

# MTK467 Nesneye Yönelik Programlama

---

Ders 5 - Metodlar2


Zümra Kavafoğlu  
*<https://zumrakavafoglu.github.io/>*

# Metodlar: Yerel değişkenler (Local variables)

---

- Metodun içinde tanımlanan değişkenler o metodun yerel değişkenleridir.

```
public static int findMaximum(int number1, int number2, int number3){  
    int maximum;  
    maximum = number1;  
    if(number2 > maximum)  
        maximum = number2;  
    if(number3 > maximum)  
        maximum = number3;  
    return maximum;  
}
```




- Yerel değişkenlere metodun dışından ulaşım yoktur.

# Metodlar: Yerel değişkenler (Local variables)

---

```
public static double calculateAverageBetweenNumbers(int firstNumber, int lastNumber){  
    double sum = 0;  
    double average;  
  
    for(int i=firstNumber; i<=lastNumber; i++){  
        sum += i;  
    }  
  
    average = sum / (lastNumber-firstNumber+1);  
  
    return average;  
}
```



calculateAverageBetweenNumbers  
metodunun yerel değişkenleri

The diagram illustrates the local variables of the `calculateAverageBetweenNumbers` method. A blue arrow points from the local variable declarations `double sum = 0;` and `double average;` to a blue box containing the text "calculateAverageBetweenNumbers metodunun yerel değişkenleri".

# Metodlar: Pass by value

---

- Java dilinde, primitif tipteki metod argümanlarının değerinin metodun içinde değiştirilmesi, o argümanın metod sonrasındaki değerini etkilemez.

# Metodlar: PassByValueDemo.java

---

```
public class PassByValueDemo {  
  
    public static void incrementValues(int number1, int number2){  
  
        number1 ++;  
        number2 ++;  
  
        System.out.println("local number1: " + number1);  
        System.out.println("local number2: " + number2);  
    }  
  
    public static void main(String args[]){  
  
        int number1 = 3;  
        int number2 = 4;  
  
        System.out.println("Before calling IncrementValues ");  
        System.out.println("number1: " + number1);  
        System.out.println("number2: " + number2);  
  
        System.out.println("-----");  
  
        incrementValues(number1,number2);  
  
        System.out.println("-----");  
  
        System.out.println("After calling IncrementValues ");  
        System.out.println("number1: " + number1);  
        System.out.println("number2: " + number2);  
    }  
}
```

# Metodlar: PassByValueDemo.java

---

- Bu metod parametre değerlerini bir arttırıp ekrana yazdırır.

```
public static void incrementValues(int number1, int number2){  
    number1 ++;  
    number2 ++;  
  
    System.out.println("local number1: " + number1);  
    System.out.println("local number2: " + number2);  
}
```

# Metodlar: PassByValueDemo.java

---

- Peki bu metodun çağırılması, metodun argümanlarının main içindeki değerini de değiştirir mi?

```
public static void main(String args[]){  
  
    int number1 = 3;  
    int number2 = 4;  
  
    System.out.println("Before calling IncrementValues ");  
    System.out.println("number1: " + number1);  
    System.out.println("number2: " + number2);  
  
    System.out.println("-----");  
  
    incrementValues(number1,number2);  
  
    System.out.println("-----");  
  
    System.out.println("After calling IncrementValues ");  
    System.out.println("number1: " + number1);  
    System.out.println("number2: " + number2);  
  
}
```

# Metodlar: PassByValueDemo.java

---

- Hayır değıştirmez.

## Çıktı

Before calling IncrementValues

number1: 3

number2: 4

-----

local number1: 4

local number2: 5

-----

After calling IncrementValues

number1: 3

number2: 4



# Metodlar: PassByValueDemo.java

```
public static void incrementValues(int number1, int number2){  
    number1 ++;  
    number2 ++;  
  
    System.out.println("local number1: " + number1);  
    System.out.println("local number2: " + number2);  
}
```

```
public static void main(String args[]){  
    int number1 = 3;  
    int number2 = 4;  
  
    incrementValues(number1, number2);  
}
```

main metodu için  
gerekli alan

number2 4

number1 3

***main metodu çağırıldı***

# Metodlar: PassByValueDemo.java

```
public static void incrementValues(int number1, int number2){  
    number1 ++;  
    number2 ++;  
  
    System.out.println("local number1: " + number1);  
    System.out.println("local number2: " + number2);  
}
```

```
public static void main(String args[]){  
    int number1 = 3;  
    int number2 = 4;  
    incrementValues(number1, number2);  
}
```

incrementValues  
metodu için gerekli  
alan

number2 4

number1 3

main metodu için  
gerekli alan

number2 4

number1 3

**incrementValues metodu  
çağırıldı**

# Metodlar: PassByValueDemo.java

```
public static void incrementValues(int number1, int number2){  
    number1 ++;  
    number2 ++;  
  
    System.out.println("local number1: " + number1);  
    System.out.println("local number2: " + number2);  
}
```

```
public static void main(String args[]){  
  
    int number1 = 3;  
    int number2 = 4;  
  
    incrementValues(number1,number2);  
}
```

incrementValues  
metodu için gerekli  
alan

number2 4

number1 4

main metodu için  
gerekli alan

number2 4

number1 3

**incrementValues metodu  
çağırıldı**

# Metodlar: PassByValueDemo.java

```
public static void incrementValues(int number1, int number2){  
    number1 ++;  
    number2 ++;  
  
    System.out.println("local number1: " + number1);  
    System.out.println("local number2: " + number2);  
}
```

```
public static void main(String args[]){  
  
    int number1 = 3;  
    int number2 = 4;  
  
    incrementValues(number1,number2);  
}
```

incrementValues  
metodu için gerekli  
alan

number2 5

number1 4

main metodu için  
gerekli alan

number2 4

number1 3

**incrementValues metodu  
çağırıldı**

# Metodlar: PassByValueDemo.java

```
public static void incrementValues(int number1, int number2){  
    number1 ++;  
    number2 ++;  
  
    System.out.println("local number1: " + number1);  
    System.out.println("local number2: " + number2);  
}
```

```
public static void main(String args[]){  
    int number1 = 3;  
    int number2 = 4;  
  
    incrementValues(number1,number2);  
}
```

incrementValues  
metodu için gerekli  
alan

number2 5

number1 4

main metodu için  
gerekli alan

number2 4

number1 3

***incrementValues metodu***

# Metodlar: PassByValueDemo.java

```
public static void incrementValues(int number1, int number2){  
    number1 ++;  
    number2 ++;  
  
    System.out.println("local number1: " + number1);  
    System.out.println("local number2: " + number2);  
}
```

```
public static void main(String args[]){  
    int number1 = 3;  
    int number2 = 4;  
  
    incrementValues(number1,number2);  
}
```

main metodu için  
gerekli alan

number2

4

number1

3

*incrementValues metodu  
bitti*

# Boş parametre listesi

---

- Metodlar parametre listeleri boş olacak biçimde tanımlanabilirler.
- Örneğin farklı kullanıcılar için rasgele zarların atıldığı bir zar oyunu programında zar atma görevi için rollTheDice adında bir metod yazabiliriz. Bu metod her çağırılışında 1 ile 6 arasında rasgele bir tamsayı döndürmeli.

```
public static int rollTheDice(){  
    double rand = Math.random();  
  
    int diceNumber = (int)(6 * rand + 1);  
  
    return diceNumber;  
}
```

# Boş parametre listesi

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

```
    int dice1;
```

```
    int dice2;
```

```
    Scanner input = new Scanner(System.in);
```

```
    System.out.println("First player! Press 1 to roll the dice!");
```

```
    if(input.nextInt() == 1){
```

```
        dice1 = rollTheDice();
```

boş parametre listesiyle çağırılır.

```
        System.out.println("Your dice is: " + dice1);
```

```
        System.out.println("Second player! Press 2 to roll the dice!");
```

```
        if(input.nextInt() == 2){
```

```
            dice2 = rollTheDice();
```

parametre listesi boş da olsa  
parantezler yazılmalı

```
            System.out.println("Your dice is: " + dice2);
```

```
            if(dice1 > dice2){
```

```
                System.out.println("First player wins!!");
```

```
            }else if(dice2 > dice1){
```

```
                System.out.println("Second player wins!!");
```

```
            }else{
```

```
                System.out.println("Draw!");
```

```
            }
```

```
        }
```

```
    }
```

```
}
```



# Metod overload

---

- Aynı isme ama farklı parametre listesine sahip metodlar tanımlamaya metod overloading denir.
- Daha önceden bahsettiğimiz verilen 3 tamsayının en büyüğünü bulan findMaximum metodunu hatırlayalım:

```
public static int findMaximum(int number1, int number2, int number3){  
    int maximum;  
  
    maximum = number1;  
  
    if(number2 > maximum)  
        maximum = number2;  
  
    if(number3 > maximum)  
        maximum = number3;  
  
    return maximum;  
}
```

# Metod overload

---

- Aynı metodu 2 parametreyle çalıştırdığımızda da bu sefer girilen 2 değerın maksimumunu bulması için metod overloading yapmalıyız.

metod ismi aynı

parametre listesi farklı

```
public static int findMaximum(int number1, int number2){  
    int maximum;  
  
    maximum = number1;  
  
    if(number2 > maximum)  
        maximum = number2;  
  
    return maximum;  
}
```

# Metod overload

3 adet int tipinde argüman verildiği için bu metodu çağırır

- Programda metoda verilen argümanların metodun çağırıldığını gösterir

```
public static void main(String arg
```

```
    int x = 10;  
    int y = 8;  
    int z = 25;
```

```
    int maxOfThree = findMaximum(x,y,z);
```

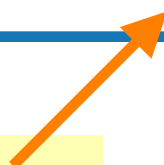
```
    System.out.printf("Maximum of %d, %d and %d is %d", x,y,z,maxOfThree);
```

```
    int maxOfTwo = findMaximum(x,y);
```

```
    System.out.printf("\nMaximum of %d and %d is %d", x,y,maxOfTwo);
```

```
}
```

```
public static int findMaximum(int number1, int number2, int number3){  
    int maximum;  
    maximum = number1;  
    if(number2 > maximum)  
        maximum = number2;  
    if(number3 > maximum)  
        maximum = number3;  
    return maximum;  
}
```



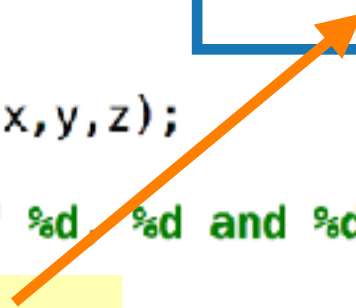
# Metod overload

2 adet int tipinde argüman verildiği için bu metodu çağırır

- Programda metoda verilen a metodun çağırıldığını otom

```
public static void main(String args[]){  
  
    int x = 10;  
    int y = 8;  
    int z = 25;  
  
    int maxOfThree = findMaximum(x,y,z);  
  
    System.out.printf("Maximum of %d, %d and %d is %d", x,y,z,maxOfThree);  
  
    int maxOfTwo = findMaximum(x,y);  
  
    System.out.printf("\nMaximum of %d and %d is %d", x,y,maxOfTwo);  
  
}
```

```
public static int findMaximum(int number1, int number2){  
  
    int maximum;  
  
    maximum = number1;  
  
    if(number2 > maximum)  
        maximum = number2;  
  
    return maximum;  
  
}
```



# Metod overload

---

- findMaximum metodunu başka tipte değişkenlerle de overload edebiliriz

```
public static double findMaximum(double number1, double number2){  
    double maximum;  
  
    maximum = number1;  
  
    if(number2 > maximum)  
        maximum = number2;  
  
    return maximum;  
}
```

---

# Metod overload: Ambiguous invocation (Belirsiz metod çağırma)

---

- Ancak aynı parametre listesine fakat farklı dönüş tipine sahip metod overloading, verilen parametrelerle hangi metodun çağırılacağına dair bir belirsizlik oluşturur. Bu bir derleyici hatasıdır.
- Örneğin bu başlıklara sahip metodların ikisinin de tanımlanması derleyici hatası verir.

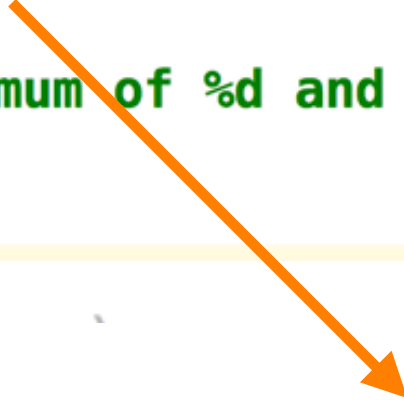
```
public static int findMaximum(int number1, int number2)
```

```
public static double findMaximum(int number1, int number2).
```

# Metod overload: Ambiguous invocation (Belirsiz metod çağırma)

---

```
public static void main(String args[]){  
  
    int x = 10;  
    int y = 8;  
  
    double maxOfTwo = findMaximum(x,y);  
  
    System.out.printf("\nMaximum of %d and %d is %f", x,y,maxOfTwo);  
  
}
```



```
public static int findMaximum(int number1, int number2)
```

# Metod overload: Ambiguous invocation (Belirsiz metod çağırma)

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

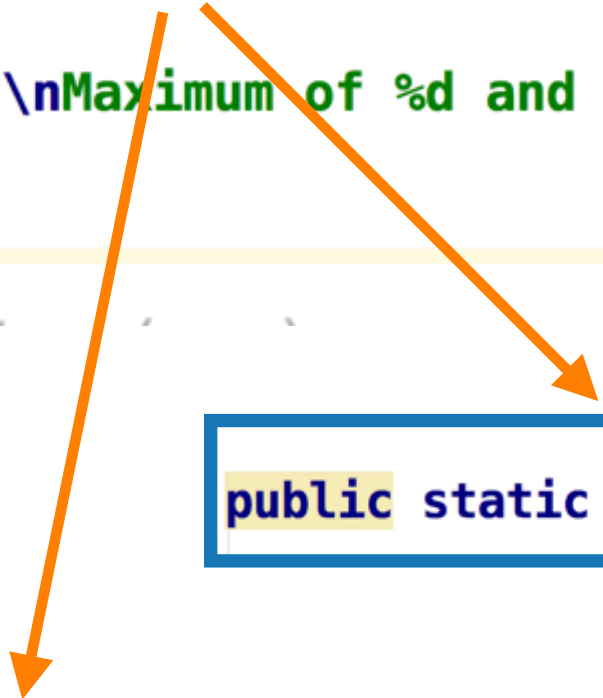
```
    int x = 10;
```

```
    int y = 8;
```

```
    double maxOfTwo = findMaximum(x,y);
```

```
    System.out.printf("\nMaximum of %d and %d is %f", x,y,maxOfTwo);
```

```
}
```



```
public static int findMaximum(int number1, int number2)
```

```
public static double findMaximum(int number1, int number2)
```



# Metod overload: Ambiguous invocation (Belirsiz metod çağırma)

```
public static void main(String args[]){  
  
    int x = 10;  
    int y = 8;  
  
    double maxOfTwo = findMaximum(x,y);  
  
    System.out.printf("\nMaximum of %d and %d is %f", x,y,maxOfTwo);  
  
}
```

hangisi çağırılacak?

`public static int findMaximum(int number1, int number2)`

`public static double findMaximum(int number1, int number2)`

# Başka bir sınıfın statik metodunu çağırma

---

- Kullanıcıdan tenisçiler Federer, Nadal ve Murray'nin puanlarını isteyen ve hangisinin Dünya şampiyonu olduğunu ekrana yazdıran bir program yazınız.
- Kullanıcı 3 tamsayı değer girecektir. Bunların maksimumunu bulmak için daha önce *MaximumFinder* sınıfında tanımladığımız *findMaximum* metodunu kullanabiliriz.

```
import java.util.Scanner;

public class FindTheChampion {

    public static void main(String args[]) {

        int pointsOfFederer;

        int pointsOfNadal;

        int pointsOfMurray;

        Scanner input = new Scanner(System.in);

        System.out.print("Enter points of Federer: ");

        pointsOfFederer = input.nextInt();

        System.out.print("Enter points of Nadal: ");

        pointsOfNadal = input.nextInt();

        System.out.print("Enter points of Murray: ");

        pointsOfMurray = input.nextInt();

        int maxPoint = MaximumFinder.findMaximum(pointsOfFederer, pointsOfMurray, pointsOfNadal);

        if(maxPoint == pointsOfFederer)
            System.out.printf("Federer is the champion with %d points ", maxPoint);

        if(maxPoint == pointsOfNadal)
            System.out.printf("Nadal is the champion with %d points ", maxPoint);

        if(maxPoint == pointsOfMurray)
            System.out.printf("Murray is the champion with %d points ", maxPoint);

    }
}
```

MaximumFinder sınıfının findMaximum yönteminin çağırılması



# Metodlar: Avantajlar

---

- Programda hedeflenen görevleri küçük parçalara ayırarak çözer ve böylece program modüler hale gelir.
- Böl ve yönet yaklaşımına olanak verir.
- Başka programlarda da kullanılabilirler, dolayısıyla kod yeniden kullanımına olanak sağlarlar
- Kodun tekrarlanmasını engellerler.

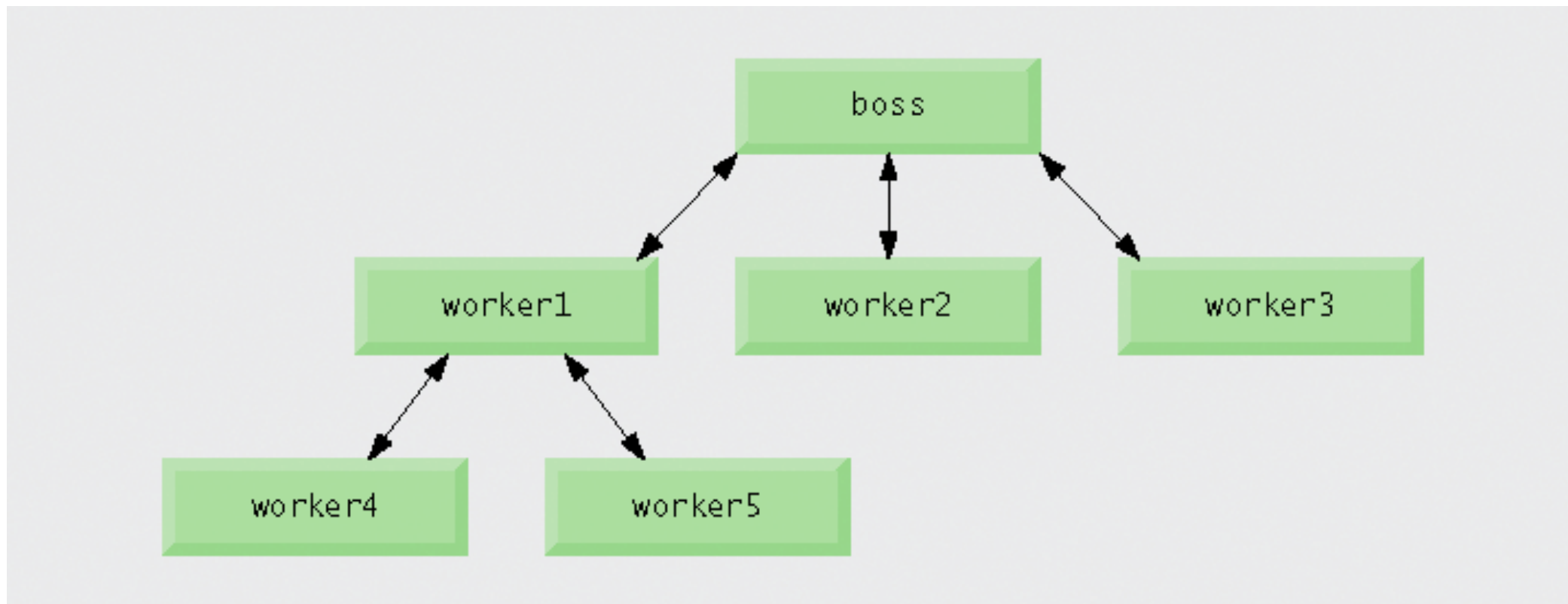
# Metodlar: Modülerlik

---

- Kod yeniden kullanılabilirliğini arttırmak için her metod yalnızca bir adet, iyi tanımlanmış bir görevi çalıştırmaya sınırlandırılmalıdır ve metodun adı bu görevi iyi tanımlayacak bir biçimde seçilmelidir. Bu şekilde metod yazımı, programın daha kolay yazılmasını ve değiştirilmesini ve hatalarının daha kolay ayıklanmasını sağlar.
- Tek bir göreve özelleşmiş küçük bir metod test ve debug(hata ayıklama) bakımından bir çok görevi kapsayan büyük bir metottan daha kullanışlıdır.

# Hiyerarşik patron metod / çalışan metod ilişkisi

---



# Statik metot (Sınıf metodu)

---

- Çağırılması için sınıfın bir nesnesinin oluşturulması gerekmez. Direk sınıfın ismiyle çağırılır.

ClassName.methodName(arguments)

örnek:

Math.sqrt() → sqrt() statik metod

input.nextInt(); → nextInt() statik olmayan metod

# Math Sınıfı

---

- Math sınıfı java.lang paketinin bir parçasıdır. Bu paket derleyici tarafından otomatik olarak import edilir.
- Math sınıfının tüm metotları statiktir.



# Math Sınıfı metodları

Method Name	Description	Returned Value
<code>abs(x)</code>	absolute value	same data type as argument
<code>pow(x1,x2)</code>	$x_1$ raised to the $x_2$ power	double
<code>sqrt(x)</code>	square root of $x$	double
<code>log(x)</code>	natural logarithm of $x$	double
<code>exp(x)</code>	$e$ raised to the $x$ power	double
<code>ceil(x)</code>	smallest integer value that is not less than $x$	double
<code>floor(x)</code>	largest integer value that is not greater than $x$	double
<code>min(x,y)</code>	smaller of its two arguments	same data type as arguments
<code>max(x,y)</code>	larger of its two arguments	same data type as arguments
<code>rint(x)</code>	closest integer value to the argument (in case of two closest integers, the even integer is returned)	double
<code>round(x)</code>	rounded value	integer
<code>random( )</code>	random number between 0.0 inclusive and 1.0 exclusive	double
<code>sin(x)</code>	sine of $x$ ( $x$ in radians)	double
<code>cos(x)</code>	cosine of $x$ ( $x$ in radians)	double
<code>tan(x)</code>	tangent of $x$ ( $x$ in radians)	double
<code>asin(x)</code>	arcsin of $x$	double
<code>acos(x)</code>	arccos of $x$	double
<code>atan(x)</code>	arctan of $x$	double

# Metodlar: Derleyici hatalarına sebep olan yanlışlar

---

- Metod parametrelerinden farklı tipte argümanlarla metodu çağırmak
- Bir metodu sınıf gövdesinin dışında tanımlamak
- Bir metodu başka bir metod gövdesi içinde tanımlamak
- Bir metod deklarasyonunda return tipini yazmamak
- Metod deklarasyonunda parametre listesini kapatan parantezden sonra noktalı virgöl koymak
- Bir metod parametresini yerel bir parametre olarak metod gövdesinde yeniden tanımlamak
- void dışında bir dönüş tipine sahip bir metod gövdesinde return komutunu yazmamak
- return komutundan sonra ifade yazmak.

# Metodlar: Derleyici Hataları

- return komutundan sonra başka bir ifade yazmak bir derleyici hatasıdır.

```
1 ▶ public class ReturnErrorDemo {  
2  
3     public static double calculateAndPrintMax(int x1, int x2){  
4  
5         int max = x1;  
6  
7         if(x2>x1)  
8             max = x2;  
9  
10        return max;  
11  
12        System.out.println("Max is: "+ max);  
13    }  
14  
15    public static void main(String args[]){  
16  
17        int x1 = 3;  
18        int x2 = 5;  
19  
20        calculateAndPrintMax(x1, x2);  
21    }  
22 }  
23 }
```

Unreachable statement

# Metodlar: Derleyici Hataları

---

- Bazı durumlarda mantıksal bir hata olmasa da derleyici return komutuna hiç bir zaman erişilemeyeceğini düşünebilir, aşağıdaki örnekte olduğu gibi.

```
public static int sign(int n) {  
    if (n > 0)  
        return 1;  
    else if (n == 0)  
        return 0;  
    else if (n < 0)  
        return -1;  
}
```

(a)

# Metodlar: Derleyici Hataları

- Bazı durumlarda mantıksal bir hata olmasa da derleyici return komutuna hiç bir zaman erişilemeyeceğini düşünebilir, aşağıdaki örnekte olduğu gibi.
- Bu hatanın düzeltilmesi için (a)'daki kodda **if(n<0)** komutu silinmelidir. Böylece derleyici if bloğunun sonucu ne olursa olsun return komutunun çalıştırılacağını düşünür ve hata vermez.

```
public static int sign(int n) {  
    if (n > 0)  
        return 1;  
    else if (n == 0)  
        return 0;  
    else if (n < 0)  
        return -1;  
}
```

(a)

Should be

```
public static int sign(int n) {  
    if (n > 0)  
        return 1;  
    else if (n == 0)  
        return 0;  
    else  
        return -1;  
}
```

(b)