

Bölüm 1: Akış Diyagramları

Java ile Nesne Yönelimli Programlama

Algoritma ve akış diyagramı

Temel semboller

Çizim kuralları

Sıralı akış

Koşullu akış

Tekrarlı akış

Döngü sembolü

Diyagramdan Java'ya

Sercan KÜLCÜ · Giresun Üniversitesi · Elektrik-Elektronik Mühendisliği

GİRİŞ

Öğrenme Çıktıları

Kavram

Algoritma ile akış diyagramı arasındaki ilişkiyi açıklamak; akış diyagramının neden ve nerede kullanıldığını söylemek.

Semboller

Başla/Bitir, Giriş/Çıkış, İşlem, Karar, Döngü, Yazdır ve bağlantı noktası sembollerini tanımak ve doğru yerde kullanmak.

Kontrol yapıları

Sıralı, koşullu ve tekrarlı akışı ayırt etmek; her biri için akış diyagramı çizmek.

İzleme

Bir diyagramı verilen girdiyle adım adım izleyip çıktığı bulmak; sınır hatalarını ve sonsuz döngüleri fark etmek.

Diyagramdan Java'ya

Her sembolün Java karşılığını bilmek:

`nextInt()`, atama, `if/else`, `while`,
`for`, `println`.

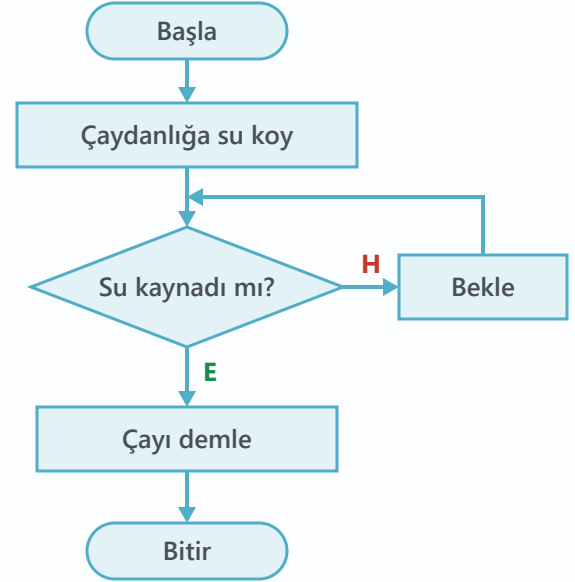
Tasarım

Basit bir problemi (toplam, tek/çift, sayma) önce diyagramla planlayıp sonra programa dönüştürmek.

Akış Diyagramları (Flowcharts)

- **Algoritma (Algorithm):** bir problemi çözmek veya belirli bir görevi gerçekleştirmek için adım adım talimatlar içeren plan.
- **Akış diyagramı (Flowchart):** algoritmanın adımlarını, adımlar arasındaki bağlantıları ve kontrol mekanizmalarını **görsel** olarak gösterir.
- Her adım bir **sembolle**, adımların sırası **oklarla** (akış çizgileri) çizilir.

Benzetme: Yemek tarifi bir algoritmadır; aynı tarifi kutular ve oklarla çizilmiş hâli akış diyagramıdır. Program yazmadan önce çözümü kâğıt üzerinde planlamanın en kolay yoludur.



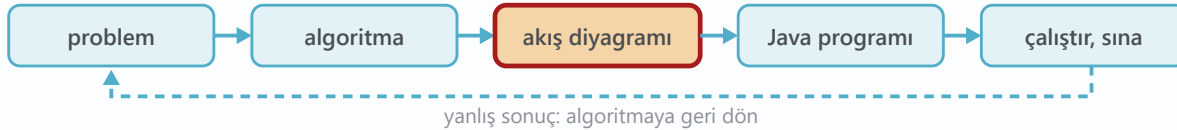
Faydaları ve Sınırlılıkları

Faydaları

- + Karmaşık süreçleri **basitleştirir**: çözümün tamamı tek bakışta görülür.
- + **Hataları** ve iyileştirme noktalarını kod yazmadan önce belirlemeye yardımcı olur.
- + Süreçleri **belgelemede** ve raporlamada kullanılır.
- + Programlama dilinden **bağımsızdır**; ekip içinde ortak bir dil sağlar.

Sınırlılıkları

- Büyük programlarda diyagram çok büyür ve okunması zorlaşır.
- Algoritma değiştiğinde diyagramın yeniden çizilmesi gerekir.
- Ayrıntı arttıkça sembol sayısı artar; bu durumda **sözde kod** (pseudocode) daha pratiktir.



Tipik çalışma sırası: **problem** → **algoritma** (adımlar) → **akış diyagramı** → **Java programı** → çalıştırıp sına.

Temel Öğeler (Semboller)

Başla

Başlangıç / Bitiş

Oval. Algoritmanın başladığı ve bittiği yer.

Oku x

Giriş / Çıkış

Paralelkenar. Klavyeden veri okuma (girdi).

$x = x + 1$

İşlem

Dikdörtgen. Hesaplama, işlem, atama.

$x > 0 ?$

Karar

Eşkenar dörtgen. Evet/Hayır sorusu, iki çıkış.

$i = 1; i \leq 5; i++$

Döngü

Altıgen. Belirli sayıda tekrar.

Yaz x

Yazdır

Dalgalı alt kenar. Sonucu ekrana / kâğıda yazma.

A

Bağlantı noktası

Küçük çember. Çizgilerin birleştiği nokta.



Akış çizgisi

Ok. Adımların sırası; okun yönünde ilerlenir.

Semboller ISO 5807 standardına dayanır. Bazı kaynaklar giriş ve çıkışın ikisi için de paralelkenar kullanır; ekrana yazma için "Yazdır" sembolü de yaygındır.

Her Sembolün Java Karşılığı

Sembol	Diyagramdaki yazı	Java karşılığı
 Başla / Bitir	Başla ... Bitir	<pre>public static void main(String[] args) { ... }</pre>
 Giriş	Oku sayı	<pre>int sayi = giris.nextInt();</pre>
 İşlem	sonuç = sayı1 + sayı2	<pre>int sonuc = sayi1 + sayi2;</pre>
 Karar	sonuç = 0 ?	<pre>if (sonuc == 0) { ... } else { ... }</pre>
 Karar + geri ok	sayaç ≤ 10 ?	<pre>while (sayac <= 10) { ... }</pre>
 Döngü	i = 1; i ≤ 10; i++	<pre>for (int i = 1; i <= 10; i++) { ... }</pre>
 Yazdır	Yaz sonuç	<pre>System.out.println(sonuc);</pre>

Dikkat: diyagramdaki işlem kutusunda = **atama**, karar kutusunda = **karşılaştırmadır**. Java'da atama = , karşılaştırma == ile yazılır. Diyagramdaki "mod" Java'da % operatörüdür.

Akış Diyagramı Nasıl Oluşturulur?

Adımlar

1. İşlemler adım adım belirlenir: girdiler, yapılacak hesaplar, çıktılar.
2. Adımlar mantıklı bir sırayla yerleştirilir.
3. Tüm işlemler **akış çizgileri** ile birbirine bağlanır.
4. Koşullu ifadelerle **karar noktaları** belirlenir.

Çizim kuralları

- Tek bir **Başla** ve en az bir **Bitir** bulunur.
- Akış yukarıdan aşağıya, soldan sağa okunur.
- Her sembolün bir çıkışı vardır; yalnızca **Karar** sembolünün iki çıkışı (E/H) olur.
- Karar çıkışları mutlaka **Evet/Hayır** diye etiketlenir.
- Çizgiler kesişmemeli; gerekirse bağlantı noktası kullanılır.
- Kutulara kısa ve açık ifadeler yazılır.

Adım Adım Çizim: Dikdörtgenin Alanı

Başla

Bitir

Problem: **dikdörtgenin alanını** bulmak. Önce iskelet: bir **Başla** ve bir **Bitir**.

Adım 1 / 6

Adım Adım Çizim: Dikdörtgenin Alanı

Başla

Oku a, b

Bitir

Girdiler belirlenir: kenar uzunlukları a ve b klavyeden okunur (paralelkenar).

Adım Adım Çizim: Dikdörtgenin Alanı

Başla

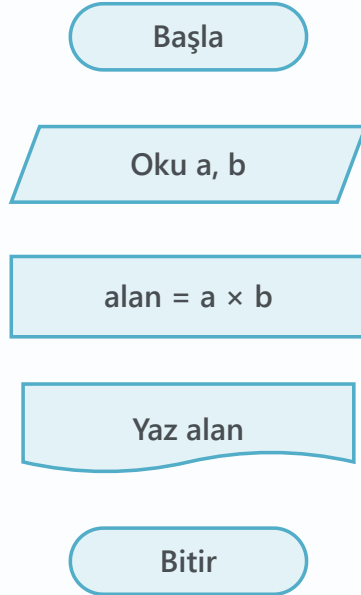
Oku a, b

$alan = a \times b$

Bitir

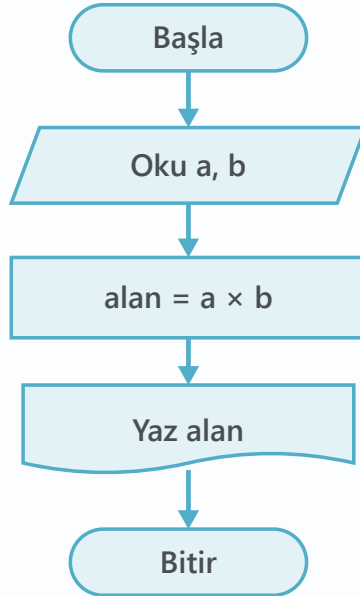
İşlem eklenir: $alan = a \times b$ hesaplanır ve alan değişkenine atanır (dikdörtgen).

Adım Adım Çizim: Dikdörtgenin Alanı



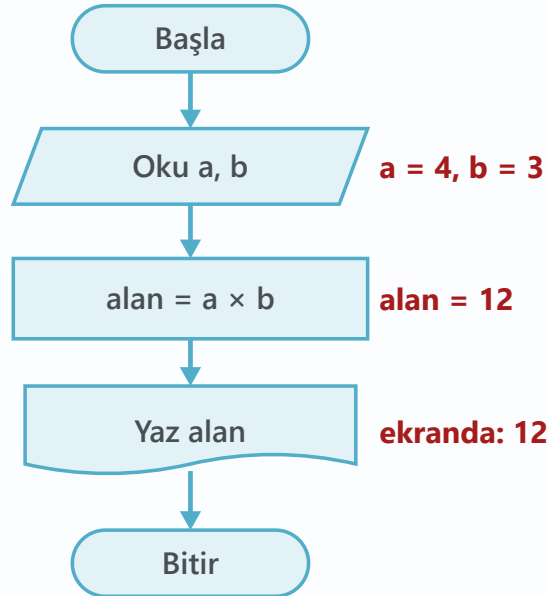
Çıktı eklenir: hesaplanan alan ekrana yazılır (Yazdır).

Adım Adım Çizim: Dikdörtgenin Alanı



Semboller mantıklı sırayla **akış çizgileri** ile bağlanır. Bu problemde karar noktası yok: akış **sıralıdır**.

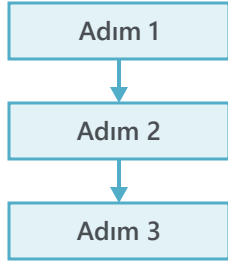
Adım Adım Çizim: Dikdörtgenin Alanı



Sınama: $a = 4$, $b = 3$ girilirse $\text{alan} = 12$ hesaplanır ve ekranda **12** görülür. Diyagram doğru çalışıyor.

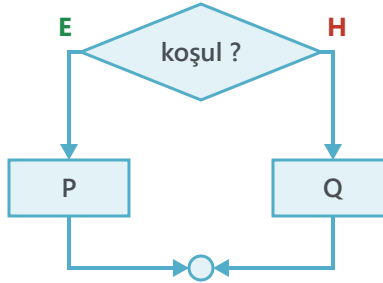
Kontrol Mekanizmaları

Sıralı akış



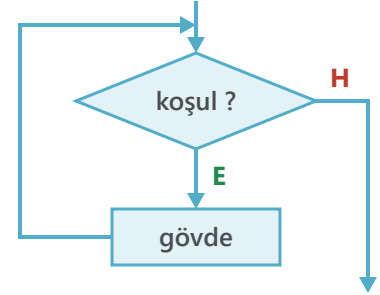
İşlem adımları sırayla, birer kez gerçekleşir.

Koşullu akış



Koşula bağlı olarak farklı işlem adımlarına yönelinir.

Tekrarlı akış



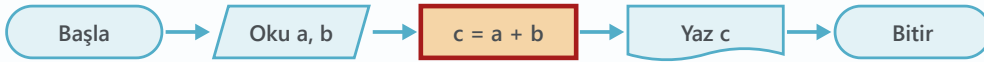
Koşul sağlandığı sürece bazı adımlar tekrarlanır.

Yapısal programlama teoremi (Böhm ve Jacopini, 1966): her algoritma yalnızca bu üç yapı (sıra, seçim, tekrar) iç içe kullanılarak yazılabilir. Java'da karşılıkları: art arda deyimler, `if/else`, `while` / `for`.

SIRALI AKIŞ

Sıralı Akış (Sequential Flow)

Yapılacak işlerin (komutların) art arda, mantıksal bir sıra ile dizilmesidir. Akış yukarıdan aşağıya ilerler.



Basit

Sade ve anlaşılır bir akışı temsil eder; dallanma ve tekrar yoktur.

Kolay bakım

Hata ayıklamak ve kodu anlamak kolaydır: adımlar yazıldığı sırayla izlenir.

Sıra önemlidir

Bir adım öncekinin sonucunu kullanıyorsa yer değiştiremez. Birbirinden bağımsız adımların (ör. iki ayrı okuma) sırası ise değiştirilebilir.

Geri dönüş yok

Her adım tam olarak **bir kez** çalışır; önceki bir adıma geri dönülmez, hiçbir adım atlanmaz.

Yanlış sıra örneği: `sonuç = sayı1 + sayı2` adımı sayılar okunmadan önce yazılırsa, toplama henüz bilinmeyen değerlerle yapılmaya çalışılır. Java'da değişkenler tanımlanmış ama değer almamışsa derleyici bunu yakalar: `variable sayı1 might not have been initialized`.

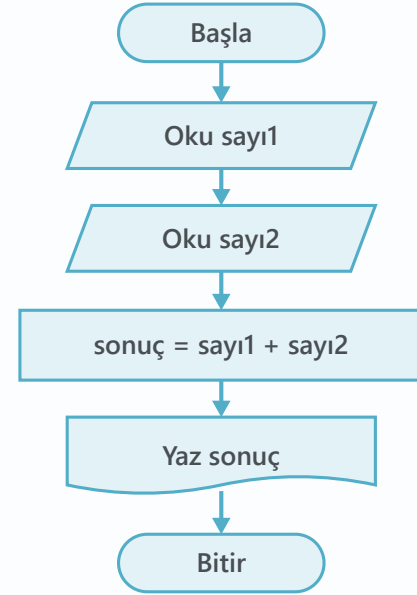
Örnek: İki Sayının Toplamı

Problem: klavyeden girilen iki sayının toplamını ekrana yazmak.

Algoritma

1. Başla
2. İlk sayıyı oku (sayı1)
3. İkinci sayıyı oku (sayı2)
4. İki sayıyı topla, sonuç'a ata
5. Sonuç'u ekrana yaz
6. Bitir

Her algoritma adımı diyagramda bir sembole karşılık gelir.



Diyagramdan Java'ya: Toplam Programı

```
1 import java.util.Scanner;
2
3 public class IkiSayiToplami {
4     ▶ public static void main(String[] args) {
5     ▶     Scanner giris = new Scanner(System.in);
6         System.out.print("İlk sayı: ");
7         int sayi1 = giris.nextInt();
8         System.out.print("İkinci sayı: ");
9         int sayi2 = giris.nextInt();
10        int sonuc = sayi1 + sayi2;
11        System.out.println("Sonuç: " + sonuc);
12        giris.close();
13    }
14 }
```

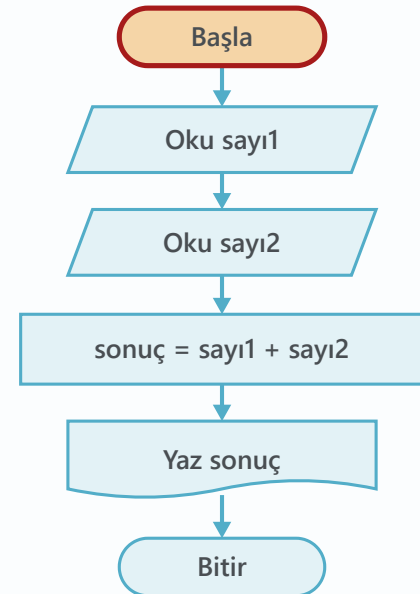
DEĞİŞKENLER

sayi1 = -

sayi2 = -

sonuc = -

ÇIKTI



Program `main` 'den başlar (**Başla**). Klavyeden okumak için bir `Scanner` oluşturulur.

Diyagramdan Java'ya: Toplam Programı

```
1 import java.util.Scanner;
2
3 public class IkiSayiToplami {
4     public static void main(String[] args) {
5         Scanner giris = new Scanner(System.in);
6         ▶ System.out.print("İlk sayı: ");
7         ▶ int sayi1 = giris.nextInt();
8         System.out.print("İkinci sayı: ");
9         int sayi2 = giris.nextInt();
10        int sonuc = sayi1 + sayi2;
11        System.out.println("Sonuç: " + sonuc);
12        giris.close();
13    }
14 }
```

DEĞİŞKENLER

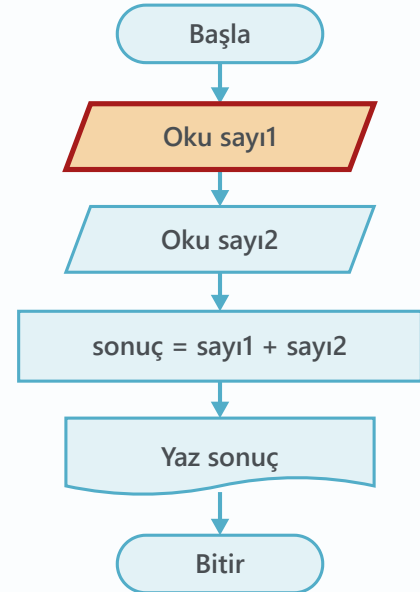
sayi1 = 7

sayi2 = -

sonuc = -

ÇIKTI

İlk sayı: 7



Mesaj yazılır, kullanıcı **7** girer:

`nextInt()` değeri `sayi1` 'e atar.

Diyagramdan Java'ya: Toplam Programı

```
1 import java.util.Scanner;
2
3 public class IkiSayiToplami {
4     public static void main(String[] args) {
5         Scanner giris = new Scanner(System.in);
6         System.out.print("İlk sayı: ");
7         int sayi1 = giris.nextInt();
8         System.out.print("İkinci sayı: ");
9         int sayi2 = giris.nextInt();
10        int sonuc = sayi1 + sayi2;
11        System.out.println("Sonuç: " + sonuc);
12        giris.close();
13    }
14 }
```

DEĞİŞKENLER

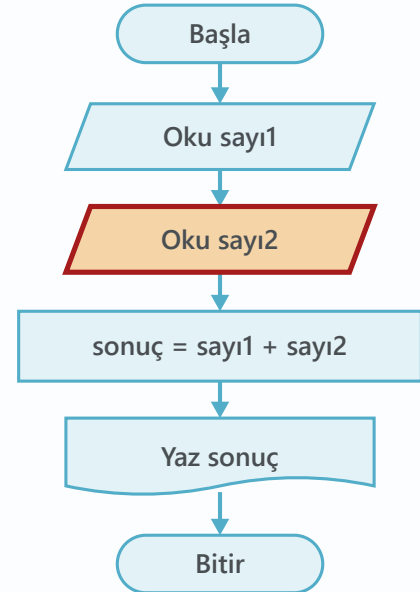
sayi1 = 7

sayi2 = 5

sonuc = -

ÇIKTI

```
İlk sayı: 7
İkinci sayı: 5
```



Kullanıcı 5 girer: **sayi2 = 5** .

Diyagramdan Java'ya: Toplam Programı

```
1 import java.util.Scanner;
2
3 public class İkiSayiToplami {
4     public static void main(String[] args) {
5         Scanner giris = new Scanner(System.in);
6         System.out.print("İlk sayı: ");
7         int sayi1 = giris.nextInt();
8         System.out.print("İkinci sayı: ");
9         int sayi2 = giris.nextInt();
10        ► int sonuc = sayi1 + sayi2;
11        System.out.println("Sonuç: " + sonuc);
12        giris.close();
13    }
14 }
```

DEĞİŞKENLER

sayi1 = 7

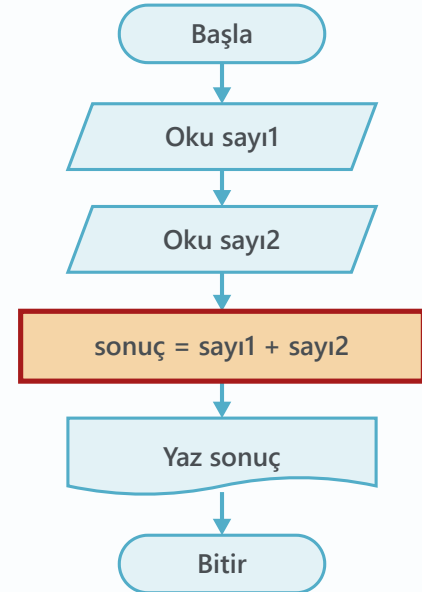
sayi2 = 5

sonuc = 12

ÇIKTI

İlk sayı: 7

İkinci sayı: 5



İşlem kutusu: **sonuc = 7 + 5 = 12**

Adım 4 / 6

Diyagramdan Java'ya: Toplam Programı

```
1 import java.util.Scanner;
2
3 public class IkiSayiToplami {
4     public static void main(String[] args) {
5         Scanner giris = new Scanner(System.in);
6         System.out.print("İlk sayı: ");
7         int sayi1 = giris.nextInt();
8         System.out.print("İkinci sayı: ");
9         int sayi2 = giris.nextInt();
10        int sonuc = sayi1 + sayi2;
11        System.out.println("Sonuç: " + sonuc);
12        giris.close();
13    }
14 }
```

DEĞİŞKENLER

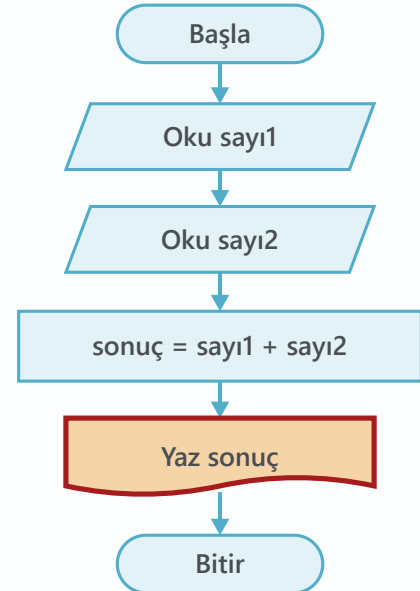
sayi1 = 7

sayi2 = 5

sonuc = 12

ÇIKTI

```
İlk sayı: 7
İkinci sayı: 5
Sonuç: 12
```



Yazdır kutusu: `println` sonucu ekrana yazar.

Diyagramdan Java'ya: Toplam Programı

```
1 import java.util.Scanner;
2
3 public class IkiSayiToplami {
4     public static void main(String[] args) {
5         Scanner giris = new Scanner(System.in);
6         System.out.print("İlk sayı: ");
7         int sayi1 = giris.nextInt();
8         System.out.print("İkinci sayı: ");
9         int sayi2 = giris.nextInt();
10        int sonuc = sayi1 + sayi2;
11        System.out.println("Sonuç: " + sonuc);
12        giris.close();
13    }
14 }
```

DEĞİŞKENLER

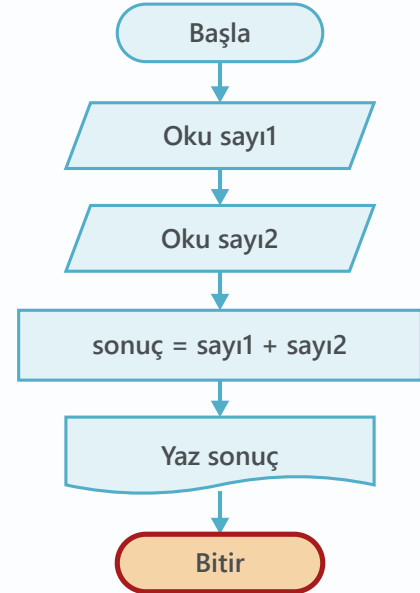
sayi1 = 7

sayi2 = 5

sonuc = 12

ÇIKTI

```
İlk sayı: 7
İkinci sayı: 5
Sonuç: 12
```



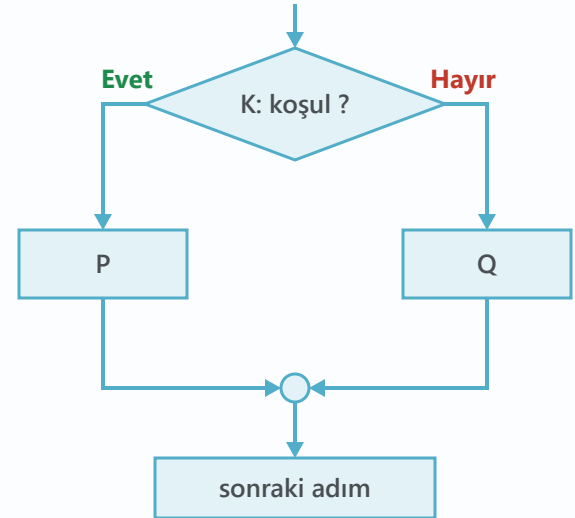
`Scanner` kapatılır, `main` biter (**Bitir**).
Her satır tam bir kez çalışır.

Koşullu Akış (Conditional Flow)

İki (veya daha çok) ihtimalden **birinin seçildiği** akış türüdür.

- **Karar noktası (K):** bir koşul sorusu bulunur; cevabı Evet ya da Hayır'dır.
- **Evet'e göre akış (P):** cevap evet ise P işlemi yönünde ilerlenir.
- **Hayır'a göre akış (Q):** cevap hayır ise Q işlemi yönünde ilerlenir.
- **Esneklik** (koşullara göre farklı davranış), **karar verme yetisi** (koşula bağlı işlem) ve **kod kontrolü** (durumu denetleyip ona göre işlem) sağlar.

```
if (K) {  
    P; // koşul doğru  
} else {  
    Q; // koşul yanlış  
}
```

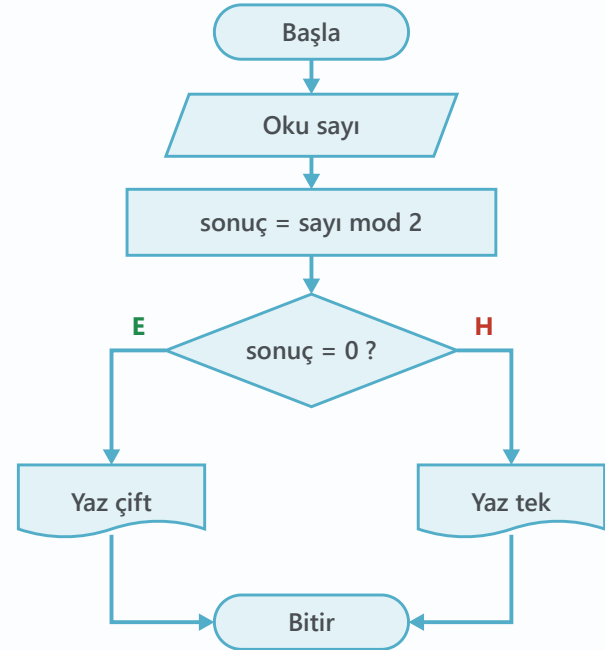


Örnek: Sayı Tek mi Çift mi?

Problem: verilen sayının tek mi çift mi olduğunu ekrana yazmak.

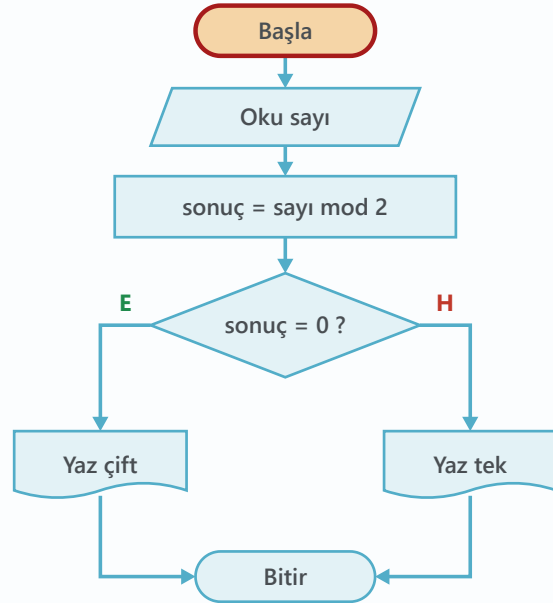
1. Başla
2. Sayıyı oku
3. Sayı mod 2'yi al, sonuç'a ata
4. Sonuç 0 ise 5'e git, değilse 6'ya git
5. Ekrana "çift" yaz, 7'ye git
6. Ekrana "tek" yaz
7. Bitir

mod: bölümden kalan. $7 \bmod 2 = 1$, $8 \bmod 2 = 0$. Çift sayının 2'ye bölümünden kalan 0'dır. Java'da `sayi % 2`.



Diyagramdan Java'ya: Tek mi Çift mi?

```
1 import java.util.Scanner;
2
3 public class TekMiCiftMi {
4     ▶ public static void main(String[] args) {
5     ▶     Scanner giris = new
        Scanner(System.in);
6         System.out.print("Sayı: ");
7         int sayi = giris.nextInt();
8         int sonuc = sayi % 2;
9         if (sonuc == 0) {
10             System.out.println("çift");
11         } else {
12             System.out.println("tek");
13         }
14         giris.close();
15     }
16 }
```



Başla: Scanner oluşturulur.

DEĞİŞKENLER

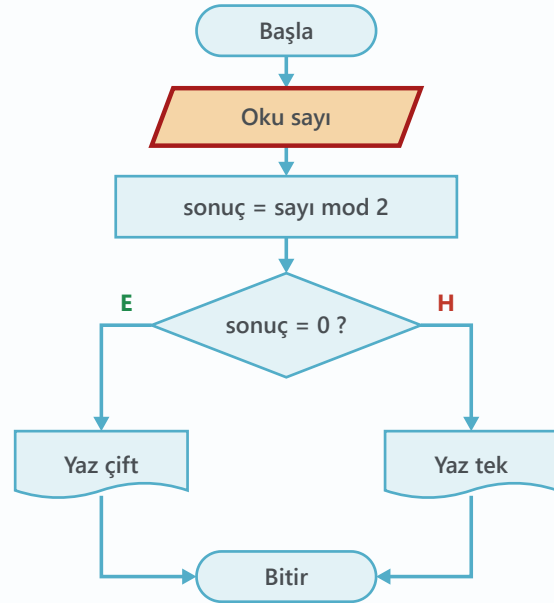
sayi = -

sonuc = -

ÇIKTI

Diyagramdan Java'ya: Tek mi Çift mi?

```
1 import java.util.Scanner;
2
3 public class TekMiCiftMi {
4     public static void main(String[] args) {
5         Scanner giris = new
Scanner(System.in);
6         System.out.print("Sayı: ");
7         int sayi = giris.nextInt();
8         int sonuc = sayi % 2;
9         if (sonuc == 0) {
10             System.out.println("çift");
11         } else {
12             System.out.println("tek");
13         }
14         giris.close();
15     }
16 }
```



Kullanıcı **7** girer: `sayi = 7` .

Adım 2 / 6

DEĞİŞKENLER

`sayi = 7`

`sonuc = -`

ÇIKTI

Sayı: 7

Diyagramdan Java'ya: Tek mi Çift mi?

```
1 import java.util.Scanner;
2
3 public class TekMiCiftMi {
4     public static void main(String[] args) {
5         Scanner giris = new
Scanner(System.in);
6         System.out.print("Sayı: ");
7         int sayi = giris.nextInt();
8         ▶ int sonuc = sayi % 2;
9         if (sonuc == 0) {
10             System.out.println("çift");
11         } else {
12             System.out.println("tek");
13         }
14         giris.close();
15     }
16 }
```

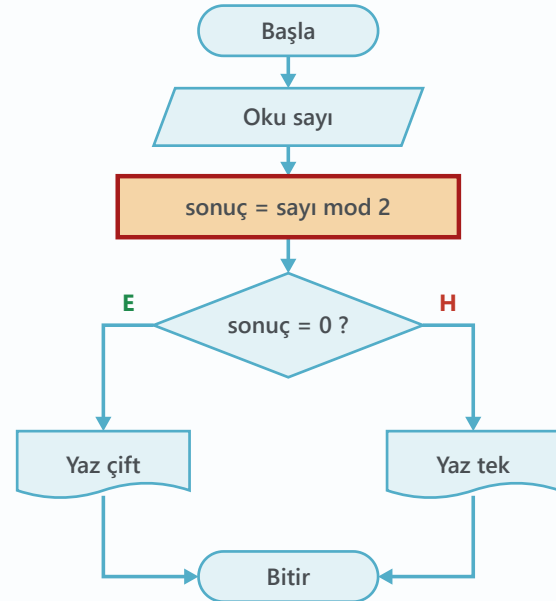
DEĞİŞKENLER

sayi = 7

sonuc = 1

ÇIKTI

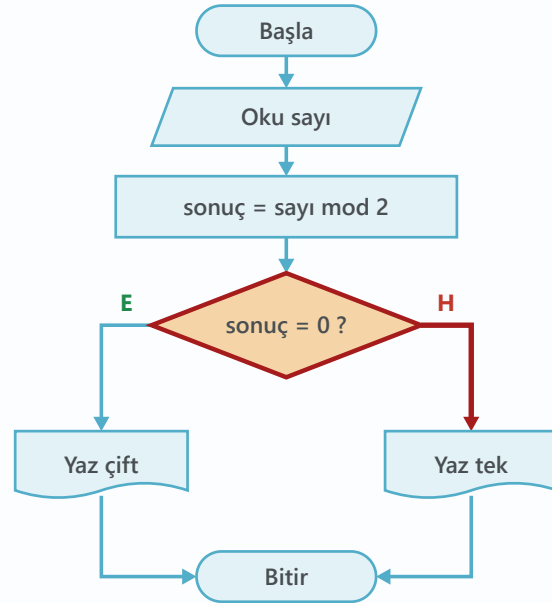
Sayı: 7


$$\text{sonuc} = 7 \% 2 = 1 \quad (7 = 3 \times 2 + 1).$$

Adım 3 / 6

Diyagramdan Java'ya: Tek mi Çift mi?

```
1 import java.util.Scanner;
2
3 public class TekMiCiftMi {
4     public static void main(String[] args) {
5         Scanner giris = new
Scanner(System.in);
6         System.out.print("Sayı: ");
7         int sayi = giris.nextInt();
8         int sonuc = sayi % 2;
9         if (sonuc == 0) {
10             System.out.println("çift");
11         } else {
12             System.out.println("tek");
13         }
14         giris.close();
15     }
16 }
```



Karar: `1 == 0` **yanlış** → Hayır (H) yolu, `else` bloğu.

DEĞİŞKENLER

sayi = 7

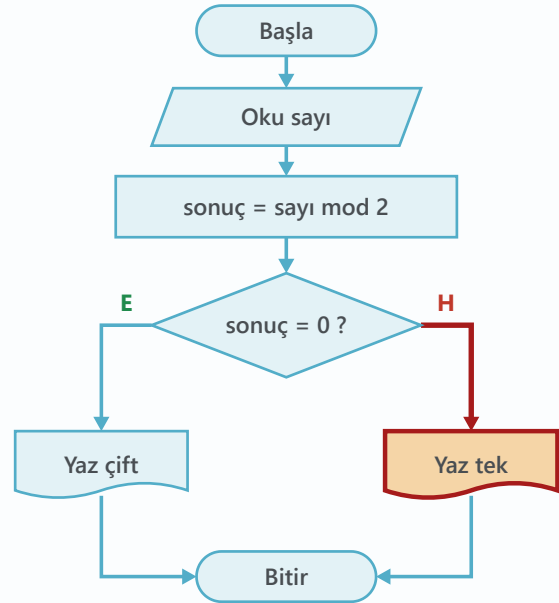
sonuc = 1

ÇIKTI

Sayı: 7

Diyagramdan Java'ya: Tek mi Çift mi?

```
1 import java.util.Scanner;
2
3 public class TekMiCiftMi {
4     public static void main(String[] args) {
5         Scanner giris = new
Scanner(System.in);
6         System.out.print("Sayı: ");
7         int sayi = giris.nextInt();
8         int sonuc = sayi % 2;
9         if (sonuc == 0) {
10             System.out.println("çift");
11         } else {
12             System.out.println("tek");
13         }
14         giris.close();
15     }
16 }
```



"tek" yazılır. `if` bloğundaki 10. satır **atlandı**.

DEĞİŞKENLER

sayi = 7

sonuc = 1

ÇIKTI

Sayı: 7
tek

Diyagramdan Java'ya: Tek mi Çift mi?

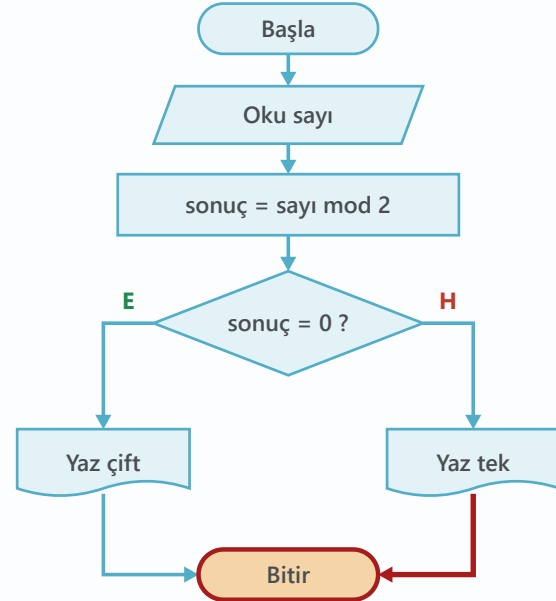
```
1 import java.util.Scanner;
2
3 public class TekMiCiftMi {
4     public static void main(String[] args) {
5         Scanner giris = new
Scanner(System.in);
6         System.out.print("Sayı: ");
7         int sayi = giris.nextInt();
8         int sonuc = sayi % 2;
9         if (sonuc == 0) {
10             System.out.println("çift");
11         } else {
12             System.out.println("tek");
13         }
14         giris.close();
15     }
16 }
```

DEĞİŞKENLER

sayi = 7

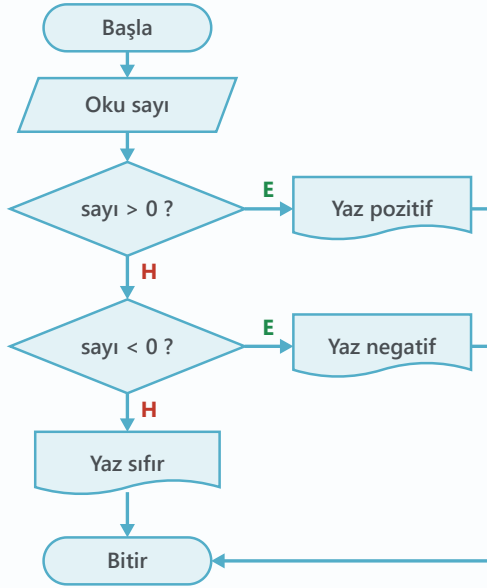
sonuc = 1

ÇIKTI

Sayı: 7
tek

İki yol **Bitir**'de birleşir. 8 girilseydi `sonuc = 0` olur, E yolundan "çift" yazılırdı.

İç İçe Karar: Pozitif, Negatif, Sıfır



Üç ihtimal varsa iki karar kutusu art arda bağlanır: ilk sorunun **Hayır** çıkışı ikinci soruya gider.

```

int sayi = giris.nextInt();
if (sayi > 0) {
    System.out.println("pozitif");
} else if (sayi < 0) {
    System.out.println("negatif");
} else {
    System.out.println("sıfır");
}
  
```

girdi	izlenen yol	çıktı
5	1. karar E	pozitif
-2	1. karar H, 2. karar E	negatif
0	1. karar H, 2. karar H	sıfır

Tek yönlü karar (yalnız Evet kolu): Java'da

`else` 'siz `if` .

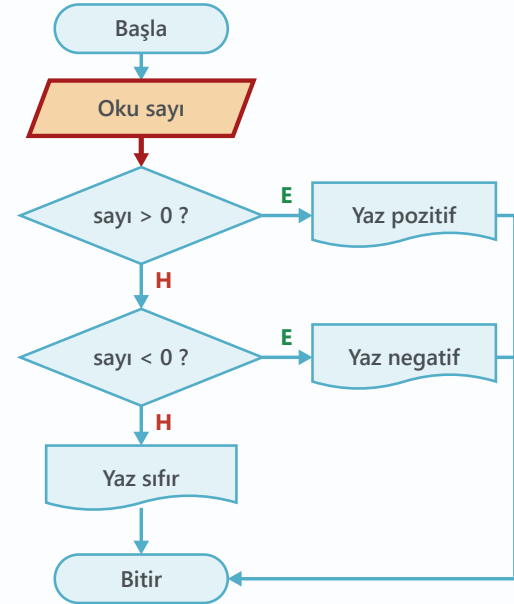
İç İçe Karar: Kod İzleme

```
1 Scanner giris = new Scanner(System.in);  
2 int sayi = giris.nextInt();  
3 if (sayi > 0) {  
4     System.out.println("pozitif");  
5 } else if (sayi < 0) {  
6     System.out.println("negatif");  
7 } else {  
8     System.out.println("sıfır");  
9 }
```

DEĞİŞKENLER

sayi = -4

ÇIKTI



girdi: -4

1. çalıştırma: kullanıcı -4 girer, **sayi** = -4.

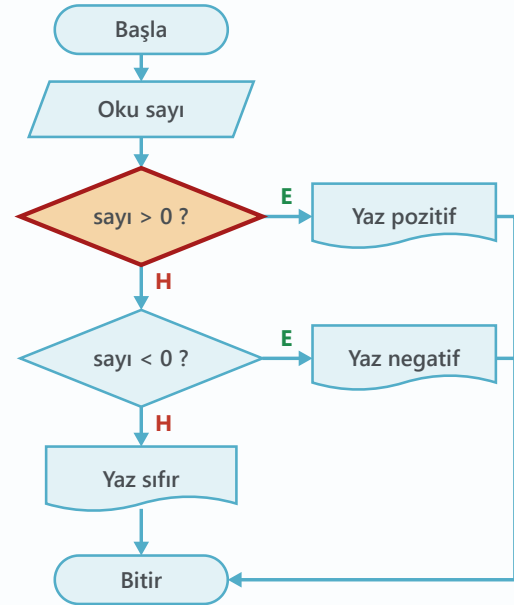
İç İçe Karar: Kod İzleme

```
1 Scanner giris = new Scanner(System.in);
2 int sayi = giris.nextInt();
3 ▶ if (sayi > 0) {
4     System.out.println("pozitif");
5 } else if (sayi < 0) {
6     System.out.println("negatif");
7 } else {
8     System.out.println("sıfır");
9 }
```

DEĞİŞKENLER

sayi = -4

ÇIKTI



-4 > 0 → yanlış

1. karar: -4 > 0 yanlış → Hayır çıkışı.

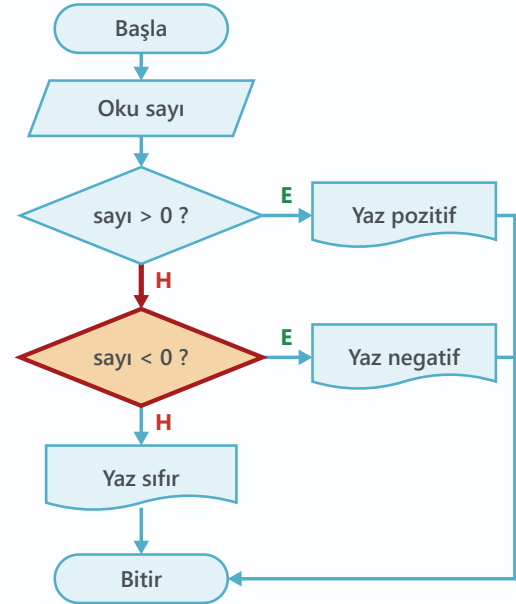
İç İçe Karar: Kod İzleme

```
1 Scanner giris = new Scanner(System.in);
2 int sayi = giris.nextInt();
3 if (sayi > 0) {
4     System.out.println("pozitif");
5 } else if (sayi < 0) {
6     System.out.println("negatif");
7 } else {
8     System.out.println("sıfır");
9 }
```

DEĞİŞKENLER

sayi = -4

ÇIKTI



-4 < 0 → doğru

Hayır çıkışı 2. karara gider: -4 < 0 **doğru** → Evet.

İç İçe Karar: Kod İzleme

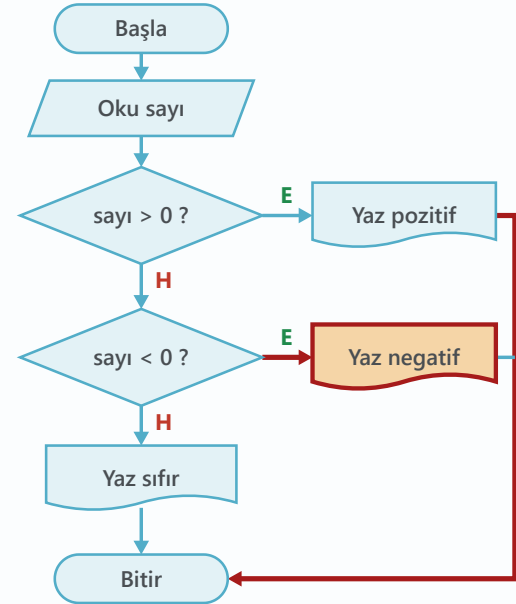
```
1 Scanner giris = new Scanner(System.in);
2 int sayi = giris.nextInt();
3 if (sayi > 0) {
4     System.out.println("pozitif");
5 } else if (sayi < 0) {
6     System.out.println("negatif");
7 } else {
8     System.out.println("sıfır");
9 }
```

DEĞİŞKENLER

sayi = -4

ÇIKTI

negatif



çıktı: negatif

negatif yazılır; else kolu atlanır, akış Bitir'e gider.

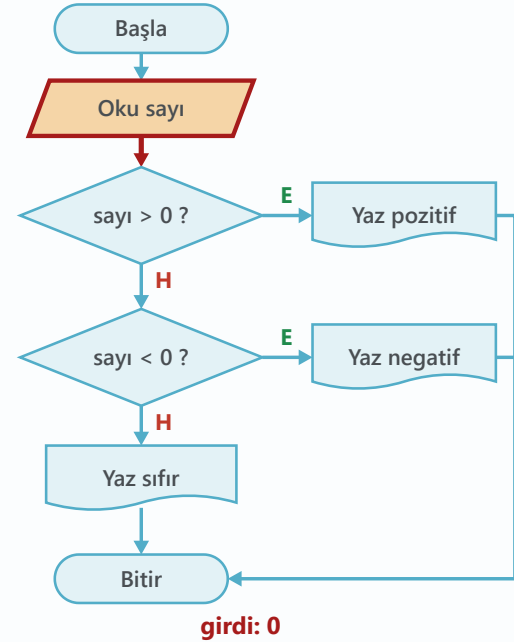
İç İçe Karar: Kod İzleme

```
1 Scanner giris = new Scanner(System.in);  
2 int sayi = giris.nextInt();  
3 if (sayi > 0) {  
4     System.out.println("pozitif");  
5 } else if (sayi < 0) {  
6     System.out.println("negatif");  
7 } else {  
8     System.out.println("sıfır");  
9 }
```

DEĞİŞKENLER

sayi = 0

ÇIKTI



2. çalıştırma: bu kez kullanıcı 0 girer (çıkı paneli yeni çalıştırmayı gösterir).

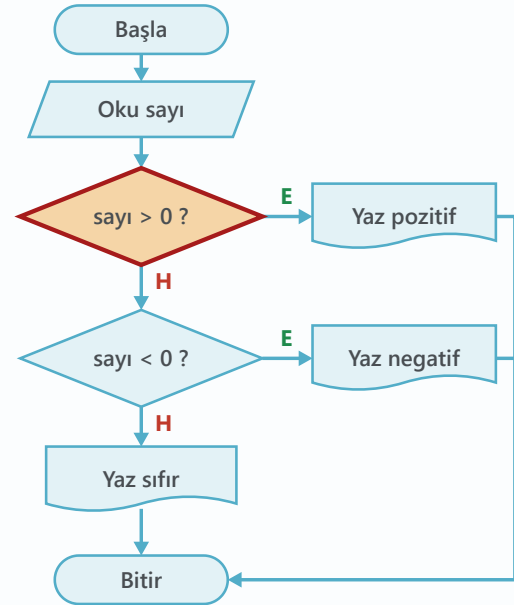
İç İçe Karar: Kod İzleme

```
1 Scanner giris = new Scanner(System.in);
2 int sayi = giris.nextInt();
3 ▶ if (sayi > 0) {
4     System.out.println("pozitif");
5 } else if (sayi < 0) {
6     System.out.println("negatif");
7 } else {
8     System.out.println("sıfır");
9 }
```

DEĞİŞKENLER

sayi = 0

ÇIKTI



0 > 0 → yanlış

0 > 0 yanlış: 0 pozitif değildir.

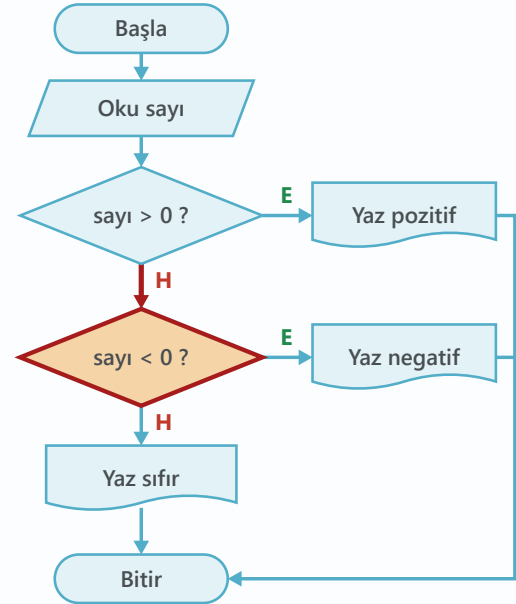
İç İçe Karar: Kod İzleme

```
1 Scanner giris = new Scanner(System.in);
2 int sayi = giris.nextInt();
3 if (sayi > 0) {
4     System.out.println("pozitif");
5 } else if (sayi < 0) {
6     System.out.println("negatif");
7 } else {
8     System.out.println("sıfır");
9 }
```

DEĞİŞKENLER

sayi = 0

ÇIKTI



0 < 0 → yanlış

0 < 0 da **yanlış** → ikinci kararın Hayır çıkışı.

İç İçe Karar: Kod İzleme

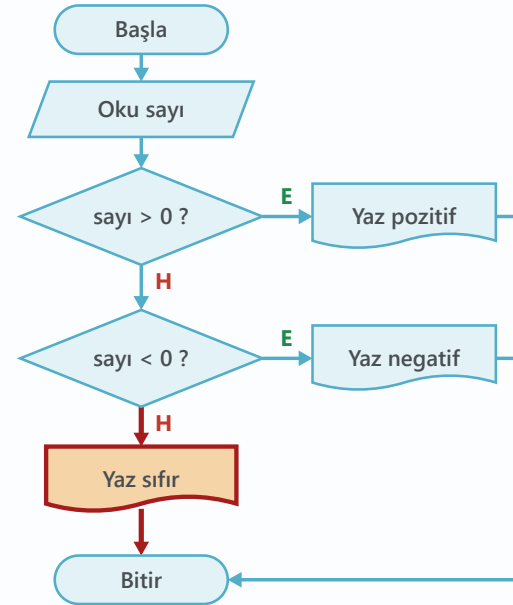
```
1 Scanner giris = new Scanner(System.in);
2 int sayi = giris.nextInt();
3 if (sayi > 0) {
4     System.out.println("pozitif");
5 } else if (sayi < 0) {
6     System.out.println("negatif");
7 } else {
8     System.out.println("sıfır");
9 }
```

DEĞİŞKENLER

sayi = 0

ÇIKTI

sıfır



çıktı: sıfır

İki koşul da yanlış: son `else` çalışır, `sıfır` yazılır. Üç koldan her seferinde yalnız **biri** çalışır.

Tekrarlı Akış (Döngü, Loop)

İşlemlerin **birden çok kez** tekrar ettiği akış şemalarıdır. Diyagramda geriye doğru giden bir ok, karar kutusuna döner.

Koşula bağlı döngü



Koşul sağlandığı sürece tekrar eder. Kaç kez döneceği önceden bilinmeyebilir (ör. kullanıcı 0 girene kadar oku). Java:

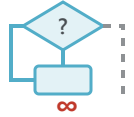
```
while .
```

Belirli sayıda tekrar



Tekrar sayısı baştan bellidir (ör. 1'den 10'a kadar). Java: `for` .

Sonsuz döngü



Koşul hep sağlandığı için işlemler durmadan tekrarlanır. Çoğunlukla bir **hatadır**; sunucu gibi sürekli çalışan programlarda bilinçli kullanılır.

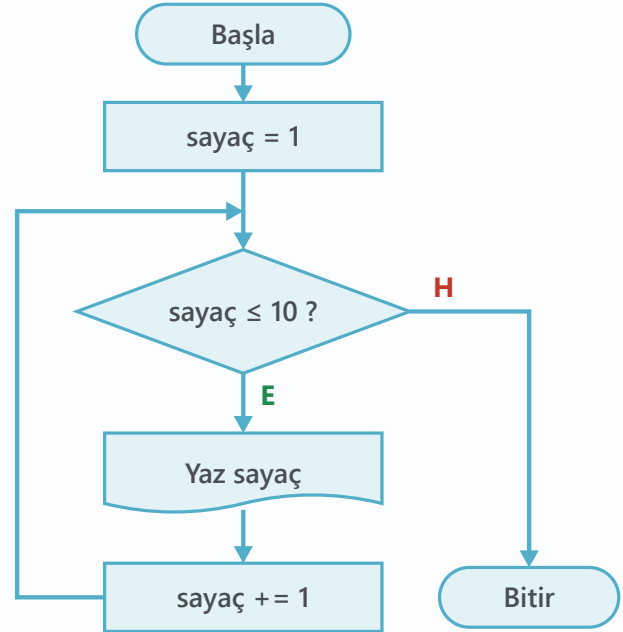
- + **Verimli:** aynı işlem, kod tekrar tekrar yazılmadan istenen sayıda yapılır.
- + **Dinamik:** tekrar sayısı girdiye göre değişebilir; program değişen koşullara uyum sağlar.
- + **Kod tekrarını önler:** 1000 sayıyı yazdırmak için 1000 satır değil, 3–4 satır yeterlidir.

Örnek: 1'den 10'a Kadar Sayıları Yazdırma

Algoritma

1. Başla
2. Sayaç = 1
3. $\text{Sayaç} \leq 10$ ise 4'e git, değilse 7'ye git
4. Sayaç'ı yaz
5. Sayaç'ı 1 artır
6. 3'e git
7. Bitir

6. adımdaki "3'e git", diyagramda karar kutusunun üstüne dönen **geri oktur**. Bu ok döngüyü oluşturur; karar kutusu döngüden çıkış kapısıdır.



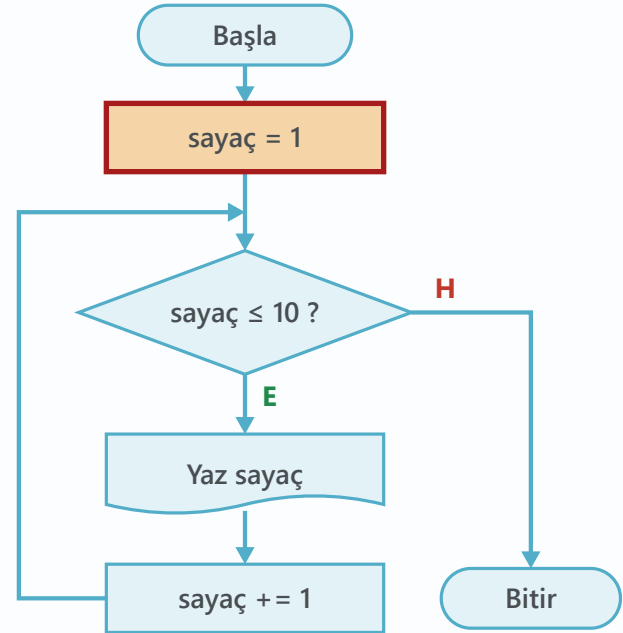
Diyagramdan Java'ya: while Döngüsü

```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 1

ÇIKTI



sayac = 1 : döngü değişkenine başlangıç değeri verilir.

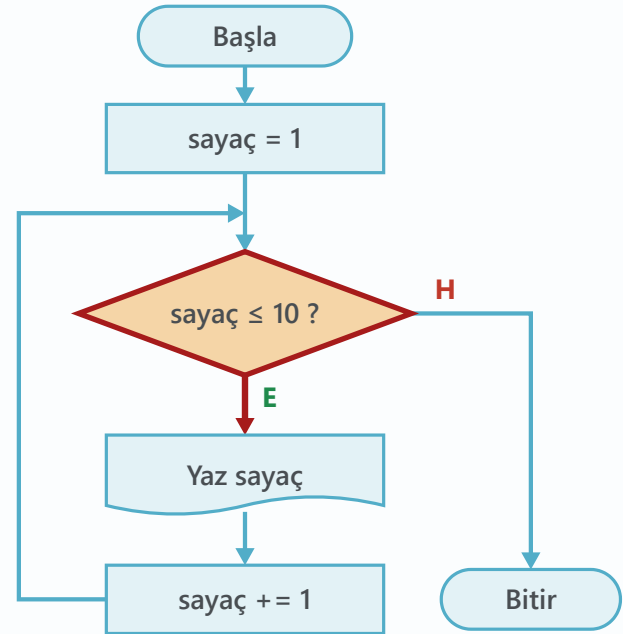
Diyagramdan Java'ya: while Döngüsü

```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 1

ÇIKTI



Koşul: `1 <= 10` **doğru** → Evet, döngü gövdesine girilir.

Diyagramdan Java'ya: while Döngüsü

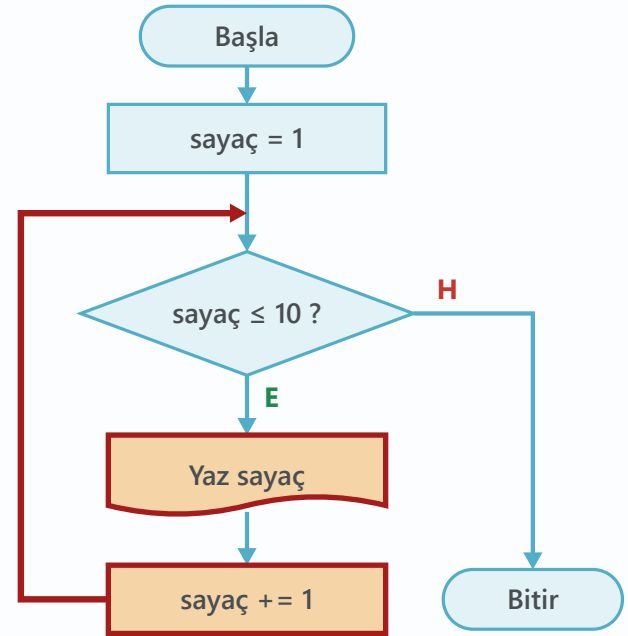
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 2

ÇIKTI

1



1 yazılır, **sayac** 2 olur; geri okla koşula dönülür.

Diyagramdan Java'ya: while Döngüsü

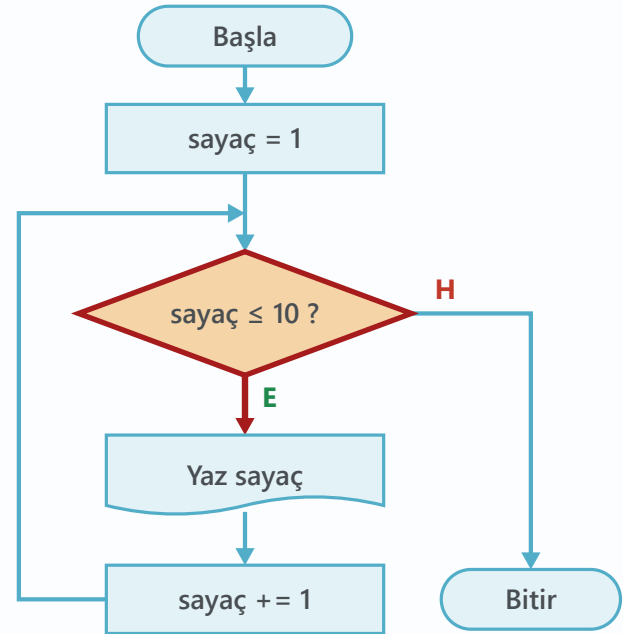
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 2

ÇIKTI

1



Koşul: `2 <= 10` **doğru** → Evet, döngü gövdesine girilir.

Diyagramdan Java'ya: while Döngüsü

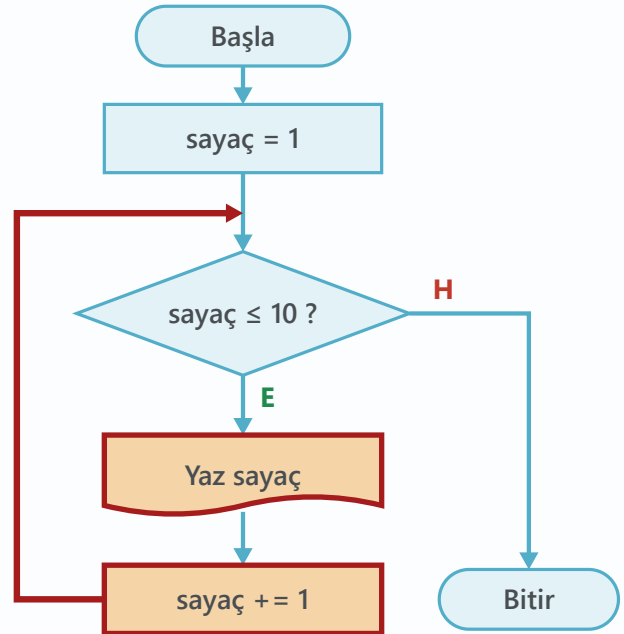
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 3

ÇIKTI

1 2



2 yazılır, sayac 3 olur; geri okla koşula dönülür.

Diyagramdan Java'ya: while Döngüsü

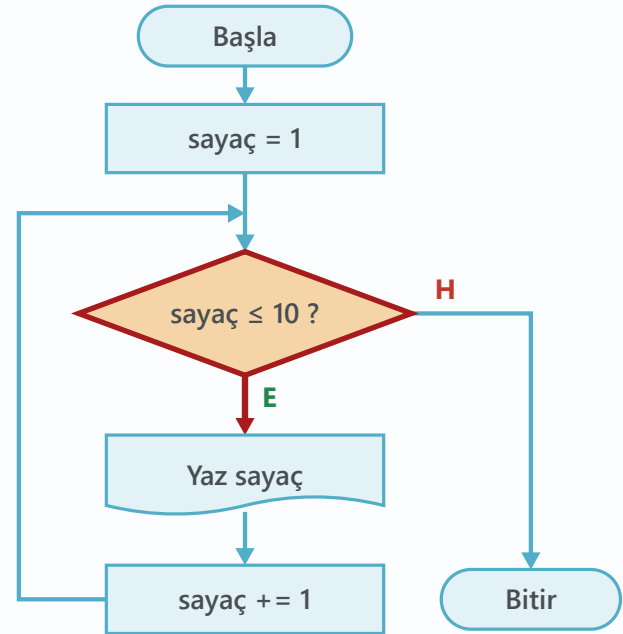
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 3

ÇIKTI

1 2



Koşul: `3 <= 10` **doğru** → Evet, döngü gövdesine girilir.

Diyagramdan Java'ya: while Döngüsü

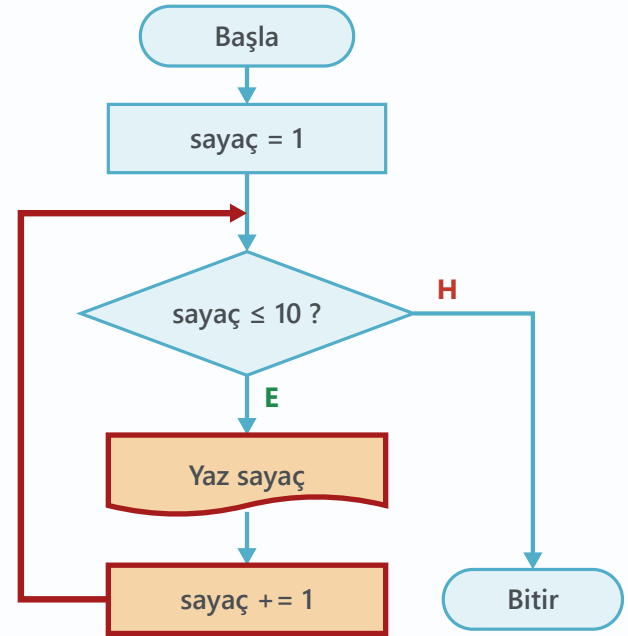
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 4

ÇIKTI

1 2 3



3 yazılır, sayac 4 olur; geri okla koşula dönülür.

Diyagramdan Java'ya: while Döngüsü

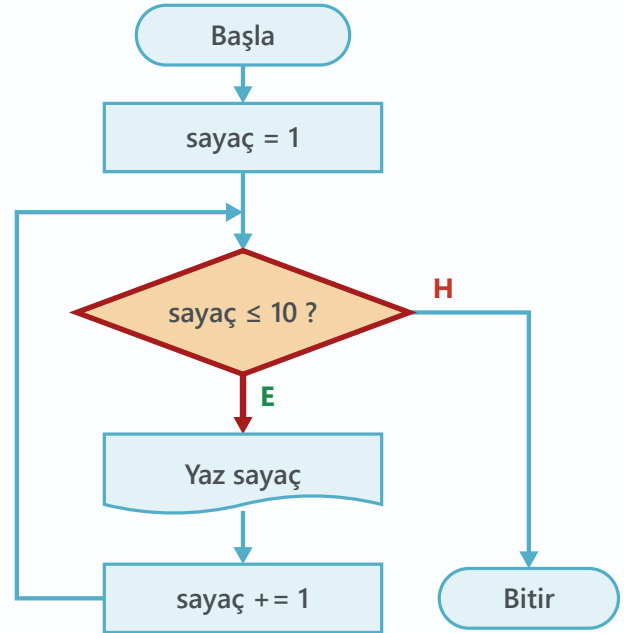
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 4

ÇIKTI

1 2 3



Koşul: `4 <= 10` **doğru** → Evet, döngü gövdesine girilir.

Diyagramdan Java'ya: while Döngüsü

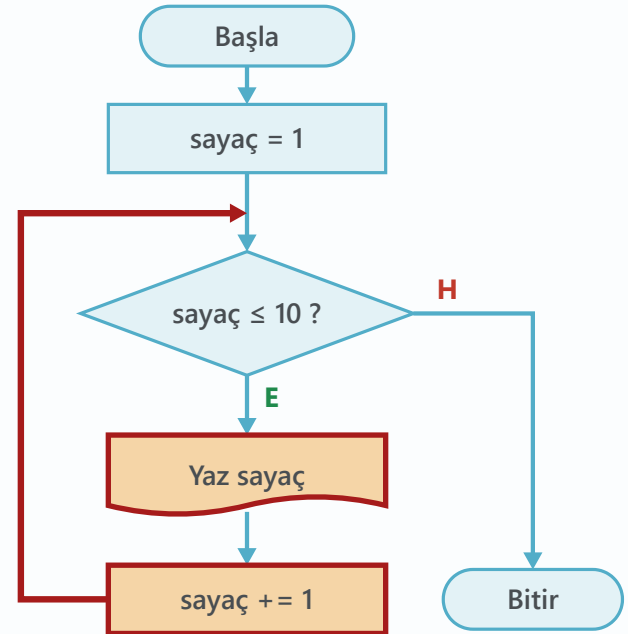
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 5

ÇIKTI

1 2 3 4



4 yazılır, sayac 5 olur; geri okla koşula dönülür.

Diyagramdan Java'ya: while Döngüsü

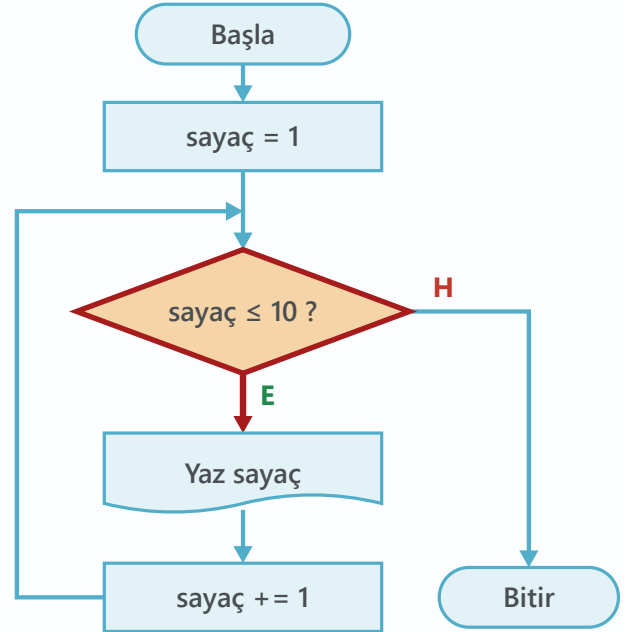
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 5

ÇIKTI

1 2 3 4



Koşul: `5 <= 10` **doğru** → Evet, döngü gövdesine girilir.

Diyagramdan Java'ya: while Döngüsü

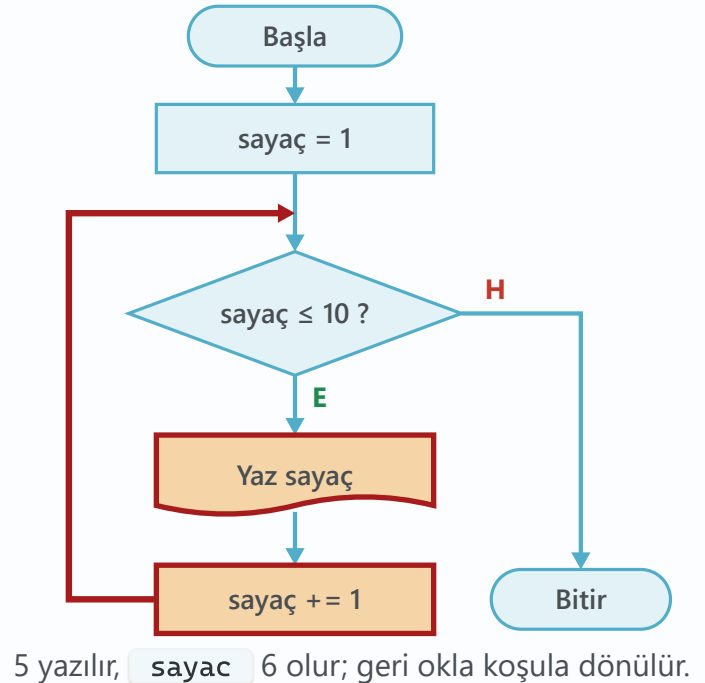
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 6

ÇIKTI

1 2 3 4 5



Diyagramdan Java'ya: while Döngüsü

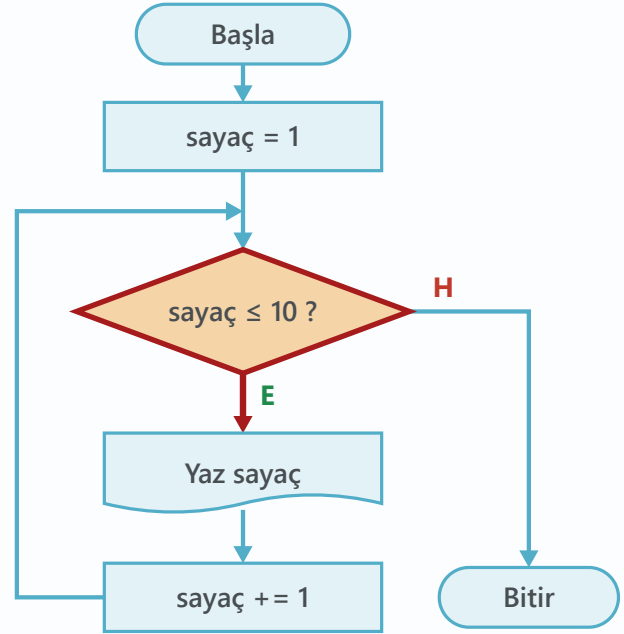
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 6

ÇIKTI

1 2 3 4 5



Koşul: `6 <= 10` **doğru** → Evet, döngü gövdesine girilir.

Diyagramdan Java'ya: while Döngüsü

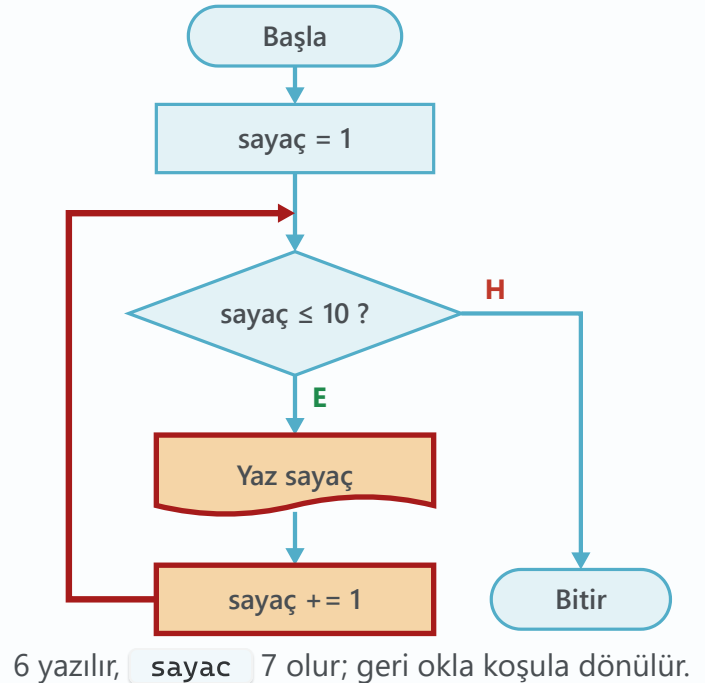
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 7

ÇIKTI

1 2 3 4 5 6



Diyagramdan Java'ya: while Döngüsü

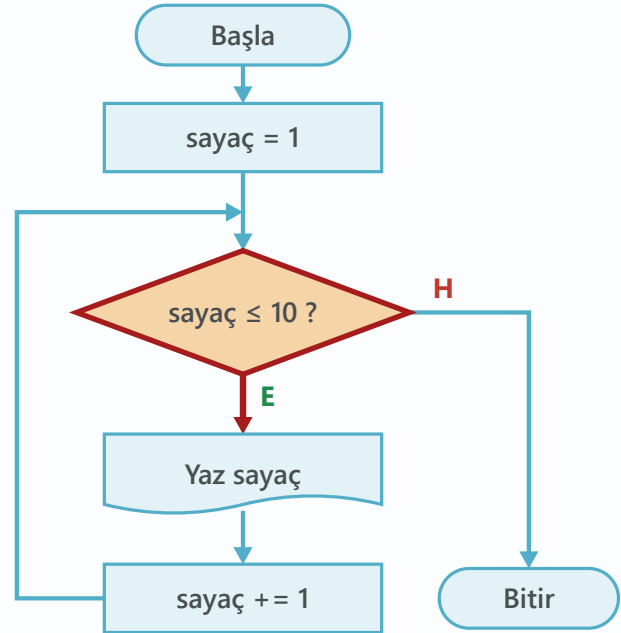
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 7

ÇIKTI

1 2 3 4 5 6



Koşul: **7 <= 10** **doğru** → Evet, döngü gövdesine girilir.

Diyagramdan Java'ya: while Döngüsü

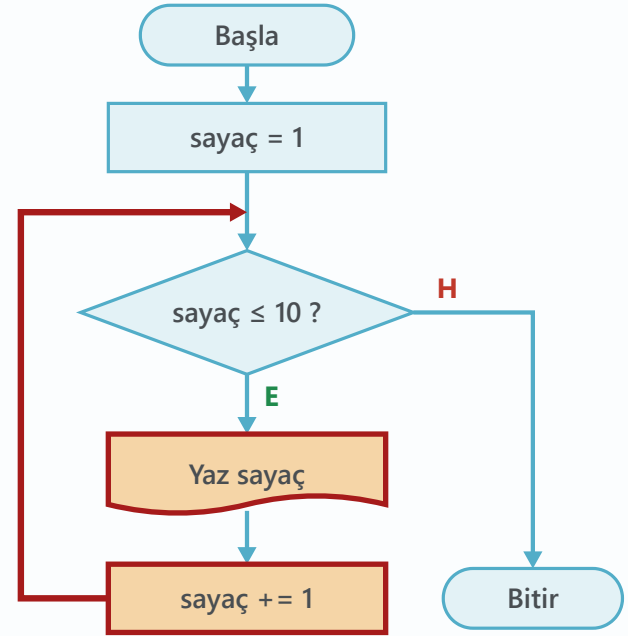
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 8

ÇIKTI

1 2 3 4 5 6 7



7 yazılır, sayac 8 olur; geri okla koşula dönülür.

Diyagramdan Java'ya: while Döngüsü

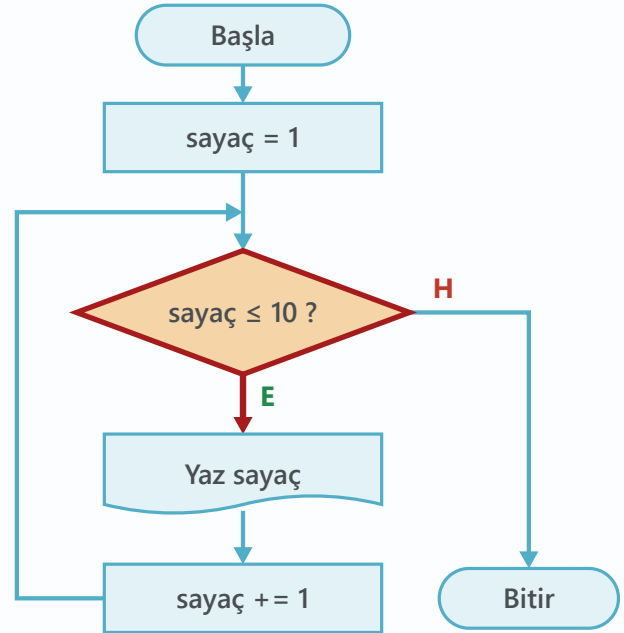
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 8

ÇIKTI

1 2 3 4 5 6 7



Koşul: **8 <= 10** **doğru** → Evet, döngü gövdesine girilir.

Diyagramdan Java'ya: while Döngüsü

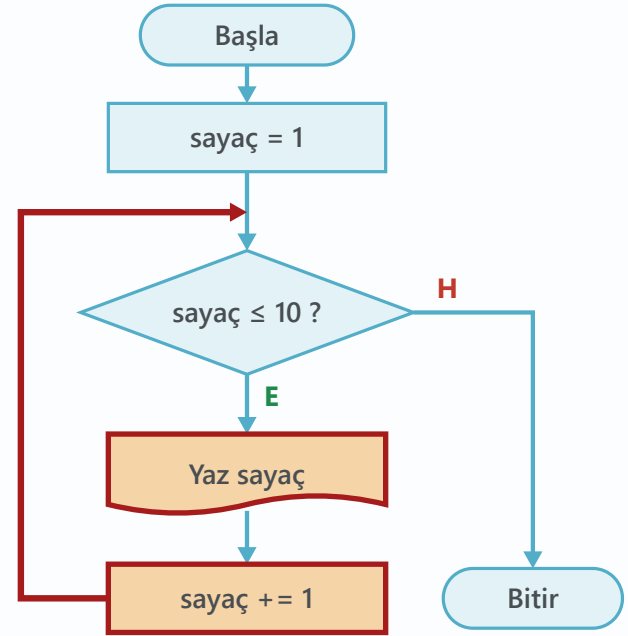
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 9

ÇIKTI

1 2 3 4 5 6 7 8



8 yazılır, sayac 9 olur; geri okla koşula dönülür.

Diyagramdan Java'ya: while Döngüsü

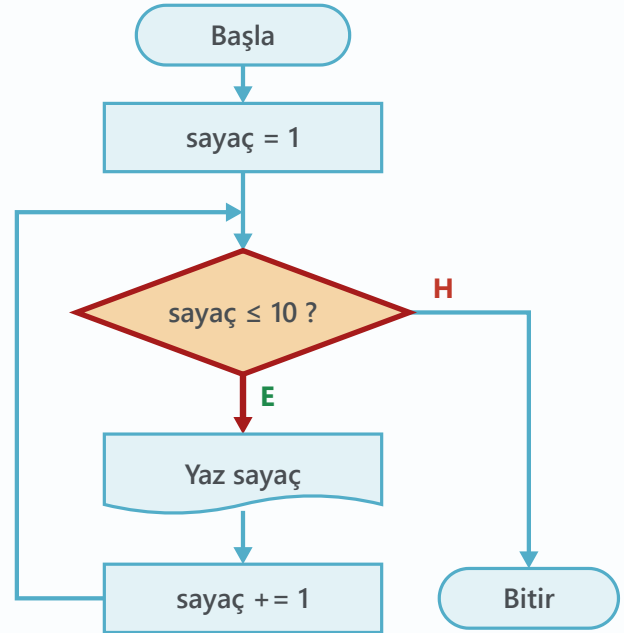
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 9

ÇIKTI

1 2 3 4 5 6 7 8



Koşul: **9 <= 10** **doğru** → Evet, döngü gövdesine girilir.

Diyagramdan Java'ya: while Döngüsü

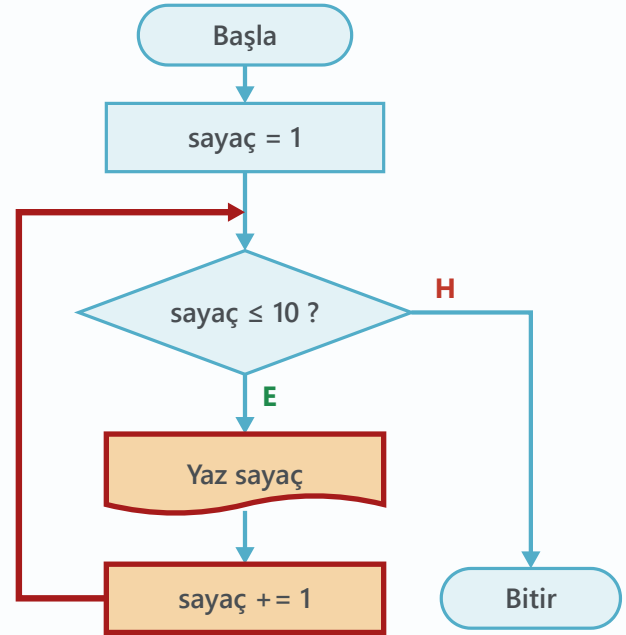
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 10

ÇIKTI

1 2 3 4 5 6 7 8 9



9 yazılır, sayac 10 olur; geri okla koşula dönülür.

Diyagramdan Java'ya: while Döngüsü

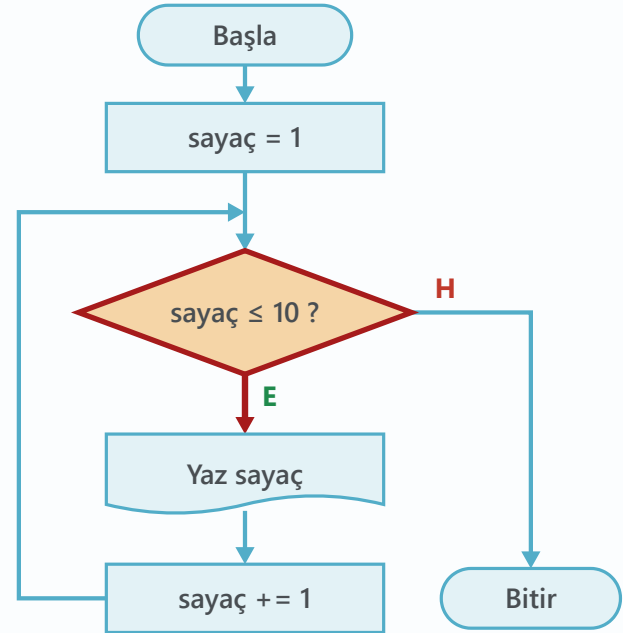
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 10

ÇIKTI

1 2 3 4 5 6 7 8 9



Koşul: `10 <= 10` **doğru** → Evet, döngü gövdesine girilir.

Diyagramdan Java'ya: while Döngüsü

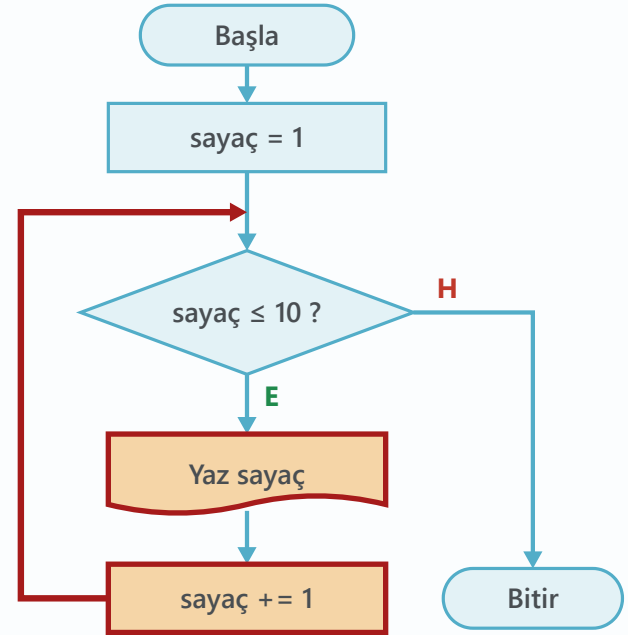
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 11

ÇIKTI

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



10 yazılır, `sayac` 11 olur; geri okla koşula dönülür.

Diyagramdan Java'ya: while Döngüsü

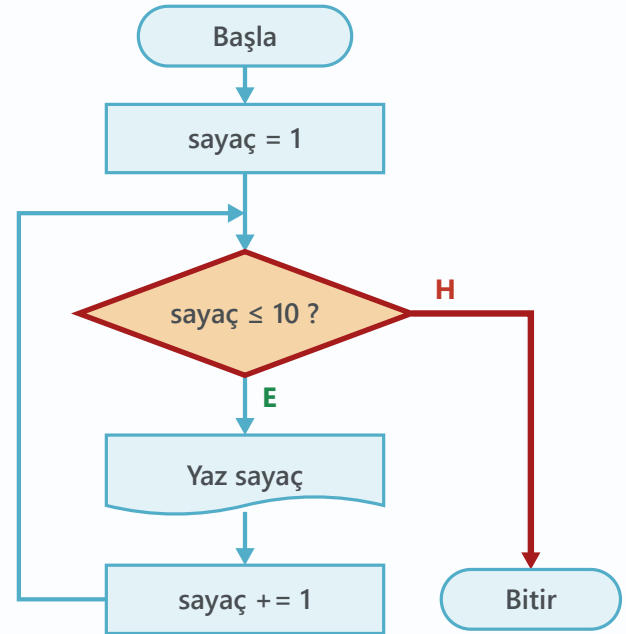
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 11

ÇIKTI

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



Koşul: `11 <= 10` **yanlış** → Hayır, döngüden çıkılır.

Diyagramdan Java'ya: while Döngüsü

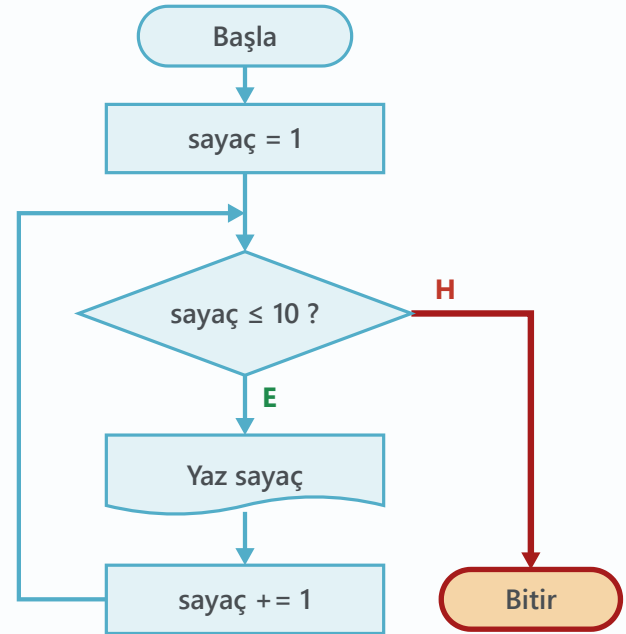
```
1 public class BirdenOna {  
2     public static void main(String[] args)  
3     {  
4         int sayac = 1;  
5         while (sayac <= 10) {  
6             System.out.print(sayac + " ");  
7             sayac += 1;  
8         }  
9     }
```

DEĞİŞKENLER

sayac = 11

ÇIKTI

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



Bitir. Gövde 10 kez, koşul 11 kez çalıştı.

TEKRARLI AKIŞ

Sınır Koşulu: $<$ mı, $\leq$ mü?

Döngü koşulundaki tek bir karakter, tekrar sayısını değiştirir. Sayaç 1'den başlıyor ve her turda 1 artıyor:

Koşul	Gövde kaç kez?	Çıktı	1'den 10'a yazar mı?
<code>sayac < 10</code>	9	1 2 3 4 5 6 7 8 9	Hayır: 10 eksik
<code>sayac <= 10</code>	10	1 2 3 4 5 6 7 8 9 10	Evet
<code>sayac < 11</code>	10	1 2 3 4 5 6 7 8 9 10	Evet (aynı anlam)
<code>sayac <= 11</code>	11	1 2 ... 10 11	Hayır: 11 fazla

Bu hataya **bir eksik / bir fazla hatası** (off-by-one error) denir. Önlemek için koşulu **son değerle** deneyin: `sayac = 10` iken koşul doğru mu? "1'den 10'a **kadar**" 10'u da kapsadığı için koşul `sayac  $\leq$  10` olmalıdır.

TEKRARLI AKIŞ

Döngüye Özel Akış (Döngü Sembolü)

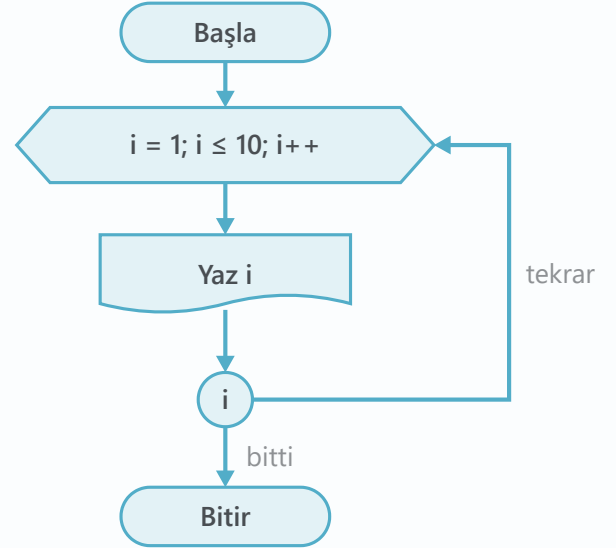
Aynı problem: 1'den 10'a kadar sayıları yazdırma.

1. Başla
2. Döngü: 1'den 10'a kadar
3. Döngü değişkenini yaz
4. Döngü sonu
5. Bitir

Altıgen sembol döngünün üç parçasını tek kutuda toplar:

- **başlangıç:** `i = 1` (bir kez)
- **koşul:** `i ≤ 10` (her turdan önce)
- **güncelleme:** `i++`, yani i'yi 1 artır (her turdan sonra)

"i" yazılı küçük çember **döngü sonudur**: akış oradan altıgene döner, döngü bitince Bitir'e geçer.



Aynı Döngü İki Biçimde: while ve for

while

```
public class BirdenOna {  
    public static void main(String[] args) {  
        int sayac = 1;  
        while (sayac <= 10) {  
            System.out.print(sayac + " ");  
            sayac += 1;  
        }  
    }  
}
```

for

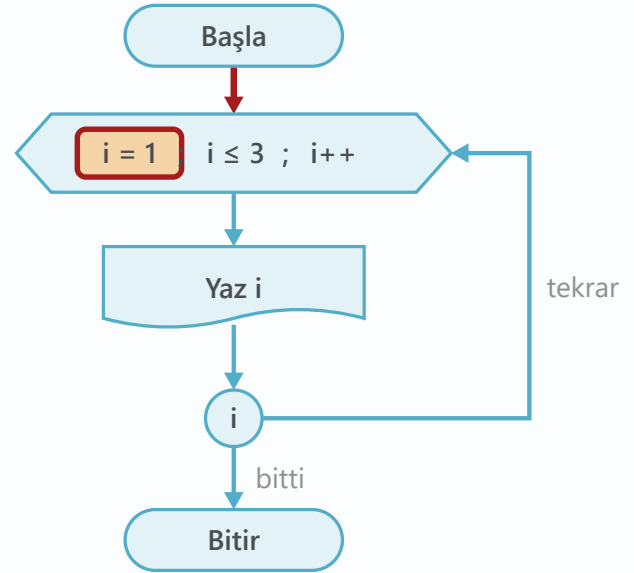
```
public class BirdenOnaFor {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 10; i++) {  
            System.out.print(i + " ");  
        }  
    }  
}
```

Parça	while	for	Diyagramda
başlangıç	<code>int sayac = 1;</code> (döngüden önce)	<code>int i = 1</code>	işlem kutusu / altıgen
koşul	<code>sayac <= 10</code>	<code>i <= 10</code>	karar kutusu / altıgen
güncelleme	<code>sayac += 1;</code> (gövdenin sonunda)	<code>i++</code>	işlem kutusu / altıgen

İki programın çıktısı da aynıdır: `1 2 3 4 5 6 7 8 9 10`. Tekrar sayısı belliyse `for`, koşula bağlıysa `while` daha okunaklıdır.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

i = 1

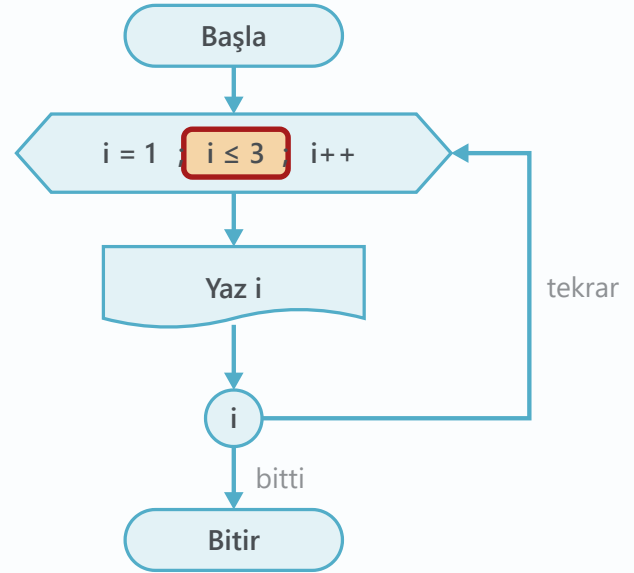
ÇIKTI

başlangıç: i = 1 (yalnız bir kez)

Başlangıç kısmı yalnız bir kez çalışır: **i** = 1.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

i = 1

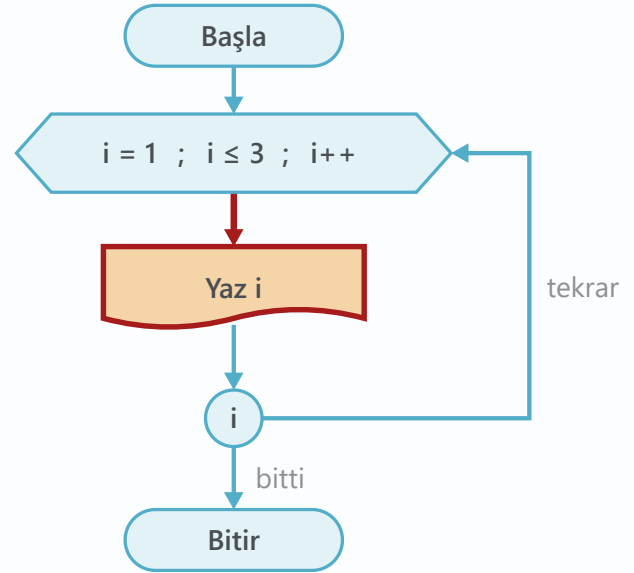
ÇIKTI

koşul: $1 \leq 3 \rightarrow$ doğru

Koşul: $1 \leq 3$ doğru $\rightarrow$ gövdeye girilir.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

i = 1

ÇIKTI

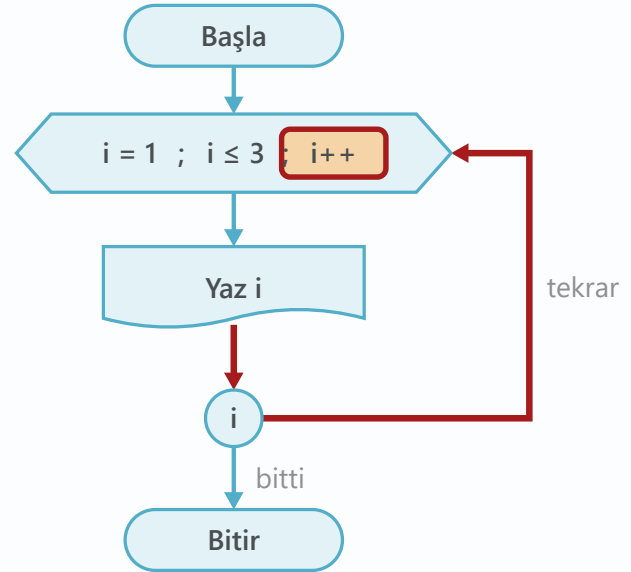
1

gövde: 1 yazılır

Gövde: 1 yazılır.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

i = 2

ÇIKTI

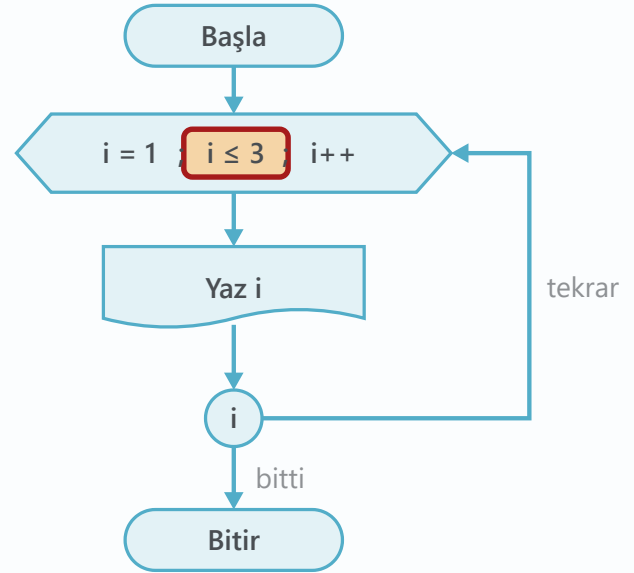
1

güncelleme: i++ → 2

Güncelleme gövdeden **sonra** çalışır: **i** = 2.
Geri okla koşula dönülür.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

i = 2

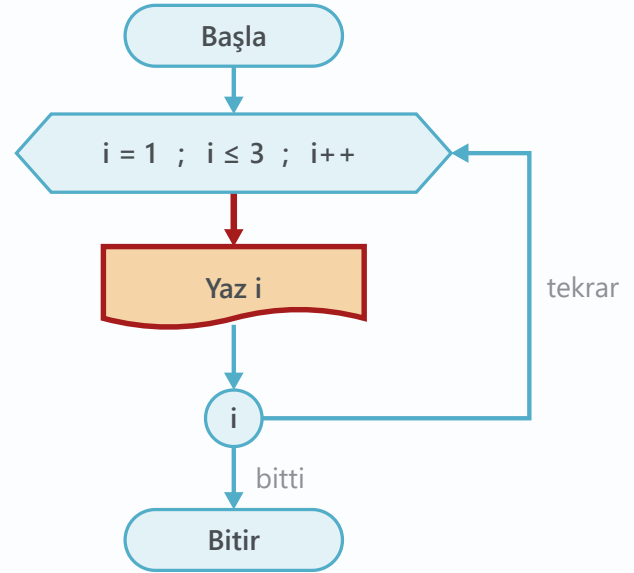
ÇIKTI

1

koşul: $2 \leq 3 \rightarrow$ doğru**Koşul:** $2 \leq 3$ doğru $\rightarrow$ gövdeye girilir.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

i = 2

ÇIKTI

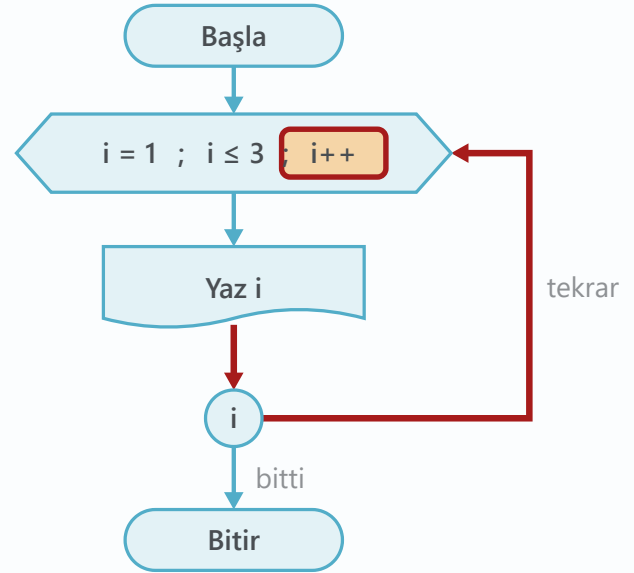
1 2

gövde: 2 yazılır

Gövde: 2 yazılır.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

i = 3

ÇIKTI

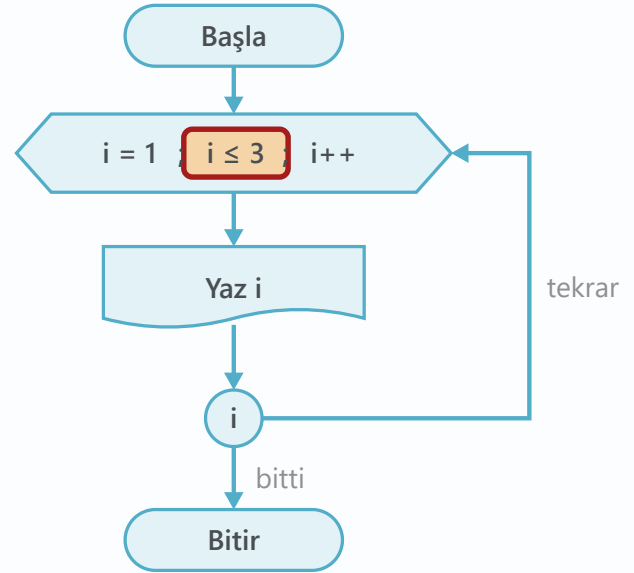
1 2

güncelleme: i++ → 3

Güncelleme gövdeden **sonra** çalışır: **i** = 3.
Geri okla koşula dönülür.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

i = 3

ÇIKTI

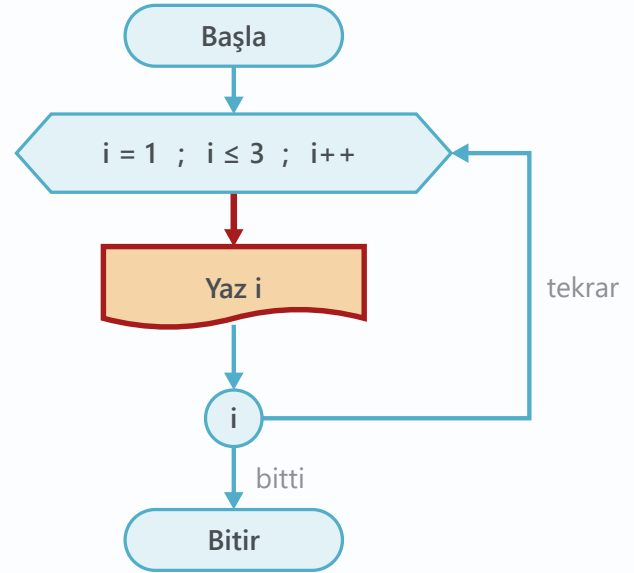
1 2

koşul: $3 \leq 3 \rightarrow$ doğru

Koşul: $3 \leq 3$ doğru $\rightarrow$ gövdeye girilir.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

i = 3

ÇIKTI

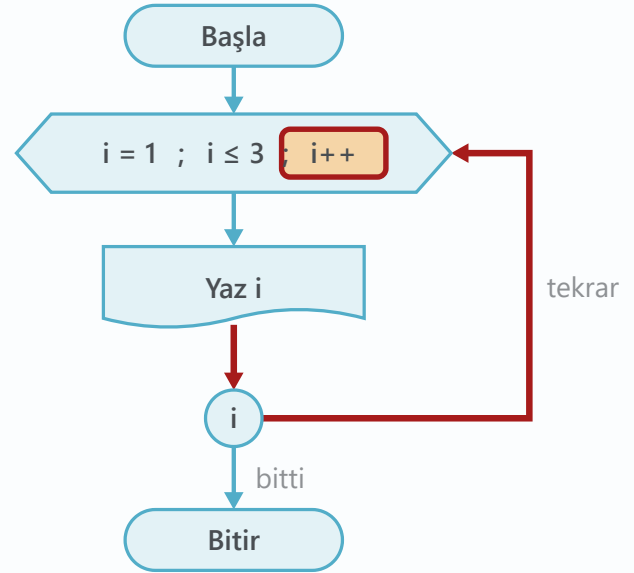
1 2 3

gövde: 3 yazılır

Gövde: 3 yazılır.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

i = 4

ÇIKTI

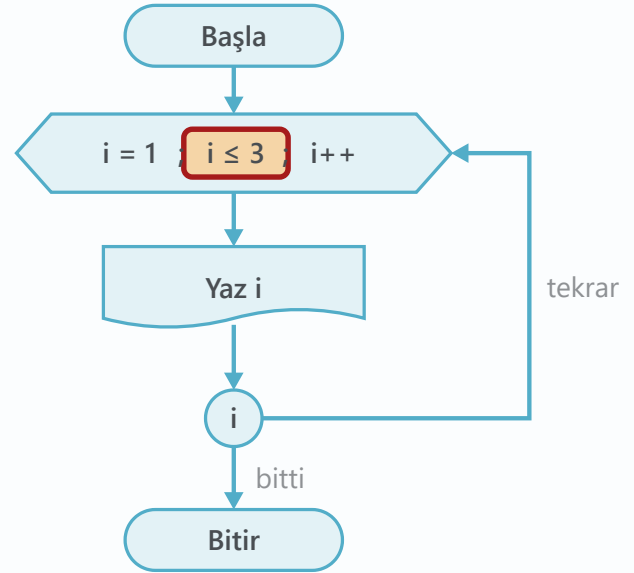
1 2 3

güncelleme: i++ → 4

Güncelleme gövdeden **sonra** çalışır: **i** = 4.
Geri okla koşula dönülür.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

i = 4

ÇIKTI

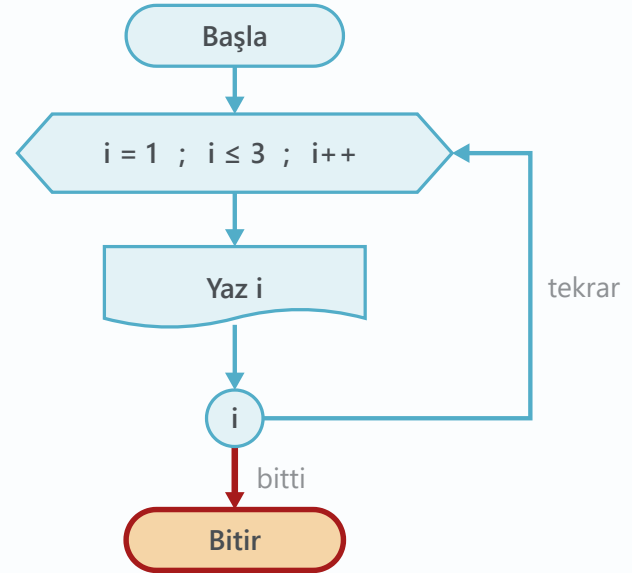
1 2 3

koşul: $4 \leq 3 \rightarrow$ yanlış

Koşul: $4 \leq 3$ yanlış $\rightarrow$ döngü biter. **i** yalnız döngü içinde tanımlıdır.

for Döngüsü: Kod İzleme

```
1 public class UcKez {  
2     public static void main(String[] args) {  
3         for (int i = 1; i <= 3; i++) {  
4             System.out.print(i + " ");  
5         }  
6         System.out.println("| döngü bitti");  
7     }  
8 }
```



DEĞİŞKENLER

—

ÇIKTI

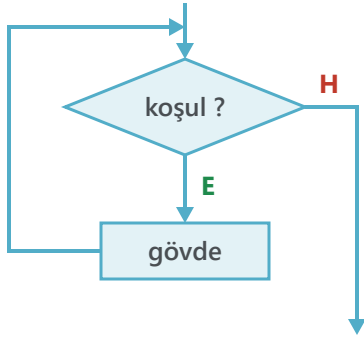
1 2 3 | döngü bitti

Sıra: başlangıç, (koşul, gövde, güncelleme) × 3, koşul

Döngüden sonraki satır çalışır. Sıra: başlangıç 1 kez, koşul 4 kez, gövde ve güncelleme 3'er kez.

Koşul Nerede? Döngü Türleri

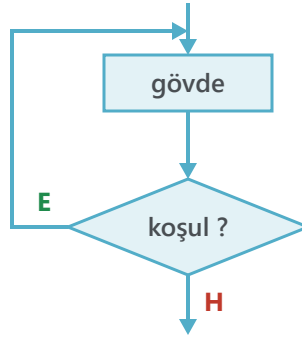
Önce test



Koşul başta sınanır; ilk seferde yanlışsa gövde **hiç** çalışmaz.

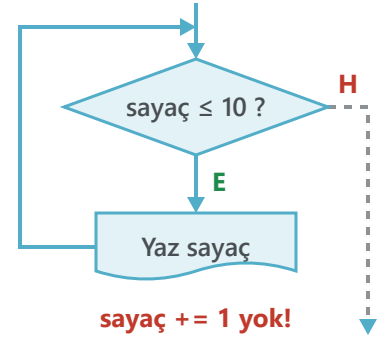
Java: `while` , `for` .

Sonra test



Gövde önce çalışır, koşul sonra sınanır: gövde **en az bir kez** çalışır. Java: `do { ... } while (koşul);`

Sonsuz döngü

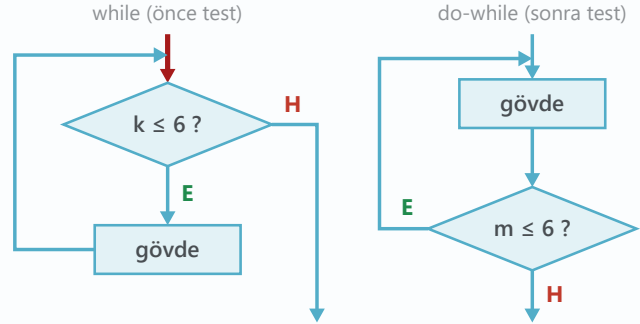


Güncelleme adımı unutulmuş: sayaç hep 1 kalır, Hayır çıkışına **hiç** gidilmez; ekrana durmadan 1 yazılır.

Her döngüde koşulu bir gün **yanlış** yapacak bir adım bulunmalıdır. Sonsuz döngüye giren bir Java programı, konsolda **Ctrl + C** ile ya da IDE'deki durdur düğmesiyle sonlandırılır.

Önce Test ve Sonra Test: Kod İzleme

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

DEĞİŞKENLER**k = 8****ÇIKTI****k = 8**

İki döngü de aynı yanlış koşulla başlıyor. Önce

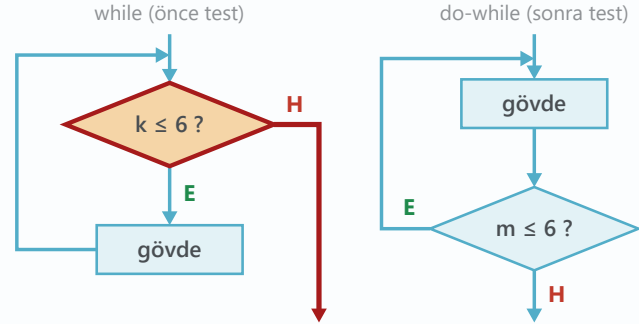
while : k = 8.

Önce Test ve Sonra Test: Kod İzleme

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

DEĞİŞKENLER

k = 8

ÇIKTI**8 ≤ 6 → yanlış**

Koşul **başta** sınanır: 8 ≤ 6 yanlış → gövde **hiç** çalışmaz.

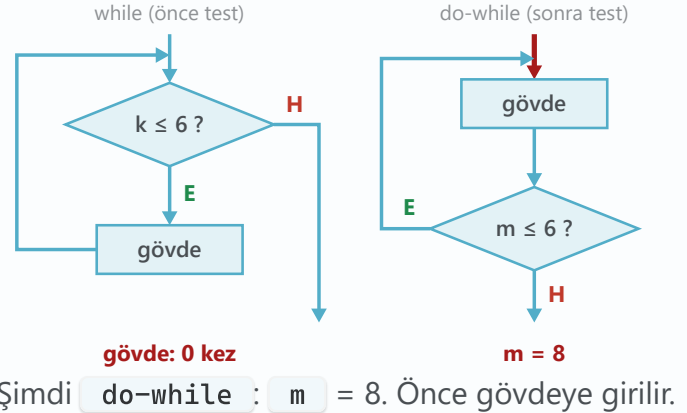
Önce Test ve Sonra Test: Kod İzleme

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

DEĞİŞKENLER

k = 8

m = 8

ÇIKTI

Adım 3 / 7

Önce Test ve Sonra Test: Kod İzleme

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

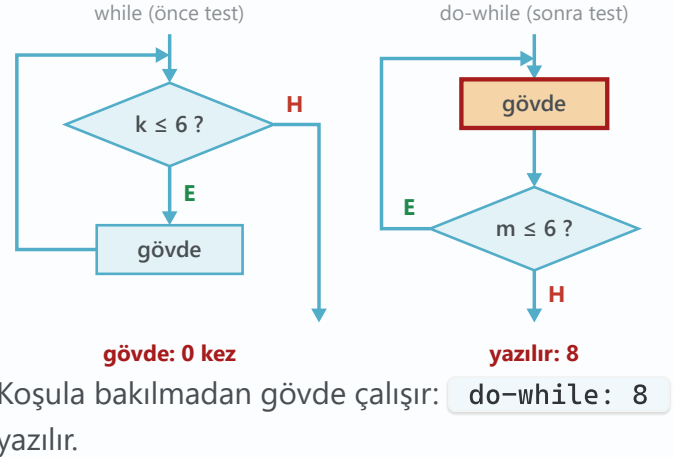
DEĞİŞKENLER

k = 8

m = 8

ÇIKTI

do-while: 8



Önce Test ve Sonra Test: Kod İzleme

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

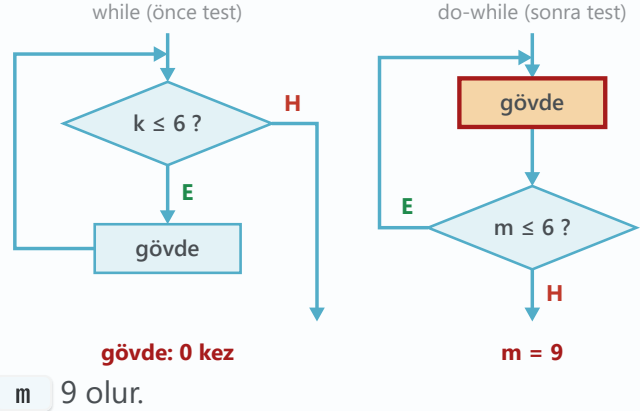
DEĞİŞKENLER

k = 8

m = 9

ÇIKTI

do-while: 8



Adım 5 / 7

Önce Test ve Sonra Test: Kod İzleme

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

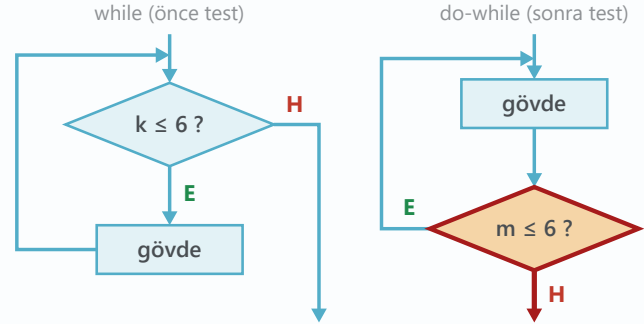
DEĞİŞKENLER

k = 8

m = 9

ÇIKTI

do-while: 8

**gövde: 0 kez****9 ≤ 6 → yanlış**

Koşul **sonda** sınanır: 9 ≤ 6 yanlış → çıkılır. Gövde 1 kez çalıştı.

Önce Test ve Sonra Test: Kod İzleme

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

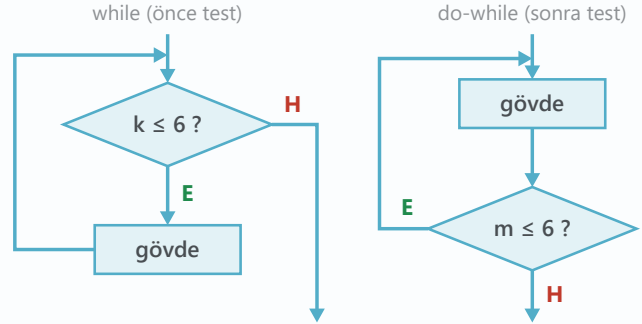
DEĞİŞKENLER

k = 8

m = 9

ÇIKTI

```
do-while: 8
8 9
```



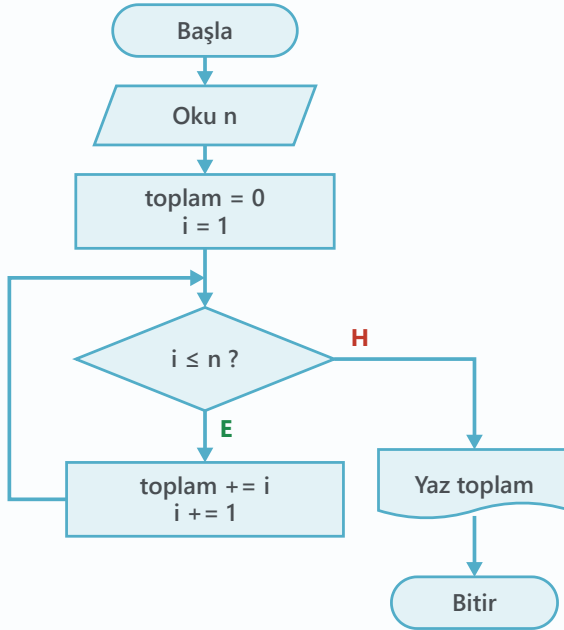
gövde: 0 kez

gövde: 1 kez

8 9 : while gövdesi 0, do-while gövdesi en az 1 kez çalışır.

UYGULAMA

Uygulama: 1'den n'e Kadar Toplam



Sıralı (okuma, başlangıç değerleri), tekrarlı (toplama) ve yine sıralı (yazdırma) akış bir arada. **toplam** değişkeni bir **biriktiricidir** (accumulator): 0'dan başlar, her turda büyür.

```
System.out.print("n: ");
int n = giris.nextInt();
int toplam = 0;
int i = 1;
while (i <= n) {
    toplam += i;
    i += 1;
}
System.out.println("Toplam: " + toplam);
```

i	1	2	3	4	5	6
toplam	1	3	6	10	15	15 (çıkış)

n = 5 için çıktı: **Toplam: 15** (1 + 2 + 3 + 4 + 5).

1'den n'e Toplam: Kod İzleme

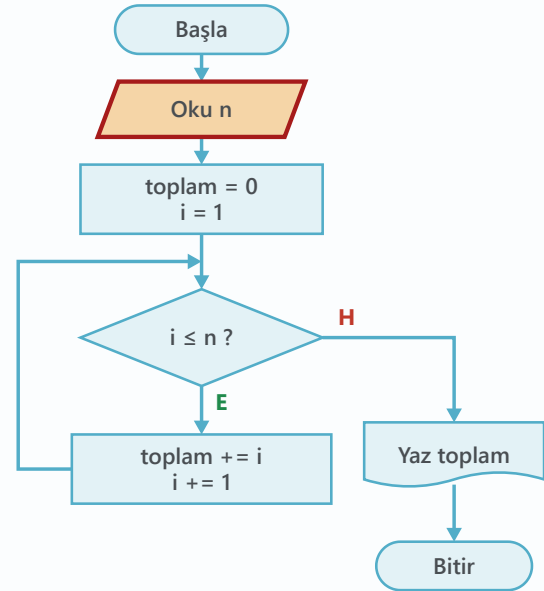
```
▶ System.out.print("n: ");  
▶ int n = giris.nextInt();  
3 int toplam = 0;  
4 int i = 1;  
5 while (i <= n) {  
6     toplam += i;  
7     i += 1;  
8 }  
9 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

n = 4

ÇIKTI

n: 4



n = 4

Mesaj yazılır, kullanıcı 4 girer: n = 4.

1'den n'e Toplam: Kod İzleme

```
1 System.out.print("n: ");
2 int n = giris.nextInt();
▶ int toplam = 0;
▶ int i = 1;
5 while (i <= n) {
6     toplam += i;
7     i += 1;
8 }
9 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

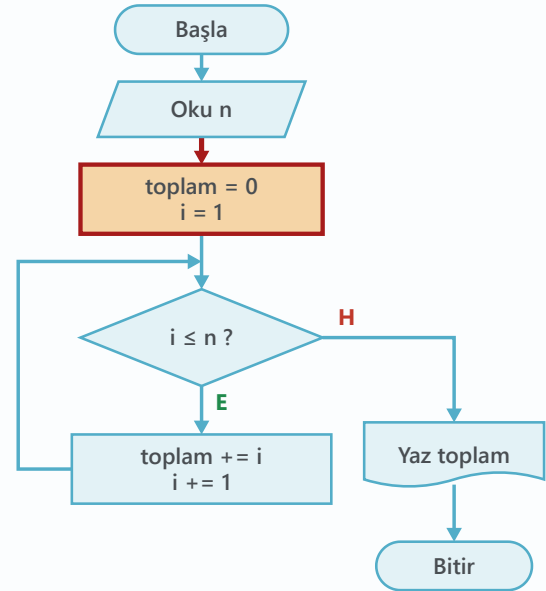
n = 4

toplam = 0

i = 1

ÇIKTI

n: 4

**toplam = 0, i = 1**

Başlangıç değerleri: biriktirici **toplam** = 0, sayacı **i** = 1.

1'den n'e Toplam: Kod İzleme

```
1 System.out.print("n: ");
2 int n = giris.nextInt();
3 int toplam = 0;
4 int i = 1;
5 ► while (i <= n) {
6     toplam += i;
7     i += 1;
8 }
9 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

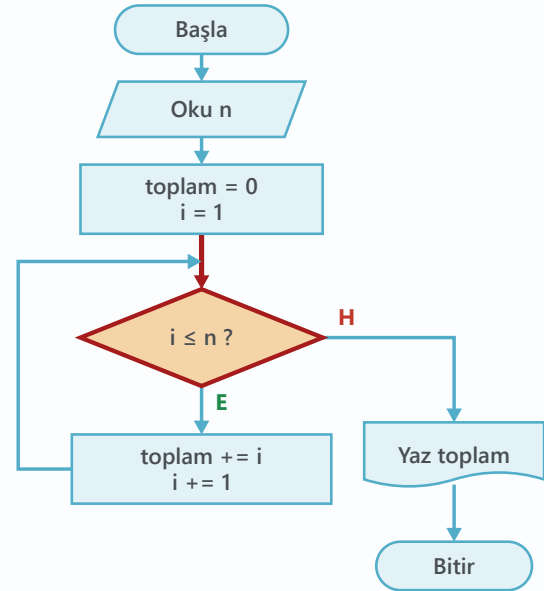
n = 4

toplam = 0

i = 1

ÇIKTI

n: 4

 $1 \leq 4 \rightarrow E$

Koşul: $1 \leq 4$ **doğru** → Evet, gövdeye girilir.

1'den n'e Toplam: Kod İzleme

```
1 System.out.print("n: ");
2 int n = giris.nextInt();
3 int toplam = 0;
4 int i = 1;
5 while (i <= n) {
6     toplam += i;
7     i += 1;
8 }
9 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

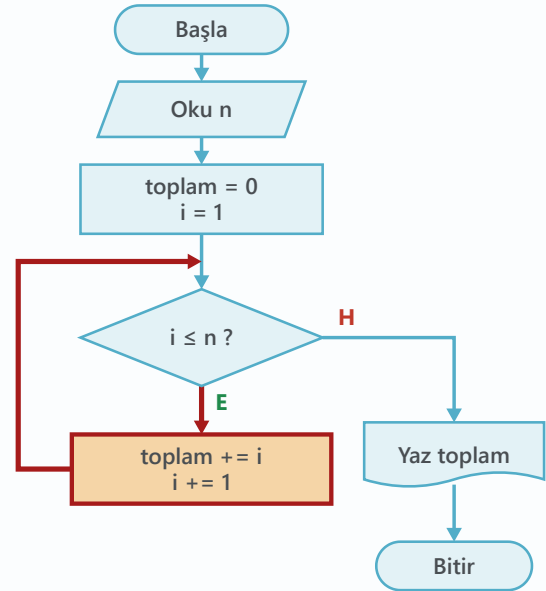
n = 4

toplam = 1

i = 2

ÇIKTI

n: 4



toplam = 0 + 1 = 1, i = 2

toplam = 0 + 1 = 1, i = 2; geri okla karara dönülür.

1'den n'e Toplam: Kod İzleme

```

1 System.out.print("n: ");
2 int n = giris.nextInt();
3 int toplam = 0;
4 int i = 1;
5 while (i <= n) {
6     toplam += i;
7     i += 1;
8 }
9 System.out.println("Toplam: " + toplam);
    
```

DEĞİŞKENLER

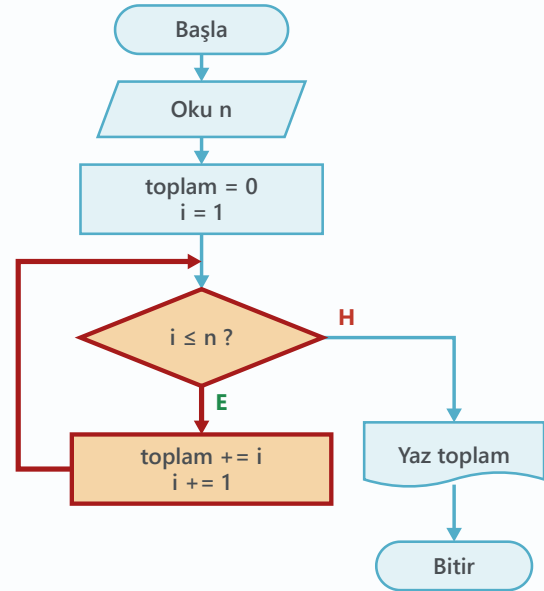
n = 4

toplam = 3

i = 3

ÇIKTI

n: 4



$2 \leq 4 \rightarrow \text{toplam} = 3, i = 3$

$2 \leq 4$ doğru: $\text{toplam} = 1 + 2 = 3, i = 3$.

1'den n'e Toplam: Kod İzleme

```
1 System.out.print("n: ");
2 int n = giris.nextInt();
3 int toplam = 0;
4 int i = 1;
5 while (i <= n) {
6     toplam += i;
7     i += 1;
8 }
9 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

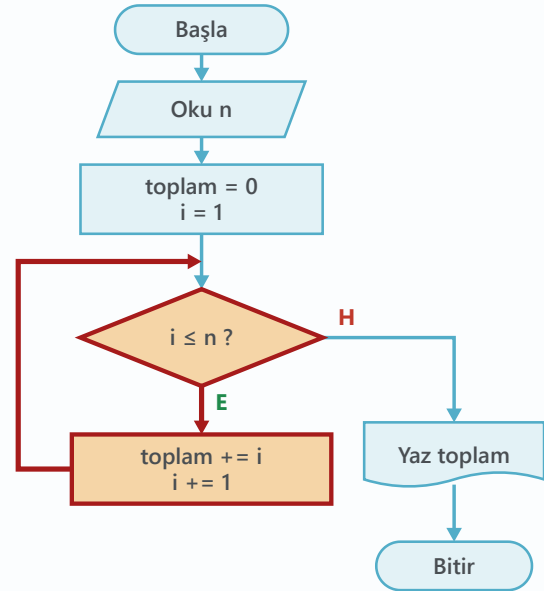
n = 4

toplam = 6

i = 4

ÇIKTI

n: 4



$3 \leq 4 \rightarrow \text{toplam} = 6, i = 4$

$3 \leq 4$ doğru: $\text{toplam} = 3 + 3 = 6$, $i = 4$.

1'den n'e Toplam: Kod İzleme

```
1 System.out.print("n: ");
2 int n = giris.nextInt();
3 int toplam = 0;
4 int i = 1;
5 while (i <= n) {
6     toplam += i;
7     i += 1;
8 }
9 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

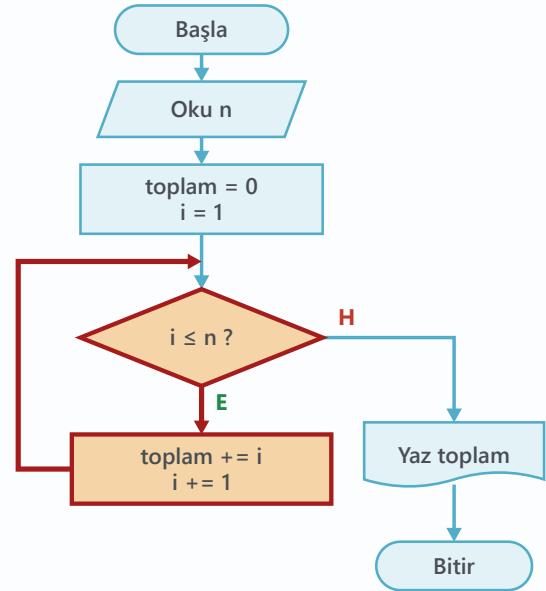
n = 4

toplam = 10

i = 5

ÇIKTI

n: 4



4 ≤ 4 → toplam = 10, i = 5

4 ≤ 4 doğru: **toplam** = 6 + 4 = 10, **i** = 5.

1'den n'e Toplam: Kod İzleme

```
1 System.out.print("n: ");
2 int n = giris.nextInt();
3 int toplam = 0;
4 int i = 1;
5 while (i <= n) {
6     toplam += i;
7     i += 1;
8 }
9 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

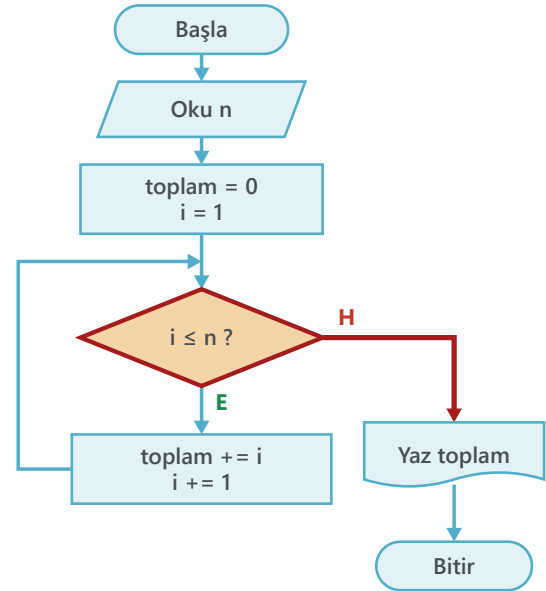
n = 4

toplam = 10

i = 5

ÇIKTI

n: 4

 $5 \leq 4 \rightarrow H$

Koşul: $5 \leq 4$ **yanlış** → Hayır çıkışı, döngüden çıkılır.

1'den n'e Toplam: Kod İzleme

```
1 System.out.print("n: ");
2 int n = giris.nextInt();
3 int toplam = 0;
4 int i = 1;
5 while (i <= n) {
6     toplam += i;
7     i += 1;
8 }
9 System.out.println("Toplam: " + toplam);
```

DEĞİŞKENLER

n = 4

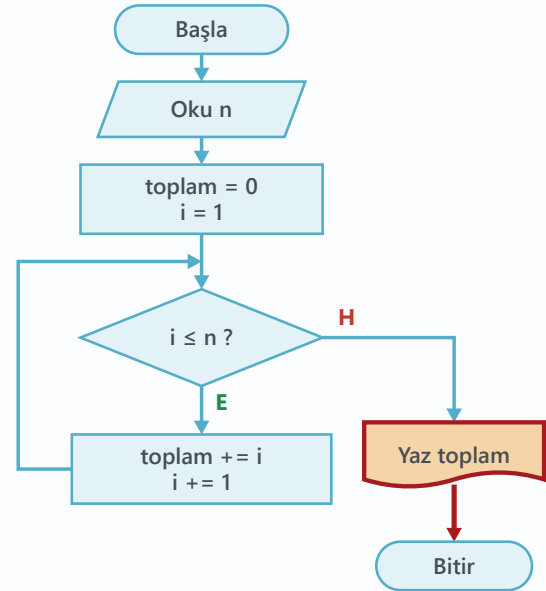
toplam = 10

i = 5

ÇIKTI

n: 4

Toplam: 10

**çıkıtı: Toplam: 10**

Sonuç yazılır: $1 + 2 + 3 + 4 = 10$. Gövde 4 kez, koşul 5 kez çalıştı.

Algoritmayı Göstermenin Yolları

Gösterim	Tek/çift örneği	Güçlü yanı	Zayıf yanı
Doğal dil	"Sayıyı oku; 2'ye bölümünden kalan 0 ise çift, değilse tek yaz."	Herkes okuyabilir	Belirsiz olabilir
Akış diyagramı	Oku → İşlem → Karar → Yaz çift / Yaz tek	Akışı görsel gösterir, dallanma ve döngü kolay görülür	Büyük problemlerde hantal
Sözde kod	<pre>OKU sayı; EĞER sayı mod 2 = 0 İSE YAZ "çift" DEĞİLSE YAZ "tek"</pre>	Hızlı yazılır, koda yakın	Standart bir söz dizimi yok
Java	<pre>if (sayı % 2 == 0) ... else ...</pre>	Bilgisayar çalıştırabilir	Söz dizimi kurallarına uymak gerekir

Yazılım mühendisliğinde akış diyagramınının modern karşılığı **UML etkinlik diyagramıdır** (activity diagram): aynı fikir; başlangıç dolu daire, karar eşkenar dörtgen, etkinlikler yuvarlatılmış kutular.

ÖZET

Özet

Konu	Hatırlanacaklar
Akış diyagramı	algoritmanın sembol ve oklarla çizimi; dilden bağımsız plan
Semboller	oval Başla/Bitir · paralelkenar giriş · dikdörtgen işlem · eşkenar dörtgen karar · altıgen döngü · dalgalı kutu yazdır · çember bağlantı
Kurallar	tek Başla; yukarıdan aşağı; karar kutusunun iki etiketli çıkışı (E/H); kesişmeyen çizgiler
Sıralı akış	adımlar art arda, her biri bir kez; bağımlı adımların sırası önemli
Koşullu akış	K sorusu → Evet: P, Hayır: Q; Java <code>if/else</code> ; iç içe kararlarla üç ve daha çok yol
Tekrarlı akış	geri ok + karar; başlangıç, koşul, güncelleme; Java <code>while</code> , <code>for</code> , <code>do-while</code>
Tuzaklar	<code><</code> / <code>≤</code> sınır hatası; güncellemeyi unutmak → sonsuz döngü

ÖZET

Sık Yapılan Hatalar

Karar kutusunda tek çıkış

Karar sembolünden iki ok çıkmalı ve ikisi de E/H diye etiketlenmeli; aksi hâlde "hayır" durumunda ne olacağı belirsiz kalır.

Karşılaştırmada = kullanmak

`if (sonuc = 0)` derlenmez: `error: incompatible types: int cannot be converted to boolean`. Doğrusu `sonuc == 0`.

Güncellemeyi unutmak

"sayac += 1" kutusu yoksa koşul hep doğru kalır: **sonsuz döngü**. Derleyici bu hatayı yakalamaz.

Döngü başlığına noktalı virgül

`while (sayac <= 10);` derlenir ama gövdesi boş bir döngü olur; sayac artmadığı için program takılır.

Yanlış sembol

Klavyeden okuma için dikdörtgen, hesaplama için paralelkenar kullanmak. Girdi = paralelkenar, işlem = dikdörtgen.

Başlangıç değerini unutmak

Diyagramda "sayac = 1" kutusu yoksa Java da şikâyet eder: `error: variable sayac might not have been initialized`.

Sınır hatası

"1'den 10'a kadar" için `sayac < 10` yazmak 10'u atlar. Koşulu son değerle deneyin.

Geri oku yanlış yere bağlamak

Geri ok "sayac = 1" kutusunun **üstüne** dönerse sayac her turda 1'e sıfırlanır; yine sonsuz döngü. Ok karar kutusuna dönmelidir.

ÖZET

Örnek Problemler

1. "Kullanıcıdan yaşını al", " $\text{yaş} \geq 18$ mi?", " $\text{vergi} = \text{fiyat} \times 0,2$ " ve "Sonucu ekrana yaz" adımları hangi sembollerle çizilir?
2. Diyagram: toplam = 0, i = 1; i ≤ 4 iken toplam += i ve i += 1; sonra Yaz toplam. Çıktı nedir? Döngüden çıkışta i kaçtır?
3. Tek/çift diyagramına **-3** girilirse ne yazılır? (Java'da `-3 % 2` kaçtır?)
4. 1'den 10'a diyagramında koşul `sayaç < 10` olsaydı kaç sayı yazılırdı? Karar kutusu kaç kez sınanırdı?
5. sayı = 1 iken " $\text{sayı} < 100$ ise $\text{sayı} = \text{sayı} \times 2$ " döngüsü kaç kez döner ve sayı kaç olur?

► Cevapları göster

ÖZET

Alıştırmalar

1. Kenarları klavyeden okunan dikdörtgenin alanını ve çevresini yazan akış diyagramını çizin (sıralı).
2. Girilen üç sayıdan en büyüğünü bulan diyagramı çizin. 3, 9, 5 için izleyin.
3. 1'den n'e kadar **çift** sayıların toplamını bulan diyagramı çizin. $n = 10$ için sonuç?
4. $n!$ (faktöriyel) hesaplayan diyagramı çizin ve Java'ya çevirin. $n = 5$ için sonuç?
5. Kullanıcı 0 girene kadar sayı okuyup toplamı yazan diyagramı çizin. Girdi: 4 7 2 0.
6. 0–100 arası notu harf notuna (≥ 85 A, ≥ 70 B, ≥ 50 C, değilse F) çeviren diyagramı çizin.

► Cevaplar / ipuçları

Kaynaklar

- ISO 5807:1985, *Information processing — Documentation symbols and conventions for data, program and system flowcharts*.
- C. Böhm ve G. Jacopini, "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules", *Communications of the ACM* 9(5), 1966.
- P. Deitel ve H. Deitel, *Java: How to Program*, 11. baskı, Pearson, 2017 (Bölüm 4–5: kontrol deyimleri ve etkinlik diyagramları).
- A. B. Downey ve C. Mayfield, *Think Java*, 2. baskı, O'Reilly, 2019.
- Oracle, *The Java Tutorials: "Control Flow Statements"* (docs.oracle.com/javase/tutorial).
- K. Sierra ve B. Bates, *Head First Java*, 3. baskı, O'Reilly, 2022.

Son

Bölüm 1: Akış Diyagramları

Sonraki bölüm: Bölüm 2: Programlama · [Terim sözlüğü](#)