

Java Programlama, Boolean Veri Türü ve Kontrol Yapıları

5. Hafta

Dr. Öğr. Üyesi BÜŞRA ÖZDENİZCİ KÖSE

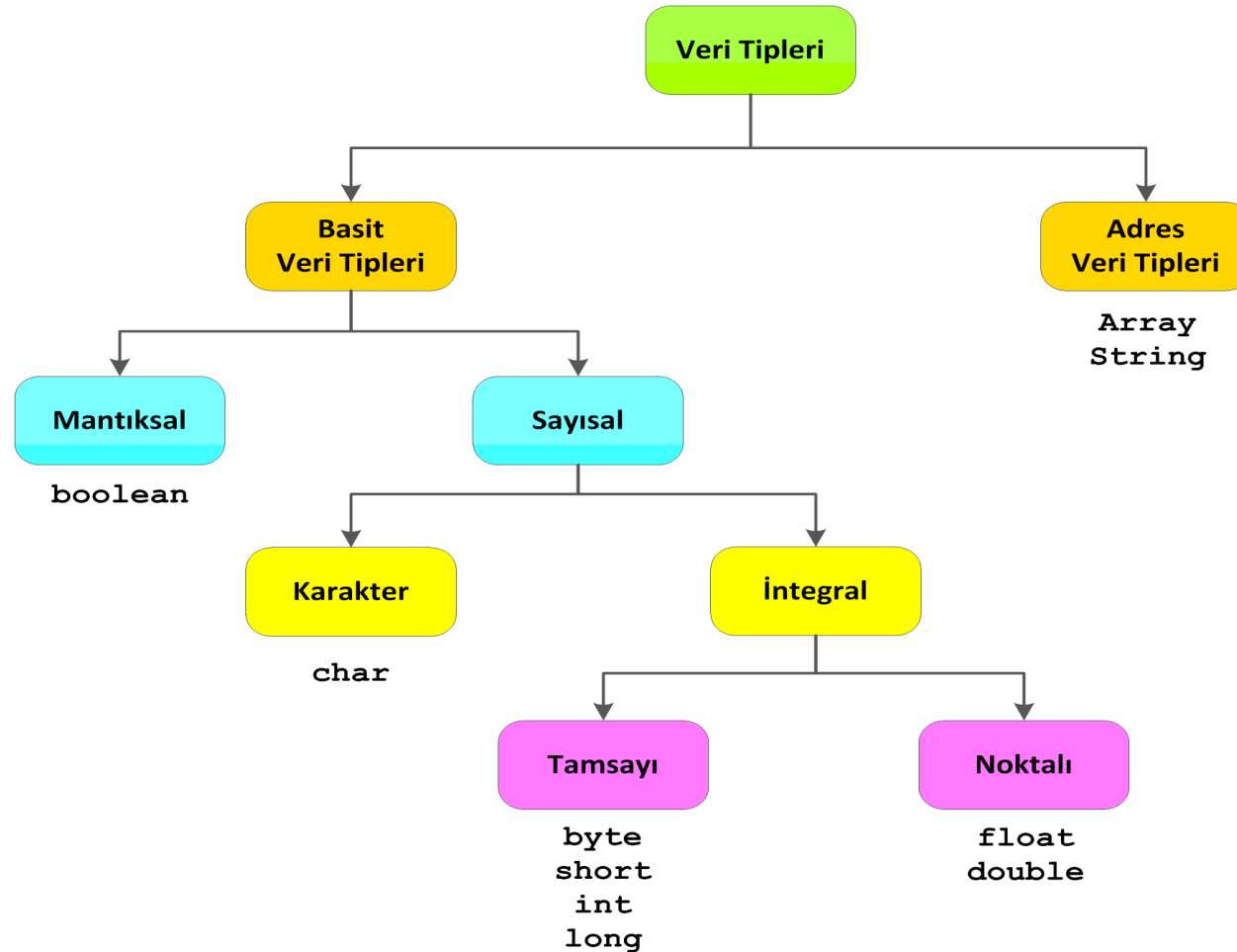
İşletme Bölümü

İşletme Fakültesi

Neler Öğrendik?

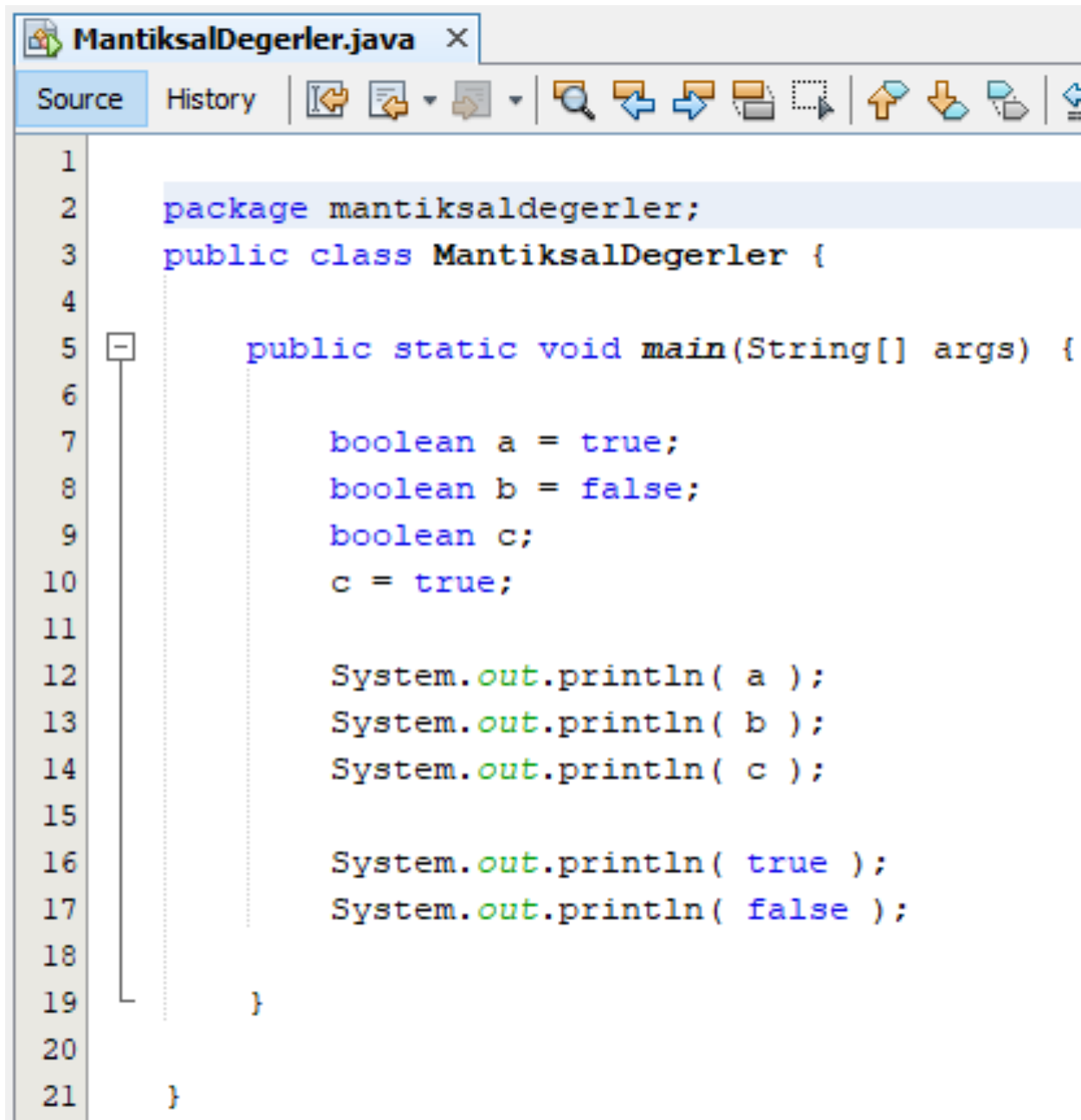
- Bellek üzerinde Okuma ve Yazma, Veri Türleri
- Sayısal Veri Türleri: Tamsayılar (byte, short, int, long)
- Tamsayılar için Scanner sınıfı (Scanner class) Metotları
- Unary Operatörler, Binary Operatörler, Operatör Öncelikleri
- Sayısal Veri Türleri: Gerçek Sayılar (float, double)
- Gerçek Sayılar için Scanner sınıfı (Scanner class) Metotları

Java'da Veri Türleri

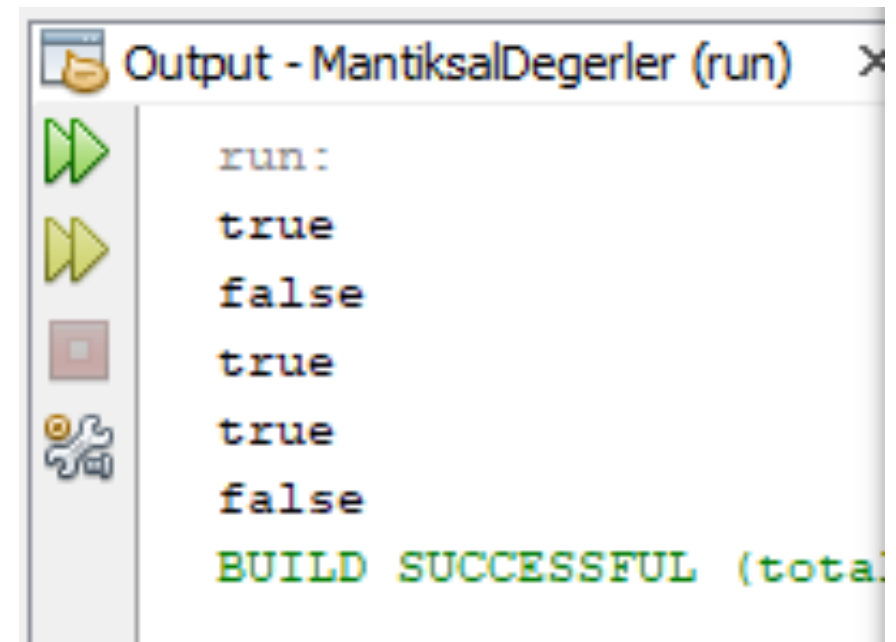


Boolean Veri Türü

- Java programlama dilinde, mantıksal değerlendirmeleri yapmak amacıyla **boolean** veri türü tanımlanmıştır
- **boolean** veri türünde tanımlanmış olan değişkenler **true** (mantık bilimindeki karşılığı doğru) ya da **false** (mantık bilimindeki karşılığı yanlış) şeklinde iki değerden birisine sahip olabilirler



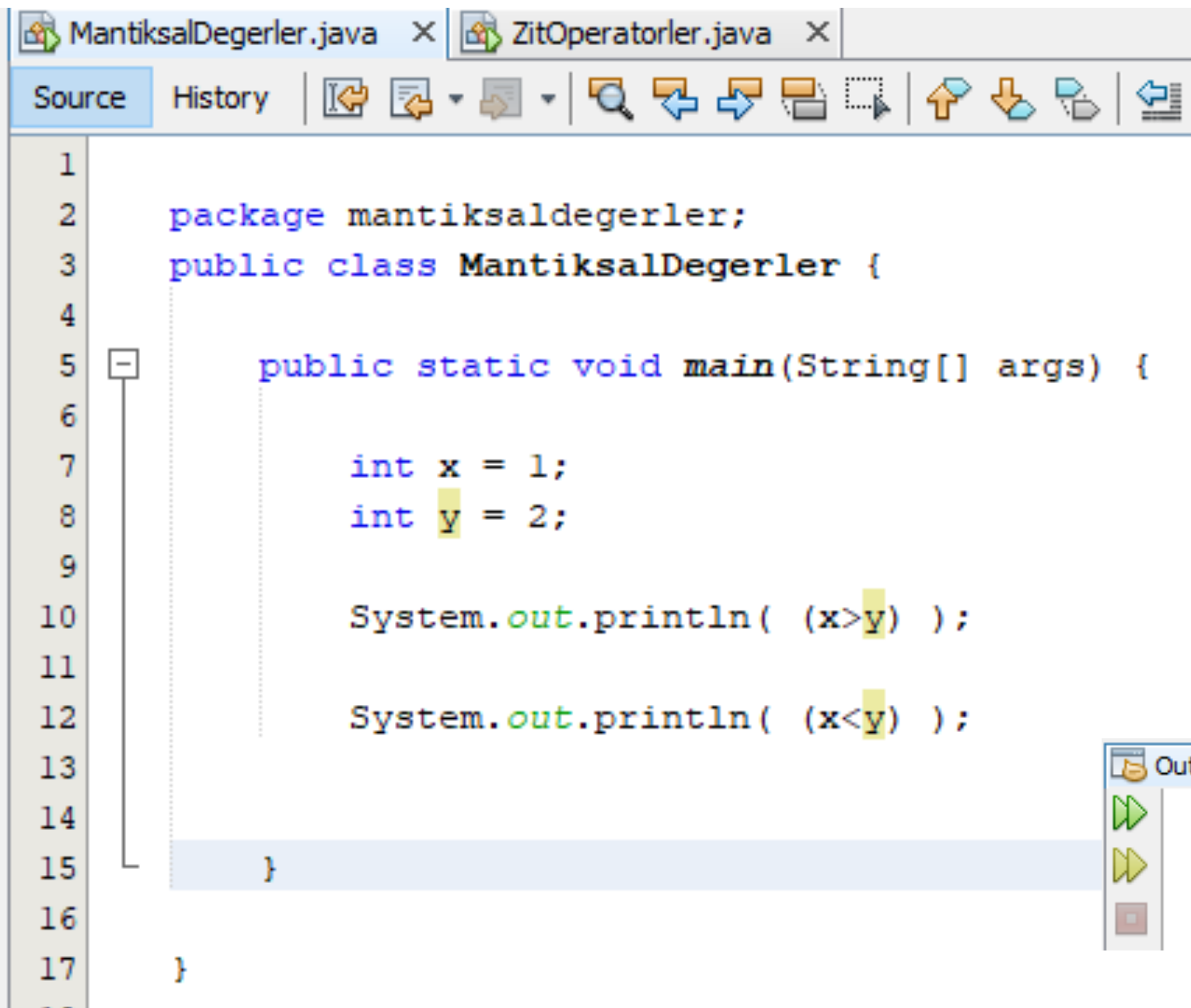
```
1
2 package mantiksaldegerler;
3 public class MantiksalDegerler {
4
5     public static void main(String[] args) {
6
7         boolean a = true;
8         boolean b = false;
9         boolean c;
10        c = true;
11
12        System.out.println( a );
13        System.out.println( b );
14        System.out.println( c );
15
16        System.out.println( true );
17        System.out.println( false );
18
19    }
20
21 }
```



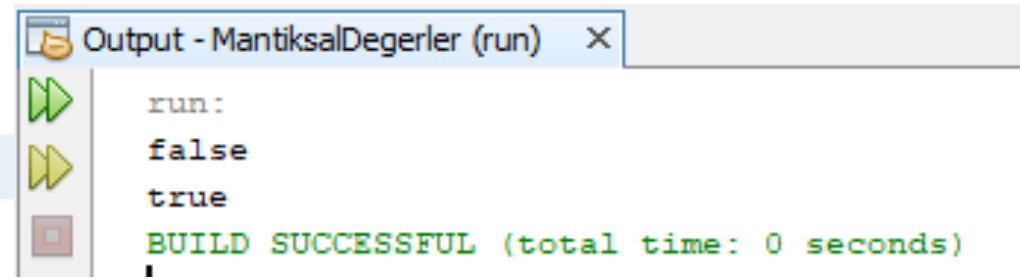
```
Output - MantiksalDegerler (run)
run:
true
false
true
true
false
BUILD SUCCESSFUL (total
```

Boolean Veri Türü

- Peki, iki değeri nasıl karşılaştırırız?
- Örneğin yarıçap değeri 0'dan büyükse, 0'dan küçükse veya 0'a eşitse?



```
1
2 package mantiksaldegerler;
3 public class MantiksalDegerler {
4
5     public static void main(String[] args) {
6
7         int x = 1;
8         int y = 2;
9
10        System.out.println( (x>y) );
11
12        System.out.println( (x<y) );
13
14
15    }
16
17 }
```



```
run:
false
true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Operatörler

- Zıt Anlam Operatörü
 - alan veya NOT alan?
- Karşılaştırma Operatörü
 - Eşittir, Küçüktür, Büyüktür vb.
- Birleştirme Operatörü
 - AND, Mantıksal VE (şartlı birleştirme)
 - OR, Mantıksal VEYA (şartsız birleştirme)
 - XOR

Zıt Anlamalı Operatör

a	!a	Açıklama
true	false	a değeri doğru ise, (!a) yanlış olur
false	true	a değeri yanlış ise, (!a) doğru olur

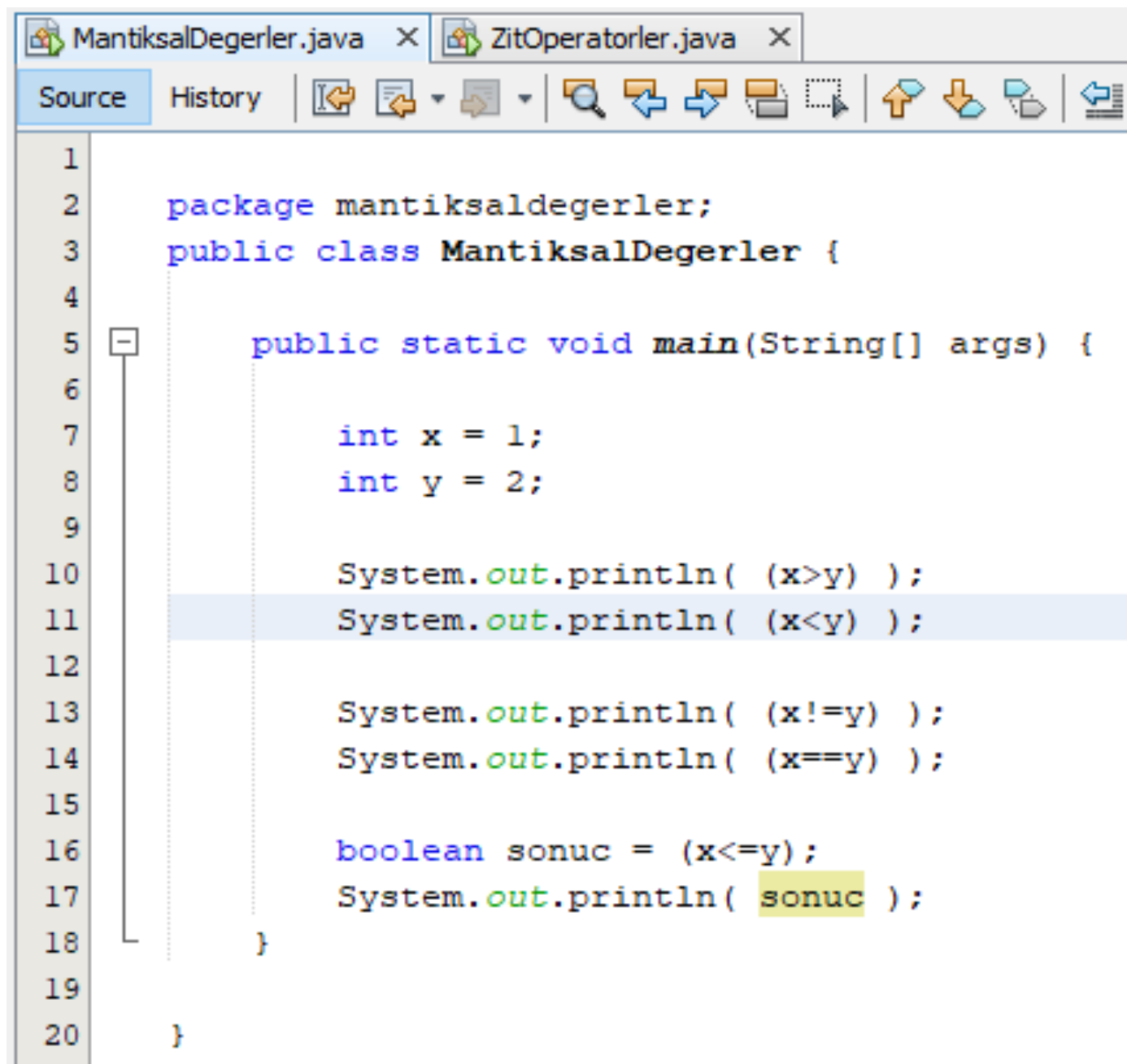
```
MantiksalDegerler.java x ZitOperatorler.java x
Source History
1
2 package zitoperatorler;
3
4 public class ZitOperatorler {
5
6     public static void main(String[] args) {
7
8         boolean a = true;
9         boolean b = false;
10        boolean c;
11        c = true;
12
13        System.out.println( (!a) );
14        System.out.println( (!b) );
15
16        System.out.println( (!c) );
17        System.out.println( !(!c) );
18
19        System.out.println( (!true) );
20        System.out.println( !(!true) );
21
22    }
23
24 }
```



```
Output - ZitOperatorler (run) x
run:
false
true
false
true
false
true
BUILD SUCCESSFUL (total
```

Karşılaştırma Operatörleri

Matematiksel Karşılığı	Java'da Karşılığı	Açıklama
=	==	Equal to?
≠	!=	Not equal to?
>	>	Greater than?
<	<	Less Than?
≥	>=	Greater than or equal to?
≤	<=	Less than or equal to?



```
1
2 package mantiksaldegerler;
3 public class MantiksalDegerler {
4
5     public static void main(String[] args) {
6
7         int x = 1;
8         int y = 2;
9
10        System.out.println( (x>y) );
11        System.out.println( (x<y) );
12
13        System.out.println( (x!=y) );
14        System.out.println( (x==y) );
15
16        boolean sonuc = (x<=y);
17        System.out.println( sonuc );
18    }
19
20 }
```



DO IT
NOW!

```
1 package mantiksaldegerlercalisma;
2
3
4 public class MantiksalDegerlerCalisma {
5
6     public static void main(String[] args) {
7
8         int x = 5;
9         int y = 3;
10        boolean a = true;
11        boolean b = false;
12
13        System.out.println( (a==b) );
14
15        System.out.println( (!a==b) );
16
17        System.out.println( (x<y) );
18
19        System.out.println( !(x<y) );
20
21    }
22
23 }
```

Output - MantiksalDegerlerCalisma (run) X

run:
false
true
false
true
BUILD SUCCESSFUL (total time: 0 seconds)

İfade	Mantıksal Değer
<pre>boolean z = true; int x = 1; int y = 2;</pre>	
$!(z)$	$!(\text{true}) \rightarrow \text{false}$
$!(!(z))$	$!(!(\text{true})) \rightarrow !(false) \rightarrow \text{true}$
$(y > x)$	true
$!(x > y)$	$!(false) \rightarrow \text{true}$
$!(y > x)$	$!(\text{true}) \rightarrow \text{false}$
$!(1 > 2)$	$!(false) \rightarrow \text{true}$

İfade	Sonuç
<pre>boolean a = true; boolean b = false;</pre>	
(a != b)	true
(a == b)	false
(a != true)	false
(a != false)	true
(a == true)	true

İfade	Sonuç
<pre>int x = 1; int y = 2;</pre>	
(x != y)	true
(x == y)	false
(x > y)	false
(x <= y)	true
(x >= y)	false
(x < y)	true

== ifadesi, kontrol edilen soldaki ve sağdaki değerlerin birbirine eşit olması durumunda false, aksi durumda true değerini döndürür. Mantıksal işlemlerde kontrol edilmekte olan sol ve sağ taraftaki ifadelerin aynı türde olmasının gerekliliğine dikkat ediniz! Örneğin bir tamsayıyla başka bir tamsayı karşılaştırılması beklenen durumdur; fakat tamsayı ile gerçel sayı; ya da gerçel sayı ile karakter karşılaştırılması mümkün değildir. Böyle bir durumda derleyici hata üretecek ve sınıf dosyasını yaratmayacaktır. Tamsayıların karşılaştırılması esnasında farklı tamsayı veri türleri -örneğin byte ile int değeri- kullanılmasında ise sakınca yoktur.

Birleştirme Operatörü: Mantıksal VE (&&)

a	b	a && b	Açıklama
true	true	true	Her iki değer de true ise sonuç true olur.
true	false	false	Değerlerden herhangi birisi false ise sonuç da false olur.
false	true		
false	false		

&& operatörü programlamada oldukça sıklıkla kullanılmaktadır. Bunun yanında sıklıkla kullanılmayan & operatörü, sırasıyla birinci ifadenin ve sonra da ikinci ifadenin değerlendirilmesini gerektirir. && operatörü ise önce birinci ifadenin değerlendirilmesini ifade eder. Eğer ilk ifade true ise diğer ifadenin değerlendirilmesine geçilebilir; fakat ilk ifade false değerini döndürüyorsa diğer ifade değerlendirilmez, sonuç doğrudan false olur.



```
2 package birlestirmeoperatoru;
3 public class BirlestirmeOperatoru {
4     public static void main(String[] args) {
5
6         int x = 1;
7         int y = 2;
8         int z = 3;
9         boolean hizli = false;
10        boolean caliskan = true;
11
12        -----
13
14        boolean sonuc1 = (3 > 2) && (3 >= 2);
15
16        boolean sonuc2 = (x > y) && (y != z);
17
18        boolean sonuc3 = (hizli && (!caliskan));
19
20        boolean sonuc4 = ((!hizli) && caliskan);
21
22        boolean sonuc5 = (3 > 5) && (3 > 2);
23
24        System.out.println ("Sonuçlar sırasıyla: ");
25        System.out.println (sonuc1);
26        System.out.println (sonuc2);
27        System.out.println (sonuc3);
28        System.out.println (sonuc4);
29        System.out.println (sonuc5);
30    }
```

```
Output - BirlestirmeOperatoru (run) X
run:
Sonuçlar sırasıyla:
true
false
false
true
false
BUILD SUCCESSFUL (total time: 0 seconds)
```

Birleştirme Operatörü: Mantıksal VEYA (||)

a	b	a b	Açıklama
true	true	true	Değerlerden herhangi birisi true ise sonuç true olur.
true	false		
false	true		
false	false	false	Her iki değer de false ise sonuç false olur.

```
2 package birlestirmeoperatoru;
3 public class BirlestirmeOperatoru {
4     public static void main(String[] args) {
5
6         int x = 1;
7         int y = 2;
8         int z = 3;
9         boolean hizli = false;
10        boolean caliskan = true;
11        boolean duzenli = false;
12
13        boolean sonuc1 = ( 3 > 2 ) || ( 3 >= 2 ) ;
14
15        boolean sonuc2 = !( x < y ) || ( x > z ) ;
16
17        boolean sonuc3 = ( caliskan || duzenli ) ;
18
19        boolean sonuc4 = ( (!hizli) || caliskan ) ;
20
21        boolean sonuc5 = ( 8 > 5 ) || ( 3 >= 0 ) ;
22
23        boolean sonuc6 = !( caliskan || duzenli );
24
25        System.out.println ("Sonuçlar sırasıyla: ");
26        System.out.println (sonuc1);
27        System.out.println (sonuc2);
28        System.out.println (sonuc3);
29        System.out.println (sonuc4);
30        System.out.println (sonuc5);
31        System.out.println (sonuc6);
32    }
33 }
```

DO IT
NOW!

Output - BirlestirmeOperatoru (run) X

run:
Sonuçlar sırasıyla:
true
false
true
true
true
false
BUILD SUCCESSFUL (total time: 0 seconds)

Birleştirme Operatörü: Exclusive OR (XOR) (^)

a	b	a ^ b	Açıklama
true	true	false	İki ifade aynı değer sahipse sonuç false olur.
false	false	false	
true	false	true	İki ifade zıt değeri sahipse sonuç true olur.

```
2 package birlestirmeoperatoru;
3 public class BirlestirmeOperatoru {
4     public static void main(String[] args) {
5
6         int x = 1;
7         int y = 2;
8         int z = 3;
9         boolean hizli = false;
10        boolean caliskan = true;
11        boolean duzenli = false;
12
13        boolean sonuc1 = ( 2 > 3 ) ^ ( 3 >= 1 ) ;
14
15        boolean sonuc2 = ( y != z ) ^ ( x < z ) ;
16
17        boolean sonuc3 = ( caliskan ^ duzenli ) ;
18
19        boolean sonuc4 = ( (!hizli) ^ caliskan ) ;
20
21        boolean sonuc5 = ( 8 > 5 ) ^ ( 3 >= 0 ) ;
22
23        System.out.println ("Sonuçlar sırasıyla: ");
24        System.out.println (sonuc1);
25        System.out.println (sonuc2);
26        System.out.println (sonuc3);
27        System.out.println (sonuc4);
28        System.out.println (sonuc5);
29    }
30 }
```

DO IT
NOW!

Output - BirlestirmeOperatoru (run)

run:

Sonuçlar sırasıyla:

true

false

true

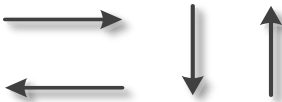

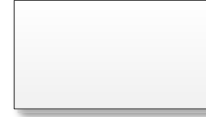
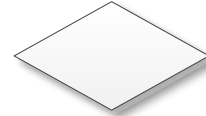
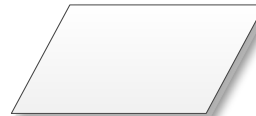

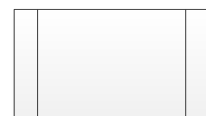
false

false

BUILD SUCCESSFUL (total

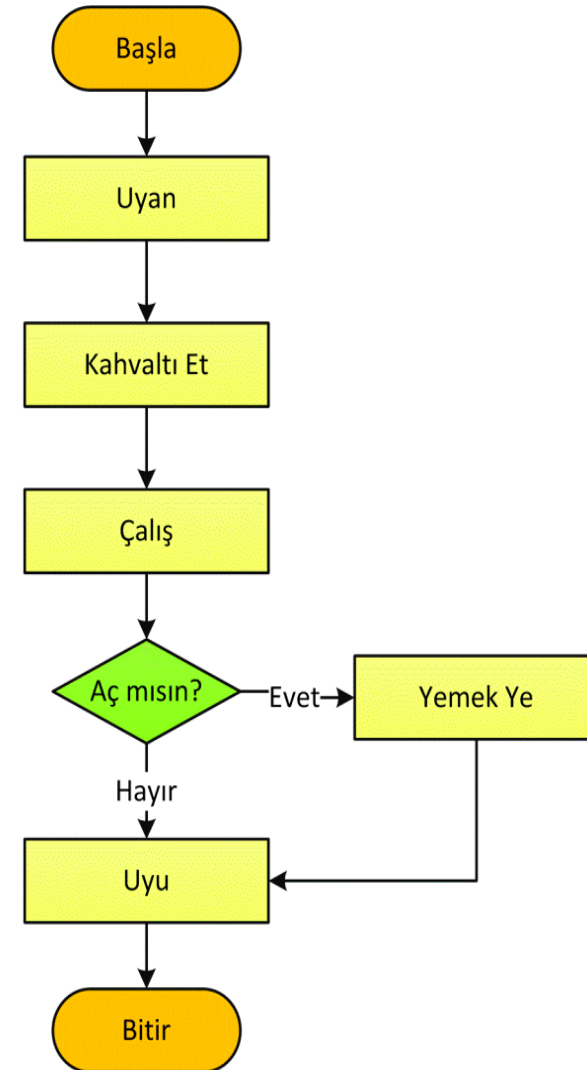
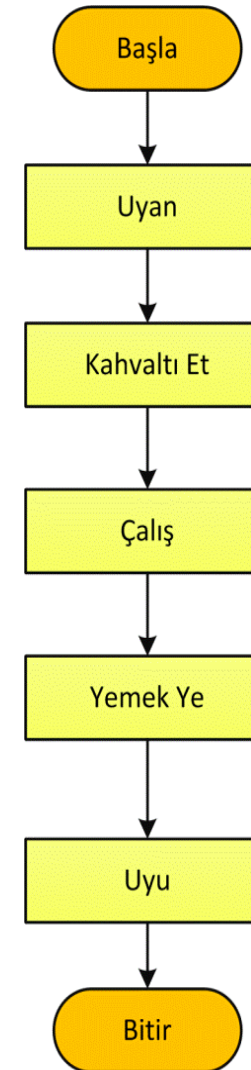
Akış Diyagramı

- Akış Diyagramı (flowchart), bir işi gerçekleştirmek üzere yapılması gereken işlem adımlarının görsel olarak ifade edilmesinde kullanılan bir modeldir.
- Her bir Akış Diyagramı, bir başlangıç noktası ile bitiş noktası arasındaki tüm adımları içerir.
- Akış diyagramlarında algoritmanın her bir işlem bir simge ile ifade edilir ve işlemler arasındaki önce – sonra ilişkisi oklar kullanılarak ifade edilir.

Simge	Simge Adı	Açıklama
	Akış Yönü	Okun yönü, bağlı olduğu işlemler arasında zaman olarak önce – sonra ilişkisini ifade eder.
	Başla / Bitir	Akış Diyagramının başlangıç adımını veya bitiş adımını ifade eder.
	İşlem	Hesaplama yolu ile değer atama ve benzeri işlemlerini içerir.
	Karar Verme	Mantıksal ifade değerinin true ya da false olması durumuna bağlı olarak akışın yönünü gösterir.
	Bilgi Girişi	Okuma yolu ile değer atama işlemlerini içerir.
	Çıktı	Bir değer ekran, dosya ve benzeri bir ortama gönderilme işlemini içerir.
	Başka Akış Diyagramı Kullanma	Bağımsız olarak çizilecek başka bir Akış Diyagramını kullanma işlemini içerir.

Akış Diyagramları ile Kontrol Yapıları (Seçimler)

- Bazı programlarda kontrol yapıları kullanarak karar alma ve seçim yapma komutları kullanabiliriz:



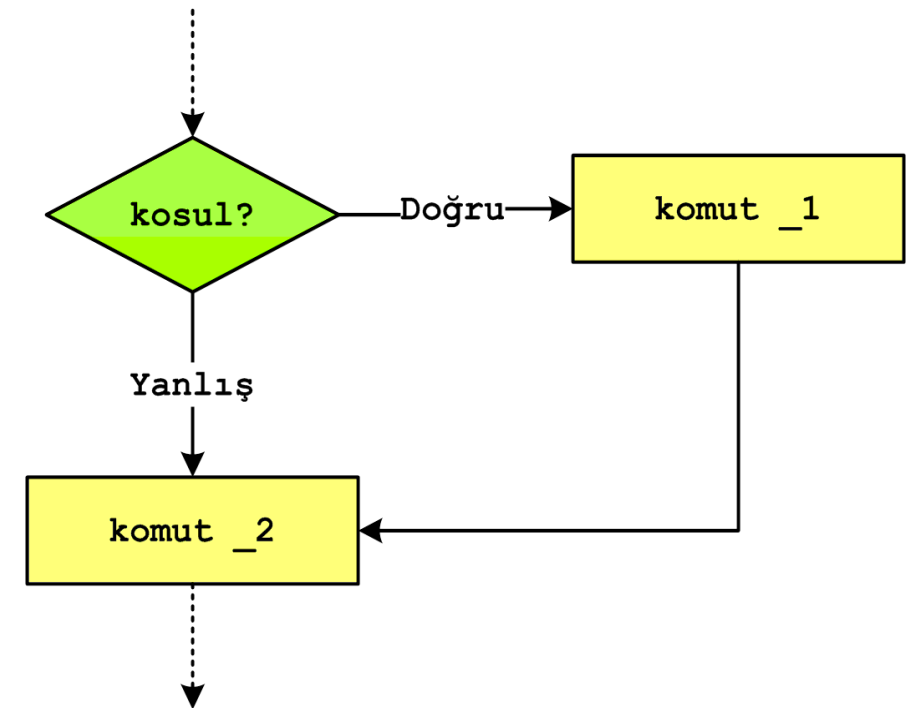
Kontrol Yapı Türleri

- IF Kontrol Yapısı (Öyle ise)
- IF-ELSE Kontrol Yapısı (Öyle ise – Değil ise)
- SWITCH Kontrol Yapısı (Seçim)
- Koşullu Operatör (Koşula Bağlı Değer Ataması)

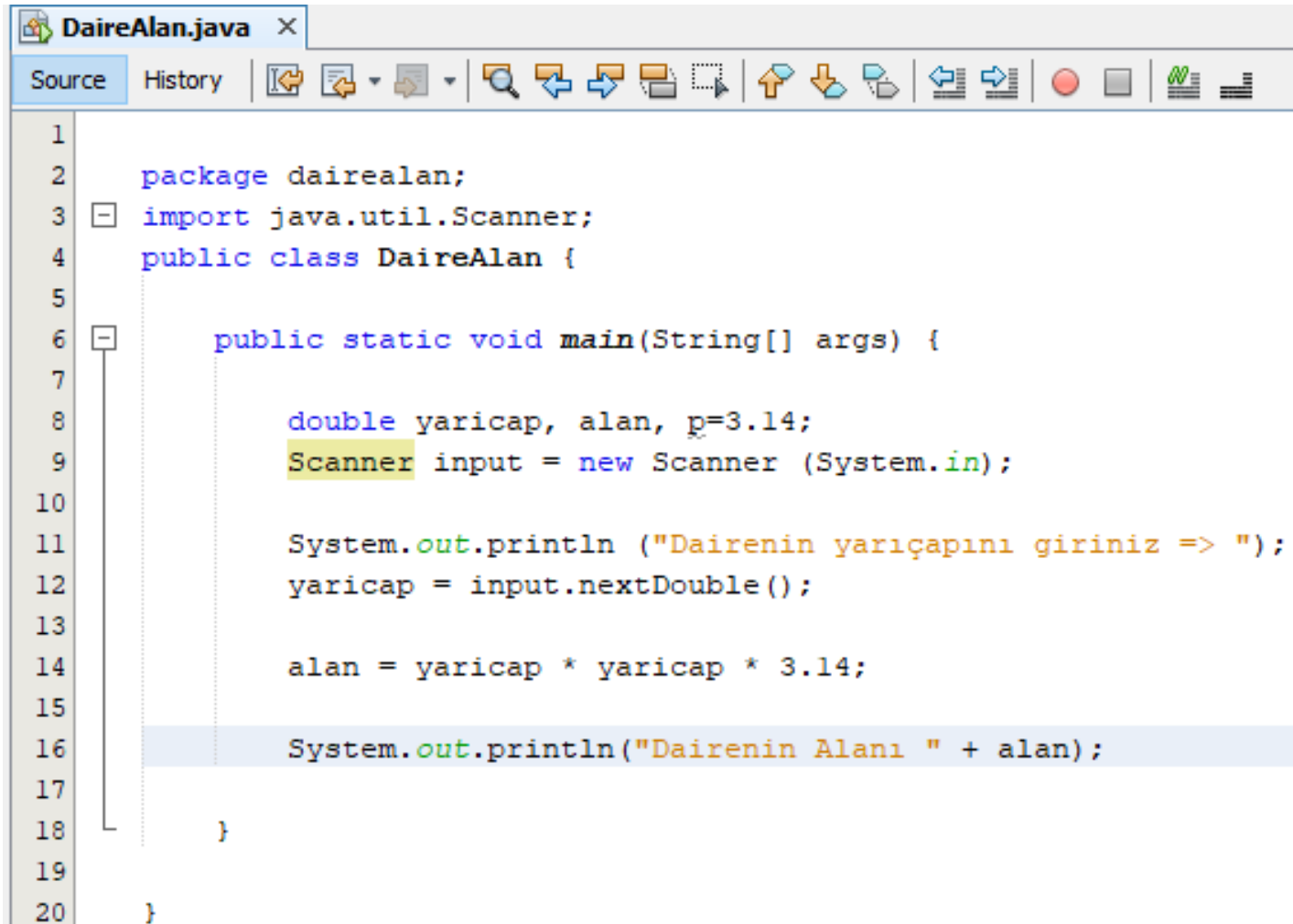
IF Kontrol Yapısı

- Parantez içerisindeki koşulun **true** olması durumunda ilgili komut gerçekleştirilir; **false** olması durumunda komut uygulanmaz
- Tek yönlü **if** kontrol ifadeleri şu şekilde kullanılır:

```
if (kosul) {  
    komut(lar);  
}
```



Hatırlayalım: DaireAlan.java



```
1
2 package dairealan;
3 import java.util.Scanner;
4 public class DaireAlan {
5
6     public static void main(String[] args) {
7
8         double yaricap, alan, p=3.14;
9         Scanner input = new Scanner (System.in);
10
11         System.out.println ("Dairenin yarıçapını giriniz => ");
12         yaricap = input.nextDouble();
13
14         alan = yaricap * yaricap * 3.14;
15
16         System.out.println("Dairenin Alanı " + alan);
17
18     }
19
20 }
```

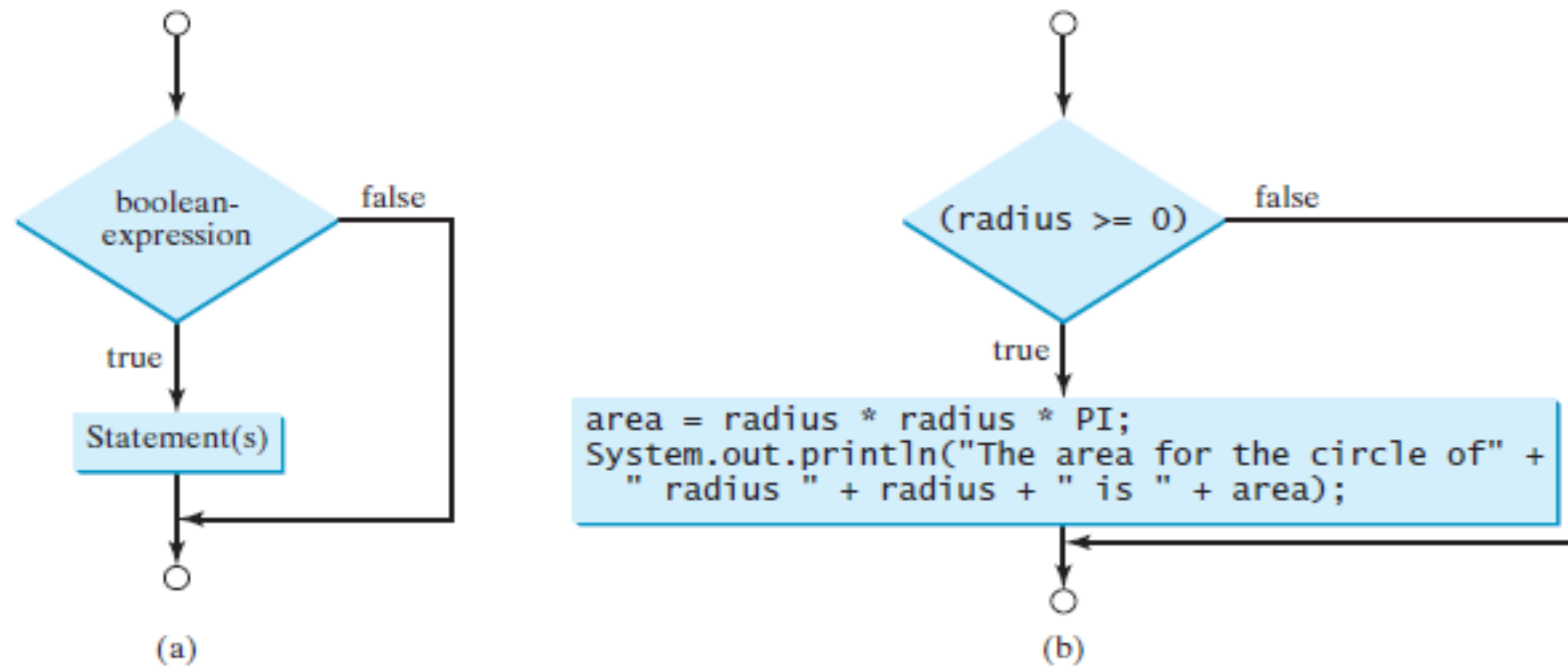


FIGURE 3.1 An **if** statement executes statements if the **boolean-expression** evaluates to **true**.

```

Output - DaireAlan (run)  X  DaireAlan.java  X
run:
Dairenin yarıçapını giriniz =>
6
Dairenin Alanı 113.04
BUILD SUCCESSFUL (total time: 2 seconds)
  
```

```

Output - DaireAlan (run)  X  DaireAlan.java  X
run:
Dairenin yarıçapını giriniz =>
-3
BUILD SUCCESSFUL (total time: 3 seconds)
  
```

```

2 package dairealan;
3 import java.util.Scanner;
4 public class DaireAlan {
5
6     public static void main(String[] args) {
7
8         double yaricap, alan, p=3.14;
9         Scanner input = new Scanner (System.in);
10
11         System.out.println ("Dairenin yarıçapını giriniz => ");
12         yaricap = input.nextDouble();
13
14         if ( yaricap >= 0 ) {
15
16             alan = yaricap * yaricap * 3.14;
17
18             System.out.println("Dairenin Alanı " + alan);
19
20         } // if bloğu
21
22     } // main bloğu
23 } // class bloğu

```

```
if i > 0 {  
    System.out.println("i is positive");  
}
```

(a) Wrong

```
if (i > 0) {  
    System.out.println("i is positive");  
}
```

(b) Correct

Mantıksal ifadenin parantez içerisinde belirtilmesi gerekir.
Örneğin program (a) yanlış bir ifadedir.

```
if (i > 0) {  
    System.out.println("i is positive");  
}
```

(a)

Equivalent

```
if (i > 0)  
    System.out.println("i is positive");
```

(b)

Küme parantez özellikle birden fazla komutlar içermesi durumunda kesinlikle kullanılmalıdır. Tek bir satır komut söz konusu ise kullanılmayabilir.

```

2 package calisma;
3 import java.util.Scanner;
4 public class Calisma {
5
6     public static void main(String[] args) {
7
8         int sayi;
9         Scanner input = new Scanner (System.in);
10
11         System.out.println ("Bir sayı giriniz => ");
12         sayi = input.nextInt();
13
14         if ( sayi > 0 ) {
15             System.out.println("Pozitif bir sayı girdiniz");
16         }
17
18         if ( sayi < 0 ) {
19             System.out.println("Negatif bir sayı girdiniz");
20         }
21
22         if ( sayi % 5 == 0 ) {
23             System.out.println("5'in katı olan bir sayı girdiniz");
24         }
25
26         if ( sayi % 2 == 0 ) {
27             System.out.println("2'nin katı olan bir sayı girdiniz; Çift sayı !! ");
28         }
29
30     } // main bloğu
31 } // class bloğu

```

Output - Calisma (run) X Calisma.java X

```

run:
Bir sayı giriniz =>
-5
Negatif bir sayı girdiniz
5'in katı olan bir sayı girdiniz
BUILD SUCCESSFUL (total time: 7 seconds)

```

Output - Calisma (run) X Calisma.java X

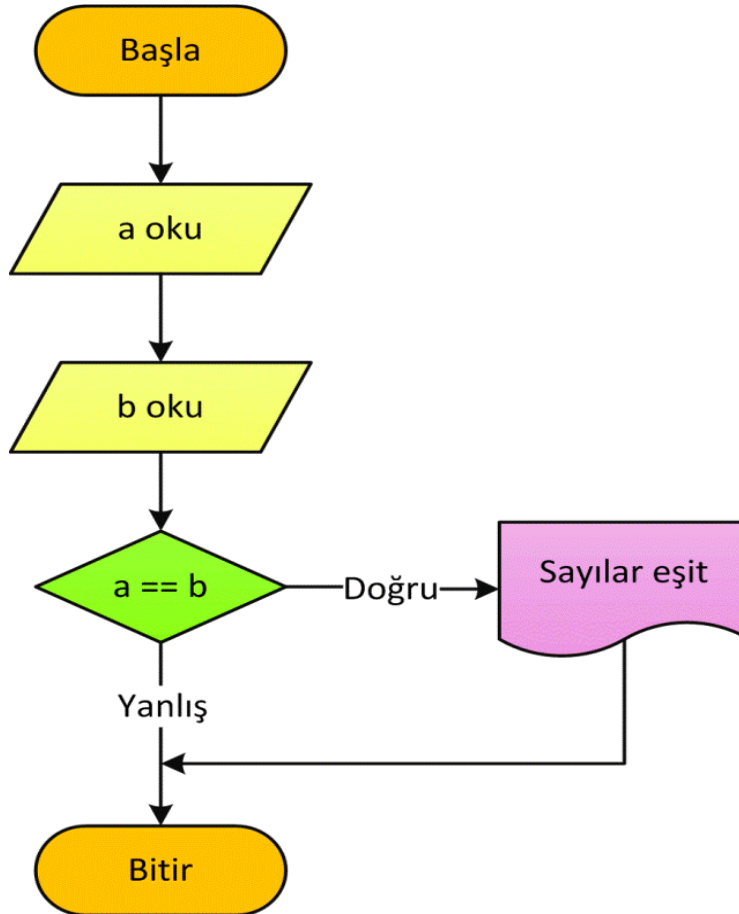
```

run:
Bir sayı giriniz =>
8
Pozitif bir sayı girdiniz
2'nin katı olan bir sayı girdiniz; Çift sayı !!
BUILD SUCCESSFUL (total time: 2 seconds)

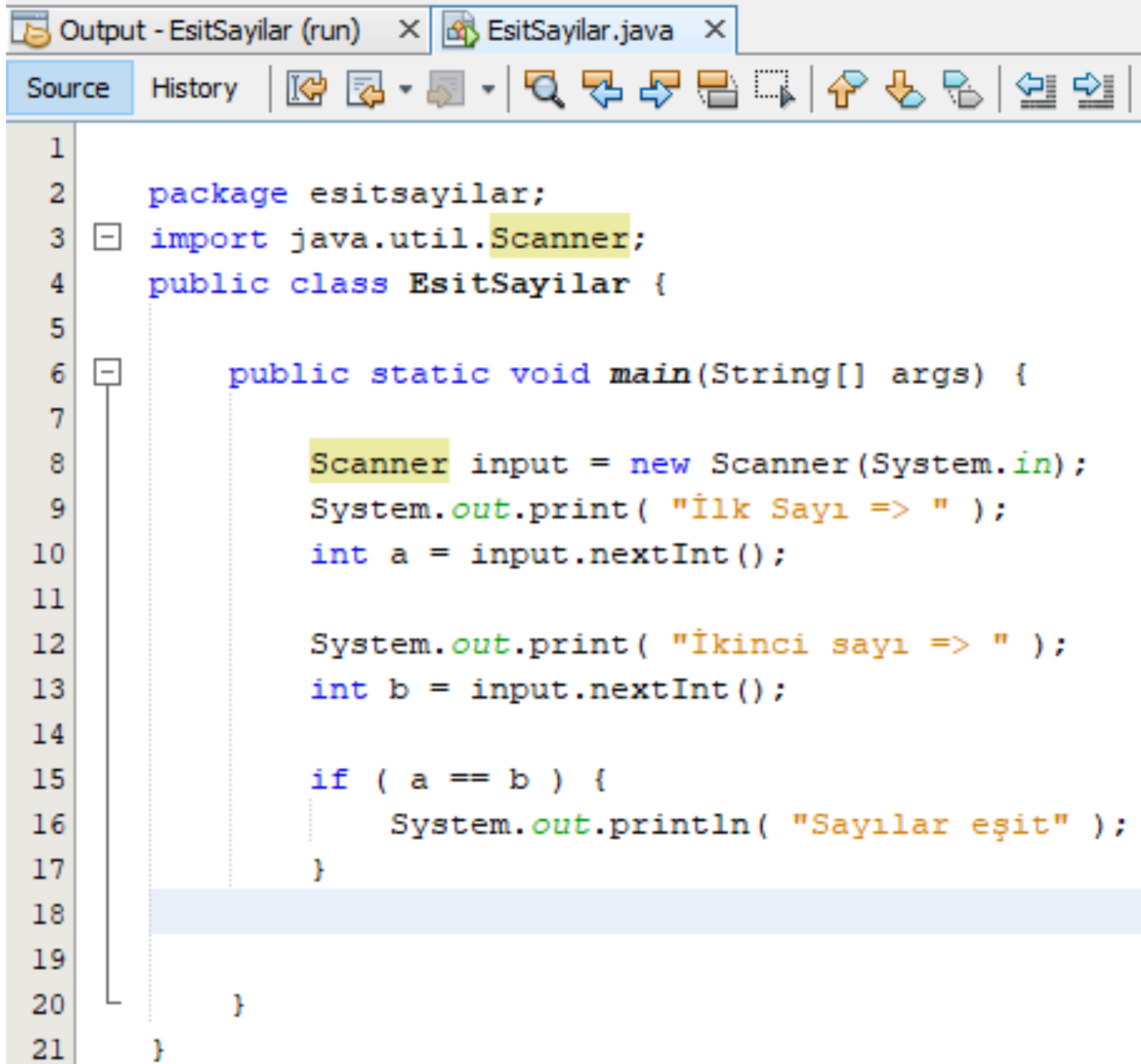
```

Şimdi, Akış Diyagramını Programlayalım...

EsitSayilar.java



```
run:
İlk Sayı => 5
İkinci sayı => 5
Sayılar eşit
BUILD SUCCESSFUL (total time: 3 seconds)
```



```
1 package esitsayilar;
2
3 import java.util.Scanner;
4 public class EsitSayilar {
5
6     public static void main(String[] args) {
7
8         Scanner input = new Scanner(System.in);
9         System.out.print( "İlk Sayı => " );
10        int a = input.nextInt();
11
12        System.out.print( "İkinci sayı => " );
13        int b = input.nextInt();
14
15        if ( a == b ) {
16            System.out.println( "Sayılar eşit" );
17        }
18
19
20    }
21 }
```



BuyukSayi.java

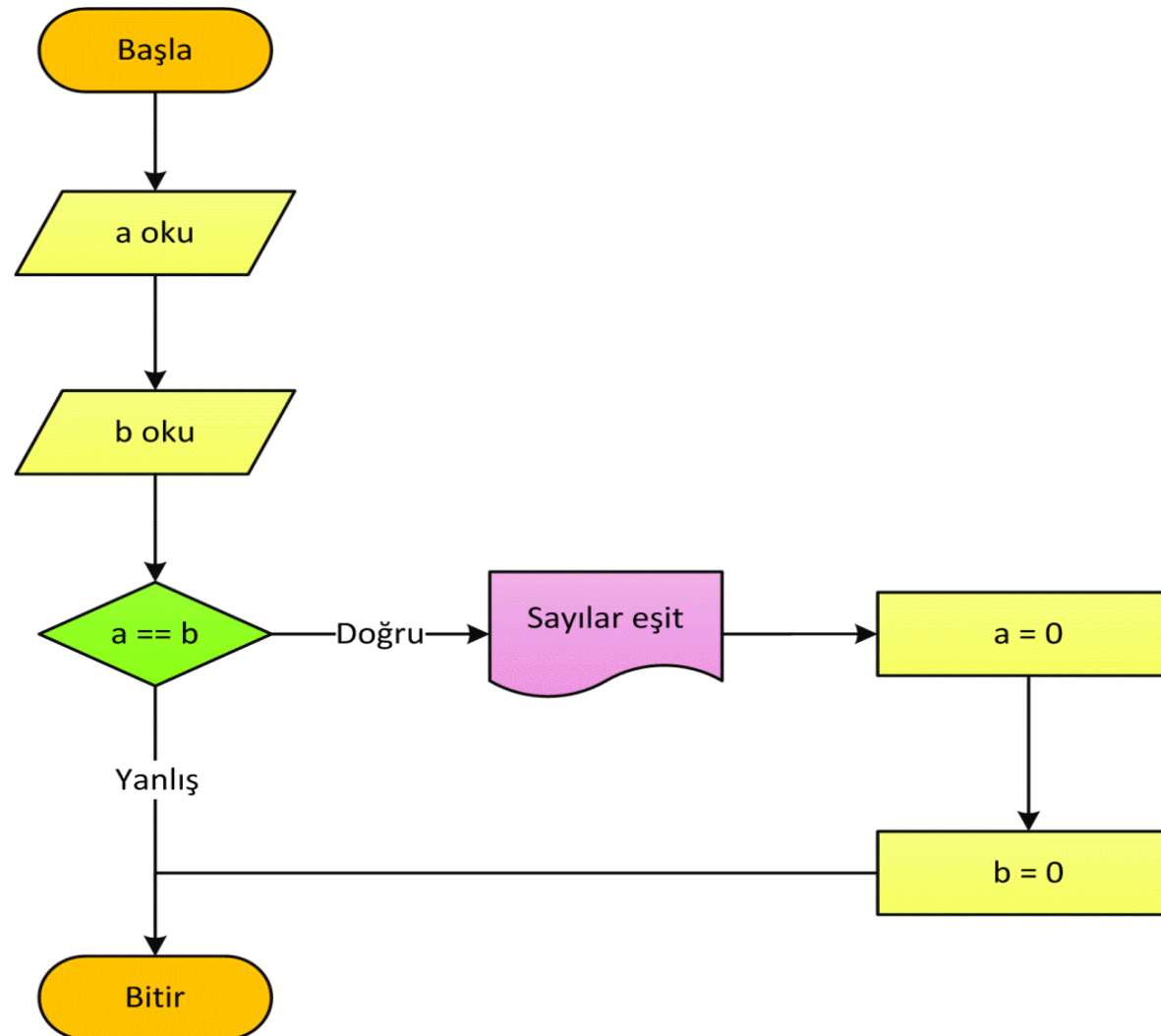


```
Output - BuyukSayi (run) X BuyukSayi.java X
run:
İlk Sayı => 6
İkinci sayı => 3
Büyük olan sayı:6
Küçük olan sayı:3
BUILD SUCCESSFUL (total time: 2 seconds)
```

```
Output - BuyukSayi (run) X BuyukSayi.java X
run:
İlk Sayı => 3
İkinci sayı => 3
Sayılar eşit
BUILD SUCCESSFUL (total time: 5 seconds)
```

```
Output - BuyukSayi (run) x BuyukSayi.java x
Source History
1
2 package buyuksayi;
3
4 import java.util.Scanner;
5
6 public class BuyukSayi {
7
8     public static void main(String[] args) {
9
10
11         Scanner input = new Scanner(System.in);
12         System.out.print( "İlk Sayı => " );
13         int a = input.nextInt();
14
15         System.out.print( "İkinci sayı => " );
16         int b = input.nextInt();
17
18         if ( a > b ) {
19             System.out.println( "Büyük olan sayı:" + a );
20             System.out.println( "Küçük olan sayı:" + b );
21         }
22
23         if ( a < b ) {
24             System.out.println( "Büyük olan sayı:" + b );
25             System.out.println( "Küçük olan sayı:" + a );
26         }
27
28         if ( a == b ) {
29             System.out.println( "Sayılar eşit" );
30         }
31
32     }
33
34 }
```

Akış Diyagramını Programlayalım...





```
import java.util.Scanner;
01 public class Sifirla {
02     public static void main( String[] args )    {
03         Scanner input = new Scanner(System.in);
04         System.out.print( "İlk sayı => " );
05         int a = input.nextInt();
06         System.out.print( "İkinci sayı => " );
07         int b = input.nextInt();
08         if ( a == b ) {
09             System.out.print( "Sayılar eşit" );
10             a = 0;
11             b = 0;
12         }
13     }
14 }
15
```

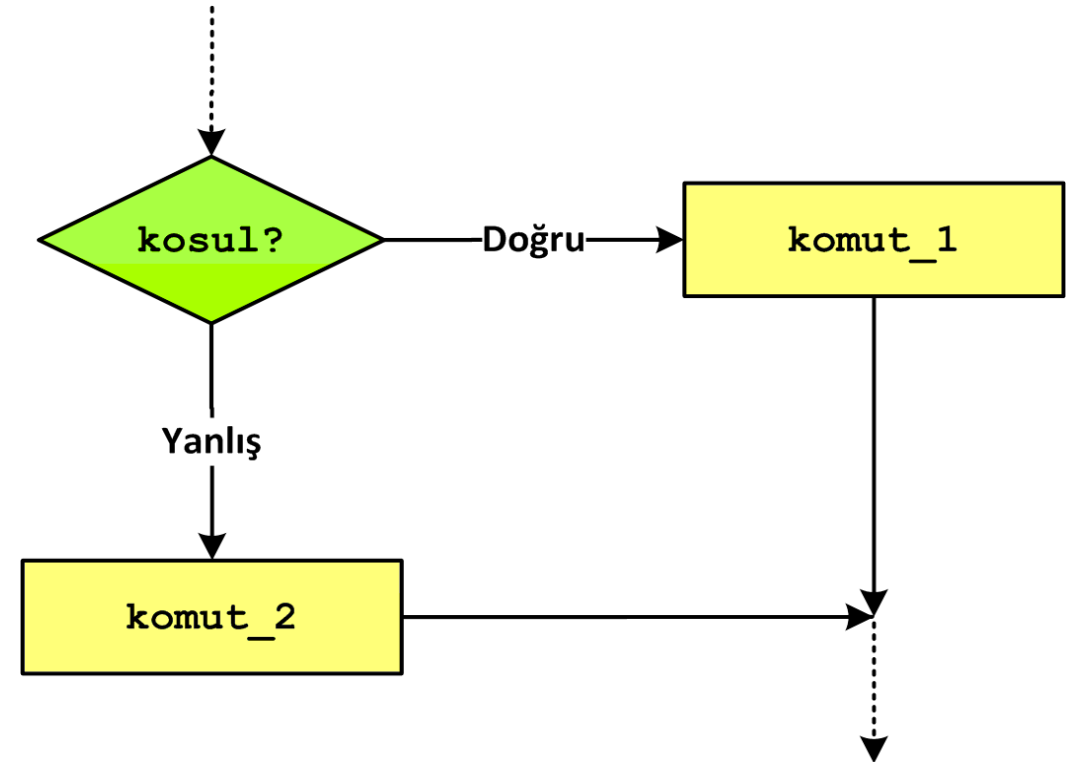
Küme parantezleri blok yapısı oluşturmak için kullanılır; birden çok komut olduğu için zorunludur

IF-ELSE Kontrol Yapısı

- Bir koşul sağlandığında bir komutun, sağlanmadığında başka bir komutun çalıştırılmasını sağlamak üzere IF-ELSE kontrol yapısı tanımlanmıştır.
- if-else kontrol ifadeleri şu şekilde kullanılır:

```
if (kosul) {  
    komut_1;  
}
```

```
else {  
    komut_2;  
}
```



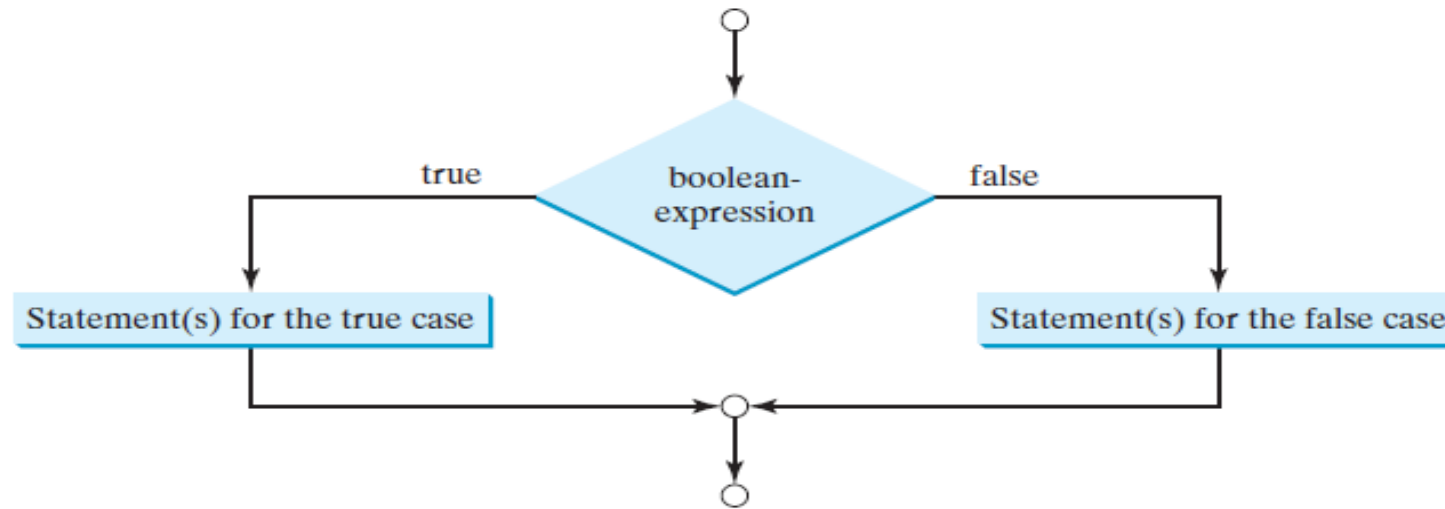
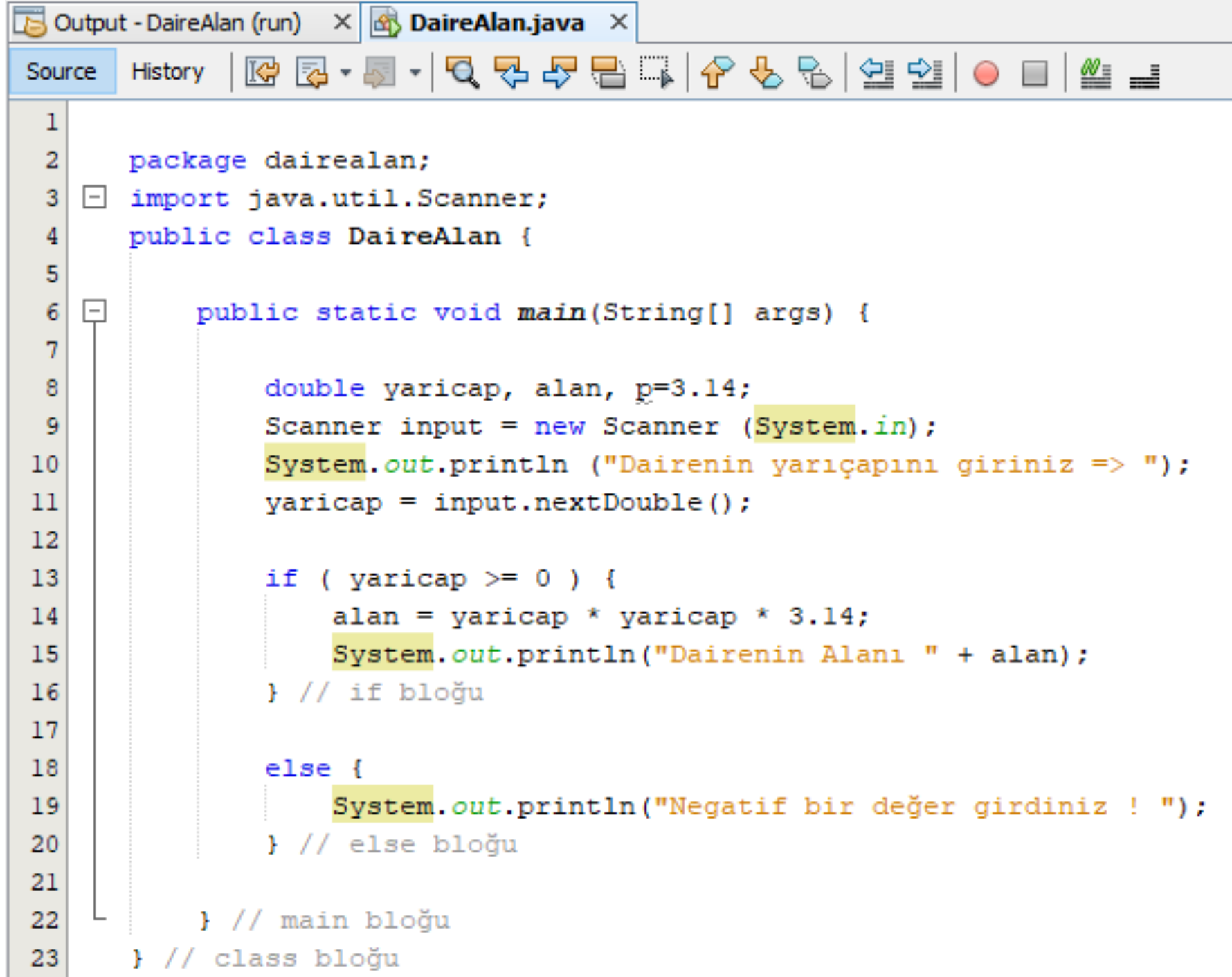


FIGURE 3.2 An **if-else** statement executes statements for the true case if the **Boolean-expression** evaluates to **true**; otherwise, statements for the false case are executed.

```
run:
Dairenin yarıçapını giriniz =>
6
Dairenin Alanı 113.04
BUILD SUCCESSFUL (total time: 1 second)
```

```
run:
Dairenin yarıçapını giriniz =>
-6
Negatif bir değer girdiniz !
BUILD SUCCESSFUL (total time: 4 seconds)
```

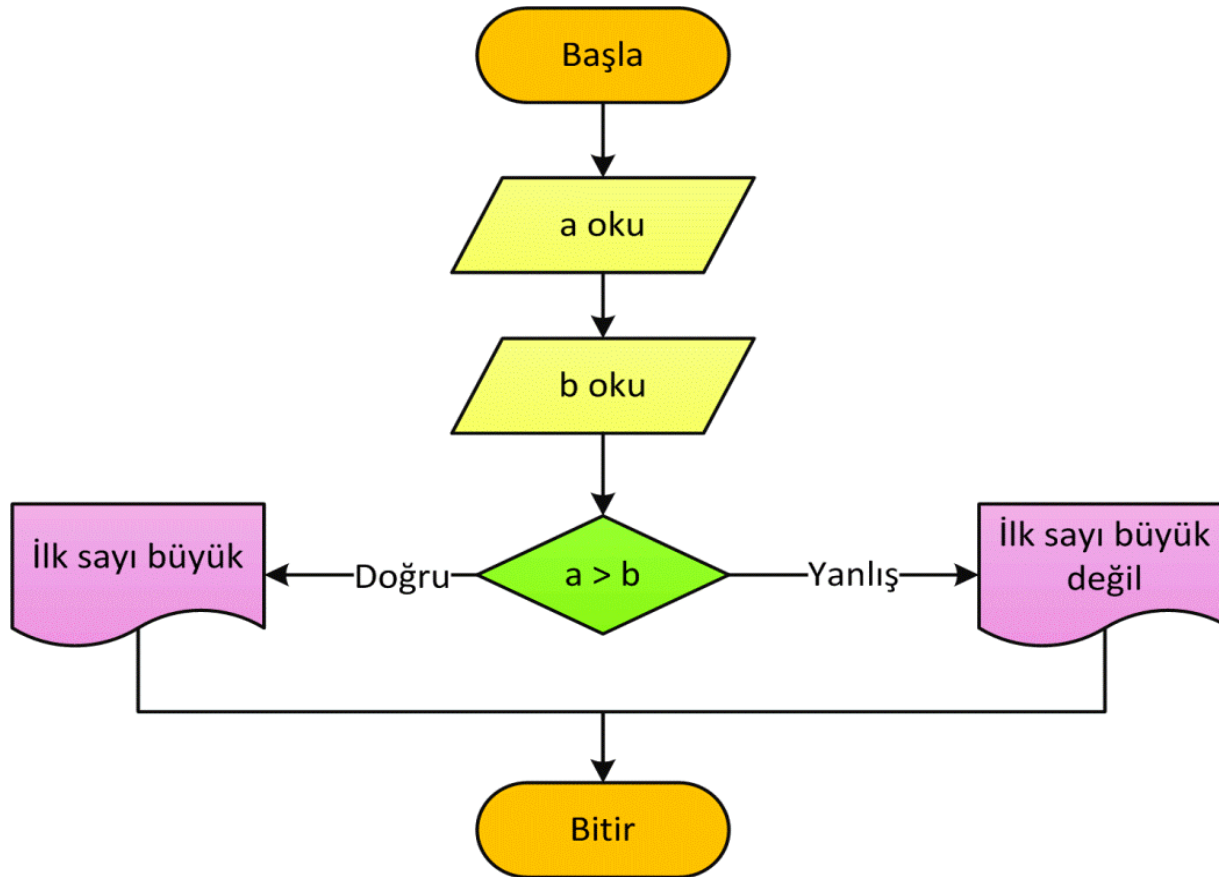


```
1
2 package dairealan;
3 import java.util.Scanner;
4 public class DaireAlan {
5
6     public static void main(String[] args) {
7
8         double yaricap, alan, p=3.14;
9         Scanner input = new Scanner (System.in);
10        System.out.println ("Dairenin yarıçapını giriniz => ");
11        yaricap = input.nextDouble();
12
13        if ( yaricap >= 0 ) {
14            alan = yaricap * yaricap * 3.14;
15            System.out.println("Dairenin Alanı " + alan);
16        } // if bloğu
17
18        else {
19            System.out.println("Negatif bir değer girdiniz ! ");
20        } // else bloğu
21
22    } // main bloğu
23 }
```



Akış Diyagramını Programlayalım...

DO IT NOW!

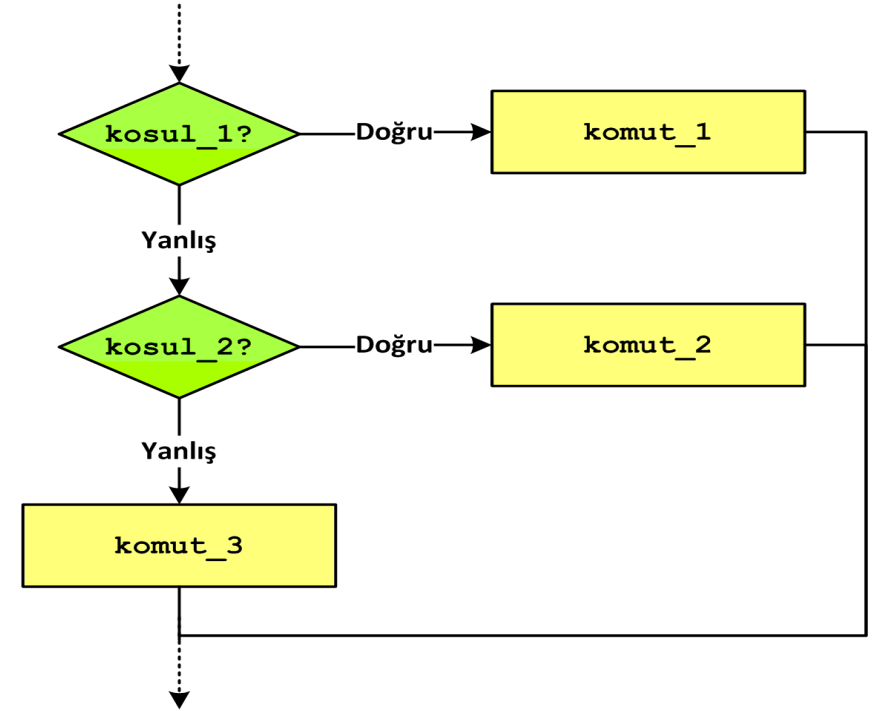


```
BuyukSayi.java x
Source History
1
2 package buyuksayi;
3 import java.util.Scanner;
4 public class BuyukSayi {
5     public static void main(String[] args) {
6         Scanner input = new Scanner(System.in);
7         System.out.print( "İlk Sayı => " );
8         int a = input.nextInt();
9
10        System.out.print( "İkinci sayı => " );
11        int b = input.nextInt();
12
13        if ( a > b ) {
14            System.out.println( "İlk sayı büyük " );
15        }
16
17        else {
18            System.out.println( "İlk sayı büyük değil" );
19        }
20
21    }
22 }
```


İç içe IF ve Çoklu IF-ELSE (IF-ELSE-IF)

```
01 if ( kosul_1 )
02     komut_1;
03 else if (kosul_2 )
04     komut_2;
05 else
06     komut_3;
```

IF-ELSE-IF yapısı, aslında Java dilinde tanımlı yeni bir yapı değildir; IF-ELSE yapısındaki ELSE'in hemen peşine yeni bir IF-ELSE eklenmesi ile oluşmaktadır. Daha fazla IF yapısı da benzer şekilde kullanılabilir.



```
01 if ( kosul_1 )
02     komut_1;
03 else if ( kosul_2 )
04     komut_2;
05 else if ( kosul_3 )
06     komut_3;
07 else
08     komut_n;
```

İç içe IF kontrol yapısı kullanmanın sınırı yoktur 😊

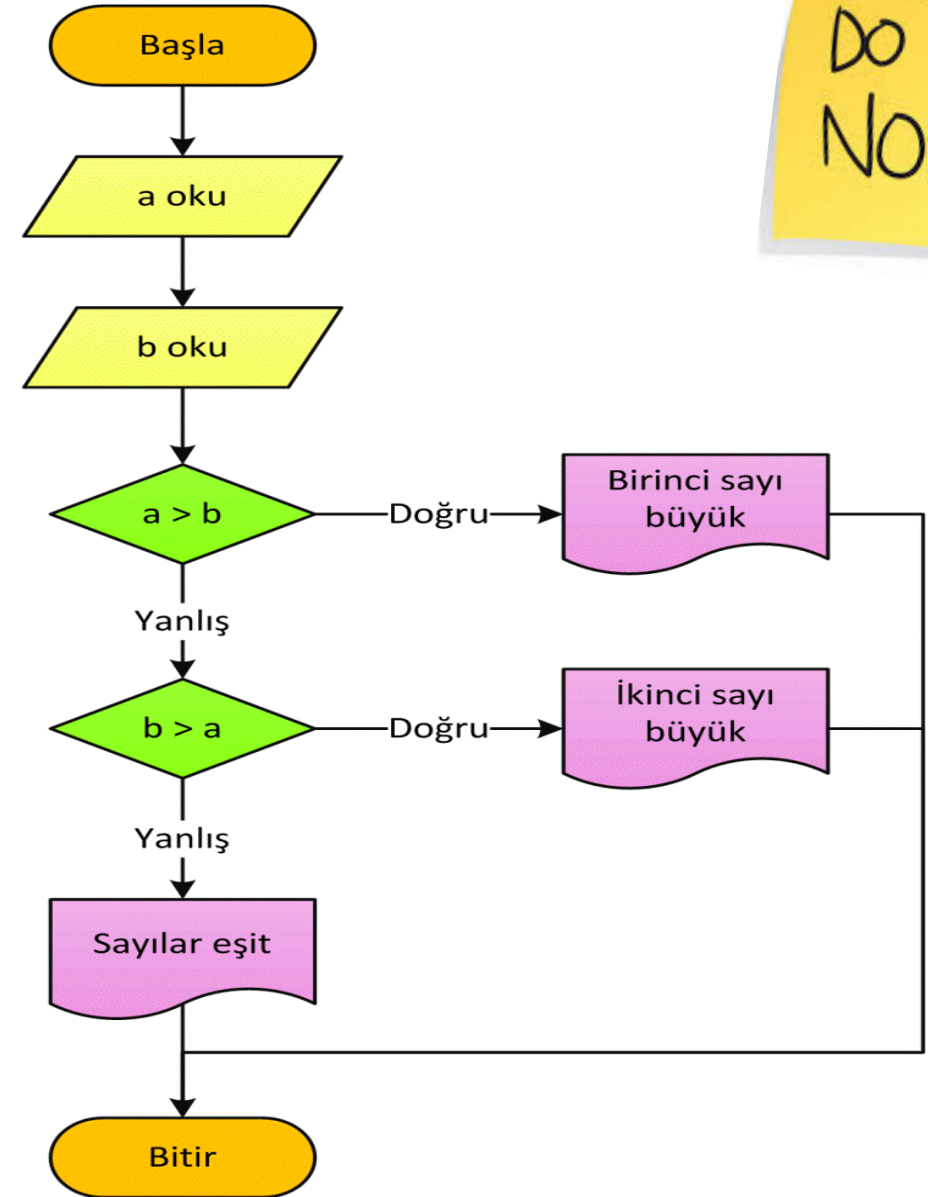
```
01 if ( kosul_1 ) {  
02     komut_1;  
03 }  
04 else if ( kosul_2 )  
05     komut_2;  
06 else {  
07     komut_3;  
08     komut_4;  
09 }
```

Birden çok komut içeriyorsa, küme parantezi kullanmayı unutmayınız



DO IT
NOW!

```
01 import java.util.Scanner;
02 public class SayiKarsilastir {
03     public static void main( String[] args ) {
04         Scanner input = new Scanner(System.in);
05         int a;
06         int b;
07         System.out.print( "İlk sayı => " );
08         a = input.nextInt();
09         System.out.print( "İkinci sayı => " );
10         b = input.nextInt();
11
12         if ( a > b )
13             System.out.print( "Birinci sayı büyük" );
14
15         else if ( a < b )
16             System.out.print( "İkinci sayı büyük" );
17
18         else
19             System.out.print( "Sayılar eşit" );
20     }
21 }
```



```
if (score >= 90.0)
    System.out.print("A");
else
    if (score >= 80.0)
        System.out.print("B");
    else
        if (score >= 70.0)
            System.out.print("C");
        else
            if (score >= 60.0)
                System.out.print("D");
            else
                System.out.print("F");
```

(a)

Equivalent

This is better

```
if (score >= 90.0)
    System.out.print("A");
else if (score >= 80.0)
    System.out.print("B");
else if (score >= 70.0)
    System.out.print("C");
else if (score >= 60.0)
    System.out.print("D");
else
    System.out.print("F");
```

(b)

FIGURE 3.3 A preferred format for multiple alternatives is shown in (b) using a multi-way **if-else** statement.

Temel Hatalar

(1) Parantez kullanımını Unutma

```
if (radius >= 0)
    area = radius * radius * PI;
    System.out.println("The area "
        + " is " + area);
```

(a) Wrong

```
if (radius >= 0) {
    area = radius * radius * PI;
    System.out.println("The area "
        + " is " + area);
}
```

(b) Correct

(2) if ifadesinin sonuna (;) kullanma

Logic error

```
if (radius >= 0);
{
    area = radius * radius * PI;
    System.out.println("The area "
        + " is " + area);
}
```

(a)

Equivalent

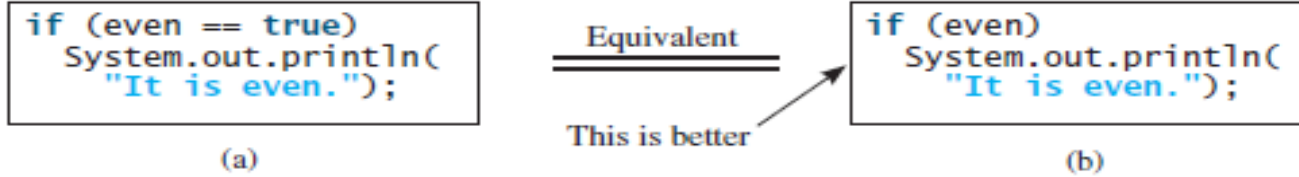
Empty block

```
if (radius >= 0) {};
{
    area = radius * radius * PI;
    System.out.println("The area "
        + " is " + area);
}
```

(b)

(3) Bir boolean Değişkeni Gereksiz Yere Koşul içerisinde Karşılaştırma

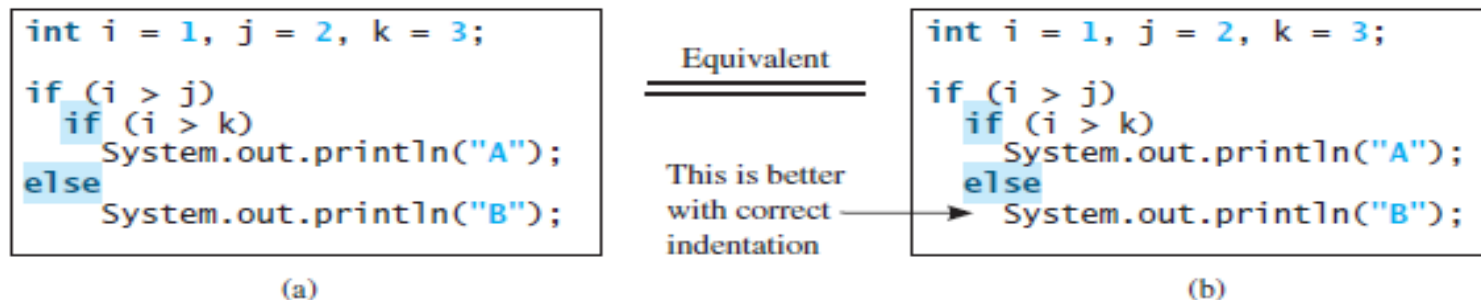
To test whether a **boolean** variable is **true** or **false** in a test condition, it is redundant to use the equality testing operator like the code in (a):



(4) Nereye ait olduğu belli olmayan else ifadeleri

The code in (a) below has two **if** clauses and one **else** clause. Which **if** clause is matched by the **else** clause? The indentation indicates that the **else** clause matches the first **if** clause.

However, the **else** clause actually matches the second **if** clause. This situation is known as the *dangling else ambiguity*. The **else** clause always matches the most recent unmatched **if** clause in the same block. So, the statement in (a) is equivalent to the code in (b).



Java'da Özel Anlamalı Kelimeler

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Any Questions?