

# MTK467 Nesneye Yönelik Programlama

---

Hafta 4 - Döngüler

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

# while döngüsü

---

```
while(koşul){  
    döngü ifadeleri  
}
```

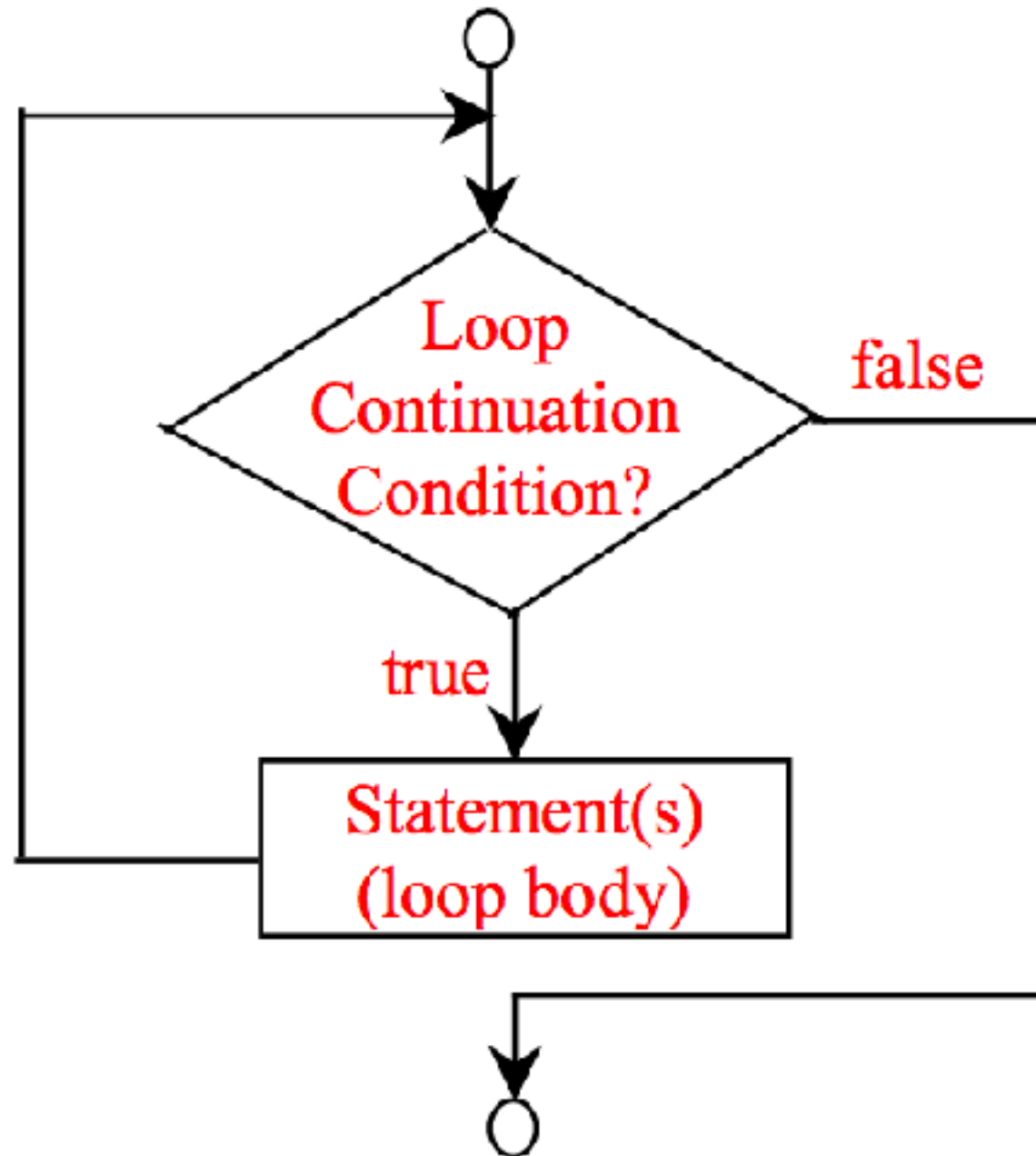
Koşul boolean değerli olmalıdır.

Koşulun değeri true olduğu sürece döngü ifadeleri tekrar tekrar çalıştırılır.

Sonsuz döngüyü engellemek için koşulun eninde sonunda false olacağından emin olun.

# while döngüsü

---



# while döngüsü

---

```
1 ▶ public class WhileDemo {
2 ▶     public static void main(String[] args){
3         int count = 1;
4         while (count < 11) {
5             System.out.println("Count is: " + count);
6             count++;
7         }
8     }
9 }
10
```

```
/Library/Java/JavaVirtualMachines/jdk-9.jdl
Connected to the target VM, address: '127.0.0.1:5055'
Disconnected from the target VM, address: '127.0.0.1:5055'
```

```
Count is: 1
Count is: 2
Count is: 3
Count is: 4
Count is: 5
Count is: 6
Count is: 7
Count is: 8
Count is: 9
Count is: 10
```

```
Process finished with exit code 0
```

# do-while döngüsü

---

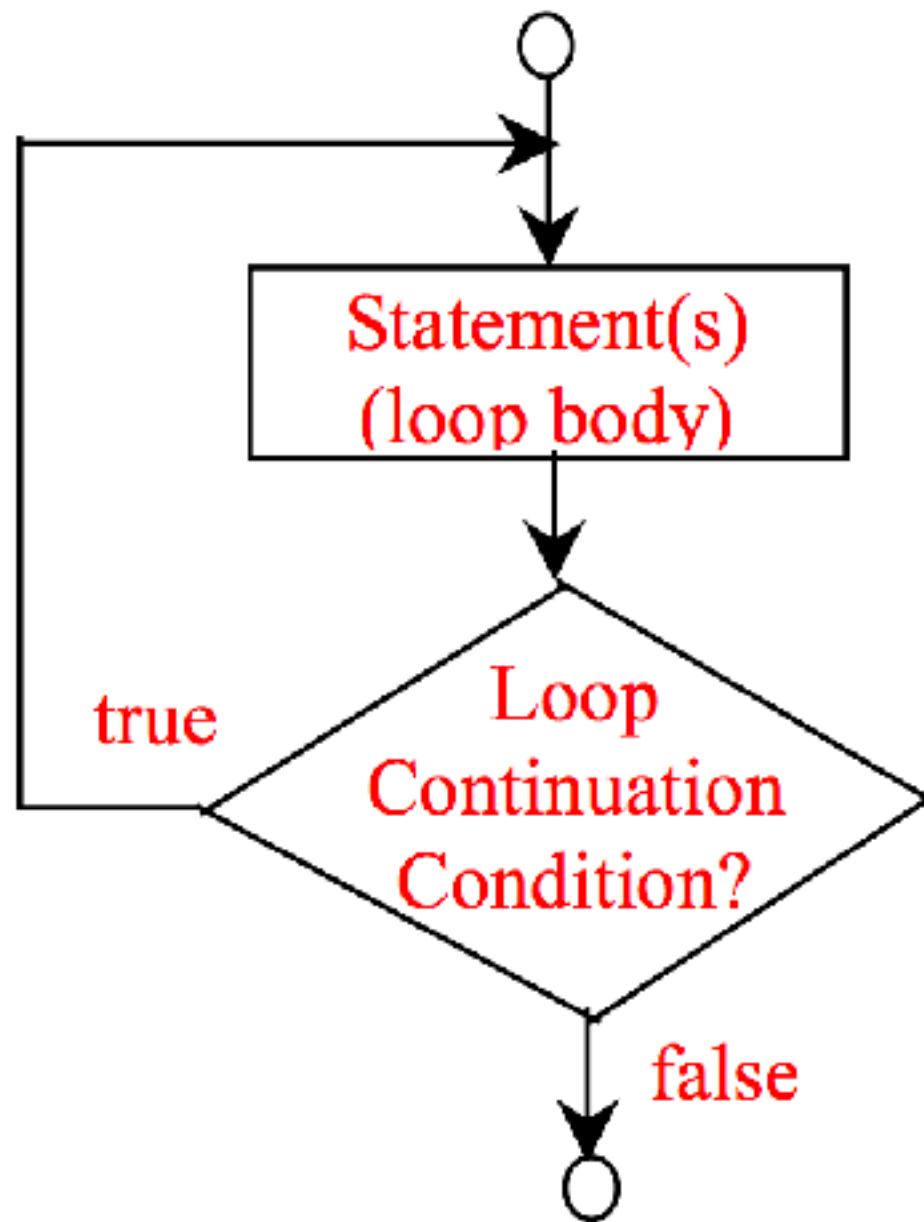
```
do{  
    döngü ifadeleri  
}while(koşul)
```

while döngüsünden farkı:

- while döngüsünde önce koşulun değerine bakılır, true ise döngü ifadeleri çalıştırılır.
- do-while döngüsünde önce döngü ifadeleri çalıştırılır, sonra koşulun değerine bakılır, true ise bir sonraki döngü ifadesi çalıştırılır. **Yani do-while döngüsünde döngü ifadeleri en az bir kez çalıştırılır.**

# do-while döngüsü

---

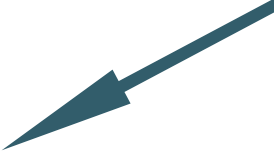


# do-while döngüsü

Kullanıcıdan negatif bir tamsayı girene kadar sürekli tamsayı girmesini istediğimiz bir programı hem while hem de do-while döngüsü kullanarak yazalım:

while döngüsü ile:

```
1  import java.util.Scanner;
2
3  public class DoWhileDemo {
4
5      public static void main(String[] args) {
6
7          int number;
8
9          Scanner input = new Scanner(System.in);
10
11          System.out.print("Input a positive number to continue, a negative number to stop: ");
12          number = input.nextInt();
13
14          while (number >= 0){
15              System.out.print("Input a positive number to continue, a negative number to stop: ");
16              number =input.nextInt();
17          }
18      }
19  }
20 }
```



ilk sayıyı döngünün dışında alıyoruz

# do-while döngüsü

Kullanıcıdan negatif bir tamsayı girene kadar sürekli tamsayı girmesini istediğimiz bir programı hem while hem de do-while döngüsü kullanarak yazalım:

do-while döngüsü ile:

```
1  import java.util.Scanner;
2
3  public class DoWhileDemo {
4
5      public static void main(String[] args) {
6
7          int number;
8
9          Scanner input = new Scanner(System.in);
10
11         do {
12             System.out.print("Input a positive number to continue, a negative number to stop: ");
13             number = input.nextInt();
14         } while(number > 0);
15     }
16 }
```

Döngünün içindeki ifade ilk sefer mutlaka çalıştırılacağı için döngünün dışında bu ifadeleri tekrar yazmaya ihtiyaç yok.



# for döngüsü

---

```
for(initialization; termination; adjustment){  
    döngü ifadeleri  
}
```

- **initialization(ilk değer verme)**: kontrol değişkenine ilk değer verilir. Sadece bir defa döngünün başlangıcında çalıştırılır.
- **termination(sonlandırma)**: değeri false olduğunda döngü sonlanır.
- **adjustment(ayarlama)**: kontrol değişkeninin değerini değiştirir.

# for döngüsü

---

Konsola alt alta yüz defa Welcome to Java yazdırmak için:

kontrol değişkeni

```
for(int i=0; i<100; i++){  
    System.out.println("Welcome to Java");  
}
```

# for döngüsü

---

Konsola alt alta yüz defa Welcome to Java yazdırmak için:

i kontrol değişkenine 0  
ilk değeri verilir.

```
for(int i=0; i<100; i++){  
    System.out.println("Welcome to Java");  
}
```

# for döngüsü

---

Konsola alt alta yüz defa Welcome to Java yazdırmak için:

(i<100) ifadesi false  
olduğunda yani (i>=100)  
olduğunda döngü sonlanır

```
for(int i=0; i<100; i++){  
    System.out.println("Welcome to Java");  
}
```

# for döngüsü

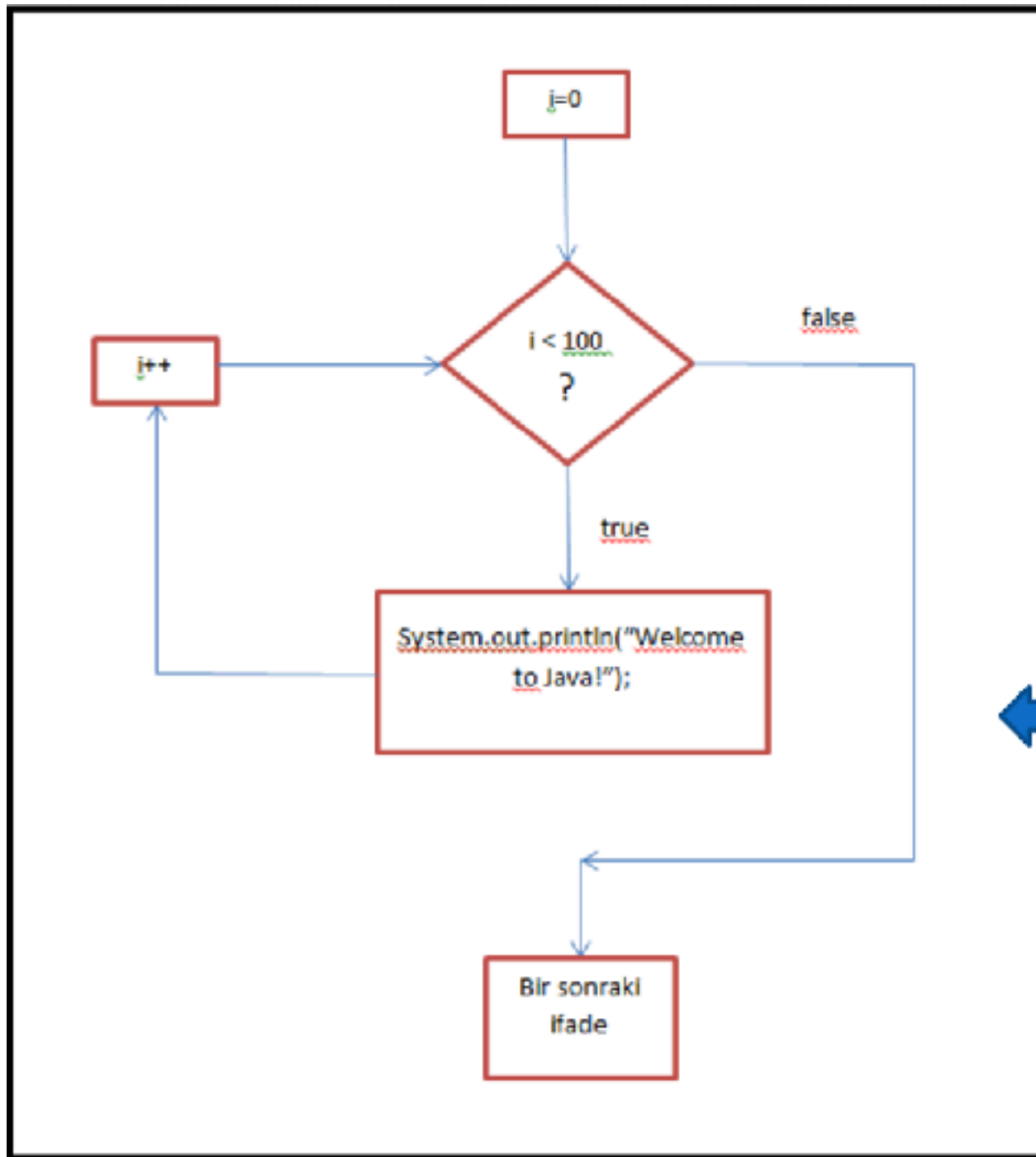
---

Konsola alt alta yüz defa Welcome to Java yazdırmak için:

i değeri döngünün her adımında 1 arttırılır

```
for(int i=0; i<100; i++){  
    System.out.println("Welcome to Java");  
}
```

# for döngüsü



```
for(int i=0; i<100; i++){  
    System.out.println("Welcome to Java");  
}
```

# for, while, do-while

---

Konsola alt alta yüz defa *Welcome to Java* yazdırmak için:

```
for(int i=0; i<100; i++){  
    System.out.println("Welcome to Java");  
}
```

=

```
int i = 0;  
  
while(i<100){  
    System.out.println("Welcome to Java");  
    i++;  
}
```

=

```
int i = 0;  
  
do{  
    System.out.println("Welcome to Java");  
    i++;  
}while(i < 100);
```

## *break ve continue*

---

**break:** Bu komut onu kapsayan en içteki döngüyü hemen sonlandırır.

**continue:** Bu komut onu kapsayan en içteki döngünün o adımını hemen sonlandırır, döngü bir sonraki adımdan devam eder.



# break

```
1
2 ▶ public class TestBreak {
3
4 ▶     public static void main(String[] args) {
5         int sum = 0;
6         int item = 0;
7
8         while(item < 5)
9         {
10             item++;
11             System.out.println("Item is now: "+item);
12             sum += item;
13             System.out.println("Sum is now: "+sum);
14             if(sum >= 6)
15                 break;
16         }
17
18         System.out.println("The last sum is: "+sum);
19     }
20 }
```

sum 6'dan büyük eşit olduğunda döngüden çık.

```
/Library/Java/JavaVirtualMachines/jdk-
Connected to the target VM, address: '
Item is now: 1
Sum is now: 1
Item is now: 2
Sum is now: 3
Item is now: 3
Sum is now: 6
The last sum is: 6
Disconnected from the target VM, address:
Process finished with exit code 0
```

# *break* komutu olmadan aynı döngü

```
1
2 ▶ public class TestBreak {
3
4 ▶     public static void main(String[] args) {
5         int sum = 0;
6         int item = 0;
7
8         while(item < 5)
9         {
10             item++;
11             System.out.println("Item is now: "+item);
12             sum += item;
13             System.out.println("Sum is now: "+sum);
14         }
15
16         System.out.println("The last sum is: "+sum);
17     }
18 }
```

```
/Library/Java/JavaVirt
Connected to the target
Item is now: 1
Sum is now: 1
Item is now: 2
Sum is now: 3
Item is now: 3
Sum is now: 6
Item is now: 4
Sum is now: 10
Item is now: 5
Sum is now: 15
The last sum is: 15
```

# *continue*

```
1
2 ▶ public class TestContinue {
3 ▶     public static void main(String[] args) {
4         for(int i=0; i<8; i++)
5         {
6             if(i==2)
7                 continue;
8             System.out.println("i is "+ i);
9         }
10    }
11
12 }
```

i 2'ye eşit olduğunda gövdedeki takip eden komutlar(bu örnekte print) atlanır ve hemen döngüdeki bir sonraki adıma geçilir

```
/Library Java/JavaVi
i is 0
i is 1
i is 3
i is 4
i is 5
i is 6
i is 7

Process finished with
```

# *continue* olmadan aynı döngü

---

```
1
2 ▶ public class TestContinue {
3 ▶     public static void main(String[] args) {
4         for(int i=0; i<8; i++)
5         {
6             System.out.println("i is "+ i);
7         }
8     }
9 }
```

/Library/Java/

i is 0  
i is 1  
i is 2  
i is 3  
i is 4  
i is 5  
i is 6  
i is 7

Process finish

# İç içe döngülerde break komutu

```
1 ▶ public class BreakNestedLoops {  
2 ▶     public static void main(String args[]){  
3  
4         for(int i=0; i<3; i++){  
5             System.out.println("\nouter value: " + i);  
6             for(int j=0; j<7; j++){  
7                 if(j==5)  
8                     break;  
9                 System.out.println("inner value: " + j);  
10            }  
11        }  
12    }  
13 }
```

ait olduğu en içteki  
döngüden çıkış sağlar

/Library/Java/JavaVi

outer value: 0  
inner value: 0  
inner value: 1  
inner value: 2  
inner value: 3  
inner value: 4

outer value: 1  
inner value: 0  
inner value: 1  
inner value: 2  
inner value: 3  
inner value: 4

outer value: 2  
inner value: 0  
inner value: 1  
inner value: 2  
inner value: 3  
inner value: 4

# Çalışma zamanı hatası(Run-time error)

- Derleyicinin algılayamadığı ama programın çalışması sırasında ortaya çıkan hatalardır.
- Bazı örnekler:
- **InputMismatchException:** Scanner ile kullanıcıdan istenen verinin tipinde uyumsuzluk olduğunda ortaya çıkar.

```
1  import java.util.Scanner;
2
3  ▶ public class RunTimeErrorDemo {
4  ▶      public static void main(String[] args) {
5
6          int inputValue;
7
8          Scanner input = new Scanner(System.in);
9
10         System.out.print("Enter an integer: ");
11
12         inputValue = input.nextInt();
13     }
14 }
```

```
/Library/Java/JavaVirtualMachines/jdk-9.jdk/Contents/Home/bin/j
Connected to the target VM, address: '127.0.0.1:55277', transpo
Enter an integer: 3.7
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:860)
    at java.base/java.util.Scanner.next(Scanner.java:1497)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2161)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2115)
    at RunTimeErrorDemo.main(RunTimeErrorDemo.java:12)
Disconnected from the target VM, address: '127.0.0.1:55277', tr
Process finished with exit code 1
|
```

# Çalışma zamanı hatası(Run-time error)

---

- **ArithmeticException:** Bir sayının 0 ile bölümünde ortaya çıkar.

```
1
2 ▶ public class ArithmeticExceptionDemo {
3 ▶     public static void main(String[] args) {
4
5         int x = 3/0;
6
7         System.out.println(x);
8     }
9 }
```

```
/Library/Java/JavaVirtualMachines/jdk-9.jdk/Contents/Home/bin/java -a
Connected to the target VM, address: '127.0.0.1:57477', transport: 's
Disconnected from the target VM, address: '127.0.0.1:57477', transpo
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at ArithmeticExceptionDemo.main(ArithmeticExceptionDemo.java:5)
```

```
Process finished with exit code 1
```



# Mantık hatası

- Mantık hataları programa yaptırmak istediğiniz bir görevin yanlış yapılması sonucu ortaya çıkar. Derleme zamanında veya çalışma zamanında herhangi bir hata verilmemesine rağmen programın çıktısı istediğiniz çıktı değildir. Bu istemsiz durumun kodun hangi bölümünden kaynaklandığını, yani mantık hatasının yerini bulmak özellikle geniş kapsamlı programlarda çok zor olabilir.
- Belli başlı mantık hatalarına örnekler:
  - Operatör önceliklerinde hata  
 $5+4*3$  ile  $(5+4)*3$  farklı sonuçlar verir
  - Bir koşulun yanlış olduğu halde doğru olduğunu varsaymak
  - Kayar noktalı sayılarla(floating point numbers, double / float) eşitlik kontrolü yapmak
  - İki tamsayı tipinde değişkenin bölümünün ondalık sayı çıkacağını varsaymak
  - Noktalı virgülü yanlış yere koymak



# float ve double tipinde değişkenlerin eşitliğinin karşılaştırılması

---

- floating-point değerleriyle aritmetik işlemler yapılırken bazı küçük yuvarlamalar sonucu teorik olarak eşit olan iki floating-point değişkeni çok küçük bir değer farkı nedeniyle eşit değilmiş gibi anlaşılabilir. Bu yanlış anlaşılma sonucu doğru olması gereken bir koşul yanlış olarak alınır ve programda ayıklanması çok güç mantık hataları oluşur.
- Dolayısıyla floating-point tipi değişkenlerle program yazarken eşitlik yerine yaklaşık eşitliği kontrol etmek daha iyi bir fikirdir. Örneğin double tipinde bir değişken olan x'in 10.0 'a eşit olup olmadığına bakmaktansa ,  $|x-10.0| \leq 1E-10$  eşitsizliğine bakmak daha mantıklıdır.

# float ve double tipinde değişkenlerin eşitliğinin karşılaştırılması

```
1 ▶ public class FloatingPointEquality {
2
3 ▶     public static void main(String[] args)
4     {
5         final double EPS = 1.0E-14;
6
7         double sinX;
8
9         sinX = Math.sin(2*Math.PI);
10
11         System.out.print("Equality test : ");
12
13         if(sinX==0)
14             System.out.println("sin(2*PI) is equal to zero");
15         else
16             System.out.println("sin(2*PI) is NOT equal to zero");
17
18         System.out.print("Approximate equality test : ");
19
20         if(Math.abs(sinX-0.0) < EPS )
21             System.out.println("sin(2*PI) is equal to zero");
22         else
23             System.out.println("sin(2*PI) is NOT equal to zero");
24
25         System.out.println("\nReal values :");
26         System.out.println("sin(2*PI) = " + sinX);
27
28     }
29 }
```

# float ve double tipinde değişkenlerin eşitliğinin karşılaştırılması

---

/Library/Java/JavaVirtualMachines/jdk-9.jdk/Contents/Home

Equality test :  $\sin(2\pi)$  is NOT equal to zero

Approximate equality test :  $\sin(2\pi)$  is equal to zero

Real values :

$\sin(2\pi) = -2.4492935982947064E-16$

Process finished with exit code 0

# Mantık hatası

- Noktalı virgülü yanlış yere koymak

```
1 ▶ public class IfError {  
2 ▶     public static void main(String args[]) {  
3  
4         int x = 10;  
5  
6         if (x < 0);  
7             System.out.println("x is negative");  
8  
9     }  
10 }
```

x, 0'dan küçükse boş satır çalıştır.

```
/Library/Java/JavaVirtualMachines/jdk-9.  
Connected to the target VM, address: '12  
Disconnected from the target VM, address:  
x is negative
```

```
Process finished with exit code 0
```

# Örnek 1

---

Kullanıcının girdiği herhangi bir  $x$  değeri için  $y(x) = \ln(1/(1-x))$  fonksiyonunu hesaplayan Java programını yazınız. Programda fonksiyonun tanımsız olduğu bir nokta girilirse ekrana hata mesajı yazdırılmalıdır.

LnFunction.java

## Örnek 2

---

Kullanıcı tarafında girilen negatif olmayan bir tamsayının faktoriyelini hesaplayan Java programını yazınız.(Kullanıcı negatif olmayan bir tamsayı girene kadar sayı girmesini istemeye devam edilmelidir.)

Factorial.java

## Örnek 3

---

Kullanıcıdan pozitif bir tamsayı isteyen ve 1'den bu sayıya kadar tüm tek sayıların çarpımını hesaplayan bir program yazınız.

OddMultiplication.java