, oluşturulacak model ya da modeller üzerinde hedef sistemin görünüşü, davranışı ya da bazı durumlarda verdiği tepkiler gözlemlenebilir.
Nesne Yönelimli Programlama ile bizler yapacağımız her şeyi bilgisayarın anlayacağı şekilde modelleyip, "**nesne**" halinde aktarıyoruz. Böylelikle gerçek hayatta bizim için geçerli olan nesneleri artık bilgisayarların anlayacağı hale getirmiş oluyoruz. Tabi ki kullanılan programlama dilinin bizlere verdiği imkanlar dahilinde. Böylelikle kodlayan kişi ile bilgisayar arasında dilden bağımsız bir anlaşma, bir felsefe ortaya çıkmış oluyor. Artık bizim için "araba" nesnesi ne anlama geliyorsa, bilgisayar için de aynı anlama geliyor.

Modelleme ve Soyut Düşünme

Gerçek hayattaki problemleri bilgisayarın sanal ortamında çözebilmek için, ilk önce problemin uygun şekilde bilgisayar ortamına aktarılması gerekmektedir. Bu işlem "soyutlama (abstraction)" ya da "modelleme (modeling)" olarak anılır.
Modelleme, insanın problem çözmek üzere eskiden beri kullandığı bir yöntemdir. Büyükçe bir problemin tamamını zihinde canlandırıp çözmeye çalışmak yerine, oluşturulacak model ya da modeller üzerinde hedef sistemin görünüşü, davranışı ya da bazı durumlarda verdiği tepkiler gözlemlenebilir.
Model, var olan ya da gerçekleştirilmesi planlanan bir sistemi anlamak ya da anlatmak üzere oluşturulabilir ve birçok farklı alanda etkili bir şekilde kullanılmaktadır. Örneğin, bir toplu konut inşaatını müşterilerine tanıtmak isteyen bir inşaat firması, binaların yerleşimlerini, renk ve göreli büyüklüklerini görsel olarak ifade eden maket ya da maketler hazırlar. Bu maketi inceleyen bir kimse, almak istediği konutun nerede olduğunu, okul binasına yakınlığını ya da anayola ulaşımın nasıl olduğunu görerek değerlendirebilir. Burada model makettir ve "hedef sistemi anlatmak" amacını yerine getirmektedir.



Bir sistemle ilgili birden çok model oluşturulabilir. Tek bir model ile sistemin tamamını görmeye çalışmak yerine, üzerinde çalışılan sistemin farklı yönlerini öne çıkaran modeller hazırlanabilir. Örneğin inşaat firması toplu konutu müşterilere anlatmak üzere estetik tasarımı ön plana çıkan bir maket hazırlarken, bu toplu konut projesindeki binaların elektrik tesisatı için farklı, su tesisatı için farklı, genel daire görünümü için farklı projeler hazırlar. Böylece aynı sistemin farklı yönleriyle ilgilenen kimseler, yalnızca kendilerini ilgilendiren yönü öne çıkaran model üzerinde çalışma olanağı bulurlar.
Modelin mutlaka elle tutulur olması da gerekmez. Bilgisayar benzetimi ile de çeşitli modeller oluşturulabilir. Örneğin bir uçağın havadaki hareketini incelemek üzere geliştirilmiş bir bilgisayar benzetimi ile uçak modellenebilir. Kanat uzunluğu ya da gövde eğimi gibi parametrelerle oynanarak uçağın farklı hava koşullarında nasıl davranacağı anlaşılmaya çalışılabilir. Burada, sistemin davranışını anlamak amacıyla, sanal ortamda oluşturulmuş bir model söz konusudur.



SINIFLAR

Sınıf Tanımları

Java Nesne Yönelimli bir programlama dilidir. Java'daki her şey, değişkenleri ve metotları ile birlikte sınıflar ve nesnelerle ilişkilidir. Örneğin: gerçek hayatta araba bir nesnedir. Otomobilin ağırlık ve renk gibi değişkenleri ve sürüş ve fren gibi metotları vardır. Nesne yönelimli programlamanın amacı yazdığımız kodlara soyut bir kavrama dönüştürmektir. Sınıflara ait nitelikler ve davranışlar vardır. Programlamada nitelikler için değişkenler (variable), davranışlar için ise metotlar (method) tanımlanır.

```java
class Car {
    // nitelikler
    String type;
    String model;
    String color;
    int speed;

    // davranışlar
    int increaseSpeed(int increment) {
        speed += increment;
        return speed;
    }

    int decreaseSpeed(int decrease) {
        if (speed > 0) {
            speed -= decrease;
        }
        return speed;
    }

    void printSpeed() {
        System.out.println("Speed : " + speed);
    }
    // ...
}
```

Sınıflar nesneler oluşturabilmek için yazılım dünyasında oluşturulmuş şablonladır. Bu şablon nesne ile ilgili modellenecek tüm özellikleri ve davranışları bir taslak halinde kodlanmasını sağlar. Böylece, tanımlanmış bir sınıftan binlerce nesne oluşturabiliriz.

Sınıflar nesneleri tarif eden şablonlardır. Nesneler ise bu şablonlardan üretilen fiziksel yapılardır. Her üretilen nesne Hesap Hafıza Bölgesi'nde tutulur. Böylece sınıftan fiziksel karşılığı olan bir yapı elde etmiş oluruz. Sınıftan onlarca, yüzlerce nesne yaratabiliriz. Hepsi de hafıza başka adresleri gösterirler.

Constructor(Yapıcı) Metot Kullanımı

Kurucu metotlar sınıf tasarlanırken yazılırlar. Sınıfınızı yazarken kurucu metotlarınızı da tanımlayabilirsiniz. Eğer sınıf içinde hiç kurucu metot tanımlamazsınız parametresiz boş bir kurucu metot Java tarafından otomatik olarak tanımlanır.
Kurucu metotlar ilgili sınıftan bir nesne üretmeye çalıştığınızda daha nesne üretme aşamasında çalıştırılan özel metotlardır (fonksiyonlardır). **Kurucu metotların isimleri Sınıf ismiyle aynı olmak zorundadır.** Dönüş tipi olarak veya void olarak herhangi bir tanımlama yapılmasına gerek yoktur.
"**new**" anahtar kelimesi ile nesne üretirken kurucu metot çağrımı yapılır.

“this” Anahtar Kelimesi: Sınıfa ait niteliklere erişmek için kullanılır.

DİZİLER

Dizilerin Genel Mantıkları

Dizi, aynı türden birden fazla değişkeni tutmamızı sağlayan hafıza birimidir. Diziler sabit boyutludur ve tanımlanırken boyutları belirtilir. Dizinin hafızada bir başlangıç adresi olur ve ardışık olan diğer elemanlar sırayla hafızaya yerleştirilir. Diziler "new" anahtar sözcüğüyle oluşturulur. Böylece, Heap Hafıza bölgesinde yer kaplarlar.

```
public static void printArray(int[] array) {
   for (int i = 0; i < array.length; i++)
   {
      System.out.print(array[i] + " ");
   }
}
```

printArray (int[ ] array) olan yer diziyi yerel değişken olarak fonksiyona gönderdiğimiz noktadır. Java'da tüm değişkenler "Pass by Value" yöntemiyle geçilir. Bu şu demektir. Sizin tanımladığınız değişkenin, nesnenin veya dizinin birebir kopyası oluşturulur. Bu kopya değer fonksiyona yerel değişken olarak gider. Bu Java mülakatlarında size sorulabilecek bir detaydır.

Çok Boyutlu Diziler

Sütun Kapasiteleri Farklı Matris Oluşturmak - Düzensiz Diziler:

Farklı sütun değerine sahip dizilere "**Ragged Arrays**" yani düzensiz diziler denir. Başka bir açıdan bakıldığında ise, 2 boyutlu dizileri, **dizilerin dizisi** (**array of arrays**) olarak düşünmek doğru olur. Yani iki boyutlu dizileri tek boyutlu diziler oluşturur.
Diziler aynı türden elemanlardan oluşmak zorundadır. int türünde bir dizi olabileceği gibi, dizinin dizisi de olabilir.
Yukarıdaki örneklerde matrisin sütun sayısını 4 olarak belirledik. Bu şekilde oluşturulursa matrisin bütün satırları 4 elemanlı olur. Fakat bu zorunlu değildir. Matris oluştururken sütun sayısı belirlemezsek, her bir satırdaki dizilerin kapasitesi farklı olabilir. Örneğin aşağıdaki kodu inceleyelim:

```
int[][] matrix = new int[3][];
matrix[0] = new int[1];
matrix[1] = new int[2];
matrix[2] = new int[3];
```

Burada önce 3 satırdan oluşan bir matris belirttik, fakat sabit bir sütun sayısı vermedik. Sonra her bir satır için ayrı ayrı sütun sayısı belirledik.

Çok boyutlu dizi oluştururken, yalnızca ilk boyutun (en soldaki) kapasitesini belirlemeniz yeterlidir. Diğer boyutların kapasitesini dinamik olarak belirleyebilirsiniz.

ForEach Kullanımı

Dizilerdeki ve listelerdeki elemanları daha hızlı şekilde ulaşmak için kullanılan kısa bir yöntemdir.

```
String[] arabalar = {"BMW", "Mercedes", "Ford", "Ferrari"};
for (String i : arabalar) {
  System.out.println(i);
}

// Çıktısı
// BMW
// Mercedes
// Ford
// Ferrari
```

```
public class Main {public static void main(String[] args) {
        int[][] matris = {
                {1, 2, 3},
                {4, 5, 6},
                {7, 8, 9},
                {10, 11, 12}
        };

        for (int[] u : matris) {
            for (int elem : u) {
                System.out.println(elem);
            }
        }
    }
}
```

Arrays Sınıfı ve Metotları

Java.util paketindeki Arrays sınıfı, Java Collection Framework'ün bir parçasıdır. Bu sınıf, Java dizilerini dinamik olarak oluşturmak ve bunlara erişmek için statik metotlar sağlar. Yalnızca statik metotlar ve Object sınıfının metotlarından oluşur. Bu sınıfın metotları, sınıf adının kendisi tarafından kullanılabilir.

```
import java.util.Arrays;
```

Arrays.toString(): Diziye ait elemanları direkt ekrana basmak için kullanılan bir metottur.

```java
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] dizi = {3, 5, 79, 12, 25, -3, 66, 82, -49, 152};
        System.out.println(Arrays.toString(dizi));
    }
}

// Çıktısı
// [3, 5, 79, 12, 25, -3, 66, 82,-49,152]
```

Arrays.fill(): Arrays.fill metodu ile dizilerimizin belirli bir bölümlerine değerler atayabiliriz.

```java
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] liste = {15, 1, 99, 7, 7, -22, 11, 2, -49, 52};

        Arrays.fill(liste, 2);
        System.out.println(Arrays.toString(liste));

        int[] liste2 = {15, 1, 99, 7, 7, -22, 11, 2, -49, 52};

        Arrays.fill(liste2, 3, 5, 7);
        System.out.println(Arrays.toString(liste2));
    }
}

// Çıktısı
// [2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
// [15, 1, 99, 7, 7, -22, 11, 2, -49, 52]
```

Arrays.sort(): Arrays.sort() metodu ile dizilerdeki elemanları sıralayabiliriz.

```java
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] liste = {6, 1, 55, 21, 33, -321, -21, 2, -11, 27};
        Arrays.sort(liste);
        System.out.println(Arrays.toString(liste));
    }
}
// Çıktısı
// [-321, -21, -11, 1, 2, 6, 21, 27, 33, 55]
```

Arrays.binarySearch(): Java'da dizideki bir elemanın indis değerini bulmak için binarySearch kullanılabilir. Ama bu metodu kullanabilmek için, dizinin sıralı olması gerekmektedir.

```
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] liste = {6, 1, 55, 21, 33, -321, -21, 2, -11, 27};

        Arrays.sort(liste);
        System.out.println(Arrays.toString(liste));

        int index = Arrays.binarySearch(liste, 33);
        System.out.println("33'ün indeksi :" + index);
    }
}
// [-321, -21, -11, 1, 2, 6, 21, 27, 33, 55]
// 33'ün indeksi :8
```

Arrays.copyOf() ve Arrays.copyOfRange(): Mevcut diziden belli bir uzunlukta yeni bir dizi oluşturmak için Arrays.copyOf() metotu kullanılır. Mevcut diziden belli bir aralıkta yeni bir dizi oluşturmak için ise Array.copyOfRange() metodu kullanılır.

```
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] liste = {6, 1, 55, 21, 33, -321, -21, 2, -11, 27};

        int[] copyArray = Arrays.copyOf(liste, 3);
        System.out.println(Arrays.toString(copyArray));

        int[] copyOfRangeArray = Arrays.copyOfRange(liste, 0,5);
        System.out.println(Arrays.toString(copyOfRangeArray));
    }
}
// [6, 1, 55]
// [6, 1, 55, 21, 33]
```

Arrays.equals(): Java'da iki dizinin eşitliğini kontrol etmek için Arrays.equals() metotu kullanılır.

```java
import java.util.Arrays;

public class Main {
    public static void main(String[] args) {
        int[] list1 = {1, 2, 3};
        int[] list2 = {1, 2, 3};
        int[] list3 = {1, 2, 10};

        System.out.println(Arrays.equals(list1, list2)); // true
        System.out.println(Arrays.equals(list2, list3)); // false
    }
}
```

DİĞER KONULAR

Math Sınıfı ve Metotları

Metot	Açıklaması	Veri Dönüş Tipi
abs(x)	X'in mutlak değerini verir	double|float|int|long
acos(x)	X'in arkkosinüsünü radyan cinsinden verir	double
asin(x)	X'in arksinüsünü radyan cinsinden verir	double
atan(x)	X'in arkini radyan cinsinden verir	double
atan2(y,x)	Dikdörtgen koordinatların (x, y) kutupsal koordinatlara (r, teta) dönüştürülmesinden teta açısını verir.	double
cbrt(x)	X'in küp kökünü verir	double
ceil(x)	En yakın tam sayıya yuvarlanmış x değerini verir	double
copySign(x, y)	İkinci kayan nokta y'nin işaretiyle birlikte ilk kayan nokta x'i verir	double
cos(x)	X'in kosinüsünü verir (x radyan cinsindendir)	double
cosh(x)	Çift değerin hiperbolik kosinüsünü verir	double
exp(x)	e üzeri x değerini verir	double
expm1(x)	e üzeri x 'in tersini verir	double
floor(x)	En yakın tam sayıya yuvarlanmış x değerini verir	double
getExponent(x)	X'te kullanılan yansız üssü verir	int
IEEEremainder(x, y)	IEEE 754 standardında belirtildiği gibi x ve y üzerindeki kalan işlemi hesaplar	double
log(x)	X'in doğal logaritmasını (e tabanında) verir	double
log10(x)	X'in 10 tabanındaki logaritmasını verir	double
log1p(x)	X ve 1 toplamının doğal logaritmasını (e tabanında) verir	double
max(x, y)	En yüksek değere sahip sayıyı verir	double|float|int|long
min(x, y)	En düşük değere sahip sayıyı verir	double|float|int|long
nextAfter(x, y)	X'in y yönünde bitişiğinde değen nokta sayısını verir	double|float
nextUp(x)	Pozitif sonsuzluk yönünde x'e bitişik kayan nokta değerini verir	double|float
pow(x, y)	X'in değerini y'nin üssüne döndürür	double
random()	0 ile 1 arasında rastgele bir sayı verir	double
round(x)	En yakın tam sayıya yuvarlanmış x değerini verir	int
rint()	X'e en yakın ve matematiksel tam sayıya eşit olan çift değeri verir	double
signum(x)	X'in işaretini verir	double
sin(x)	X'in sinüsünü verir (x radyan cinsindendir)	double
sinh(x)	Çift değerin hiperbolik sinüsünü verir	double
sqrt(x)	X'in karekökünü verir	double
tan(x)	Bir açının tanjantını verir	double
tanh(x)	Çift değerin hiperbolik tanjantını verir	double
toDegrees(x)	Radyan cinsinden ölçülen bir açıyı yakl. derece cinsinden ölçülen eşdeğer açı	double
toRadians(x)	Derece cinsinden ölçülen bir açıyı yakl. radyan cinsinden ölçülen açı	double
ulp(x)	X'in en az duyarlıklı (ulp) biriminin boyutunu verir	double|float

Math sınıfının kullanımı şu şekildedir:

```
Math.MethodAdi(Parametreler)
```

String Sınıfı ve Metotları

Metot	Açıklama	Veri Dönüş Tipi
charAt()	Belirtilen indisteki (konum) karakteri verir	char
codePointAt()	Belirtilen indisteki karakterin Unicode'unu verir	int
codePointBefore()	Belirtilen indisteki önceki karakterin Unicode'unu verir	int
codePointCount()	Bu dizenin belirtilen metin aralığındaki Unicode'u döndürür	int
compareTo()	İki dizeyi sözlükbilimsel olarak karşılaştırır	int
compareTolgnoreCase()	Büyük / küçük harf farklılıklarını göz ardı ederek iki dizgeyi sözlükbilimsel olarak karşılaştırır	int
concat()	Başka bir String'in sonuna bir karakter ekler	String
contains()	Bir dizenin bir dizi karakter içerip içermediğini kontrol eder	boolean
contentEquals()	Bir dizenin belirtilen CharSequence veya StringBuffer ile aynı karakter dizisini içerip içermediğini kontrol eder	boolean
copyValueOf()	Karakter dizisinin karakterlerini temsil eden bir Dizi döndürür	String
endsWith()	Bir dizenin belirtilen karakter (ler) ile bitip bitmediğini kontrol eder	boolean
equals()	İki dizgiyi karşılaştırır. Dizeler eşitse doğru, değilse yanlış döndürür	boolean
equalslgnoreCase()	Büyük / küçük harfe ilişkin hususları göz ardı ederek iki dizeyi karşılaştırır	boolean
format()	Belirtilen yerel ayarı, biçim dizesini ve bağımsız değişkenleri kullanarak biçimlendirilmiş bir dize döndürür	String
getBytes()	Bu dizeyi adlandırılmış karakter kümesini kullanarak bir bayt dizisine kodlar, sonucu yeni bir bayt dizisinde saklar	byte[]
getChars()	Karakterleri bir dizeden bir karakter dizisine kopyalar	void
hashCode()	Bir dizenin karma kodunu verir	int
indexOf()	Bir dizede belirtilen karakterlerin ilk bulunan oluşumunun konumunu verir	int
intern()	Aramayı belirtilen dizinde başlatarak, belirtilen karakterin ilk oluşumunun bu dizge içindeki dizini döndürür	String
isEmpty()	Bir dizenin boş olup olmadığını kontrol eder	boolean
lastIndexOf()	Bir dizede belirtilen karakterlerin son bulunan oluşumunun konumunu verir	int
length()	Belirtilen bir dizenin uzunluğunu verir	int
matches()	Normal bir ifadeye karşı bir eşleşme için bir dize arar ve eşleşmeleri döndürür	boolean
offsetByCodePoints()	CodePointOffset kod noktaları tarafından verilen dizinden uzak olan bu Dize içindeki dizini döndürür	int

regionMatches()	İki dizi bölgesinin eşit olup olmadığını test eder	boolean
replace()	Belirli bir değer için bir dize arar ve belirtilen değerlerin değiştirildiği yeni bir dize döndürür	String
replaceFirst()	Verilen normal ifadeyle eşleşen bir alt dizenin ilk oluşumunu verilen değiştirmeyle değiştirir	String
replaceAll()	Verilen normal ifadeyle eşleşen bu dizenin her bir alt dizesini verilen değiştirmeyle değiştirir	String
split()	Bir dizeyi bir alt dizeye böler	String[]
startsWith()	Bir dizenin belirtilen karakterlerle başlayıp başlamadığını kontrol eder	boolean
subSequence()	Bu dizinin bir alt dizisi olan yeni bir karakter dizisi verir	CharSequence
substring()	Karakterleri bir dizeden, belirtilen bir başlangıç konumundan başlayarak ve belirtilen karakter sayısıyla ayıklar	String
toCharArray()	Bu dizeyi yeni bir karakter dizisine dönüştürür	char[]
toLowerCase()	Bir dizeyi küçük harflere dönüştürür	String
toString()	Bir String nesnesinin değerini verir	String
toUpperCase()	Bir dizeyi büyük harflere dönüştürür	String
trim()	Bir dizenin her iki ucundaki boşluğu kaldırır	String
valueOf()	Bir String nesnesinin ilkel değerini verir	String