Bölüm 3.1: Kontrol Yapıları

Java ile Nesne Yönelimli Programlama

Sıralı işlemler | if-else | else-if zinciri | İç içe kararlar | switch-case | for döngüsü | while ve do-while | İç içe döngüler | break ve continue

Sercan KÜLCÜ · Giresun Üniversitesi · Elektrik-Elektronik Mühendisliği

GİRİŞ

Öğrenme Çıktıları

Akış kontrolü
Sıralı işlem, karar ve döngü yapılarının programın akışını nasıl yönlendirdiğini açıklamak.

Karar yapıları
`if`, `if-else`, `else if` zinciri, iç içe kararlar ve `&&`, `||` ile birleşik koşullarla koşula bağlı kod yazmak.

switch-case
Bir değeri sabitlerle karşılaştırmak; `break`'in görevini ve "aşağı düşme" davranışını açıklamak.

Döngüler
`for`, `while` ve `do-while` döngülerini doğru yerde seçmek; sonsuz döngü ve sınır hatalarından kaçınmak.

İç içe döngüler
Desen, tablo ve basamak örneklerinde iç içe döngülerin kaç kez döndüğünü hesaplamak.

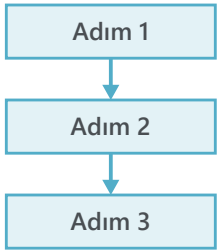
İzleme
Bir programı adım adım izleyerek değişken değerlerini ve ekran çıktısını önceden bulmak; `break` ve `continue`'nun etkisini göstermek.

GİRİŞ

Kontrol Yapıları

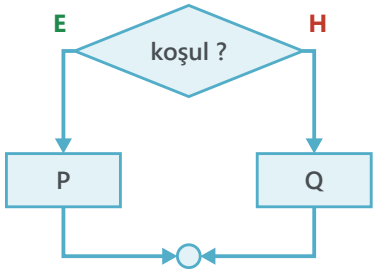
- **Kontrol yapıları** (control structures) programın **akışını**, yani deyimlerin hangi sırayla ve kaç kez çalışacağını yönlendirir.
- Java'da üç temel kontrol yapısı vardır; her program bu üçünün birleşimiyle yazılabilir:

Sıralı işlemler



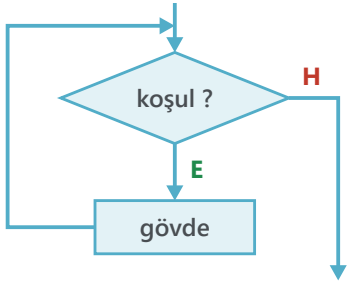
Deyimler yukarıdan aşağıya, birer kez çalışır.

Karar yapıları



Koşula göre iki (veya daha çok) yoldan biri seçilir: `if-else`, `switch`.

Döngüler



Koşul sağlandıkça işlemler tekrarlanır: `for`, `while`, `do-while`.

Önemi: kararlar sayesinde program farklı durumlara ve kullanıcı girdilerine farklı tepki verir; döngüler tekrar eden işleri otomatikleştirir. Programlar böylece esnek ve güçlü olur.

Sıralı İşlemler

Kod, **yukarıdan aşağıya doğru satır satır** sırayla çalışır; bir işlem bitmeden diğeri başlamaz. Kontrol yapısı kullanılmayan her program sıralıdır.

```
▶ int x = 5;
2 int y = 10;
3 int sonuc = x + y;
4 System.out.println(sonuc);
```



İlk satır çalışır: `x` 'e 5 yazılır. Sonraki satır **bu satır bitmeden** başlamaz.

Adım 1 / 4

DEĞİŞKENLER

x = **5**

ÇIKTI

Sıralı İşlemler

Kod, **yukarıdan aşağıya doğru satır satır** sırayla çalışır; bir işlem bitmeden diğeri başlamaz. Kontrol yapısı kullanılmayan her program sıralıdır.

```
1 int x = 5;
▶ int y = 10;
3 int sonuc = x + y;
4 System.out.println(sonuc);
```



İkinci satır: `y` 'ye 10 yazılır.

Adım 2 / 4

DEĞİŞKENLER

`x = 5` `y = 10`

ÇIKTI

Sıralı İşlemler

Kod, **yukarıdan aşağıya doğru satır satır** sırayla çalışır; bir işlem bitmeden diğeri başlamaz. Kontrol yapısı kullanılmayan her program sıralıdır.

```
1 int x = 5;
2 int y = 10;
▶ int sonuc = x + y;
4 System.out.println(sonuc);
```



Üçüncü satır, önceki iki satırın sonuçlarını kullanır: 5 + 10 = 15.

Adım 3 / 4

DEĞİŞKENLER

`x = 5` `y = 10` `sonuc = 15`

ÇIKTI

Sıralı İşlemler

Kod, **yukarıdan aşağıya doğru satır satır** sırayla çalışır; bir işlem bitmeden diğeri başlamaz. Kontrol yapısı kullanılmayan her program sıralıdır.

```
1 int x = 5;
2 int y = 10;
3 int sonuc = x + y;
▶ System.out.println(sonuc);
```



Son satır sonucu ekrana yazar. Satırların sırası değişseydi (ör. 3. satır en başta) program derlenmezdi.

Adım 4 / 4

DEĞİŞKENLER

`x = 5` `y = 10` `sonuc = 15`

ÇIKTI

```
15
```

Karar Verme İşlemi

- **Karar verme**, programın hangi yolu izleyeceğine **çalışma zamanında** karar vermesidir.
- Belirli **koşullara** bağlı olarak farklı işlemleri yürütme yeteneği sağlar.
- Java'da `if-else` ve `switch-case` ile yapılır.
- Koşul, sonucu `true` ya da `false` olan bir **mantıksal ifadedir** (boolean expression).
- Benzetme: yol ayrımındaki tabela. "Hava yağmurluysa şemsiye al, değilse alma."
- Önemi: program farklı şartlara ve kullanıcı girdilerine tepki verir, kodun akışı yönlendirilir, esneklik artar.

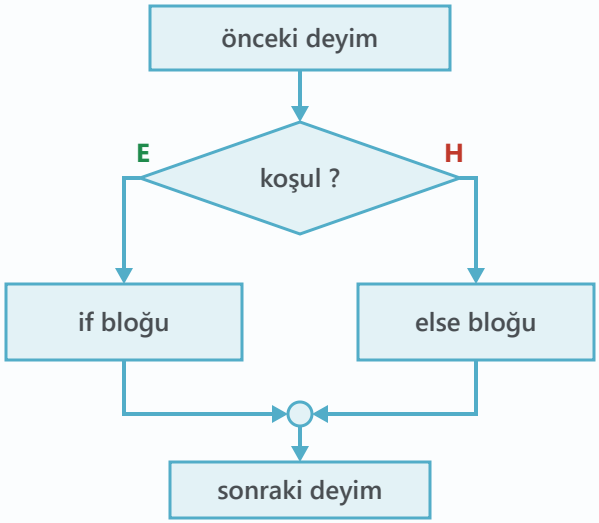
İşleç	Anlamı	Örnek (x = 7)
`==`	eşit mi	`x == 7` → true
`!=`	eşit değil mi	`x != 7` → false
`<` `>`	küçük / büyük	`x < 10` → true
`<=` `>=`	küçük-eşit / büyük-eşit	`x >= 8` → false

`=` atamadır, `==` karşılaştırmadır. `if (x = 7)` derlenmez: koşul `boolean` olmalıdır.

if-else Yapısı

```
if (koşul) {
    // Koşul doğruysa burası çalışır
} else {
    // Koşul yanlışsa burası çalışır
}
```

- Önce **koşul değerlendirilir**.
- Koşul **doğru** (`true`) ise `if` bloğu çalışır, `else` bloğu **atlanır**.
- Koşul **yanlış** (`false`) ise `if` bloğu atlanır, `else` bloğu çalışır.
- `else` kısmı **isteğe bağlıdır** (opsiyonel). İki bloktan **en fazla biri** çalışır.
- Koşul her zaman parantez içinde yazılır: `if (koşul)` .



Yalnız if ve Süslü Parantezler

ELSE OLMADAN IF

```
int sicaklik = 35;
if (sicaklik > 30) {
    System.out.println("Hava sıcak!");
}
System.out.println("Program devam ediyor.");
```

```
Hava sıcak!
Program devam ediyor.
```

Koşul yanlış olsaydı (ör. `sicaklik = 20`) yalnızca "Program devam ediyor." yazılırdı. `if`'ten sonraki deyim **her durumda** çalışır.

TUZAK: SÜSLÜ PARANTEZ YOK

```
int sicaklik = 35;
if (sicaklik > 40)
    System.out.println("Çok sıcak!");
    System.out.println("Bol su için.");
```

```
Bol su için.
```

Parantez yoksa `if`'e **yalnızca ilk deyim** bağlıdır. Girinti Java için anlamsızdır: "Bol su için." koşuldan bağımsız çalışır. Bu yüzden tek satırlık blokta bile `{ }` kullanılır.

Örnek: Yaş Kontrolü

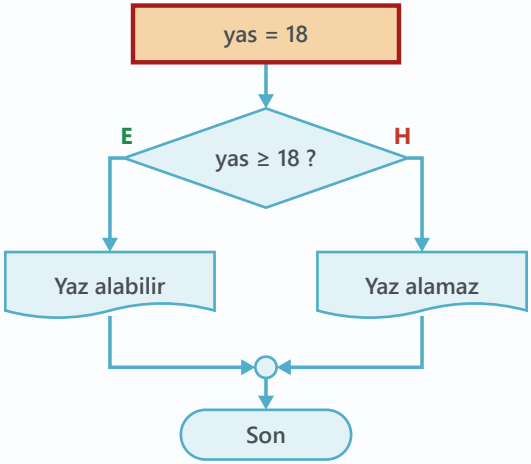
```
▶ int yas = 18;
2 if (yas >= 18) {
3     System.out.println("Ehliyet alabilirsiniz.");
4 } else {
5     System.out.println("Ehliyet alamazsınız.");
6 }
```

DEĞİŞKENLER

`yas = 18`

ÇIKTI





`yas` değişkenine 18 atanır.

Adım 1 / 4

Örnek: Yaş Kontrolü

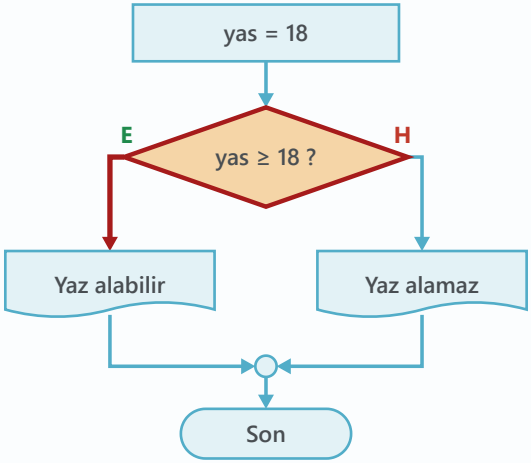
```
1 int yas = 18;
▶ if (yas >= 18) {
3     System.out.println("Ehliyet alabilirsiniz.");
4 } else {
5     System.out.println("Ehliyet alamazsınız.");
6 }
```

DEĞİŞKENLER

`yas = 18`

ÇIKTI





Koşul değerlendirilir: 18 >= 18 → **true**. Akış “E” (evet) koluna gider.

Örnek: Yaş Kontrolü

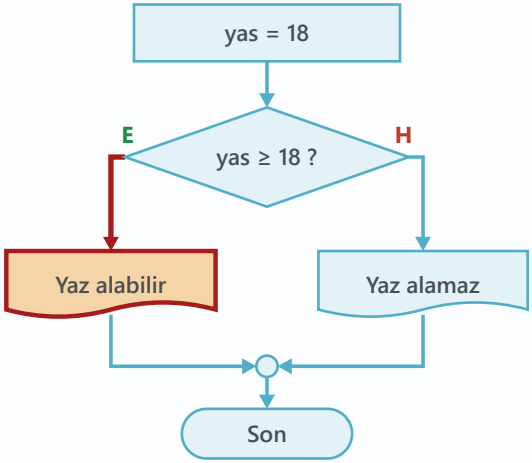
```
1 int yas = 18;
2 if (yas >= 18) {
      System.out.println("Ehliyet alabilirsiniz.");
4 } else {
5     System.out.println("Ehliyet alamazsınız.");
6 }
```

DEĞİŞKENLER

`yas = 18`

ÇIKTI

```
Ehliyet alabilirsiniz.
```



`if` bloğu çalışır ve mesaj yazdırılır.

Örnek: Yaş Kontrolü

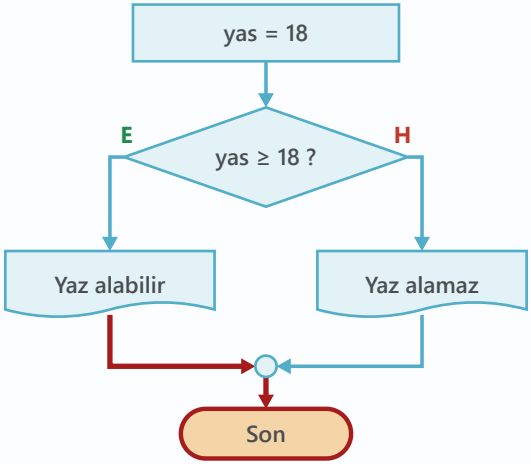
```
1 int yas = 18;
2 if (yas >= 18) {
3     System.out.println("Ehliyet alabilirsiniz.");
4 } else {
5     System.out.println("Ehliyet alamazsınız.");
▶ }
```

DEĞİŞKENLER

`yas = 18`

ÇIKTI

```
Ehliyet alabilirsiniz.
```



`else` bloğu (satır 5) **atlanır**. `yas = 16` olsaydı koşul false olur, yalnızca "Ehliyet alamazsınız." yazılırdı.

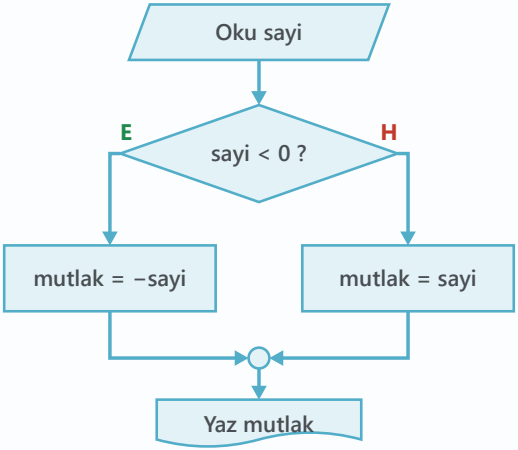
Örnek: Mutlak Değer

```
Scanner scanner = new Scanner(System.in);
System.out.print("Bir sayı girin: ");
double sayi = scanner.nextDouble();
double mutlakDeger;
if (sayi < 0) {
    mutlakDeger = -sayi;
} else {
    mutlakDeger = sayi;
}
System.out.println("Mutlak değer: " + mutlakDeger);
```

DEĞİŞKENLER

—

ÇIKTI



Klavyeden okumak için bir `Scanner` nesnesi oluşturulur (dosyanın başında `import java.util.Scanner;` gerekir).

Örnek: Mutlak Değer

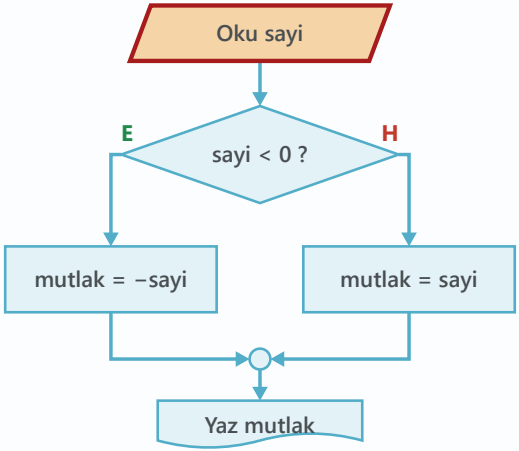
```
1  Scanner scanner = new Scanner(System.in);
▶  System.out.print("Bir sayı girin: ");
3  double sayi = scanner.nextDouble();
4  double mutlakDeger;
5  if (sayi < 0) {
6      mutlakDeger = -sayi;
7  } else {
8      mutlakDeger = sayi;
9  }
10 System.out.println("Mutlak değer: " + mutlakDeger);
```

DEĞİŞKENLER

—

ÇIKTI

```
Bir sayı girin:
```



Kullanıcıya ne gireceği söylenir. `print` satır atlamaz; imleç aynı satırda bekler.

Örnek: Mutlak Değer

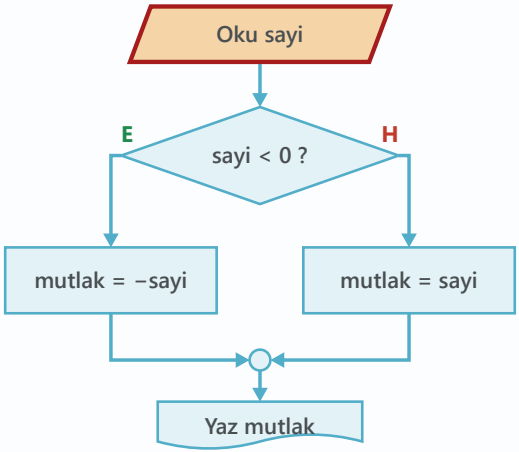
```
1  Scanner scanner = new Scanner(System.in);
2  System.out.print("Bir sayı girin: ");
▶  double sayi = scanner.nextDouble();
4  double mutlakDeger;
5  if (sayi < 0) {
6      mutlakDeger = -sayi;
7  } else {
8      mutlakDeger = sayi;
9  }
10 System.out.println("Mutlak değer: " + mutlakDeger);
```

DEĞİŞKENLER

`sayi = -7.5`

ÇIKTI

```
Bir sayı girin: -7,5
```



Kullanıcı **-7,5** yazar. Türkçe bölge ayarlı bilgisayarda ondalık ayırıcı virgüldür; `-7.5` yazmak `InputMismatchException` verir.

Örnek: Mutlak Değer

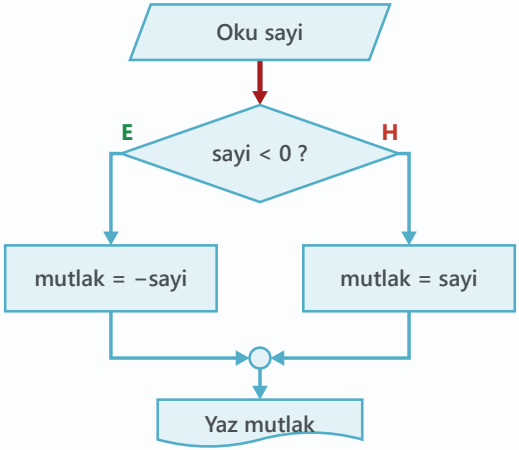
```
1  Scanner scanner = new Scanner(System.in);
2  System.out.print("Bir sayı girin: ");
3  double sayi = scanner.nextDouble();
▶  double mutlakDeger;
5  if (sayi < 0) {
6      mutlakDeger = -sayi;
7  } else {
8      mutlakDeger = sayi;
9  }
10 System.out.println("Mutlak değer: " + mutlakDeger);
```

DEĞİŞKENLER

`sayi = -7.5` `mutlakDeger = ?`

ÇIKTI

```
Bir sayı girin: -7,5
```



`mutlakDeger` tanımlanır ama henüz değeri yoktur.

Örnek: Mutlak Değer

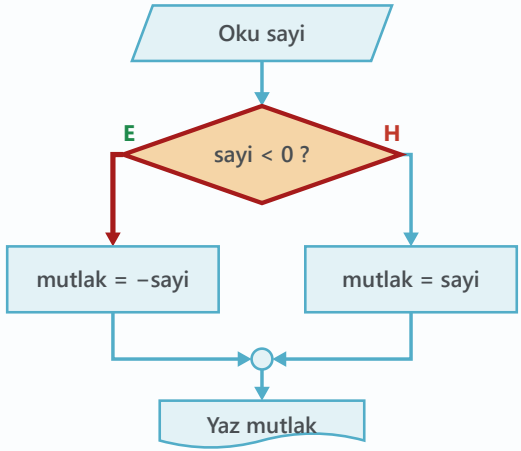
```
1  Scanner scanner = new Scanner(System.in);
2  System.out.print("Bir sayı girin: ");
3  double sayi = scanner.nextDouble();
4  double mutlakDeger;
▶  if (sayi < 0) {
6      mutlakDeger = -sayi;
7  } else {
8      mutlakDeger = sayi;
9  }
10 System.out.println("Mutlak değer: " + mutlakDeger);
```

DEĞİŞKENLER

`sayi = -7.5` `mutlakDeger = ?`

ÇIKTI

```
Bir sayı girin: -7,5
```



Koşul: -7.5 < 0 → **true**.

Örnek: Mutlak Değer

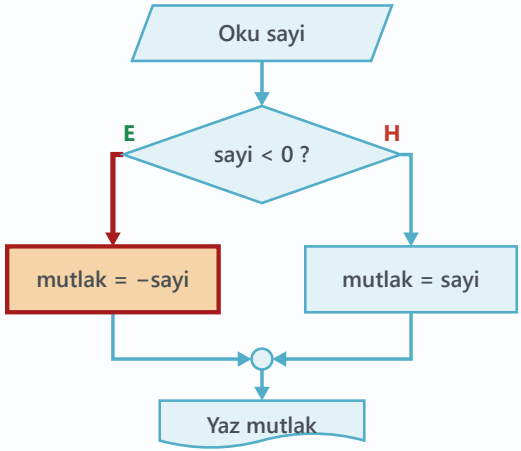
```
1  Scanner scanner = new Scanner(System.in);
2  System.out.print("Bir sayı girin: ");
3  double sayi = scanner.nextDouble();
4  double mutlakDeger;
5  if (sayi < 0) {
▶      mutlakDeger = -sayi;
7  } else {
8      mutlakDeger = sayi;
9  }
10 System.out.println("Mutlak değer: " + mutlakDeger);
```

DEĞİŞKENLER

`sayi = -7.5` `mutlakDeger = 7.5`

ÇIKTI

```
Bir sayı girin: -7,5
```



`-sayi` = −(−7.5) = 7.5 atanır.
`else` bloğu atlanacak.

Örnek: Mutlak Değer

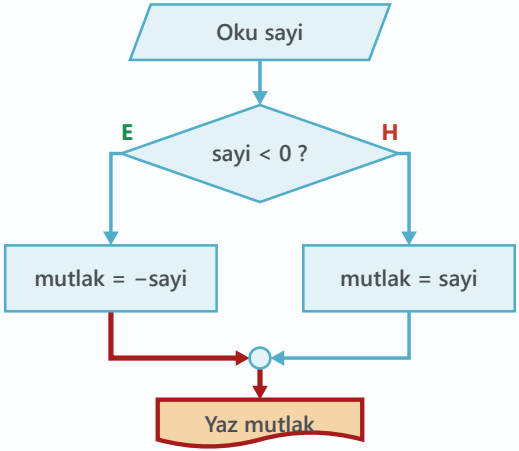
```
1 Scanner scanner = new Scanner(System.in);
2 System.out.print("Bir sayı girin: ");
3 double sayi = scanner.nextDouble();
4 double mutlakDeger;
5 if (sayi < 0) {
6     mutlakDeger = -sayi;
7 } else {
8     mutlakDeger = sayi;
9 }
▶ System.out.println("Mutlak değer: " + mutlakDeger);
```

DEĞİŞKENLER

`sayi = -7.5` `mutlakDeger = 7.5`

ÇIKTI

```
Bir sayı girin: -7,5
Mutlak değer: 7.5
```



Sonuç yazılır. `println` ise `double` değeri her zaman noktayla yazar: 7.5.

else-if Zinciri

```java
import java.util.Scanner;

public class PozitifNegatif {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Bir sayı girin: ");
        double sayi = scanner.nextDouble();
        if (sayi > 0) {
            System.out.println("Sayı pozitif.");
        } else if (sayi < 0) {
            System.out.println("Sayı negatif.");
        } else {
            System.out.println("Sayı sıfır.");
        }
    }
}
```

- İkiden çok seçenek varsa `else if` ile koşullar **art arda** sınanır.
- Koşullar **yukarıdan aşağıya** denenir; **ilk doğru** koşulun bloğu çalışır, gerisi atlanır.
- Hiçbiri doğru değilse son `else` çalışır (varsa).

ÜÇ AYRI ÇALIŞTIRMA

```
Bir sayı girin: 4
Sayı pozitif.
```

```
Bir sayı girin: -2
Sayı negatif.
```

```
Bir sayı girin: 0
Sayı sıfır.
```

else-if Zinciri: Harf Notu

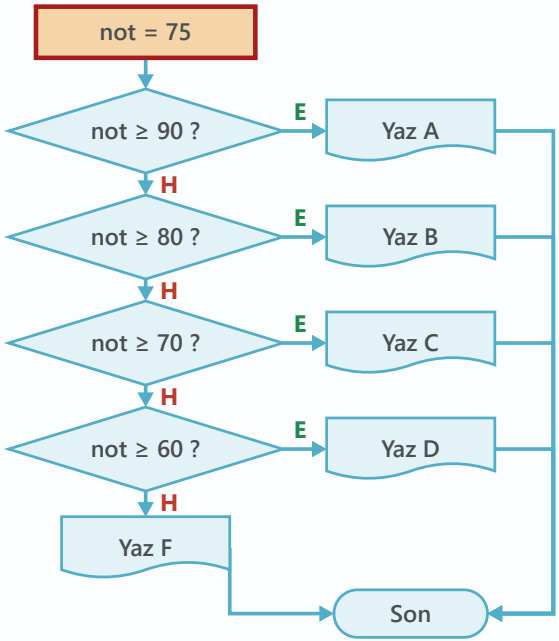
```
▶ int not = 75;
2 if (not >= 90) {
3     System.out.println("A");
4 } else if (not >= 80) {
5     System.out.println("B");
6 } else if (not >= 70) {
7     System.out.println("C");
8 } else if (not >= 60) {
9     System.out.println("D");
10 } else {
11     System.out.println("F");
12 }
```

DEĞİŞKENLER

`not = 75`

ÇIKTI





`not` = 75.

Adım 1 / 6

else-if Zinciri: Harf Notu

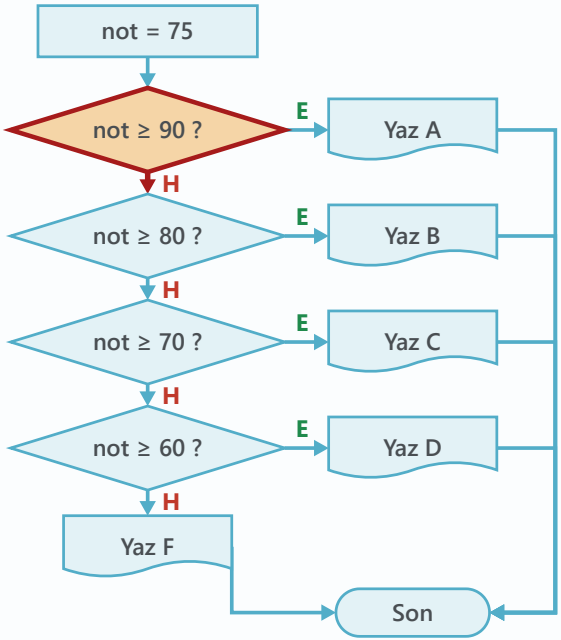
```
1  int not = 75;
2  if (not >= 90) {
3      System.out.println("A");
4  } else if (not >= 80) {
5      System.out.println("B");
6  } else if (not >= 70) {
7      System.out.println("C");
8  } else if (not >= 60) {
9      System.out.println("D");
10 } else {
11     System.out.println("F");
12 }
```

DEĞİŞKENLER

`not = 75`

ÇIKTI





75 >= 90 → false. Sonraki `else if` 'e geçilir.

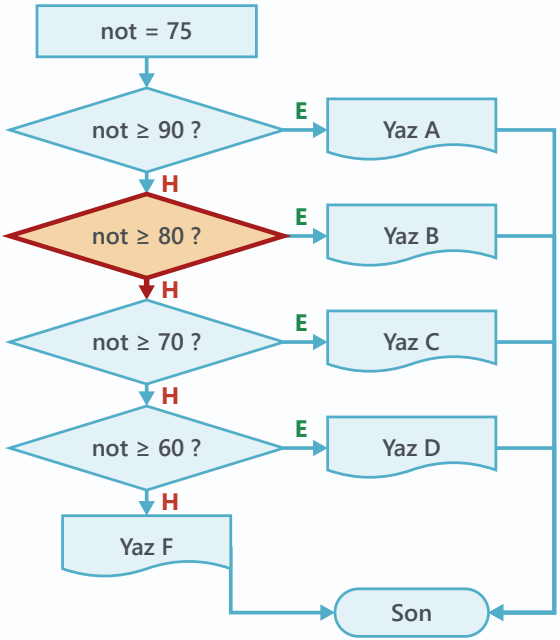
else-if Zinciri: Harf Notu

```
1  int not = 75;
2  if (not >= 90) {
3      System.out.println("A");
▶  } else if (not >= 80) {
5      System.out.println("B");
6  } else if (not >= 70) {
7      System.out.println("C");
8  } else if (not >= 60) {
9      System.out.println("D");
10 } else {
11     System.out.println("F");
12 }
```

DEĞİŞKENLER

`not = 75`

ÇIKTI



75 >= 80 → false. Sonraki koşula geçilir.

else-if Zinciri: Harf Notu

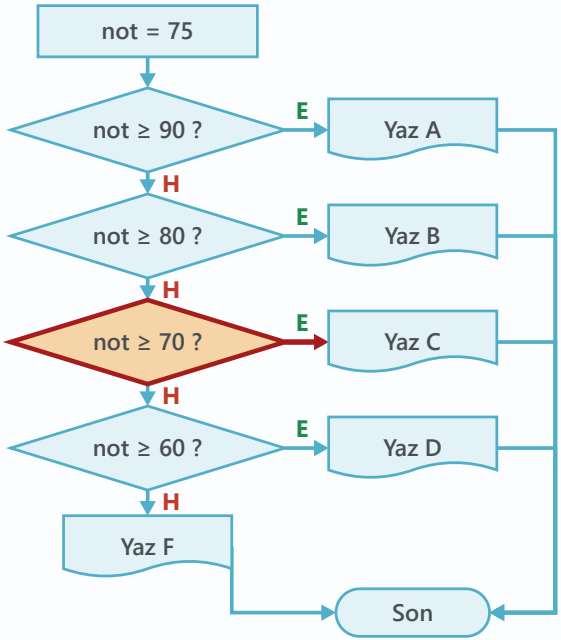
```
1  int not = 75;
2  if (not >= 90) {
3      System.out.println("A");
4  } else if (not >= 80) {
5      System.out.println("B");
▶  } else if (not >= 70) {
7      System.out.println("C");
8  } else if (not >= 60) {
9      System.out.println("D");
10 } else {
11     System.out.println("F");
12 }
```

DEĞİŞKENLER

`not = 75`

ÇIKTI





75 >= 70 → **true**: ilk doğru koşul bulundu.

else-if Zinciri: Harf Notu

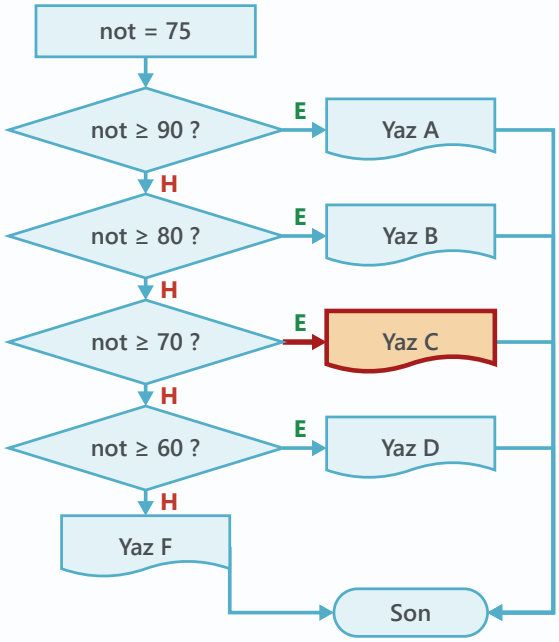
```
1  int not = 75;
2  if (not >= 90) {
3      System.out.println("A");
4  } else if (not >= 80) {
5      System.out.println("B");
6  } else if (not >= 70) {
▶      System.out.println("C");
8  } else if (not >= 60) {
9      System.out.println("D");
10 } else {
11     System.out.println("F");
12 }
```

DEĞİŞKENLER

`not = 75`

ÇIKTI

```
C
```



"C" yazılır.

else-if Zinciri: Harf Notu

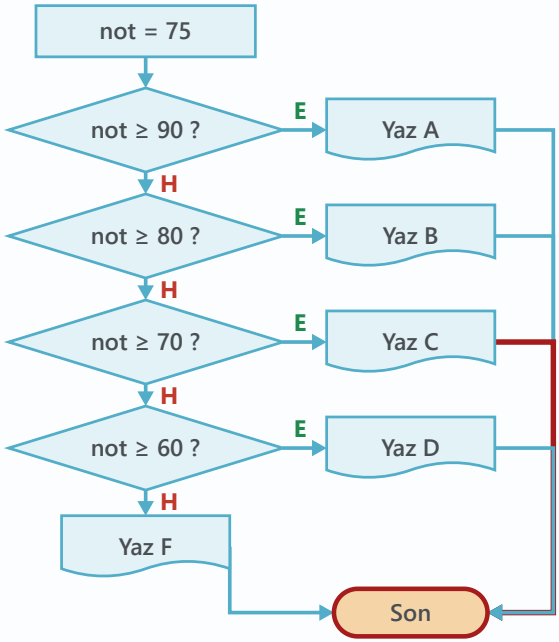
```
1  int not = 75;
2  if (not >= 90) {
3      System.out.println("A");
4  } else if (not >= 80) {
5      System.out.println("B");
6  } else if (not >= 70) {
7      System.out.println("C");
8  } else if (not >= 60) {
9      System.out.println("D");
10 } else {
11     System.out.println("F");
12 }
```

DEĞİŞKENLER

`not = 75`

ÇIKTI

```
C
```

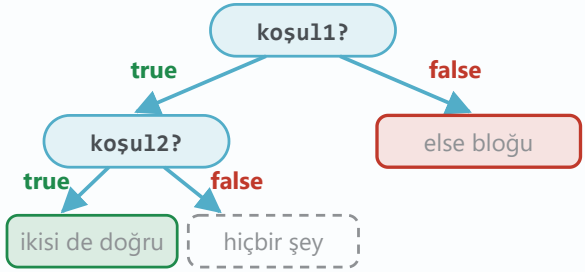


Kalan koşullar (`not >= 60` dahil) **hiç sınanmaz**. Sıra önemlidir: `>= 60` en üstte olsaydı 75 için “D” yazılırdı.

İç İçe Kararlar (Nesting)

- Bir karar yapısının bloğu içinde **başka bir karar yapısı** kullanılabilir.
- Daha karmaşık koşullar ve işlemler için kullanışlıdır: dış koşul sağlandıktan sonra ikinci bir soru sorulur.
- İç `if` yalnızca dış koşul **doğruysa** sınanır.
- Çok derin iç içe yapı kodun **okunabilirliğini olumsuz etkiler**; düzgün girinti kullanılır, mümkünse `&&` veya `else if` ile düzleştirilir: `if (yas >= 18) { if (ehliyetVar) … }` ile `if (yas >= 18 && ehliyetVar) …` aynı işi yapar.
- Aralık denetimi de `&&` ile yazılır: `not >= 60 && not < 90` ; `60 <= not < 90` derlenmez.

```
if (koşul1) {
    if (koşul2) {
        // koşul1 VE koşul2 doğru
    }
} else {
    // koşul1 yanlış
}
```



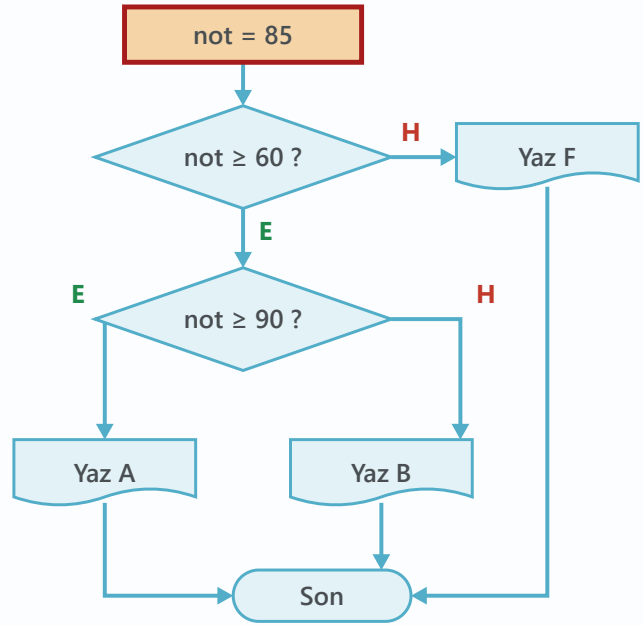
İç İçe Karar: Harf Notu

```
▶ int not = 85;
2 if (not >= 60) {
3     if (not >= 90) {
4         System.out.println("Harf notu: A");
5     } else {
6         System.out.println("Harf notu: B");
7     }
8 } else {
9     System.out.println("Harf notu: F");
10 }
```

DEĞİŞKENLER

`not = 85`

ÇIKTI



`not` = 85.

Adım 1 / 5

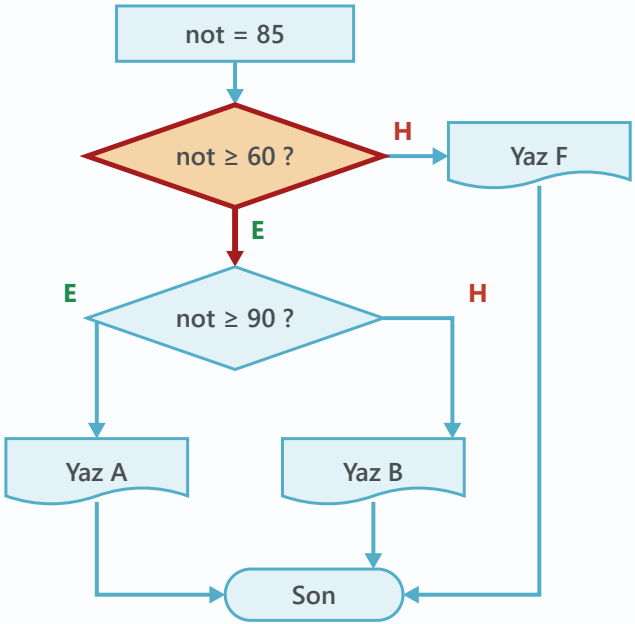
İç İçe Karar: Harf Notu

```
1  int not = 85;
▶ if (not >= 60) {
3      if (not >= 90) {
4          System.out.println("Harf notu: A");
5      } else {
6          System.out.println("Harf notu: B");
7      }
8  } else {
9      System.out.println("Harf notu: F");
10 }
```

DEĞİŞKENLER

`not = 85`

ÇIKTI



Dış koşul: 85 >= 60 → **true**. Dış `if` bloğuna girilir.

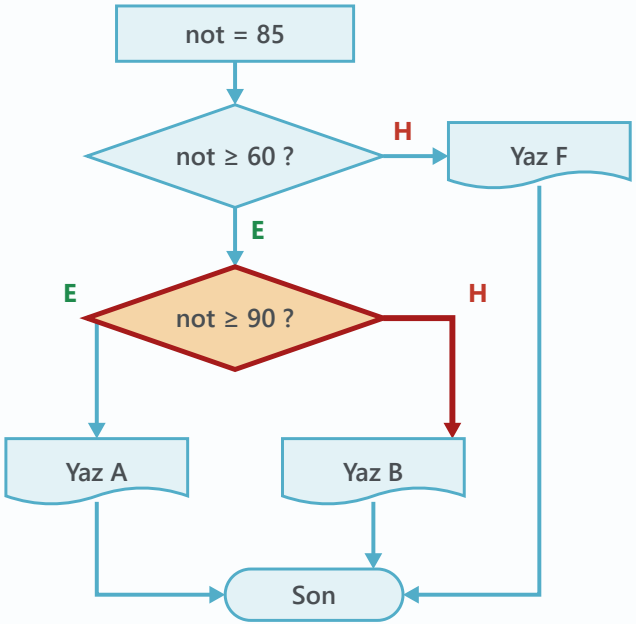
İç İçe Karar: Harf Notu

```
1  int not = 85;
2  if (not >= 60) {
▶      if (not >= 90) {
4          System.out.println("Harf notu: A");
5      } else {
6          System.out.println("Harf notu: B");
7      }
8  } else {
9      System.out.println("Harf notu: F");
10 }
```

DEĞİŞKENLER

`not = 85`

ÇIKTI



İç koşul: 85 >= 90 → false. İç `else` 'e gidilir.

İç İçe Karar: Harf Notu

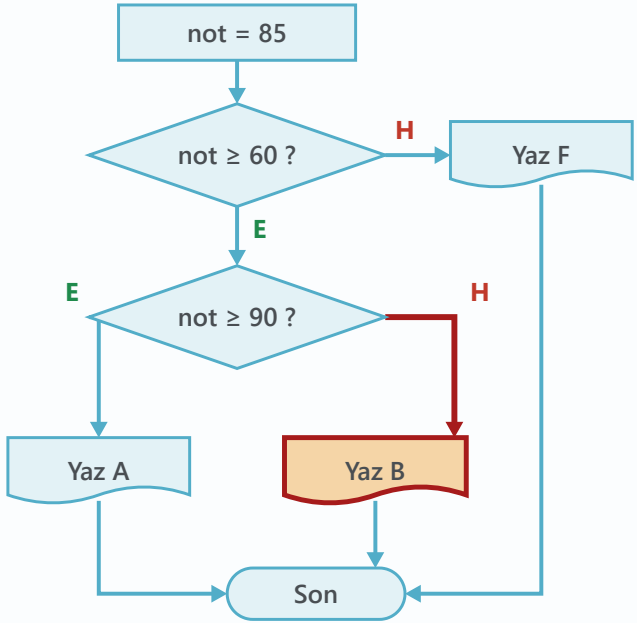
```
1  int not = 85;
2  if (not >= 60) {
3      if (not >= 90) {
4          System.out.println("Harf notu: A");
5      } else {
▶          System.out.println("Harf notu: B");
7      }
8  } else {
9      System.out.println("Harf notu: F");
10 }
```

DEĞİŞKENLER

`not = 85`

ÇIKTI

```
Harf notu: B
```



"Harf notu: B" yazılır.

İç İçe Karar: Harf Notu

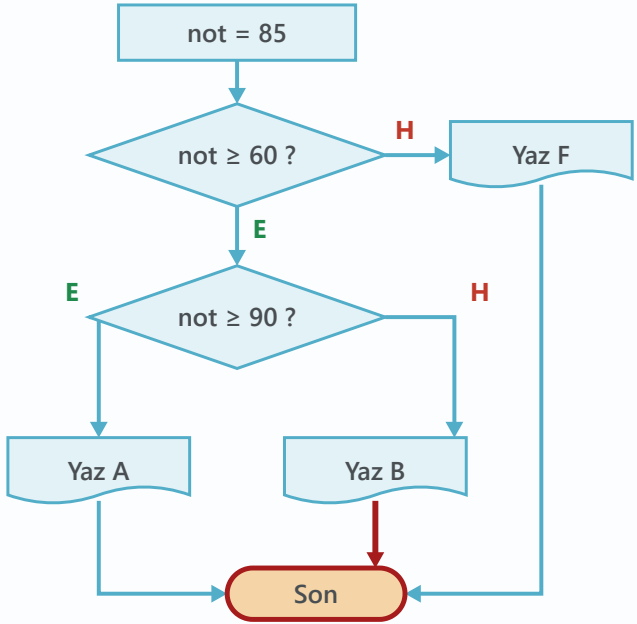
```
1 int not = 85;
2 if (not >= 60) {
3     if (not >= 90) {
4         System.out.println("Harf notu: A");
5     } else {
6         System.out.println("Harf notu: B");
7     }
8 } else {
9     System.out.println("Harf notu: F");
▶ }
```

DEĞİŞKENLER

`not = 85`

ÇIKTI

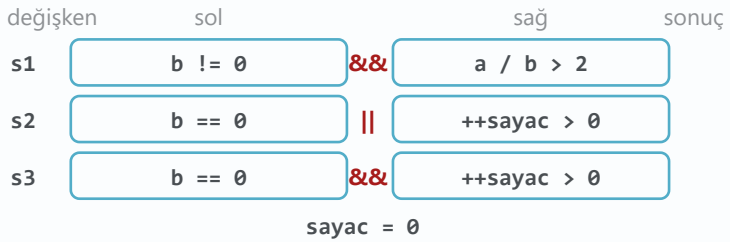
```
Harf notu: B
```



Dış `else` (satır 9) atlanır. `not = 45` olsaydı iç `if` hiç sınanmaz, doğrudan F yazılırdı.

Kısa Devre: Kodun Adım Adım İzlenmesi

```
1 int a = 10, b = 0, sayac = 0;
2 boolean s1 = b != 0 && a / b > 2;
3 boolean s2 = b == 0 || ++sayac > 0;
4 boolean s3 = b == 0 && ++sayac > 0;
5 System.out.println(s1 + " " + s2 + " " +
  s3);
6 System.out.println("sayac = " + sayac);
```



`b = 0` : `a / b` hesaplanırsa sıfıra bölme hatası (`ArithmeticException`) oluşur. `sayac` sağ tarafların çalışıp çalışmadığını sayacak.

Adım 1 / 9

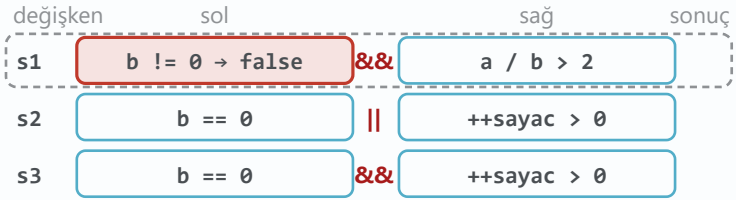
DEĞİŞKENLER

`a = 10` `b = 0` `sayac = 0`

ÇIKTI

Kısa Devre: Kodun Adım Adım İzlenmesi

```
1 int a = 10, b = 0, sayac = 0;
▶ boolean s1 = b != 0 && a / b > 2;
3 boolean s2 = b == 0 || ++sayac > 0;
4 boolean s3 = b == 0 && ++sayac > 0;
5 System.out.println(s1 + " " + s2 + " " +
  s3);
6 System.out.println("sayac = " + sayac);
```



`&&` önce **sol** tarafı hesaplar: `b != 0` → false.

Adım 2 / 9

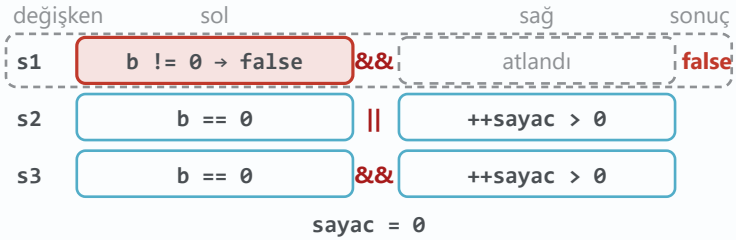
DEĞİŞKENLER

`a = 10` `b = 0` `sayac = 0`

ÇIKTI

Kısa Devre: Kodun Adım Adım İzlenmesi

```
1 int a = 10, b = 0, sayac = 0;
▶ boolean s1 = b != 0 && a / b > 2;
3 boolean s2 = b == 0 || ++sayac > 0;
4 boolean s3 = b == 0 && ++sayac > 0;
5 System.out.println(s1 + " " + s2 + " " +
  s3);
6 System.out.println("sayac = " + sayac);
```



Sol false ise `&&` 'nin sonucu zaten false: `a / b > 2` **hiç hesaplanmaz**, hata oluşmaz. `s1 = false`.

Adım 3 / 9

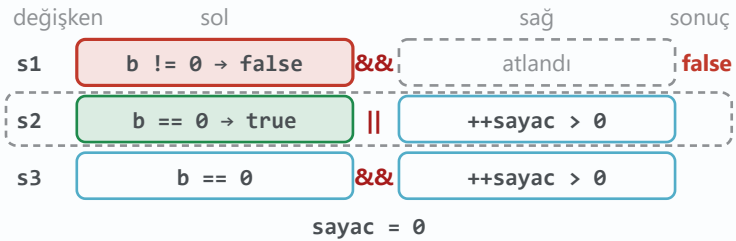
DEĞİŞKENLER

`a = 10` `b = 0` `sayac = 0` `s1 = false`

ÇIKTI

Kısa Devre: Kodun Adım Adım İzlenmesi

```
1 int a = 10, b = 0, sayac = 0;
2 boolean s1 = b != 0 && a / b > 2;
▶ boolean s2 = b == 0 || ++sayac > 0;
4 boolean s3 = b == 0 && ++sayac > 0;
5 System.out.println(s1 + " " + s2 + " " +
  s3);
6 System.out.println("sayac = " + sayac);
```



`||` için sol taraf: `b == 0` → true.

Adım 4 / 9

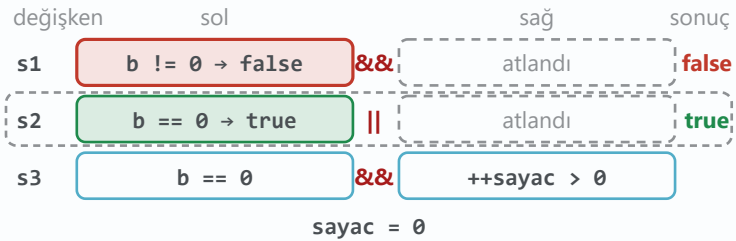
DEĞİŞKENLER

`a = 10` `b = 0` `sayac = 0` `s1 = false`

ÇIKTI

Kısa Devre: Kodun Adım Adım İzlenmesi

```
1 int a = 10, b = 0, sayac = 0;
2 boolean s1 = b != 0 && a / b > 2;
▶ boolean s2 = b == 0 || ++sayac > 0;
4 boolean s3 = b == 0 && ++sayac > 0;
5 System.out.println(s1 + " " + s2 + " " +
  s3);
6 System.out.println("sayac = " + sayac);
```



Sol true ise `||` 'nin sonucu zaten true: `++sayac` **çalışmaz**, `sayac` 0 kalır. `s2 = true`.

Adım 5 / 9

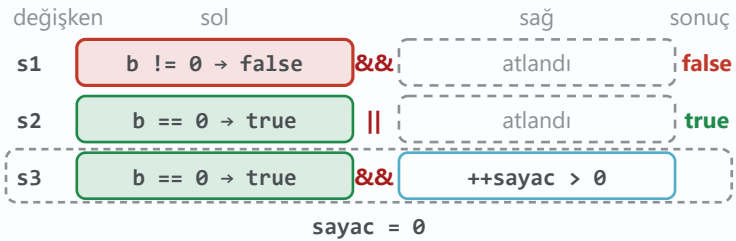
DEĞİŞKENLER

`a = 10` `b = 0` `sayac = 0` `s1 = false` `s2 = true`

ÇIKTI

Kısa Devre: Kodun Adım Adım İzlenmesi

```
1 int a = 10, b = 0, sayac = 0;
2 boolean s1 = b != 0 && a / b > 2;
3 boolean s2 = b == 0 || ++sayac > 0;
▶ boolean s3 = b == 0 && ++sayac > 0;
5 System.out.println(s1 + " " + s2 + " " +
  s3);
6 System.out.println("sayac = " + sayac);
```



Yine `&&` ; sol taraf `b == 0` → true. Sonuç henüz belli değil.

Adım 6 / 9

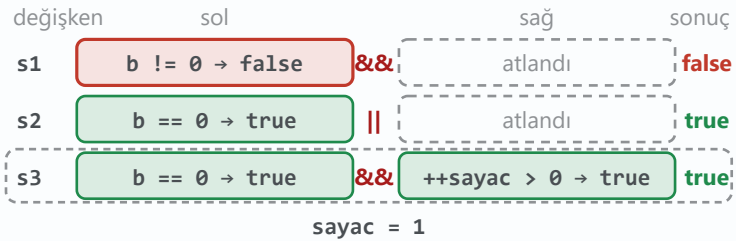
DEĞİŞKENLER

`a = 10` `b = 0` `sayac = 0` `s1 = false` `s2 = true`

ÇIKTI

Kısa Devre: Kodun Adım Adım İzlenmesi

```
1 int a = 10, b = 0, sayac = 0;
2 boolean s1 = b != 0 && a / b > 2;
3 boolean s2 = b == 0 || ++sayac > 0;
▶ boolean s3 = b == 0 && ++sayac > 0;
5 System.out.println(s1 + " " + s2 + " " +
  s3);
6 System.out.println("sayac = " + sayac);
```



Sağ taraf hesaplanmak **zorunda**: `++sayac` → 1, `1 > 0` → true. `s3 = true`.

Adım 7 / 9

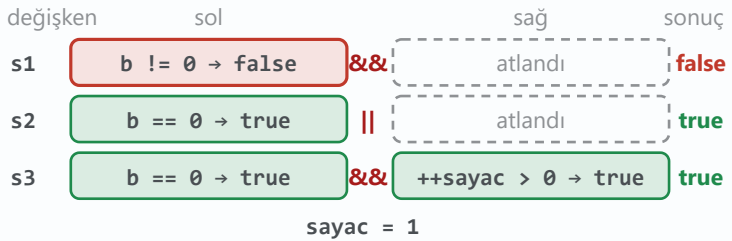
DEĞİŞKENLER

`a = 10` `b = 0` `sayac = 1` `s1 = false` `s2 = true` `s3 = true`

ÇIKTI

Kısa Devre: Kodun Adım Adım İzlenmesi

```
1 int a = 10, b = 0, sayac = 0;
2 boolean s1 = b != 0 && a / b > 2;
3 boolean s2 = b == 0 || ++sayac > 0;
4 boolean s3 = b == 0 && ++sayac > 0;
▶ System.out.println(s1 + " " + s2 + " " + s3);
6 System.out.println("sayac = " + sayac);
```



Üç sonuç yazdırılır: `false true true`.

Adım 8 / 9

DEĞİŞKENLER

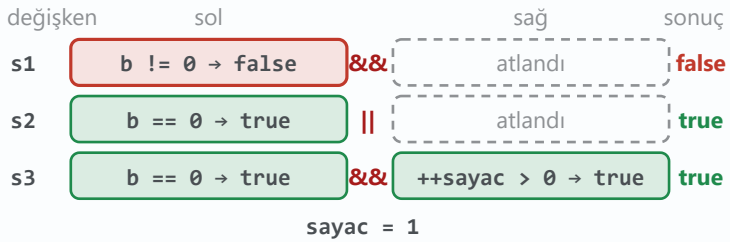
`a = 10` `b = 0` `sayac = 1` `s1 = false` `s2 = true` `s3 = true`

ÇIKTI

```
false true true
```

Kısa Devre: Kodun Adım Adım İzlenmesi

```
1 int a = 10, b = 0, sayac = 0;
2 boolean s1 = b != 0 && a / b > 2;
3 boolean s2 = b == 0 || ++sayac > 0;
4 boolean s3 = b == 0 && ++sayac > 0;
5 System.out.println(s1 + " " + s2 + " " +
  s3);
▶ System.out.println("sayac = " + sayac);
```



`sayac = 1` : üç sağ taraftan yalnızca biri çalıştı. Yan etkili işlemleri (`++`) `&&` / `||` 'nin sağına yazmak bu yüzden yanıltıcıdır.

Adım 9 / 9

DEĞİŞKENLER

`a = 10` `b = 0` `sayac = 1` `s1 = false` `s2 = true` `s3 = true`

ÇIKTI

```
false true true
sayac = 1
```

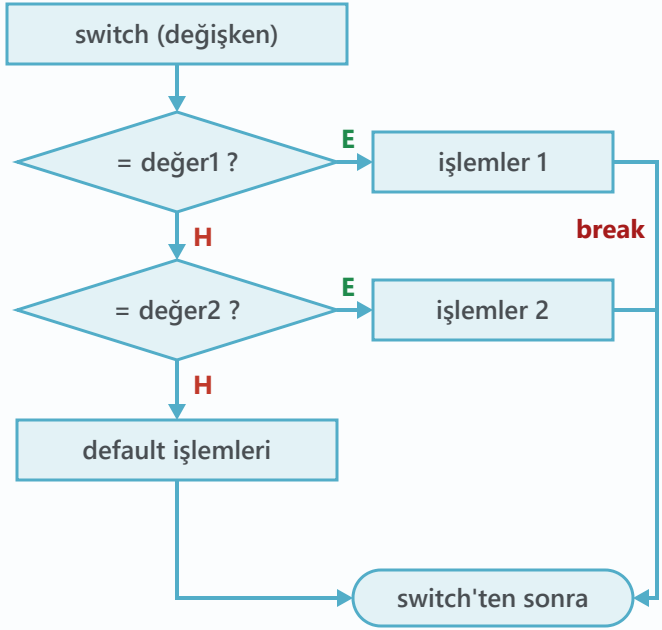
switch-case Yapısı

- **switch-case**, bir değişkenin değerini **bir dizi sabit** değerle karşılaştırır ve duruma göre farklı işlemler yapar.
- Birden çok şarta dayalı **bir yol seçmeye** yarar.
- Aynı değişken için çok sayıda `==` karşılaştırması olduğunda `else if` zincirinden daha düzenlidir.
- Benzetme: asansör düğmeleri. Basılan kat numarası hangi kata gidileceğini seçer.

```
switch (değişken) {  // kontrol edilen değer
    case değer1:     // her case bir değer
        // İşlemler 1
        break;
    case değer2:
        // İşlemler 2
        break;
    // ...
    default:         // hiçbiri eşleşmezse
        // Varsayılan işlemler (opsiyonel)
}
```

switch-case İşleyişi

- Değişken, `case` değerleriyle **yukarıdan aşağıya** karşılaştırılır.
- Eşleşen `case` bulunursa o `case` 'in işlemleri çalıştırılır, ardından `break` ile `switch` 'ten çıkılır.
- Eşleşen `case` yoksa `default` bölümü (varsa) çalışır.
- `default` da yoksa hiçbir şey yapılmadan `switch` 'ten çıkılır.
- `break` yazılmazsa çalışma **bir sonraki case'e düşer** (fall-through).



Örnek: Mevsim Belirleme

```
▶ int ay = 4;
2 String mevsim;
3 switch (ay) {
4     case 3: case 4: case 5:
5         mevsim = "İlkbahar";
6         break;
7     case 6: case 7: case 8:
8         mevsim = "Yaz";
9         break;
10    case 9: case 10: case 11:
11        mevsim = "Sonbahar";
12        break;
13    case 12: case 1: case 2:
14        mevsim = "Kış";
15        break;
16    default:
17        mevsim = "Bilinmiyor";
18 }
19 System.out.println(ay + ". ay: " + mevsim);
```

`3 4 5`	→ İlkbahar
`6 7 8`	→ Yaz
`9 10 11`	→ Sonbahar
`12 1 2`	→ Kış
`default`	→ Bilinmiyor

`ay` = 4.

Adım 1 / 7

DEĞİŞKENLER

`ay = 4`

ÇIKTI

Örnek: Mevsim Belirleme

```
1  int ay = 4;
▶  String mevsim;
3  switch (ay) {
4      case 3: case 4: case 5:
5          mevsim = "İlkbahar";
6          break;
7      case 6: case 7: case 8:
8          mevsim = "Yaz";
9          break;
10     case 9: case 10: case 11:
11         mevsim = "Sonbahar";
12         break;
13     case 12: case 1: case 2:
14         mevsim = "Kış";
15         break;
16     default:
17         mevsim = "Bilinmiyor";
18 }
19 System.out.println(ay + ". ay: " + mevsim);
```

3 4 5	→ İlkbahar
6 7 8	→ Yaz
9 10 11	→ Sonbahar
12 1 2	→ Kış
default	→ Bilinmiyor

`mevsim` tanımlanır, henüz değeri yok.

Adım 2 / 7

DEĞİŞKENLER

`ay = 4` `mevsim = ?`

ÇIKTI

Örnek: Mevsim Belirleme

```
1  int ay = 4;
2  String mevsim;
▶  switch (ay) {
4      case 3: case 4: case 5:
5          mevsim = "İlkbahar";
6          break;
7      case 6: case 7: case 8:
8          mevsim = "Yaz";
9          break;
10     case 9: case 10: case 11:
11         mevsim = "Sonbahar";
12         break;
13     case 12: case 1: case 2:
14         mevsim = "Kış";
15         break;
16     default:
17         mevsim = "Bilinmiyor";
18 }
19 System.out.println(ay + ". ay: " + mevsim);
```

`3 4 5`	→ İlkbahar
`6 7 8`	→ Yaz
`9 10 11`	→ Sonbahar
`12 1 2`	→ Kış
`default`	→ Bilinmiyor

`switch (ay)` : değer 4 olarak bir kez hesaplanır, sonra `case` etiketleri aranır.

Adım 3 / 7

DEĞİŞKENLER

`ay = 4` `mevsim = ?`

ÇIKTI

Örnek: Mevsim Belirleme

```
1  int ay = 4;
2  String mevsim;
3  switch (ay) {
▶      case 3: case 4: case 5:
5          mevsim = "İlkbahar";
6          break;
7      case 6: case 7: case 8:
8          mevsim = "Yaz";
9          break;
10     case 9: case 10: case 11:
11         mevsim = "Sonbahar";
12         break;
13     case 12: case 1: case 2:
14         mevsim = "Kış";
15         break;
16     default:
17         mevsim = "Bilinmiyor";
18 }
19 System.out.println(ay + ". ay: " + mevsim);
```

3 4 5	→ İlkbahar
6 7 8	→ Yaz
9 10 11	→ Sonbahar
12 1 2	→ Kış
default	→ Bilinmiyor

`case 4` eşleşir. Aynı satırdaki üç `case` aynı koda iner.

Adım 4 / 7

DEĞİŞKENLER

`ay = 4` `mevsim = ?`

ÇIKTI

Örnek: Mevsim Belirleme

```
1  int ay = 4;
2  String mevsim;
3  switch (ay) {
4      case 3: case 4: case 5:
▶          mevsim = "İlkbahar";
6          break;
7      case 6: case 7: case 8:
8          mevsim = "Yaz";
9          break;
10     case 9: case 10: case 11:
11         mevsim = "Sonbahar";
12         break;
13     case 12: case 1: case 2:
14         mevsim = "Kış";
15         break;
16     default:
17         mevsim = "Bilinmiyor";
18 }
19 System.out.println(ay + ". ay: " + mevsim);
```

3 4 5	→ İlkbahar
6 7 8	→ Yaz
9 10 11	→ Sonbahar
12 1 2	→ Kış
default	→ Bilinmiyor

`mevsim` = "İlkbahar".

Adım 5 / 7

DEĞİŞKENLER

`ay = 4` `mevsim = İlkbahar`

ÇIKTI

Örnek: Mevsim Belirleme

```
1  int ay = 4;
2  String mevsim;
3  switch (ay) {
4      case 3: case 4: case 5:
5          mevsim = "İlkbahar";
▶          break;
7      case 6: case 7: case 8:
8          mevsim = "Yaz";
9          break;
10     case 9: case 10: case 11:
11         mevsim = "Sonbahar";
12         break;
13     case 12: case 1: case 2:
14         mevsim = "Kış";
15         break;
16     default:
17         mevsim = "Bilinmiyor";
18 }
19 System.out.println(ay + ". ay: " + mevsim);
```

`3 4 5`	→ İlkbahar
`6 7 8`	→ Yaz
`9 10 11`	→ Sonbahar
`12 1 2`	→ Kış
`default`	→ Bilinmiyor

`break` : `switch` 'in kapanış parantezinin (satır 18) arkasına atlanır.

Adım 6 / 7

DEĞİŞKENLER

`ay = 4` `mevsim = İlkbahar`

ÇIKTI

Örnek: Mevsim Belirleme

```
1  int ay = 4;
2  String mevsim;
3  switch (ay) {
4      case 3: case 4: case 5:
5          mevsim = "İlkbahar";
6          break;
7      case 6: case 7: case 8:
8          mevsim = "Yaz";
9          break;
10     case 9: case 10: case 11:
11         mevsim = "Sonbahar";
12         break;
13     case 12: case 1: case 2:
14         mevsim = "Kış";
15         break;
16     default:
17         mevsim = "Bilinmiyor";
18 }
▶  System.out.println(ay + ". ay: " + mevsim);
```

`3 4 5`	→ İlkbahar
`6 7 8`	→ Yaz
`9 10 11`	→ Sonbahar
`12 1 2`	→ Kış
`default`	→ Bilinmiyor

`default` sayesinde `mevsim` her yolda değer alır; yoksa: *variable mevsim might not have been initialized*.

Adım 7 / 7

DEĞİŞKENLER

`ay = 4` `mevsim = İlkbahar`

ÇIKTI

```
4. ay: İlkbahar
```

break Unutulursa: Aşağı Düşme

```
1 int gun = 2;
2 switch (gun) {
3     case 1:
4         System.out.println("Pazartesi");
5     case 2:
6         System.out.println("Salı");
7     case 3:
8         System.out.println("Çarşamba");
9         break;
10    default:
11        System.out.println("Diğer gün");
12 }
13 System.out.println("switch bitti");
```

`case 1:`	Pazartesi
`case 2:`	Salı
`case 3:`	Çarşamba · break
`default:`	Diğer gün

`gun` = 2.

Adım 1 / 8

DEĞİŞKENLER

gun = **2**

ÇIKTI

break Unutulursa: Aşağı Düşme

```
1  int gun = 2;
▶  switch (gun) {
3      case 1:
4          System.out.println("Pazartesi");
5      case 2:
6          System.out.println("Salı");
7      case 3:
8          System.out.println("Çarşamba");
9          break;
10     default:
11         System.out.println("Diğer gün");
12 }
13 System.out.println("switch bitti");
```

`case 1:`	Pazartesi
`case 2:`	Salı
`case 3:`	Çarşamba · break
`default:`	Diğer gün

2 değeriyle eşleşen `case` aranır.

Adım 2 / 8

DEĞİŞKENLER

gun = **2**

ÇIKTI

break Unutulursa: Aşağı Düşme

```
1  int gun = 2;
2  switch (gun) {
3      case 1:
4          System.out.println("Pazartesi");
▶      case 2:
6          System.out.println("Salı");
7      case 3:
8          System.out.println("Çarşamba");
9          break;
10     default:
11         System.out.println("Diğer gün");
12 }
13 System.out.println("switch bitti");
```

`case 1:` Pazartesi

`case 2:` Salı

`case 3:` Çarşamba · break

`default:` Diğer gün

`case 1` atlanır, `case 2` eşleşir.

Adım 3 / 8

DEĞİŞKENLER

gun = **2**

ÇIKTI

break Unutulursa: Aşağı Düşme

```
1  int gun = 2;
2  switch (gun) {
3      case 1:
4          System.out.println("Pazartesi");
5      case 2:
6          System.out.println("Salı");
7      case 3:
8          System.out.println("Çarşamba");
9          break;
10     default:
11         System.out.println("Diğer gün");
12 }
13 System.out.println("switch bitti");
```

`case 1:`	Pazartesi
`case 2:`	Salı
`case 3:`	Çarşamba · break
`default:`	Diğer gün

"Salı" yazılır. Burada **`break` yok!**

Adım 4 / 8

DEĞİŞKENLER

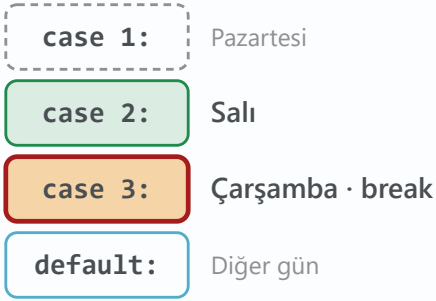
`gun = 2`

ÇIKTI

```
Salı
```

break Unutulursa: Aşağı Düşme

```
1  int gun = 2;
2  switch (gun) {
3      case 1:
4          System.out.println("Pazartesi");
5      case 2:
6          System.out.println("Salı");
▶      case 3:
8          System.out.println("Çarşamba");
9          break;
10     default:
11         System.out.println("Diğer gün");
12 }
13 System.out.println("switch bitti");
```



`case 3:` etiketi bir karşılaştırma **değildir**; çalışma alttaki koda "düşer" (fall-through).

Adım 5 / 8

DEĞİŞKENLER

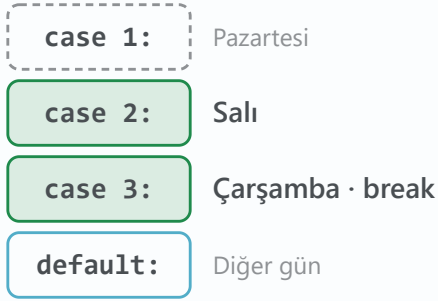
`gun = 2`

ÇIKTI

```
Salı
```

break Unutulursa: Aşağı Düşme

```
1  int gun = 2;
2  switch (gun) {
3      case 1:
4          System.out.println("Pazartesi");
5      case 2:
6          System.out.println("Salı");
7      case 3:
▶          System.out.println("Çarşamba");
9          break;
10     default:
11         System.out.println("Diğer gün");
12 }
13 System.out.println("switch bitti");
```



"Çarşamba" da yazılır, oysa gün 3 değil.

Adım 6 / 8

DEĞİŞKENLER

gun = **2**

ÇIKTI

```
Salı
Çarşamba
```

break Unutulursa: Aşağı Düşme

```
1  int gun = 2;
2  switch (gun) {
3      case 1:
4          System.out.println("Pazartesi");
5      case 2:
6          System.out.println("Salı");
7      case 3:
8          System.out.println("Çarşamba");
▶          break;
10     default:
11         System.out.println("Diğer gün");
12 }
13 System.out.println("switch bitti");
```

`case 1:` Pazartesi

`case 2:` Salı

`case 3:` Çarşamba · break

`default:` Diğer gün

`break` ile `switch`'ten çıkılır; `default` çalışmaz.

Adım 7 / 8

DEĞİŞKENLER

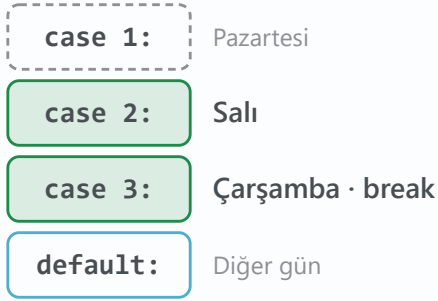
`gun = 2`

ÇIKTI

```
Salı
Çarşamba
```

break Unutulursa: Aşağı Düşme

```
1  int gun = 2;
2  switch (gun) {
3      case 1:
4          System.out.println("Pazartesi");
5      case 2:
6          System.out.println("Salı");
7      case 3:
8          System.out.println("Çarşamba");
9          break;
10     default:
11         System.out.println("Diğer gün");
12 }
▶  System.out.println("switch bitti");
```



`switch`'ten sonra devam edilir. Aşağı düşme istenmiyorsa her `case` `break` ile biter.

Adım 8 / 8

DEĞİŞKENLER

`gun = 2`

ÇIKTI

```
Salı
Çarşamba
switch bitti
```

switch: char ve String

CHAR İLE BASİT HESAP MAKİNESİ

```
int a = 12, b = 4;
char islem = '*';
int sonuc;
switch (islem) {
    case '+':  sonuc = a + b;  break;
    case '-':  sonuc = a - b;  break;
    case '*':  sonuc = a * b;  break;
    case '/':  sonuc = a / b;  break;
    default:
        System.out.println("Geçersiz işlem");
        sonuc = 0;
}
System.out.println(a + " " + islem + " " + b
        + " = " + sonuc);
```

```
12 * 4 = 48
```

STRING İLE KOMUT SEÇİMİ (JAVA 7+)

```
String komut = "dur";
switch (komut) {
    case "basla":
        System.out.println("Motor çalıştı");
        break;
    case "dur":
        System.out.println("Motor durdu");
        break;
    default:
        System.out.println("Bilinmeyen komut");
}
```

```
Motor durdu
```

Kısa `case`'ler tek satıra yazılabilir: `case '+': sonuc = a + b; break;`. `String` karşılaştırması büyük-küçük harfe duyarlıdır: `"Dur"` ≠ `"dur"`.

switch Kuralları

- Seçici (parantez içindeki ifade) şu türlerden olabilir: `byte`, `short`, `char`, `int` (ve sarmalayıcıları), `String`, `enum`.
- `long`, `float`, `double`, `boolean` ile klasik `switch` **yazılamaz**.
- `case` değerleri **derleme zamanında bilinen sabitler** olmalıdır (`5`, `'a'`, `"dur"`, `final` sabitler).
- Aynı değer iki `case`'te kullanılamaz.
- Aralık (`ay >= 3 && ay <= 5`) veya karmaşık koşul için `if-else` gerekir.

```
double d = 2.5;
switch (d) { ... }
```

```
error: selector type double is not allowed
```

```
int b = 2;
switch (a) { case b: break; }
```

```
error: constant expression required
```

```
switch (a) { case 1: break; case 1: break; }
```

```
error: duplicate case label
```

switch-case ve if-else

	`if-else / else if`	`switch-case`
Ne sınanır?	Her türlü `boolean` koşul: aralık, birleşik koşul, farklı değişkenler	Tek bir ifadenin **sabit** değerlerle eşitliği
Esneklik	Daha esnek	Yalnızca belirli tipler ve sabit `case` değerleri
Çok seçenekte	Uzun zincirler karmaşık görünebilir	Düzenli ve okunaklı bir liste
Tuzak	Koşul sırası yanlışsa yanlış dal seçilir	`break` unutulursa alttaki case'lere düşülür
Örnek	`if (not >= 90) … else if (not >= 80) …`	`switch (ay) { case 3: … }`

Her `switch`, bir `else if` zinciriyle yazılabilir; tersi her zaman mümkün değildir (ör. `not >= 90` gibi aralıklar).

Yeni switch (Java 14+)

```
int ay = 11;
String mevsim = switch (ay) {
    case 3, 4, 5 -> "İlkbahar";
    case 6, 7, 8 -> "Yaz";
    case 9, 10, 11 -> "Sonbahar";
    case 12, 1, 2 -> "Kış";
    default -> "Bilinmiyor";
};
System.out.println(mevsim);
```

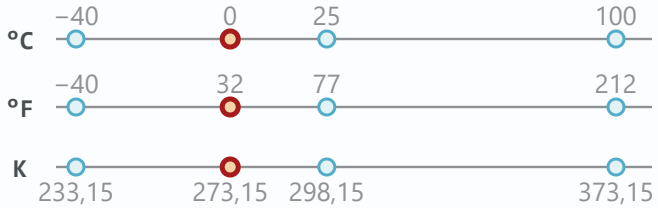
```
Sonbahar
```

- Java 14 ile gelen **ok (arrow) biçimi**: `case 3, 4, 5 -> …`
- Bir `case` virgülle birden çok değer alabilir.
- Ok biçiminde **aşağı düşme yoktur**; `break` gerekmez.
- **switch ifadesi** bir değer üretir ve doğrudan atanabilir; sonuna `;` konur.
- İfade biçimi tüm olasılıkları kapsamalıdır: `int` için `default` zorunludur.
- Bu derste klasik biçim esastır; ok biçimi Java 14 ve sonrasında kullanılabilir.

UYGULAMA: SICAKLIK

Sıcaklık Dönüşümü: Tablo ve Formüller

Celsius (°C)	Fahrenheit (°F)	Kelvin (K)
−40	−40	233,15
0	32	273,15
25	77	298,15
100	212	373,15



Celsius → Fahrenheit

```
°F = (°C × 1.8) + 32
```

Fahrenheit → Celsius

```
°C = (°F − 32) / 1.8
```

Celsius → Kelvin

```
K = °C + 273.15
```

Hedef: formülleri Java'ya çevirmek ve tek bir değer için hesaplamak.

UYGULAMA: SICAKLIK

Sıcaklık Dönüşümü: Tek Değer

```java
public class Program {
    public static void main(String[] args) {
        double celsius = 25.0;
        double fahrenheit = (celsius * 1.8) + 32;
        double kelvin = celsius + 273.15;
        System.out.println(celsius + " °C = " + fahrenheit + " °F = " + kelvin + " K");
    }
}
```

```
25.0 °C = 77.0 °F = 298.15 K
```

- Sıralı bir programdır: üç satır hesaplama, bir satır yazdırma; karar ya da döngü yoktur.
- Ondalıklı değerler için `double` kullanılır. Parantez önceliği netleştirir: önce `celsius * 1.8`, sonra `+ 32`.
- `+` bir dizgiyle (String) kullanılınca **birleştirme** yapar; `double` değerler `25.0` biçiminde yazılır.
- Hesaplanan her değer kullanılmalıdır: `kelvin` yazdırılmasaydı boşuna hesaplanmış olurdu.

Döngü ve Türleri

Döngü (loop), aynı işlemi **birden çok kez** tekrarlamak için kullanılır. Benzetme: pistte 10 tur koşmak; her turda aynı hareketler, tur sayacı bir artar.

for

Tekrar sayısı belli olduğunda; belirli bir aralıktaki değerlerle çalışır.

```
for (int i = 0; i < 5; i++) { … }
```

while

Belirli bir koşul sağlandığı sürece çalışır; tekrar sayısı önceden bilinmeyebilir.

```
while (koşul) { … }
```

do-while

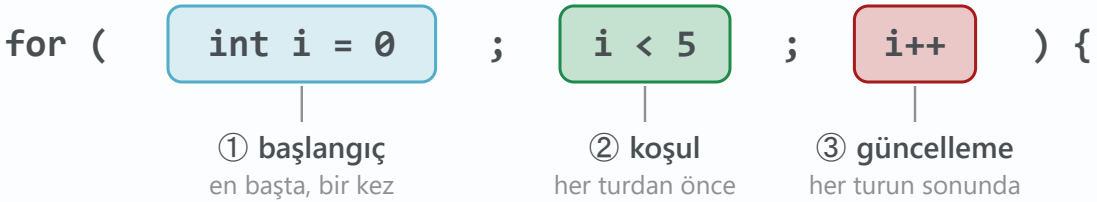
Koşul sona yazılır; gövde **en az bir kez** çalışır.

```
do { … } while (koşul);
```



- Döngüler bir **veri kümesinin sırayla işlenmesi** ve işlem tekrarları için kullanışlıdır; özellikle diziler ve listelerle çalışırken önemlidir.
- Aynı satırı 100 kez yazmak yerine tek bir döngü: kod **daha kısa, düzenli ve okunabilir** olur.

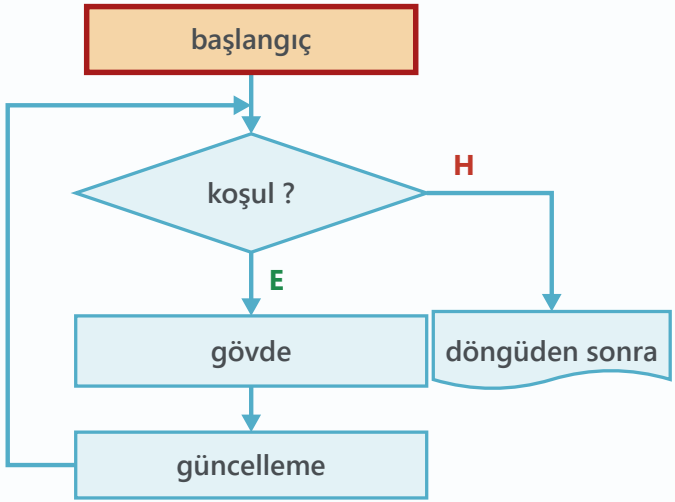
for Döngüsü



- Başlık, döngünün **başlangıç**, **koşul** ve **güncelleme** (artırma) adımlarını tek satırda belirler; aralarında `;` vardır.
- Her döngü dönüşünden (tur, iteration) önce koşul kontrol edilir; güncelleme adımı döngü değişkenini değiştirir.
- Çalışma sırası: **1** → **2** → **4** → **3** → **2** → **4** → **3** → … → **2** yanlış → döngüden çık.
- `i` = 0, 1, 2, 3, 4 için gövde **5 kez** çalışır; `i` 5 olunca koşul yanlış olur. `i` yalnızca döngünün içinde tanımlıdır.

Döngülerin İşleyişi

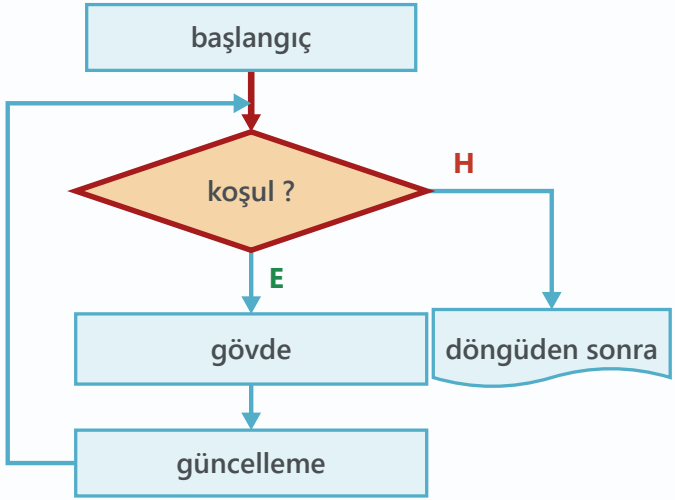
- Döngü **başlangıç adımıyla** başlar.
- **Koşul kontrol edilir.**
  - Koşul sağlanıyorsa döngü içindeki işlemler yapılır.
  - Güncelleme (artırma) adımı uygulanır (`for` döngüsü için).
  - Koşul hâlâ sağlanıyorsa tekrar döngü içine girilir.
- Koşul sağlanmazsa **döngüden çıkılır** ve sonraki deyimle devam edilir.
- Koşul ilk kontrolde yanlışsa gövde **hiç** çalışmaz.



1. Başlangıç: döngü değişkeni ilk değerini alır (bir kez).

Döngülerin İşleyişi

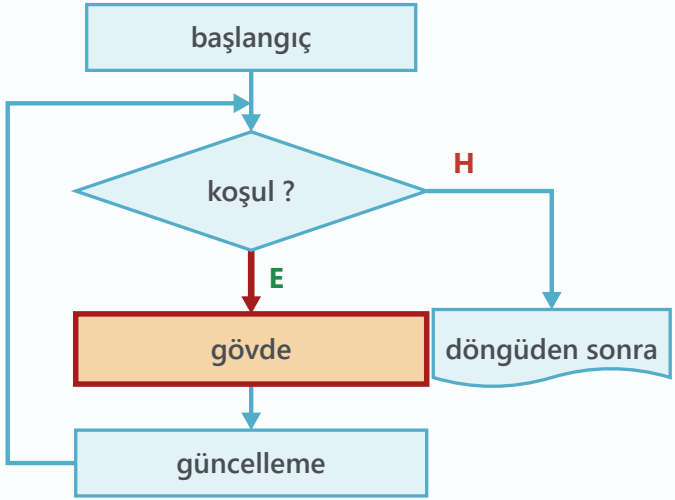
- Döngü **başlangıç adımıyla** başlar.
- **Koşul kontrol edilir.**
  - Koşul sağlanıyorsa döngü içindeki işlemler yapılır.
  - Güncelleme (artırma) adımı uygulanır (`for` döngüsü için).
  - Koşul hâlâ sağlanıyorsa tekrar döngü içine girilir.
- Koşul sağlanmazsa **döngüden çıkılır** ve sonraki deyimle devam edilir.
- Koşul ilk kontrolde yanlışsa gövde **hiç** çalışmaz.



2. Koşul kontrol edilir.

Döngülerin İşleyişi

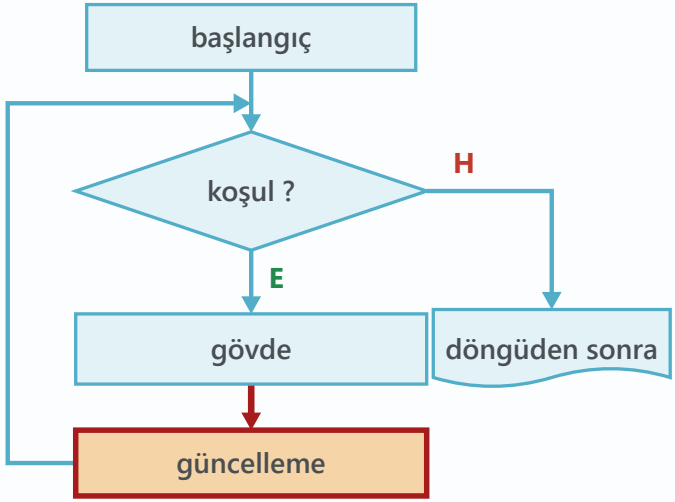
- Döngü **başlangıç adımıyla** başlar.
- **Koşul kontrol edilir.**
  - Koşul sağlanıyorsa döngü içindeki işlemler yapılır.
  - Güncelleme (artırma) adımı uygulanır (`for` döngüsü için).
  - Koşul hâlâ sağlanıyorsa tekrar döngü içine girilir.
- Koşul sağlanmazsa **döngüden çıkılır** ve sonraki deyimle devam edilir.
- Koşul ilk kontrolde yanlışsa gövde **hiç** çalışmaz.



3. Koşul doğruysa gövde çalışır.

Döngülerin İşleyişi

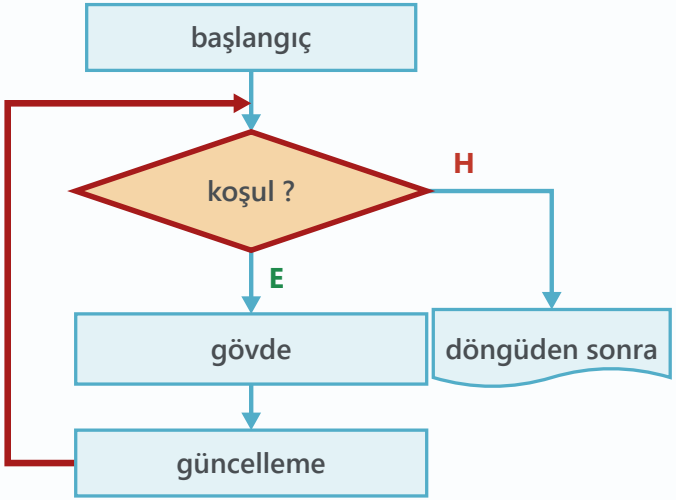
- Döngü **başlangıç adımıyla** başlar.
- **Koşul kontrol edilir.**
  - Koşul sağlanıyorsa döngü içindeki işlemler yapılır.
  - Güncelleme (artırma) adımı uygulanır (`for` döngüsü için).
  - Koşul hâlâ sağlanıyorsa tekrar döngü içine girilir.
- Koşul sağlanmazsa **döngüden çıkılır** ve sonraki deyimle devam edilir.
- Koşul ilk kontrolde yanlışsa gövde **hiç** çalışmaz.



4. Güncelleme: döngü değişkeni değişir.

Döngülerin İşleyişi

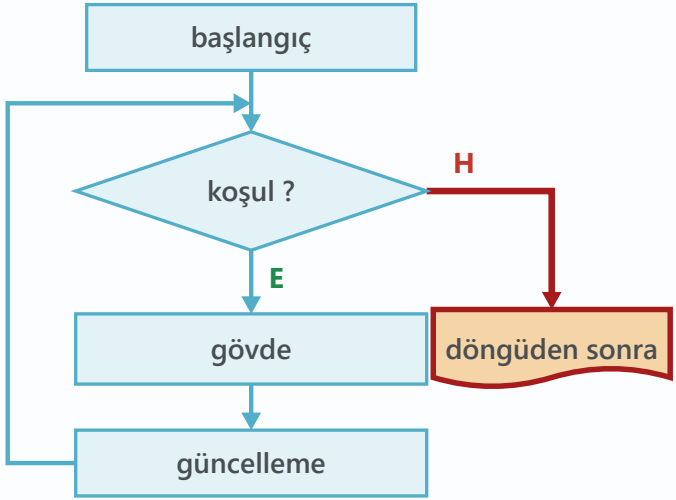
- Döngü **başlangıç adımıyla** başlar.
- **Koşul kontrol edilir.**
  - Koşul sağlanıyorsa döngü içindeki işlemler yapılır.
  - Güncelleme (artırma) adımı uygulanır (`for` döngüsü için).
  - Koşul hâlâ sağlanıyorsa tekrar döngü içine girilir.
- Koşul sağlanmazsa **döngüden çıkılır** ve sonraki deyimle devam edilir.
- Koşul ilk kontrolde yanlışsa gövde **hiç** çalışmaz.



5. Koşula geri dönülür; doğru oldukça 3–4–5 tekrarlanır.

Döngülerin İşleyişi

- Döngü **başlangıç adımıyla** başlar.
- **Koşul kontrol edilir.**
  - Koşul sağlanıyorsa döngü içindeki işlemler yapılır.
  - Güncelleme (artırma) adımı uygulanır (`for` döngüsü için).
  - Koşul hâlâ sağlanıyorsa tekrar döngü içine girilir.
- Koşul sağlanmazsa **döngüden çıkılır** ve sonraki deyimle devam edilir.
- Koşul ilk kontrolde yanlışsa gövde **hiç** çalışmaz.



6. Koşul yanlış olunca döngüden çıkılır.

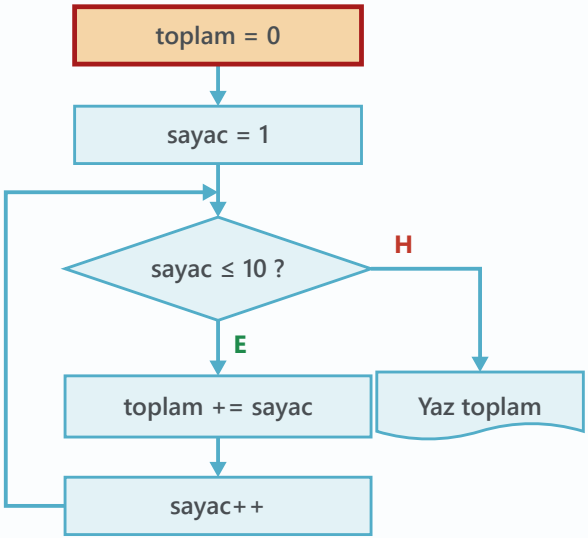
Sayı Toplama (for)

```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
3     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 0`

ÇIKTI



`toplam` 0 ile başlar (toplama işleminin etkisiz elemanı).

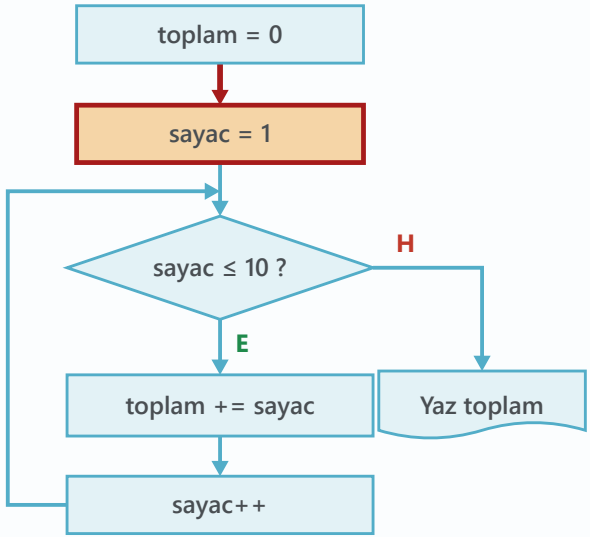
Sayı Toplama (for)

```
1 int toplam = 0;
▶ for (int sayac = 1; sayac <= 10; sayac++) {
3     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 0` `sayac = 1`

ÇIKTI



Başlangıç: `sayac` = 1 (bir kez).

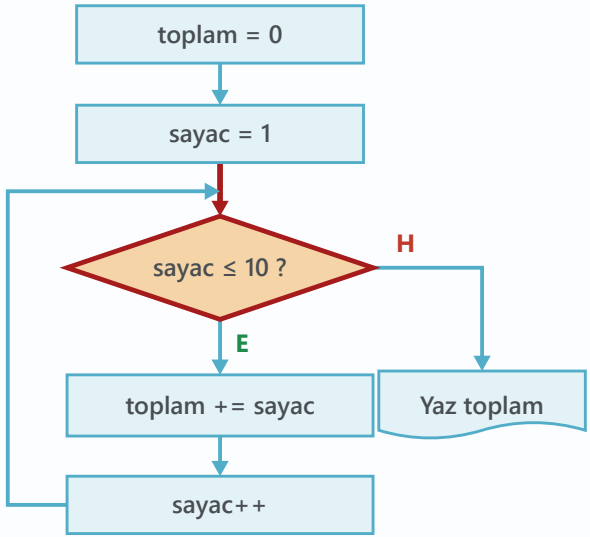
Sayı Toplama (for)

```
1 int toplam = 0;
▶ for (int sayac = 1; sayac <= 10; sayac++) {
3     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 0` `sayac = 1`

ÇIKTI



Koşul: 1 <= 10 → true.

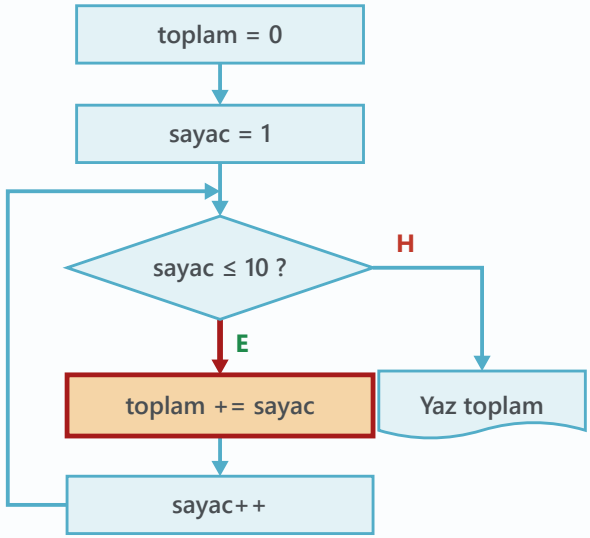
Sayı Toplama (for)

```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
▶     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 1` `sayac = 1`

ÇIKTI



1. tur: `toplam` = 0 + 1 = 1.

Sayı Toplama (for)

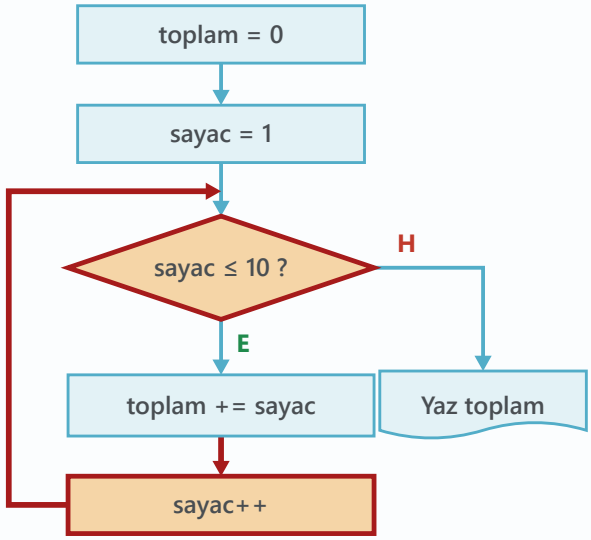
```
1 int toplam = 0;
▶ for (int sayac = 1; sayac <= 10; sayac++) {
3     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 1` `sayac = 2`

ÇIKTI





`sayac++` → 2, koşul: 2 <= 10 → true.

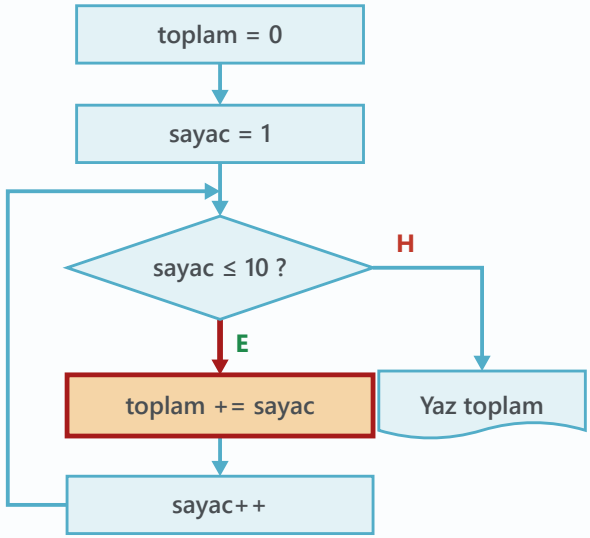
Sayı Toplama (for)

```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
▶     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 3` `sayac = 2`

ÇIKTI



2. tur: `toplam` = 1 + 2 = 3.

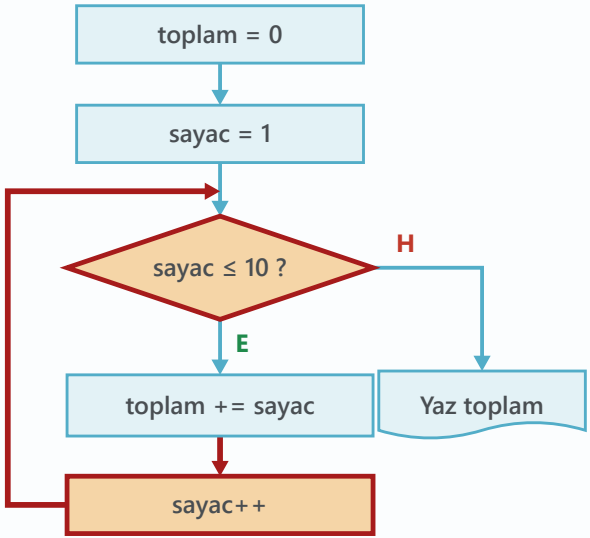
Sayı Toplama (for)

```
1 int toplam = 0;
▶ for (int sayac = 1; sayac <= 10; sayac++) {
3     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 3` `sayac = 3`

ÇIKTI



`sayac++` → 3, koşul: 3 <= 10 → true.

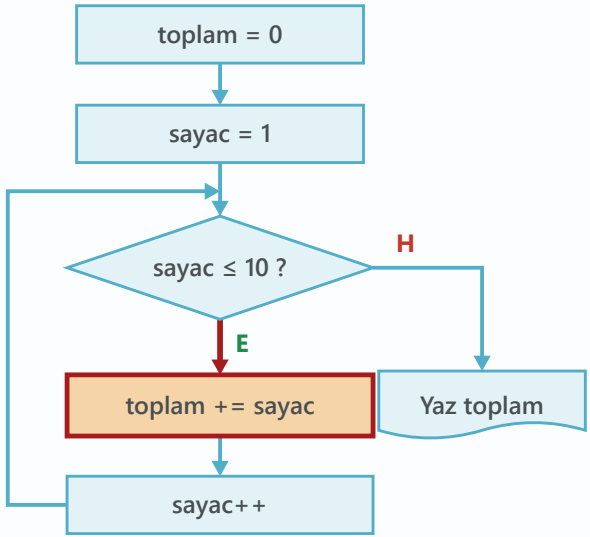
Sayı Toplama (for)

```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
▶     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 6` `sayac = 3`

ÇIKTI



3. tur: `toplam` = 3 + 3 = 6. Güncelleme ve koşul her turda aynı biçimde tekrarlanır.

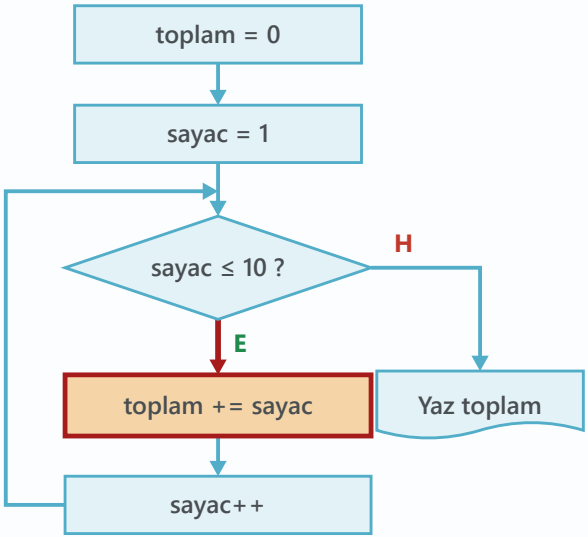
Sayı Toplama (for)

```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
▶     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 10` `sayac = 4`

ÇIKTI



4. tur: `toplam` = 6 + 4 = 10.

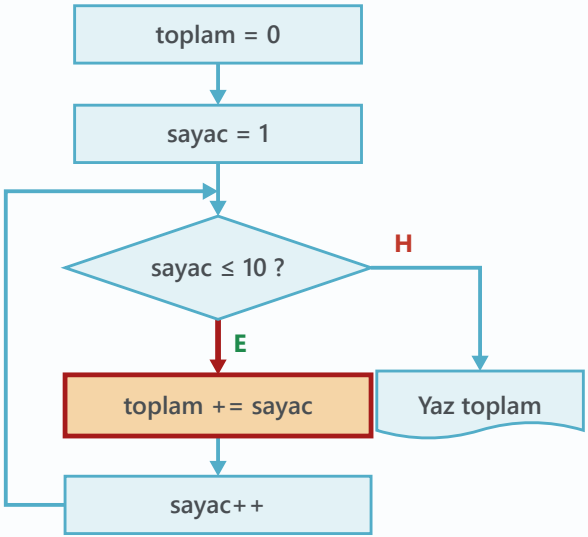
Sayı Toplama (for)

```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
▶     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 15` `sayac = 5`

ÇIKTI



5. tur: `toplam` = 10 + 5 = 15.

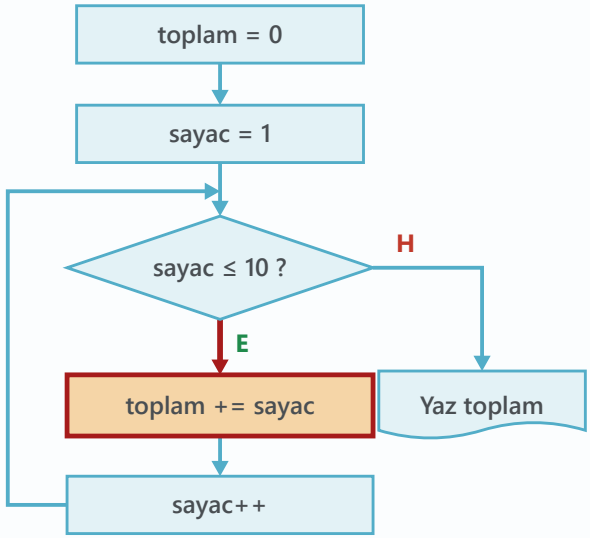
Sayı Toplama (for)

```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
▶     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 21` `sayac = 6`

ÇIKTI



6. tur: `toplam` = 15 + 6 = 21.

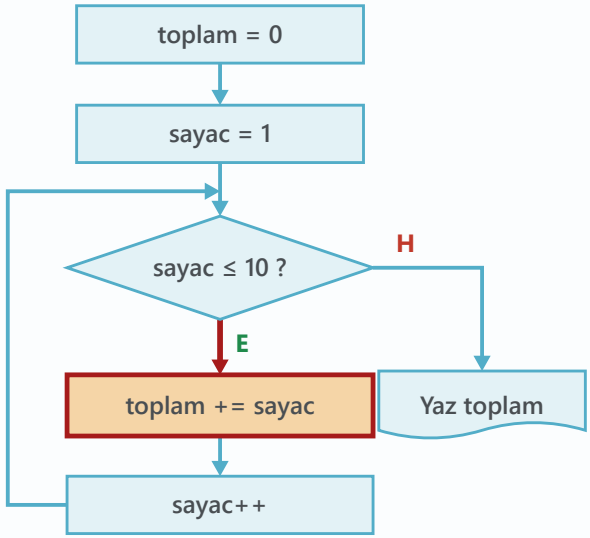
Sayı Toplama (for)

```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
▶     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 28` `sayac = 7`

ÇIKTI



7. tur: `toplam` = 21 + 7 = 28.

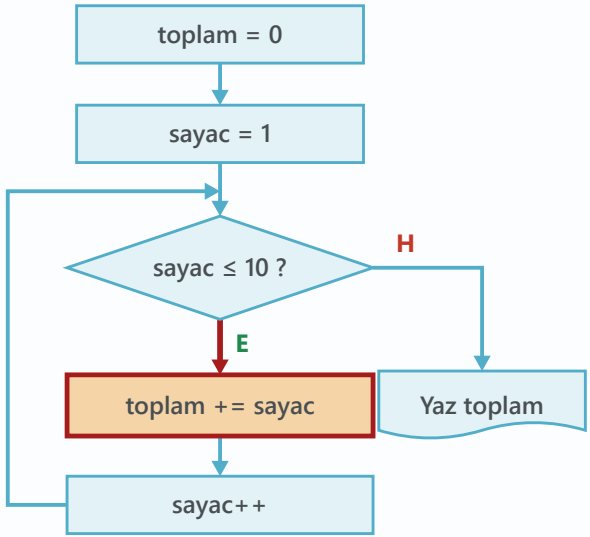
Sayı Toplama (for)

```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
▶     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 36` `sayac = 8`

ÇIKTI



8. tur: `toplam` = 28 + 8 = 36.

Sayı Toplama (for)

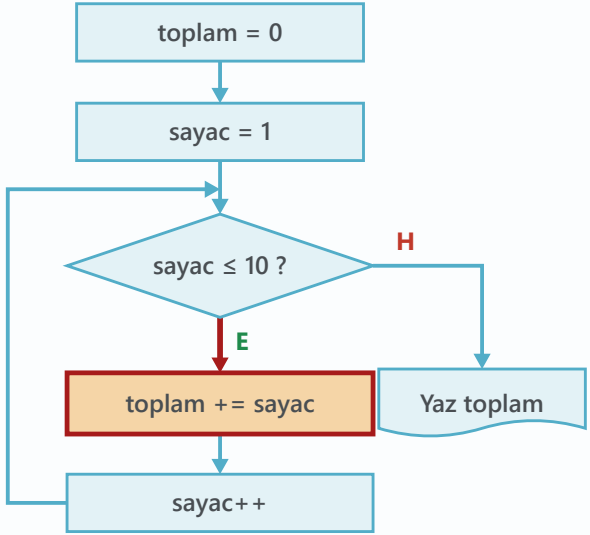
```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
▶     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 45` `sayac = 9`

ÇIKTI





9. tur: `toplam` = 36 + 9 = 45.

Sayı Toplama (for)

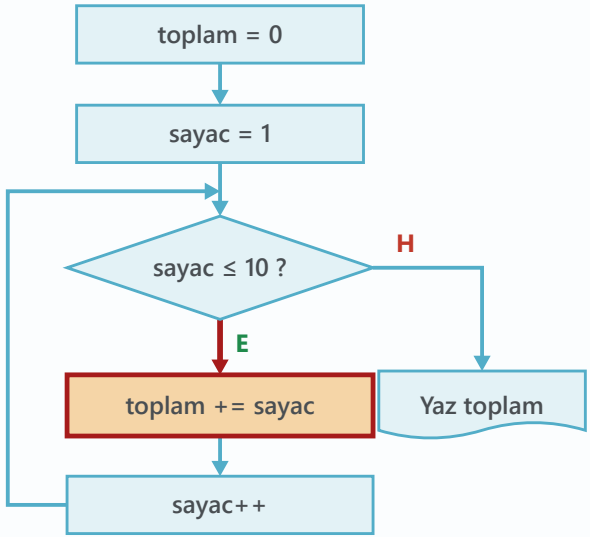
```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
▶     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 55` `sayac = 10`

ÇIKTI





10. tur: `toplam` = 45 + 10 = 55.

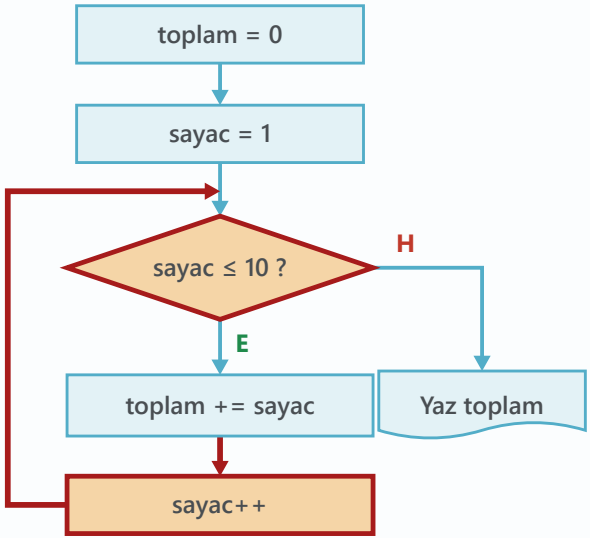
Sayı Toplama (for)

```
1 int toplam = 0;
▶ for (int sayac = 1; sayac <= 10; sayac++) {
3     toplam += sayac;
4 }
5 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 55` `sayac = 11`

ÇIKTI



`sayac++` → 11, koşul: 11 <= 10 → **false**. Döngü biter; gövde tam 10 kez çalıştı.

Sayı Toplama (for)

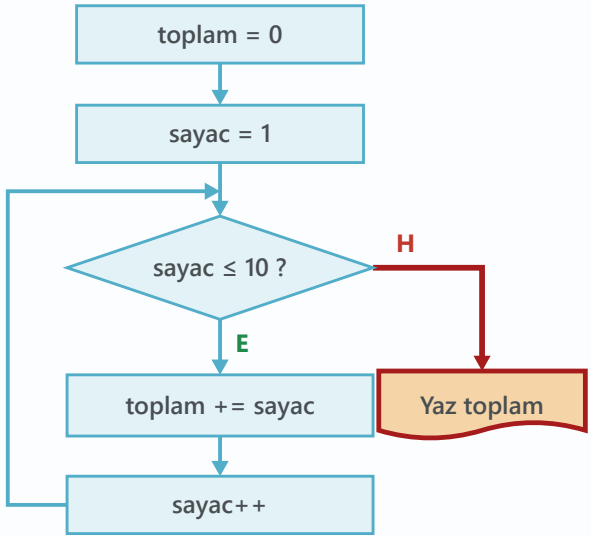
```
1 int toplam = 0;
2 for (int sayac = 1; sayac <= 10; sayac++) {
3     toplam += sayac;
4 }
▶ System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

`toplam = 55`

ÇIKTI

```
Toplam: 55
```



Döngüden sonra toplam yazılır. `sayac` artık tanımlı değil: yalnızca döngünün içinde yaşar.

Aralık Toplama

```java
import java.util.Scanner;

public class AralikToplama {
    public static void main(String[] args) {
        Scanner klavye = new Scanner(System.in);
        System.out.print("Başlangıç değeri (a): ");
        int a = klavye.nextInt();
        System.out.print("Bitiş değeri (b): ");
        int b = klavye.nextInt();
        int toplam = 0;
        for (int i = a; i <= b; i++) {
            toplam += i;
        }
        System.out.println("Sayıların toplamı: " + toplam);
    }
}
```

GİRDİLER: 3 VE 7

```
Başlangıç değeri (a): 3
Bitiş değeri (b): 7
Sayıların toplamı: 25
```

Döngü değişkeni `i` sabit bir sayıdan değil, **kullanıcının girdiği** `a` 'dan başlar: 3 + 4 + 5 + 6 + 7 = 25.

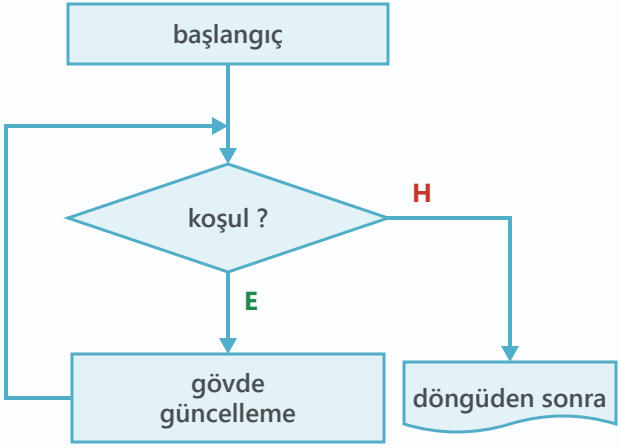
GİRDİLER: 7 VE 3

```
Başlangıç değeri (a): 7
Bitiş değeri (b): 3
Sayıların toplamı: 0
```

İlk kontrolde 7 <= 3 yanlıştır: gövde **hiç** çalışmaz, toplam 0 kalır. Çözüm: `a > b` ise önce ikisi yer değiştirilir (`if` ile).

while Döngüsü

```
int i = 0;
while (i < 5) {
    // Döngü içinde yapılacak işlemler
    i++;
}
```



- **Yalnızca bir koşulu** kontrol eder; başlangıç ve güncelleme döngünün dışında/içinde ayrıca yazılır.
- Koşul **sağlandığı sürece** çalışır; koşul gövdeden **önce** sınanır.
- Tekrar sayısı önceden belli değilse doğal seçimdir (ör. "kullanıcı 0 girene kadar").
- **Sonsuz döngülere dikkat!** `i++` unutulursa `i` hep 0 kalır, koşul hep doğrudur ve program hiç bitmez.

for ve while Karşılaştırması

FOR

```
int toplam = 0;
for (int sayac = 1; sayac <= 10; sayac++) {
    toplam += sayac;
}
System.out.println("Toplam: " + toplam);
```

WHILE

```
int toplam = 0;
int sayac = 1;
while (sayac <= 10) {
    toplam += sayac;
    sayac++;
}
System.out.println("Toplam: " + toplam);
```

```
Toplam: 55
```

Aynı üç parça iki döngüde de var: **başlangıç** `sayac = 1`, **koşul** `sayac <= 10`, **güncelleme** `sayac++`. `for` bunları tek satırda toplar; `while` 'da dağınıktır ve güncelleme unutulabilir. Her ikisi de 55 yazar.

Faktöriyel Hesaplama (while)

```
▶ int n = 5;
▶ int faktoriyel = 1;
▶ int sayac = 1;
4 while (sayac <= n) {
5     faktoriyel *= sayac;
6     sayac++;
7 }
8 System.out.println(n + " faktöriyeli: " + faktoriyel);
```

5! = 1 × 2 × 3 × 4 × 5

Hedef: 5! = 1 × 2 × 3 × 4 × 5. Çarpımın başlangıç değeri **1**'dir (0 olsaydı sonuç hep 0 olurdu).

Adım 1 / 10

DEĞİŞKENLER

`n = 5` `faktoriyel = 1` `sayac = 1`

ÇIKTI

Faktöriyel Hesaplama (while)

```
1 int n = 5;
2 int faktoriyel = 1;
3 int sayac = 1;
▶ while (sayac <= n) {
▶     faktoriyel *= sayac;
6     sayac++;
7 }
8 System.out.println(n + " faktöriyeli: " + faktoriyel);
```

faktoriyel = 1 × 1 = 1

1 <= 5 → true. `faktoriyel *= sayac` : 1 × 1 = 1.

Adım 2 / 10

DEĞİŞKENLER

`n = 5` `faktoriyel = 1` `sayac = 1`

ÇIKTI

Faktöriyel Hesaplama (while)

```
1 int n = 5;
2 int faktoriyel = 1;
3 int sayac = 1;
4 while (sayac <= n) {
5     faktoriyel *= sayac;
▶     sayac++;
7 }
8 System.out.println(n + " faktöriyeli: " + faktoriyel);
```

sayac = 2

`sayac++` → 2. Koşula geri dönülür.

Adım 3 / 10

DEĞİŞKENLER

`n = 5` `faktoriyel = 1` `sayac = 2`

ÇIKTI

Faktöriyel Hesaplama (while)

```
1 int n = 5;
2 int faktoriyel = 1;
3 int sayac = 1;
▶ while (sayac <= n) {
▶     faktoriyel *= sayac;
6     sayac++;
7 }
8 System.out.println(n + " faktöriyeli: " + faktoriyel);
```

faktoriyel = 1 × 2 = 2

2 <= 5 → true. `faktoriyel *= sayac` : 1 × 2 = 2.

Adım 4 / 10

DEĞİŞKENLER

`n = 5` `faktoriyel = 2` `sayac = 2`

ÇIKTI

Faktöriyel Hesaplama (while)

```
1 int n = 5;
2 int faktoriyel = 1;
3 int sayac = 1;
▶ while (sayac <= n) {
▶     faktoriyel *= sayac;
6     sayac++;
7 }
8 System.out.println(n + " faktöriyeli: " + faktoriyel);
```

faktoriyel = 2 × 3 = 6

3 <= 5 → true. `faktoriyel *= sayac` : 2 × 3 = 6.

Adım 5 / 10

DEĞİŞKENLER

`n = 5` `faktoriyel = 6` `sayac = 3`

ÇIKTI

Faktöriyel Hesaplama (while)

```
1 int n = 5;
2 int faktoriyel = 1;
3 int sayac = 1;
▶ while (sayac <= n) {
▶     faktoriyel *= sayac;
6     sayac++;
7 }
8 System.out.println(n + " faktöriyeli: " + faktoriyel);
```

faktoriyel = 6 × 4 = 24

4 <= 5 → true. `faktoriyel *= sayac` : 6 × 4 = 24.

Adım 6 / 10

DEĞİŞKENLER

`n = 5` `faktoriyel = 24` `sayac = 4`

ÇIKTI

Faktöriyel Hesaplama (while)

```
1 int n = 5;
2 int faktoriyel = 1;
3 int sayac = 1;
▶ while (sayac <= n) {
▶     faktoriyel *= sayac;
6     sayac++;
7 }
8 System.out.println(n + " faktöriyeli: " + faktoriyel);
```

faktoriyel = 24 × 5 = 120

5 <= 5 → true. `faktoriyel *= sayac` : 24 × 5 = 120.

Adım 7 / 10

DEĞİŞKENLER

`n = 5` `faktoriyel = 120` `sayac = 5`

ÇIKTI

Faktöriyel Hesaplama (while)

```
1 int n = 5;
2 int faktoriyel = 1;
3 int sayac = 1;
4 while (sayac <= n) {
5     faktoriyel *= sayac;
▶     sayac++;
7 }
8 System.out.println(n + " faktöriyeli: " + faktoriyel);
```

sayac = 6

`sayac++` → 6. Koşula geri dönülür.

Adım 8 / 10

DEĞİŞKENLER

`n = 5` `faktoriyel = 120` `sayac = 6`

ÇIKTI

Faktöriyel Hesaplama (while)

```
1 int n = 5;
2 int faktoriyel = 1;
3 int sayac = 1;
▶ while (sayac <= n) {
5     faktoriyel *= sayac;
6     sayac++;
7 }
8 System.out.println(n + " faktöriyeli: " + faktoriyel);
```

6 <= 5 → false

Koşul yanlış: döngüden çıkılır.

Adım 9 / 10

DEĞİŞKENLER

`n = 5` `faktoriyel = 120` `sayac = 6`

ÇIKTI

Faktöriyel Hesaplama (while)

```
1 int n = 5;
2 int faktoriyel = 1;
3 int sayac = 1;
4 while (sayac <= n) {
5     faktoriyel *= sayac;
6     sayac++;
7 }
▶ System.out.println(n + " faktöriyeli: " + faktoriyel);
```

5! = 120

Sonuç yazılır. `int` 12!'ye kadar yeterlidir; 13! taşar, `long` gerekir.

Adım 10 / 10

DEĞİŞKENLER

`n = 5` `faktoriyel = 120` `sayac = 6`

ÇIKTI

```
5 faktöriyeli: 120
```

Üs Alma (for)

```java
import java.util.Scanner;

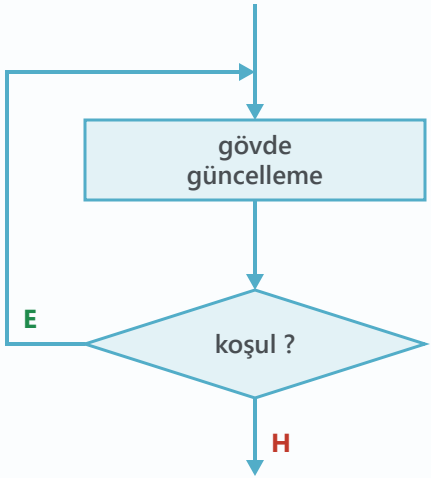
public class UsAlma {
    public static void main(String[] args) {
        Scanner tarayici = new Scanner(System.in);
        System.out.print("Taban ve üs girin: ");
        int taban = tarayici.nextInt();
        int us = tarayici.nextInt();
        long sonuc = 1;
        for (int i = 1; i <= us; i++) {
            sonuc = sonuc * taban;
        }
        System.out.println("Sonuç: " + sonuc);
        System.out.println("Math.pow: " + Math.pow(taban, us));
        tarayici.close();
    }
}
```

```
Taban ve üs girin: 3 4
Sonuç: 81
Math.pow: 81.0
```

Java'nın hazır `Math.pow(taban, us)` metodu aynı işi yapar ama sonucu `double` döndürür (81.0). Kendi döngümüz algoritmanın nasıl çalıştığını gösterir; gerçek programlarda hazır metotlar tercih edilir.

do-while Döngüsü

```
int i = 0;
do {
    // Döngü içinde yapılacak işlemler
    i++;
} while (i < 5);
```



- **Önce işlem yapar**, ardından koşulu kontrol eder.
- Koşul sağlandığı sürece tekrar eder.
- Gövde koşuldan bağımsız olarak **en az bir kez** çalışır.
- `while (koşul)` 'dan sonra **noktalı virgül** zorunludur.
- Tipik kullanım: kullanıcıdan geçerli bir değer alınana kadar tekrar sormak, menüler.

do-while: Girdi Doğrulama

```java
import java.util.Scanner;

public class DoWhileGirdi {
    public static void main(String[] args) {
        Scanner klavye = new Scanner(System.in);
        int sayi;
        do {
            System.out.print("1-10 arası bir sayı girin: ");
            sayi = klavye.nextInt();
        } while (sayi < 1 || sayi > 10);
        System.out.println("Kabul edilen sayı: " + sayi);
    }
}
```

GİRDİLER: 15, 0, 7

```
1-10 arası bir sayı girin: 15
1-10 arası bir sayı girin: 0
1-10 arası bir sayı girin: 7
Kabul edilen sayı: 7
```

- Soru **en az bir kez** sorulmalıdır: `do-while` tam bu iş içindir.
- 15 ve 0 aralık dışında: koşul `sayi < 1 || sayi > 10` true → tekrar sor.
- 7 geçerli: koşul false → döngüden çık.
- `sayi` döngüden önce tanımlanır; çünkü koşulda ve döngüden sonra da kullanılır.

Girdi Doğrulama: Kodun Adım Adım İzlenmesi

```
Scanner klavye = new Scanner(System.in);
int sayi;
do {
    System.out.print("1-10 arası bir sayı girin: ");
    sayi = klavye.nextInt();
} while (sayi < 1 || sayi > 10);
System.out.println("Kabul edilen sayı: " + sayi);
```

klavye girdileri

`12` `-3` `4`

Kullanıcı sırayla **12**, **−3**, **4** yazacak. `sayi` döngüden önce bildirilir; koşulda ve döngüden sonra da kullanılır.

Adım 1 / 8

DEĞİŞKENLER

—

ÇIKTI

Girdi Doğrulama: Kodun Adım Adım İzlenmesi

```
1 Scanner klavye = new Scanner(System.in);
2 int sayi;
3 do {
▶     System.out.print("1-10 arası bir sayı girin: ");
▶     sayi = klavye.nextInt();
6 } while (sayi < 1 || sayi > 10);
7 System.out.println("Kabul edilen sayı: " + sayi);
```

klavye girdileri

12 | -3 | 4

`sayi = 12`

Soru koşula bakılmadan sorulur. `nextInt()` ilk girdiyi okur: `sayi = 12`.

Adım 2 / 8

DEĞİŞKENLER

`sayi = 12`

ÇIKTI

```
1-10 arası bir sayı girin: 12
```

Girdi Doğrulama: Kodun Adım Adım İzlenmesi

```
1 Scanner klavye = new Scanner(System.in);
2 int sayi;
3 do {
4     System.out.print("1-10 arası bir sayı girin: ");
5     sayi = klavye.nextInt();
▶ } while (sayi < 1 || sayi > 10);
7 System.out.println("Kabul edilen sayı: " + sayi);
```

klavye girdileri

12 | -3 | 4

sayi = 12

12 < 1 || 12 > 10 → true

tekrar sor

`12 < 1 || 12 > 10` : sağ taraf true → koşul **true**, soru yeniden sorulur.

Adım 3 / 8

DEĞİŞKENLER

`sayi = 12`

ÇIKTI

```
1-10 arası bir sayı girin: 12
```

Girdi Doğrulama: Kodun Adım Adım İzlenmesi

```
1 Scanner klavye = new Scanner(System.in);
2 int sayi;
3 do {
▶     System.out.print("1-10 arası bir sayı girin: ");
▶     sayi = klavye.nextInt();
6 } while (sayi < 1 || sayi > 10);
7 System.out.println("Kabul edilen sayı: " + sayi);
```

klavye girdileri



`sayi = -3`

İkinci girdi okunur: `sayi = −3`.

Adım 4 / 8

DEĞİŞKENLER

`sayi = −3`

ÇIKTI

```
1-10 arası bir sayı girin: 12
1-10 arası bir sayı girin: -3
```

Girdi Doğrulama: Kodun Adım Adım İzlenmesi

```
1 Scanner klavye = new Scanner(System.in);
2 int sayi;
3 do {
4     System.out.print("1-10 arası bir sayı girin: ");
5     sayi = klavye.nextInt();
▶ } while (sayi < 1 || sayi > 10);
7 System.out.println("Kabul edilen sayı: " + sayi);
```

klavye girdileri

12 | -3 | 4 — sayi = -3

-3 < 1 || -3 > 10 → true — tekrar sor

`-3 < 1` true; `||` sağ tarafa bakmaz → koşul **true**, tekrar.

Adım 5 / 8

DEĞİŞKENLER

`sayi = -3`

ÇIKTI

```
1-10 arası bir sayı girin: 12
1-10 arası bir sayı girin: -3
```

Girdi Doğrulama: Kodun Adım Adım İzlenmesi

```
1 Scanner klavye = new Scanner(System.in);
2 int sayi;
3 do {
▶     System.out.print("1-10 arası bir sayı girin: ");
▶     sayi = klavye.nextInt();
6 } while (sayi < 1 || sayi > 10);
7 System.out.println("Kabul edilen sayı: " + sayi);
```

klavye girdileri



sayi = 4

Üçüncü girdi: `sayi = 4` .

Adım 6 / 8

DEĞİŞKENLER

`sayi = 4`

ÇIKTI

```
1-10 arası bir sayı girin: 12
1-10 arası bir sayı girin: -3
1-10 arası bir sayı girin: 4
```

Girdi Doğrulama: Kodun Adım Adım İzlenmesi

```
1 Scanner klavye = new Scanner(System.in);
2 int sayi;
3 do {
4     System.out.print("1-10 arası bir sayı girin: ");
5     sayi = klavye.nextInt();
▶ } while (sayi < 1 || sayi > 10);
7 System.out.println("Kabul edilen sayı: " + sayi);
```

klavye girdileri

12 | -3 | **4**

sayi = 4

4 < 1 || 4 > 10 → false

çık

`4 < 1` false, `4 > 10` false → koşul **false**: döngüden çıkılır.

Adım 7 / 8

DEĞİŞKENLER

`sayi = 4`

ÇIKTI

```
1-10 arası bir sayı girin: 12
1-10 arası bir sayı girin: -3
1-10 arası bir sayı girin: 4
```

Girdi Doğrulama: Kodun Adım Adım İzlenmesi

```
1 Scanner klavye = new Scanner(System.in);
2 int sayi;
3 do {
4     System.out.print("1-10 arası bir sayı girin: ");
5     sayi = klavye.nextInt();
6 } while (sayi < 1 || sayi > 10);
▶ System.out.println("Kabul edilen sayı: " + sayi);
```

klavye girdileri



sayi = 4

Geçerli sayı kabul edilir. Soru toplam 3 kez soruldu; geçerli bir sayı gelene kadar döngü sürer.

Adım 8 / 8

DEĞİŞKENLER

`sayi = 4`

ÇIKTI

```
1-10 arası bir sayı girin: 12
1-10 arası bir sayı girin: -3
1-10 arası bir sayı girin: 4
Kabul edilen sayı: 4
```

while ve do-while Farkı

```
▶ int i = 10;
2 while (i < 5) {
3     System.out.println("while: " + i);
4     i++;
5 }
6 int j = 10;
7 do {
8     System.out.println("do-while: " + j);
9     j++;
10 } while (j < 5);
11 System.out.println("i = " + i + ", j = " +
   j);
```

i = 10

`i` = 10.

Adım 1 / 8

DEĞİŞKENLER

`i = 10`

ÇIKTI

while ve do-while Farkı

```
1  int i = 10;
▶ while (i < 5) {
3      System.out.println("while: " + i);
4      i++;
5  }
6  int j = 10;
7  do {
8      System.out.println("do-while: " + j);
9      j++;
10 } while (j < 5);
11 System.out.println("i = " + i + ", j = " +
   j);
```

i < 5 → 10 < 5 → false

`while` koşulu **önce** sınar: false. Gövde (satır 3–4) **hiç** çalışmaz.

Adım 2 / 8

DEĞİŞKENLER

`i = 10`

ÇIKTI

while ve do-while Farkı

```
1  int i = 10;
2  while (i < 5) {
3      System.out.println("while: " + i);
4      i++;
5  }
   int j = 10;
7  do {
8      System.out.println("do-while: " + j);
9      j++;
10 } while (j < 5);
11 System.out.println("i = " + i + ", j = " +
   j);
```

j = 10

`j` = 10.

Adım 3 / 8

DEĞİŞKENLER

`i = 10` `j = 10`

ÇIKTI

while ve do-while Farkı

```
1  int i = 10;
2  while (i < 5) {
3      System.out.println("while: " + i);
4      i++;
5  }
6  int j = 10;
▶  do {
8      System.out.println("do-while: " + j);
9      j++;
10 } while (j < 5);
11 System.out.println("i = " + i + ", j = " +
   j);
```

do: koşula bakmadan gir

`do` bloğuna koşul sınanmadan girilir.

Adım 4 / 8

DEĞİŞKENLER

`i = 10` `j = 10`

ÇIKTI

while ve do-while Farkı

```
1  int i = 10;
2  while (i < 5) {
3      System.out.println("while: " + i);
4      i++;
5  }
6  int j = 10;
7  do {
▶      System.out.println("do-while: " + j);
9      j++;
10 } while (j < 5);
11 System.out.println("i = " + i + ", j = " +
   j);
```

do-while: 10

Gövde bir kez çalışır.

Adım 5 / 8

DEĞİŞKENLER

`i = 10` `j = 10`

ÇIKTI

```
do-while: 10
```

while ve do-while Farkı

```
1  int i = 10;
2  while (i < 5) {
3      System.out.println("while: " + i);
4      i++;
5  }
6  int j = 10;
7  do {
8      System.out.println("do-while: " + j);
▶      j++;
10 } while (j < 5);
11 System.out.println("i = " + i + ", j = " +
   j);
```

j = 11

`j++` → 11.

Adım 6 / 8

DEĞİŞKENLER

`i = 10` `j = 11`

ÇIKTI

```
do-while: 10
```

while ve do-while Farkı

```
1  int i = 10;
2  while (i < 5) {
3      System.out.println("while: " + i);
4      i++;
5  }
6  int j = 10;
7  do {
8      System.out.println("do-while: " + j);
9      j++;
▶ } while (j < 5);
11 System.out.println("i = " + i + ", j = " +
   j);
```

j < 5 → 11 < 5 → false

Koşul **sonra** sınanır: false → çıkılır.

Adım 7 / 8

DEĞİŞKENLER

`i = 10` `j = 11`

ÇIKTI

```
do-while: 10
```

while ve do-while Farkı

```
1  int i = 10;
2  while (i < 5) {
3      System.out.println("while: " + i);
4      i++;
5  }
6  int j = 10;
7  do {
8      System.out.println("do-while: " + j);
9      j++;
10 } while (j < 5);
▶  System.out.println("i = " + i + ", j = " + j);
```

while: 0 tur · do-while: 1 tur

Aynı koşul, farklı sonuç: `while` 0 kez, `do-while` 1 kez çalıştı.

Adım 8 / 8

DEĞİŞKENLER

`i = 10` `j = 11`

ÇIKTI

```
do-while: 10
i = 10, j = 11
```

Döngü Türlerinin Karşılaştırılması

	`for`	`while`	`do-while`
Koşul nerede sınanır?	Her turdan önce	Her turdan önce	Her turdan **sonra**
En az kaç tur?	0	0	1
Başlangıç / güncelleme	Başlıkta, tek satırda	Döngü dışında / gövdede	Döngü dışında / gövdede
Ne zaman?	Tekrar sayısı belli: "1'den n'e", "dizinin her öğesi"	Bir koşul değişene kadar: "sayı 0 olana kadar"	İşlem en az bir kez yapılmalı: girdi doğrulama, menü
Söz dizimi	`for (bas; koşul; günc) { }`	`while (koşul) { }`	`do { } while (koşul);`

Üç döngü birbirine dönüştürülebilir; seçim okunabilirliğe göre yapılır.

DÖNGÜLER

for Döngüsü: Deneme Alanı

başlangıç `1` koşul i `<=` `10` güncelleme `i++` **çalıştır**

```
for (int i = 1; i <= 10; i++) { toplam += i; }
```

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

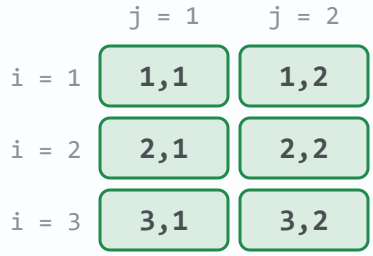
Gövde **10** kez çalıştı · toplam = **55** · çıkışta i = **11** (11 <= 10 yanlış).

Kutular, gövdenin çalıştığı her turdaki `i` değeridir. Örnek ayarlar: başlangıç 10, koşul `<=`, bitiş 1 (hiç dönmez); güncelleme `i--` ile `<=` (sonsuz döngü); başlangıç 1, `!=` ve `i += 2` ile bitiş 9 (durur) ve 10 (hiç eşit olmaz: sonsuz). 200 turdan sonra simülasyon durdurulur.

İç İçe Döngüler

- **İç içe döngü** (nested loop): bir döngünün gövdesinde başka bir döngü kullanmaktır.
- Daha karmaşık işlemler için kullanılır: **matris** ve çok boyutlu diziler, tablolar, **desenler**.
- Benzetme: saat. Akrep (dış döngü) bir ilerlediğinde yelkovan (iç döngü) tam bir tur atar.
- Toplam tur sayısı çarpımdır: dış 3 tur × iç 2 tur = **6** kez gövde.
- İç içe döngülerin sayısı ve koşulları kodun karmaşıklığını etkiler. **Fazla iç içe döngüler** karmaşık ve yavaş kodlara yol açabilir: üç döngü × 1000 tur = 1 milyar işlem!

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 2; j++) {
        // İç içe döngü gövdesi
    }
}
```



İç İçe Döngülerin İşleyişi

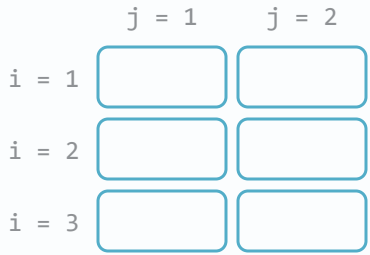
```
▶ for (int i = 1; i <= 3; i++) {
2     for (int j = 1; j <= 2; j++) {
3         System.out.println("i = " + i + ", j = " + j);
4     }
5 }
```

DEĞİŞKENLER

`i = 1`

ÇIKTI





Dış döngü başlar: `i` = 1, 1 <= 3 → true.

Adım 1 / 10

İç İçe Döngülerin İşleyişi

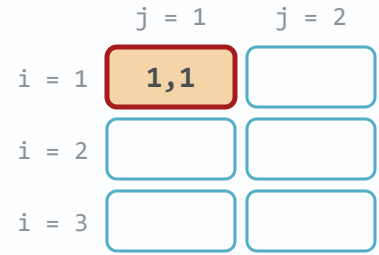
```
1 for (int i = 1; i <= 3; i++) {
2     for (int j = 1; j <= 2; j++) {
▶         System.out.println("i = " + i + ", j = " + j);
4     }
5 }
```

DEĞİŞKENLER

`i = 1` `j = 1`

ÇIKTI

```
i = 1, j = 1
```



İç döngü başlar: `j` = 1. Gövde: `i` = 1, `j` = 1 yazılır.

Adım 2 / 10

İç İçe Döngülerin İşleyişi

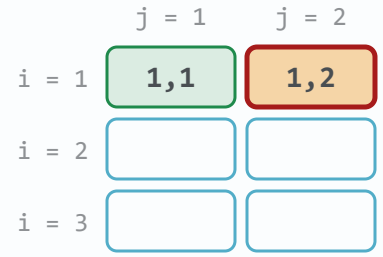
```
1 for (int i = 1; i <= 3; i++) {
2     for (int j = 1; j <= 2; j++) {
▶         System.out.println("i = " + i + ", j = " + j);
4     }
5 }
```

DEĞİŞKENLER

`i = 1` `j = 2`

ÇIKTI

```
i = 1, j = 1
i = 1, j = 2
```



Gövde: `i` = 1, `j` = 2 yazılır.

Adım 3 / 10

İç İçe Döngülerin İşleyişi

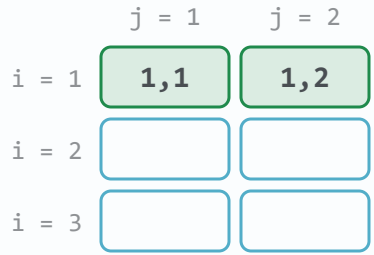
```
1 for (int i = 1; i <= 3; i++) {
▶       for (int j = 1; j <= 2; j++) {
3           System.out.println("i = " + i + ", j = " + j);
4       }
5   }
```

DEĞİŞKENLER

`i = 1` `j = 3`

ÇIKTI

```
i = 1, j = 1
i = 1, j = 2
```



`j` = 3 → 3 <= 2 false: iç döngü **tamamlandı**. Dışa dönülür: `i++` → 2, 2 <= 3 → true; iç döngü **baştan** başlayacak.

İç İçe Döngülerin İşleyişi

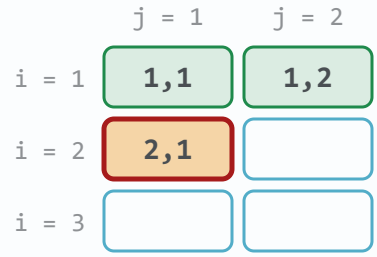
```
1 for (int i = 1; i <= 3; i++) {
2     for (int j = 1; j <= 2; j++) {
▶         System.out.println("i = " + i + ", j = " + j);
4     }
5 }
```

DEĞİŞKENLER

`i = 2` `j = 1`

ÇIKTI

```
i = 1, j = 1
i = 1, j = 2
i = 2, j = 1
```



İç döngü başlar: `j` = 1. Gövde: `i` = 2, `j` = 1 yazılır.

Adım 5 / 10

İç İçe Döngülerin İşleyişi

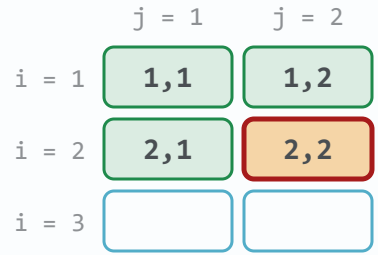
```
1 for (int i = 1; i <= 3; i++) {
2     for (int j = 1; j <= 2; j++) {
▶         System.out.println("i = " + i + ", j = " + j);
4     }
5 }
```

DEĞİŞKENLER

`i = 2` `j = 2`

ÇIKTI

```
i = 1, j = 1
i = 1, j = 2
i = 2, j = 1
i = 2, j = 2
```



Gövde: `i` = 2, `j` = 2 yazılır.

Adım 6 / 10

İç İçe Döngülerin İşleyişi

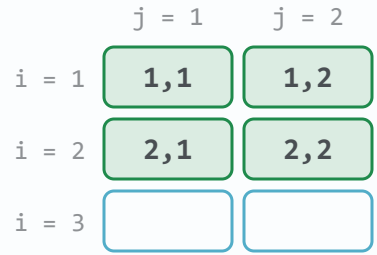
```
1 for (int i = 1; i <= 3; i++) {
▶       for (int j = 1; j <= 2; j++) {
3           System.out.println("i = " + i + ", j = " + j);
4       }
5 }
```

DEĞİŞKENLER

`i = 2` `j = 3`

ÇIKTI

```
i = 1, j = 1
i = 1, j = 2
i = 2, j = 1
i = 2, j = 2
```



`j` = 3 → 3 <= 2 false: iç döngü **tamamlandı**. Dışa dönülür: `i++` → 3, 3 <= 3 → true; iç döngü **baştan** başlayacak.

İç İçe Döngülerin İşleyişi

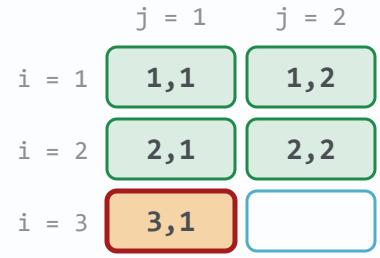
```
1 for (int i = 1; i <= 3; i++) {
2     for (int j = 1; j <= 2; j++) {
▶         System.out.println("i = " + i + ", j = " + j);
4     }
5 }
```

DEĞİŞKENLER

`i = 3` `j = 1`

ÇIKTI

```
i = 1, j = 1
i = 1, j = 2
i = 2, j = 1
i = 2, j = 2
i = 3, j = 1
```



İç döngü başlar: `j` = 1. Gövde: `i` = 3, `j` = 1 yazılır.

İç İçe Döngülerin İşleyişi

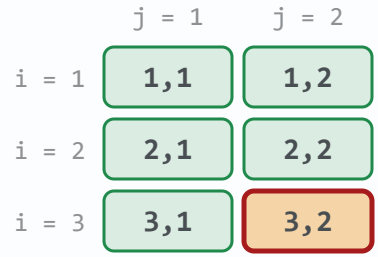
```
1 for (int i = 1; i <= 3; i++) {
2     for (int j = 1; j <= 2; j++) {
▶         System.out.println("i = " + i + ", j = " + j);
4     }
5 }
```

DEĞİŞKENLER

`i = 3` `j = 2`

ÇIKTI

```
i = 1, j = 1
i = 1, j = 2
i = 2, j = 1
i = 2, j = 2
i = 3, j = 1
i = 3, j = 2
```



Gövde: `i` = 3, `j` = 2 yazılır.

Adım 9 / 10

İç İçe Döngülerin İşleyişi

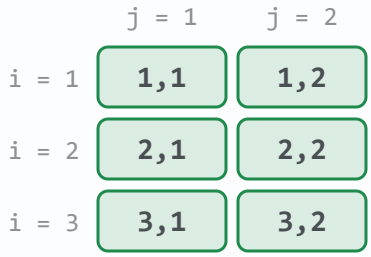
```
▶ for (int i = 1; i <= 3; i++) {
2     for (int j = 1; j <= 2; j++) {
3         System.out.println("i = " + i + ", j = " + j);
4     }
5 }
```

DEĞİŞKENLER

`i = 4`

ÇIKTI

```
i = 1, j = 1
i = 1, j = 2
i = 2, j = 1
i = 2, j = 2
i = 3, j = 1
i = 3, j = 2
```



`j` = 3 → 3 <= 2 false: iç döngü **tamamlandı**. `i++` → 4, 4 <= 3 false: dış döngü de biter. Gövde 3 × 2 = 6 kez çalıştı.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

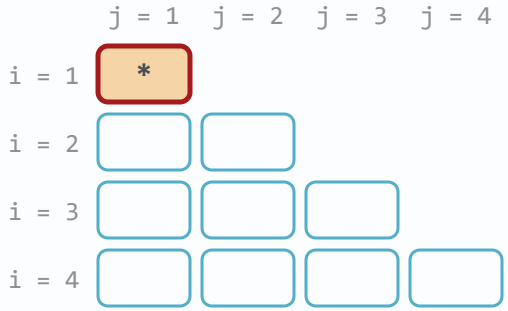
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
▶         System.out.print("* ");
4     }
5     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 1` `j = 1`

ÇIKTI

```
*
```



1. satır: iç döngü `j` = 1'den `i` = 1'e kadar döner. `print("* ")` : aynı satırda kalınır.

Adım 1 / 14

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

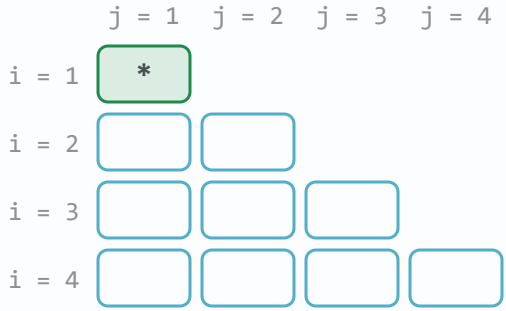
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
3         System.out.print("* ");
4     }
▶     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 1` `j = 2`

ÇIKTI

```
*
```



İç döngü bitti (`j` = 2 > 1). `println()` alt satıra geçer.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

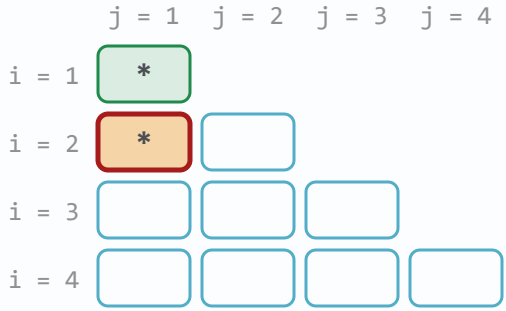
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
          System.out.print("* ");
4     }
5     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 2` `j = 1`

ÇIKTI

```
*
*
```



2. satır: iç döngü `j` = 1'den `i` = 2'ye kadar döner. `print("* ")` : aynı satırda kalınır.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

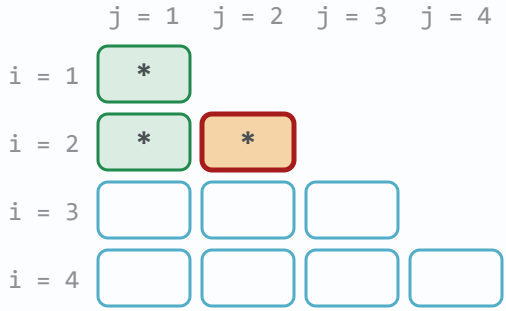
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
          System.out.print("* ");
4     }
5     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 2` `j = 2`

ÇIKTI

```
*
* *
```



`print("* ")` : aynı satırda kalınır.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

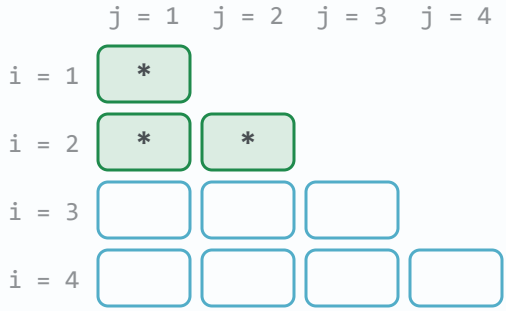
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
3         System.out.print("* ");
4     }
▶     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 2` `j = 3`

ÇIKTI

```
*
* *
```



İç döngü bitti (`j` = 3 > 2). `println()` alt satıra geçer.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

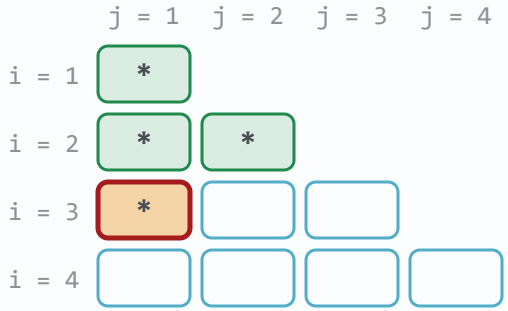
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
▶         System.out.print("* ");
4     }
5     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 3` `j = 1`

ÇIKTI

```
*
* *
*
```



3. satır: iç döngü `j` = 1'den `i` = 3'e kadar döner. `print("* ")` : aynı satırda kalınır.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

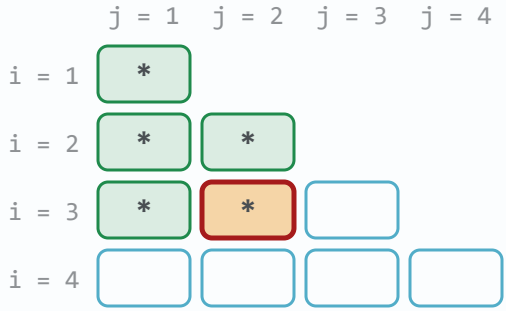
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
▶         System.out.print("* ");
4     }
5     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 3` `j = 2`

ÇIKTI

```
*
* *
* *
```



`print("* ")` : aynı satırda kalınır.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

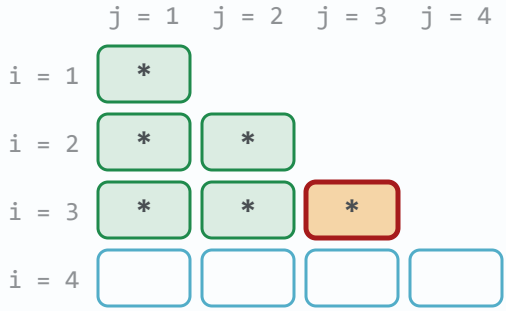
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
▶         System.out.print("* ");
4     }
5     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 3` `j = 3`

ÇIKTI

```
*
* *
* * *
```



`print("* ")` : aynı satırda kalınır.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

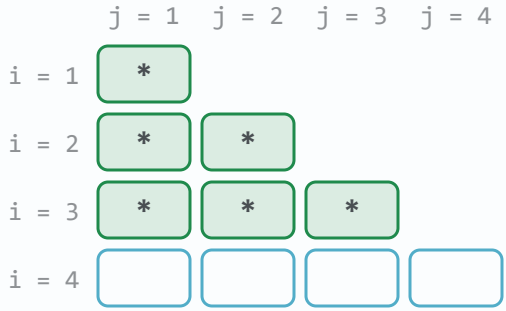
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
3         System.out.print("* ");
4     }
▶     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 3` `j = 4`

ÇIKTI

```
*
* *
* * *
```



İç döngü bitti (`j` = 4 > 3). `println()` alt satıra geçer.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

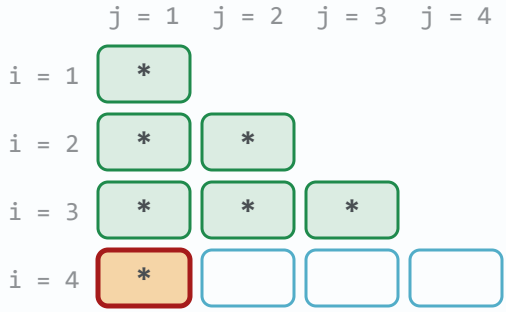
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
          System.out.print("* ");
4     }
5     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 4` `j = 1`

ÇIKTI

```
*
* *
* * *
*
```



4. satır: iç döngü `j` = 1'den `i` = 4'e kadar döner. `print("* ")` : aynı satırda kalınır.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

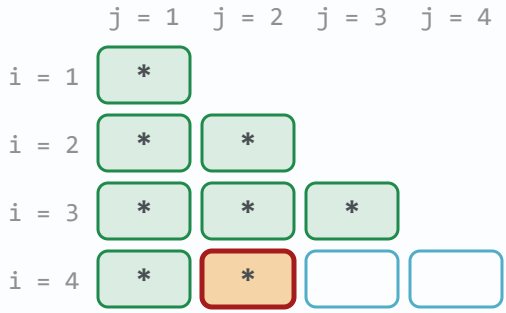
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
          System.out.print("* ");
4     }
5     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 4` `j = 2`

ÇIKTI

```
*
* *
* * *
* *
```



`print("* ")` : aynı satırda kalınır.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

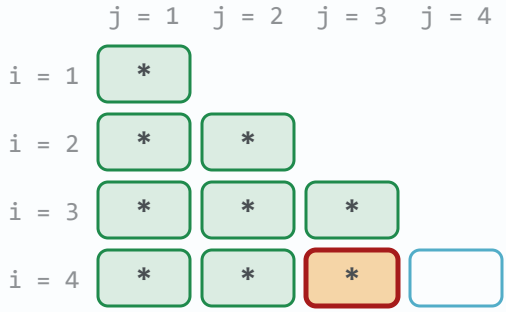
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
▶         System.out.print("* ");
4     }
5     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 4` `j = 3`

ÇIKTI

```
*
* *
* * *
* * *
```



`print("* ")` : aynı satırda kalınır.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

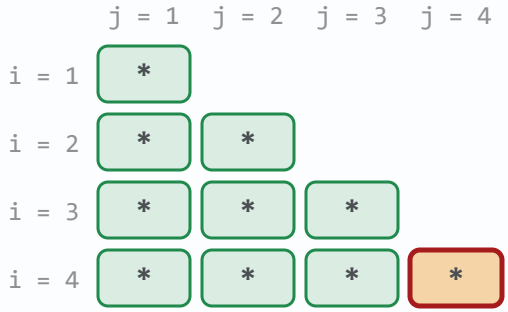
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
▶         System.out.print("* ");
4     }
5     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 4` `j = 4`

ÇIKTI

```
*
* *
* * *
* * * *
```



`print("* ")` : aynı satırda kalınır.

Örnek: Yıldız Deseni

İç döngünün sınırı dış döngünün değişkenine bağlıdır (`j <= i`): her satırda bir yıldız fazla yazılır.

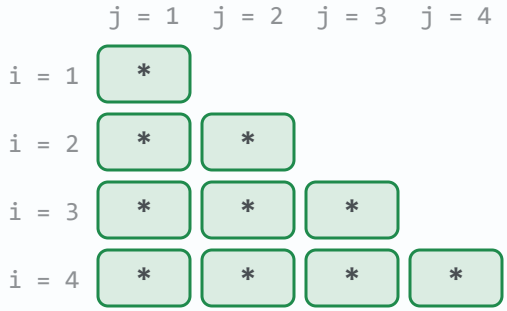
```
1 for (int i = 1; i <= 4; i++) {
2     for (int j = 1; j <= i; j++) {
3         System.out.print("* ");
4     }
▶     System.out.println();
6 }
```

DEĞİŞKENLER

`i = 4` `j = 5`

ÇIKTI

```
*
* *
* * *
* * * *
```



İç döngü bitti (`j` = 5 > 4). `println()` alt satıra geçer. Toplam 1 + 2 + 3 + 4 = 10 yıldız.

Basamak Değeri: Çıktı Ne Olur?

```
for (int onlar = 0; onlar <= 4; onlar++) {
    for (int birler = 1; birler <= 9; birler++) {
        int sayi = onlar * 10 + birler;
        System.out.println("Onlar basamağı: " + onlar
                + ", Birler basamağı: " + birler + " --> " + sayi);
    }
}
```

Soru: Program kaç satır yazar? İlk ve son satır nedir? Hangi sayılar **hiç** yazılmaz?

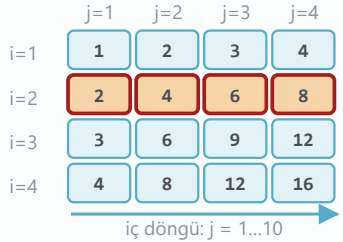
▶ **Cevap**

İÇ İÇE DÖNGÜLER

Çarpım Tablosu

```java
for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 10; j++) {
        int carpim = i * j;
        System.out.print(i + "x" + j + "=" + carpim + "\t");
    }
    System.out.println(); // Yeni satır
}
```

- Dış döngü **satırları** (`i`), iç döngü **sütunları** (`j`) gezer: 10 × 10 = 100 çarpım.
- `print` satır atlamaz; `"\t"` (sekme, tab) sütunları hizalar.
- Satır bitince `println()` ile alt satıra geçilir. Bu satır iç döngünün **dışında** ama dış döngünün **içindedir**.



ÇIKTIDAN KESİT (SEKMELER BOŞLUKLA GÖSTERİLDİ)

```
1x1=1   1x2=2   1x3=3   1x4=4   1x5=5   1x6=6   …
2x1=2   2x2=4   2x3=6   2x4=8   2x5=10   2x6=12   …
3x1=3   3x2=6   3x3=9   3x4=12   3x5=15   3x6=18   …
…
10x1=10   10x2=20   10x3=30   10x4=40   10x5=50   …
```

Döngülerde break

`break`, döngüyü **sonlandırmak** için kullanılır: koşul sağlandığında döngüden çıkılır, döngü sona erer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i == 5) {
3         break;
4     }
▶     System.out.println("Döngü adımı: " + i);
6 }
7 System.out.println("Döngüden çıkıldı.");
```

i

1	2	3	4	5	6	7	8	9	10

`i` = 1: 1 == 5 false, `break` atlanır; adım yazılır.

Adım 1 / 6

DEĞİŞKENLER

`i = 1`

ÇIKTI

```
Döngü adımı: 1
```

Döngülerde break

`break`, döngüyü **sonlandırmak** için kullanılır: koşul sağlandığında döngüden çıkılır, döngü sona erer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i == 5) {
3         break;
4     }
▶     System.out.println("Döngü adımı: " + i);
6 }
7 System.out.println("Döngüden çıkıldı.");
```

i

1	2	3	4	5	6	7	8	9	10

✓

`i` = 2: 2 == 5 false → yazılır.

Adım 2 / 6

DEĞİŞKENLER

`i = 2`

ÇIKTI

```
Döngü adımı: 1
Döngü adımı: 2
```

Döngülerde break

`break`, döngüyü **sonlandırmak** için kullanılır: koşul sağlandığında döngüden çıkılır, döngü sona erer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i == 5) {
3         break;
4     }
▶     System.out.println("Döngü adımı: " + i);
6 }
7 System.out.println("Döngüden çıkıldı.");
```

i



`i` = 3: 3 == 5 false → yazılır.

Adım 3 / 6

DEĞİŞKENLER

`i = 3`

ÇIKTI

```
Döngü adımı: 1
Döngü adımı: 2
Döngü adımı: 3
```

Döngülerde break

`break`, döngüyü **sonlandırmak** için kullanılır: koşul sağlandığında döngüden çıkılır, döngü sona erer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i == 5) {
3         break;
4     }
▶     System.out.println("Döngü adımı: " + i);
6 }
7 System.out.println("Döngüden çıkıldı.");
```



`i` = 4: 4 == 5 false → yazılır.

Adım 4 / 6

DEĞİŞKENLER

`i = 4`

ÇIKTI

```
Döngü adımı: 1
Döngü adımı: 2
Döngü adımı: 3
Döngü adımı: 4
```

Döngülerde break

`break`, döngüyü **sonlandırmak** için kullanılır: koşul sağlandığında döngüden çıkılır, döngü sona erer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i == 5) {
3         break;
4     }
5     System.out.println("Döngü adımı: " + i);
6 }
7 System.out.println("Döngüden çıkıldı.");
```

i



`i` = 5: koşul true → `break`. Döngü **hemen** sona erer; 5 yazılmaz, 6–10 hiç denenmez.

Adım 5 / 6

DEĞİŞKENLER

`i = 5`

ÇIKTI

```
Döngü adımı: 1
Döngü adımı: 2
Döngü adımı: 3
Döngü adımı: 4
```

Döngülerde break

`break`, döngüyü **sonlandırmak** için kullanılır: koşul sağlandığında döngüden çıkılır, döngü sona erer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i == 5) {
3         break;
4     }
5     System.out.println("Döngü adımı: " + i);
6 }
▶ System.out.println("Döngüden çıkıldı.");
```

i



Döngü dışındaki işlemler devam eder. `i` artık tanımlı değil.

Adım 6 / 6

DEĞİŞKENLER

—

ÇIKTI

```
Döngü adımı: 1
Döngü adımı: 2
Döngü adımı: 3
Döngü adımı: 4
Döngüden çıkıldı.
```

Asal Sayı: Bayrak ve `break`

```java
import java.util.Scanner;

public class AsalSayi {
    public static void main(String[] args) {
        Scanner tarayici = new Scanner(System.in);
        System.out.print("Bir sayı girin: ");
        int sayi = tarayici.nextInt();
        boolean asalMi = true;            // bayrak (flag)
        if (sayi < 2) asalMi = false;     // 0 ve 1 asal değildir
        for (int bolen = 2; bolen <= sayi / 2; bolen++) {
            if (sayi % bolen == 0) {      // tam bölünüyor
                asalMi = false;
                break;                    // döngüyü kır
            }
        }
        if (asalMi) {
            System.out.println(sayi + " bir asal sayıdır.");
        } else {
            System.out.println(sayi + " bir asal sayı değildir.");
        }
        tarayici.close();
    }
}
```

```
Bir sayı girin: 21
21 bir asal sayı değildir.

Bir sayı girin: 97
97 bir asal sayıdır.
```

Daha hızlısı: bir sayının böleni varsa biri √sayı'dan küçük veya eşittir. Koşul `bolen * bolen <= sayi` olunca 97 için 47 yerine yalnızca 8 bölen denenir.

Döngülerde continue

`continue`, o anki adımın kalanını atlar ve **bir sonraki adıma** geçer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i % 2 == 0) {
3         continue;
4     }
▶     System.out.println("Tek sayı: " + i);
6 }
```

i

1	2	3	4	5	6	7	8	9	10

`i` = 1 tek → koşul false, sayı yazılır.

Adım 1 / 11

DEĞİŞKENLER

`i = 1`

ÇIKTI

```
Tek sayı: 1
```

Döngülerde continue

`continue`, o anki adımın kalanını atlar ve **bir sonraki adıma** geçer.

```
1 for (int i = 1; i <= 10; i++) {
▶     if (i % 2 == 0) {
▶         continue;
4     }
5     System.out.println("Tek sayı: " + i);
6 }
```

i



✓

`i` = 2 çift → `continue`: satır 5 atlanır, doğrudan `i++`'ya geçilir.

Adım 2 / 11

DEĞİŞKENLER

`i = 2`

ÇIKTI

```
Tek sayı: 1
```

Döngülerde continue

`continue`, o anki adımın kalanını atlar ve **bir sonraki adıma** geçer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i % 2 == 0) {
3         continue;
4     }
▶     System.out.println("Tek sayı: " + i);
6 }
```



`i` = 3 tek → koşul false, sayı yazılır.

Adım 3 / 11

DEĞİŞKENLER

`i = 3`

ÇIKTI

```
Tek sayı: 1
Tek sayı: 3
```

Döngülerde continue

`continue`, o anki adımın kalanını atlar ve **bir sonraki adıma** geçer.

```
1 for (int i = 1; i <= 10; i++) {
▶       if (i % 2 == 0) {
▶           continue;
4       }
5       System.out.println("Tek sayı: " + i);
6   }
```



`i` = 4 çift → `continue`: satır 5 atlanır, doğrudan `i++`'ya geçilir.

Adım 4 / 11

DEĞİŞKENLER

`i = 4`

ÇIKTI

```
Tek sayı: 1
Tek sayı: 3
```

Döngülerde continue

`continue`, o anki adımın kalanını atlar ve **bir sonraki adıma** geçer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i % 2 == 0) {
3         continue;
4     }
▶     System.out.println("Tek sayı: " + i);
6 }
```



`i` = 5 tek → koşul false, sayı yazılır.

Adım 5 / 11

DEĞİŞKENLER

`i = 5`

ÇIKTI

```
Tek sayı: 1
Tek sayı: 3
Tek sayı: 5
```

Döngülerde continue

`continue`, o anki adımın kalanını atlar ve **bir sonraki adıma** geçer.

```
1 for (int i = 1; i <= 10; i++) {
▶     if (i % 2 == 0) {
▶         continue;
4     }
5     System.out.println("Tek sayı: " + i);
6 }
```



`i` = 6 çift → `continue`: satır 5 atlanır, doğrudan `i++`'ya geçilir.

Adım 6 / 11

DEĞİŞKENLER

`i = 6`

ÇIKTI

```
Tek sayı: 1
Tek sayı: 3
Tek sayı: 5
```

Döngülerde continue

`continue`, o anki adımın kalanını atlar ve **bir sonraki adıma** geçer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i % 2 == 0) {
3         continue;
4     }
▶     System.out.println("Tek sayı: " + i);
6 }
```



`i` = 7 tek → koşul false, sayı yazılır.

Adım 7 / 11

DEĞİŞKENLER

`i = 7`

ÇIKTI

```
Tek sayı: 1
Tek sayı: 3
Tek sayı: 5
Tek sayı: 7
```

Döngülerde continue

`continue`, o anki adımın kalanını atlar ve **bir sonraki adıma** geçer.

```
1 for (int i = 1; i <= 10; i++) {
▶     if (i % 2 == 0) {
▶         continue;
4     }
5     System.out.println("Tek sayı: " + i);
6 }
```



`i` = 8 çift → `continue`: satır 5 atlanır, doğrudan `i++`'ya geçilir.

Adım 8 / 11

DEĞİŞKENLER

`i = 8`

ÇIKTI

```
Tek sayı: 1
Tek sayı: 3
Tek sayı: 5
Tek sayı: 7
```

Döngülerde continue

`continue`, o anki adımın kalanını atlar ve **bir sonraki adıma** geçer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i % 2 == 0) {
3         continue;
4     }
▶     System.out.println("Tek sayı: " + i);
6 }
```

i

1	2	3	4	5	6	7	8	9	10
✓	atla	✓	atla	✓	atla	✓	atla		

`i` = 9 tek → koşul false, sayı yazılır.

Adım 9 / 11

DEĞİŞKENLER

`i = 9`

ÇIKTI

```
Tek sayı: 1
Tek sayı: 3
Tek sayı: 5
Tek sayı: 7
Tek sayı: 9
```

Döngülerde continue

`continue`, o anki adımın kalanını atlar ve **bir sonraki adıma** geçer.

```
1 for (int i = 1; i <= 10; i++) {
2     if (i % 2 == 0) {
3         continue;
4     }
5     System.out.println("Tek sayı: " + i);
6 }
```

i



`i` = 10 çift → `continue`: satır 5 atlanır, doğrudan `i++`'ya geçilir.

Adım 10 / 11

DEĞİŞKENLER

`i = 10`

ÇIKTI

```
Tek sayı: 1
Tek sayı: 3
Tek sayı: 5
Tek sayı: 7
Tek sayı: 9
```

Döngülerde continue

`continue`, o anki adımın kalanını atlar ve **bir sonraki adıma** geçer.

```
▶ for (int i = 1; i <= 10; i++) {
2     if (i % 2 == 0) {
3         continue;
4     }
5     System.out.println("Tek sayı: " + i);
6 }
```

i



11 <= 10 false: döngü **normal biçimde** biter. `continue` döngüyü bitirmez, yalnızca o turu atlar.

Adım 11 / 11

DEĞİŞKENLER

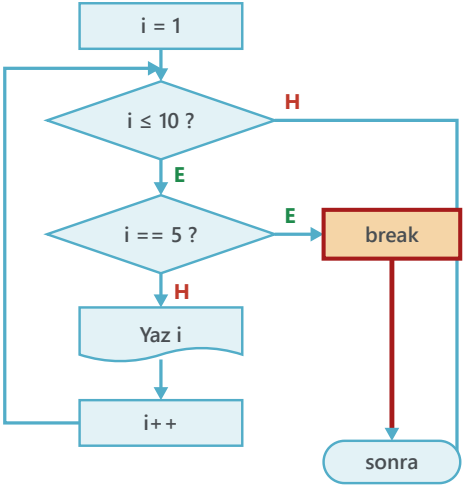
`i = 11`

ÇIKTI

```
Tek sayı: 1
Tek sayı: 3
Tek sayı: 5
Tek sayı: 7
Tek sayı: 9
```

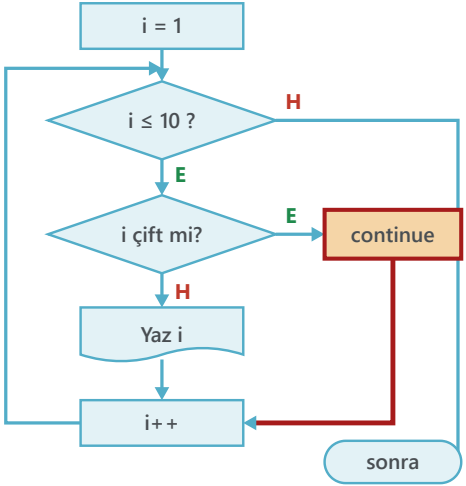
break ve continue Karşılaştırması

break: döngüyü bitirir



Döngü tamamen sona erer; döngüden sonraki deyime geçilir.

continue: turu atlar



Turun geri kalanı atlanır; güncelleme ve koşulla döngü sürer.

İkisi de genellikle bir `if` içinde, `while` ve `do-while` 'da da kullanılır; `while` 'da `continue` güncellemeyi atlarsa sonsuz döngü oluşabilir.

İç İçe Döngülerde break

BREAK YALNIZCA İÇTEKİ DÖNGÜYÜ BİTİRİR

```
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 3; j++) {
        if (j == 2) {
            break;
        }
        System.out.println(i + " " + j);
    }
}
```

```
1 1
2 1
3 1
```

ETİKETLİ BREAK: DIŞ DÖNGÜDEN ÇIKIŞ

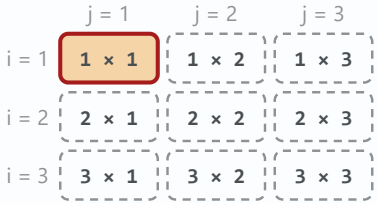
```
dis:
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 3; j++) {
        if (i * j == 4) {
            break dis;
        }
        System.out.println(i + " x " + j);
    }
}
```

```
1 x 1
1 x 2
1 x 3
2 x 1
```

Soldaki örnekte her `i` için yalnızca `j` = 1 yazılır, dış döngü sürer. Sağda `dis:` etiketi dış döngüyü adlandırır; `i * j == 4` olunca (`i` = 2, `j` = 2) `break dis;` iki döngüden birden çıkar. Etiketler seyrek kullanılmalıdır.

Etiketli break: Kodun Adım Adım İzlenmesi

```
1  dis:
2  for (int i = 1; i <= 3; i++) {
3      for (int j = 1; j <= 3; j++) {
▶          if (i * j == 4) {
5              break dis;
6          }
▶          System.out.println(i + " x " + j);
8      }
9  }
10 System.out.println("bitti");
```



`dis:` etiketi dış döngüyü adlandırır. `i = 1, j = 1` : `1 × 1 = 1 ≠ 4`, satır yazdırılır.

Adım 1 / 7

DEĞİŞKENLER

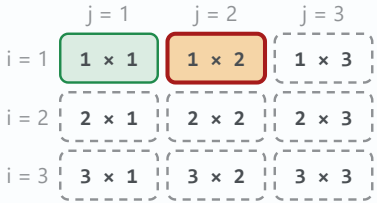
`i = 1` `j = 1`

ÇIKTI

```
1 x 1
```

Etiketli break: Kodun Adım Adım İzlenmesi

```
1  dis:
2  for (int i = 1; i <= 3; i++) {
3      for (int j = 1; j <= 3; j++) {
▶          if (i * j == 4) {
5              break dis;
6          }
▶          System.out.println(i + " x " +
   j);
8      }
9  }
10 System.out.println("bitti");
```



`j = 2` : `1 × 2 = 2` , yazdırılır.

Adım 2 / 7

DEĞİŞKENLER

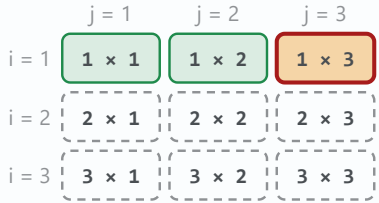
`i = 1` `j = 2`

ÇIKTI

```
1 x 1
1 x 2
```

Etiketli break: Kodun Adım Adım İzlenmesi

```
1  dis:
2  for (int i = 1; i <= 3; i++) {
3      for (int j = 1; j <= 3; j++) {
▶          if (i * j == 4) {
5              break dis;
6          }
▶          System.out.println(i + " x " + j);
8      }
9  }
10 System.out.println("bitti");
```



`j = 3` : `1 × 3 = 3` , yazdırılır. İç döngü biter; dış döngü `i = 2` 'ye geçer.

Adım 3 / 7

DEĞİŞKENLER

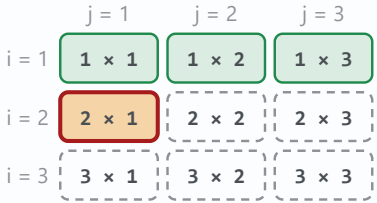
`i = 1` `j = 3`

ÇIKTI

```
1 x 1
1 x 2
1 x 3
```

Etiketli break: Kodun Adım Adım İzlenmesi

```
1  dis:
2  for (int i = 1; i <= 3; i++) {
3      for (int j = 1; j <= 3; j++) {
▶          if (i * j == 4) {
5              break dis;
6          }
▶          System.out.println(i + " x " +
   j);
8      }
9  }
10 System.out.println("bitti");
```



`i = 2, j = 1` : `2 × 1 = 2` , yazdırılır.

Adım 4 / 7

DEĞİŞKENLER

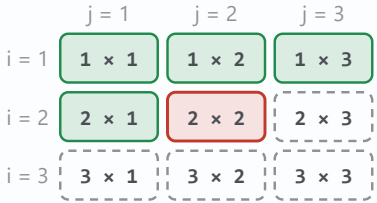
`i = 2` `j = 1`

ÇIKTI

```
1 x 1
1 x 2
1 x 3
2 x 1
```

Etiketli break: Kodun Adım Adım İzlenmesi

```
1  dis:
2  for (int i = 1; i <= 3; i++) {
3      for (int j = 1; j <= 3; j++) {
▶          if (i * j == 4) {
5              break dis;
6          }
7          System.out.println(i + " x " +
   j);
8      }
9  }
10 System.out.println("bitti");
```



`i = 2, j = 2` : `2 × 2 = 4` → koşul doğru.

Adım 5 / 7

DEĞİŞKENLER

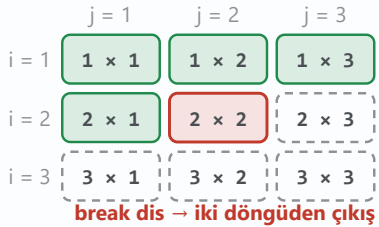
`i = 2` `j = 2`

ÇIKTI

```
1 x 1
1 x 2
1 x 3
2 x 1
```

Etiketli break: Kodun Adım Adım İzlenmesi

```
1  dis:
2  for (int i = 1; i <= 3; i++) {
3      for (int j = 1; j <= 3; j++) {
4          if (i * j == 4) {
▶              break dis;
6          }
7          System.out.println(i + " x " +
   j);
8      }
9  }
10 System.out.println("bitti");
```



`break dis;` yalnızca iç döngüden değil, **etiketli dış döngüden** de çıkar. Kalan hücreler hiç ziyaret edilmez.

Adım 6 / 7

DEĞİŞKENLER

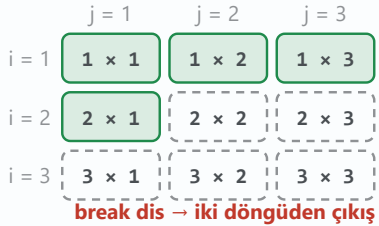
`i = 2` `j = 2`

ÇIKTI

```
1 x 1
1 x 2
1 x 3
2 x 1
```

Etiketli break: Kodun Adım Adım İzlenmesi

```
1 dis:
2 for (int i = 1; i <= 3; i++) {
3     for (int j = 1; j <= 3; j++) {
4         if (i * j == 4) {
5             break dis;
6         }
7         System.out.println(i + " x " +
  j);
8     }
9 }
▶ System.out.println("bitti");
```



Akış iki döngünün de sonrasına atlar: `bitti` yazılır. Etiketsiz `break` olsaydı `i = 3` turu da çalışırdı.

Adım 7 / 7

DEĞİŞKENLER

—

ÇIKTI

```
1 x 1
1 x 2
1 x 3
2 x 1
bitti
```

ÖZET

Özet

Yapı	Ne yapar?	Söz dizimi
Sıralı	Deyimler yukarıdan aşağıya birer kez çalışır	—
if / else	Koşul doğruysa bir blok, değilse diğeri	`if (k) { } else { }`
else if	İlk doğru koşulun bloğu çalışır	`if (k1) { } else if (k2) { } else { }`
switch	Bir değeri sabit `case`'lerle karşılaştırır; `break` ile çıkılır	`switch (x) { case 1: … break; default: … }`
for	Başlangıç, koşul, güncelleme tek satırda; sayılı tekrar	`for (int i = 0; i < n; i++) { }`
while	Koşul doğru oldukça; 0 veya daha çok tur	`while (k) { }`
do-while	Önce gövde, sonra koşul; en az 1 tur	`do { } while (k);`
break / continue	Döngüyü bitirir / o turu atlar	`break;` · `continue;`

ÖZET

Sık Yapılan Hatalar: Kararlar

= ile karşılaştırma

`if (x = 5)` → `error: incompatible types: int cannot be converted to boolean`. Doğrusu `x == 5`.

if'ten sonra noktalı virgül

`if (x > 5); { … }` derlenir ama `;` boş deyimdir: blok **her zaman** çalışır.

Parantezsiz blok

`if (k) a(); b(); else c();` → `error: 'else' without 'if'`. Birden çok deyim için `{ }` şarttır.

break'i unutmak

`switch`'te `break` yoksa alttaki `case`'ler de çalışır (aşağı düşme). Derleyici uyarmaz.

default olmadan atama

Değişken yalnızca `case`'lerde atanıyorsa: `error: variable mevsim might not have been initialized`. Çözüm: `default` eklemek.

Ondalık girişte nokta

Türkçe bölge ayarında `nextDouble()` için `-7.5` → `InputMismatchException`. `-7,5` yazılmalı.

ÖZET

Sık Yapılan Hatalar: Döngüler

Sonsuz döngü

`while`'da güncellemeyi (`i++`) unutmak ya da yanlış yönde güncellemek (`i--` ile `i <= 10`). Derleyici yakalamaz; program durmaz.

Döngü başlığında ;

`for (int i = 0; i < 3; i++);` veya `while (i < 5);` gövdesi boş bir döngüdür; alttaki blok döngüye ait değildir.

Sınır hatası (off-by-one)

"1'den 10'a kadar" için `i < 10` yazmak 10'u atlar; `i = 0` ile `i <= 10` 11 tur döner. İlk ve son tur ayrıca denetlenmelidir.

Döngü değişkenini dışarıda kullanmak

`for (int i …) { }` sonrası `println(i)` → `error: cannot find symbol`. Gerekirse `i` döngüden önce tanımlanır.

break'ten sonra kod

`break; System.out.println(i);` → `error: unreachable statement`. `break`'ten sonraki satıra asla ulaşılmaz.

Toplam/çarpım başlangıcı

Toplam 0, çarpım **1** ile başlar. `faktoriyel = 0` ile sonuç hep 0 olur. Değişkeni döngünün **içinde** sıfırlamak da yanlıştır.

ÖZET

Örnek Problemler (1): Ne Yazar?

PROBLEM 1

```
int x = 7;
if (x > 5)
    if (x > 10)
        System.out.println("A");
    else
        System.out.println("B");
```

PROBLEM 2

```
int s = 0;
for (int i = 1; i <= 10; i += 3) {
    s += i;
}
System.out.println(s);
```

PROBLEM 3

```
int k = 2;
int puan = 0;
switch (k) {
    case 1:  puan += 10;
    case 2:  puan += 20;
    case 3:  puan += 30;  break;
    case 4:  puan += 40;
}
System.out.println(puan);
```

▶ **Cevaplar**

ÖZET

Örnek Problemler (2)

Problem 4

1'den 100'e kadar olan sayılardan **3'e veya 5'e** tam bölünenlerin toplamını bulan kodu yazın.

Problem 5

Bir tam sayının (ör. 2024) **kaç basamaklı** olduğunu `while` döngüsüyle bulun (`sayi /= 10` son basamağı atar).

▶ **Cevaplar**

ÖZET

Alıştırmalar (1)

Alıştırma 1

```
int n = 1;
while (n < 100) {
    n *= 3;
}
System.out.println(n);
```

Alıştırma 2

```
int k = 5;
do {
    k -= 2;
} while (k > 0);
System.out.println(k);
```

Alıştırma 3

```
int adet = 0;
for (int i = 1; i <= 4; i++) {
    for (int j = i; j <= 4; j++) {
        adet++;
    }
}
System.out.println(adet);
```

▶ **Cevaplar (önce kâğıt üzerinde çözün, sonra çalıştırıp doğrulayın)**

ÖZET

Alıştırmalar (2)

Alıştırma 4

```
int t = 0;
for (int i = 1; i <= 10; i++) {
    if (i % 3 == 0) {
        continue;
    }
    t += i;
}
System.out.println(t);
```

Alıştırma 5

```
int i = 10;
while (i >= 1) {
    System.out.print(i + " ");
    i -= 2;
}
System.out.println();
```

Alıştırma 6

```
int a = 17, b = 42;
int enBuyuk = (a > b) ? a : b;
System.out.println(enBuyuk);
```

▶ **Cevaplar**

Kaynaklar

- Oracle, *The Java Tutorials*: "Control Flow Statements" (if-then-else, switch, while, do-while, for, branching statements).
- J. Gosling ve diğ., *The Java Language Specification*, Java SE 21 Edition (Bölüm 14: Blocks, Statements, and Patterns).
- P. Deitel ve H. Deitel, *Java: How to Program*, 11. baskı, Pearson (Bölüm 4–5: Control Statements).
- A. B. Downey ve C. Mayfield, *Think Java*, 2. baskı, O'Reilly (Bölüm 5: Conditionals and Logic, Bölüm 6: Loops).
- K. Sierra ve B. Bates, *Head First Java*, 3. baskı, O'Reilly.
- C. S. Horstmann, *Core Java, Volume I: Fundamentals*, 12. baskı (3.8: Control Flow).

Son

Bölüm 3.1: Kontrol Yapıları

Sonraki bölüm: Bölüm 3.2: Fonksiyonlar · Terim sözlüğü