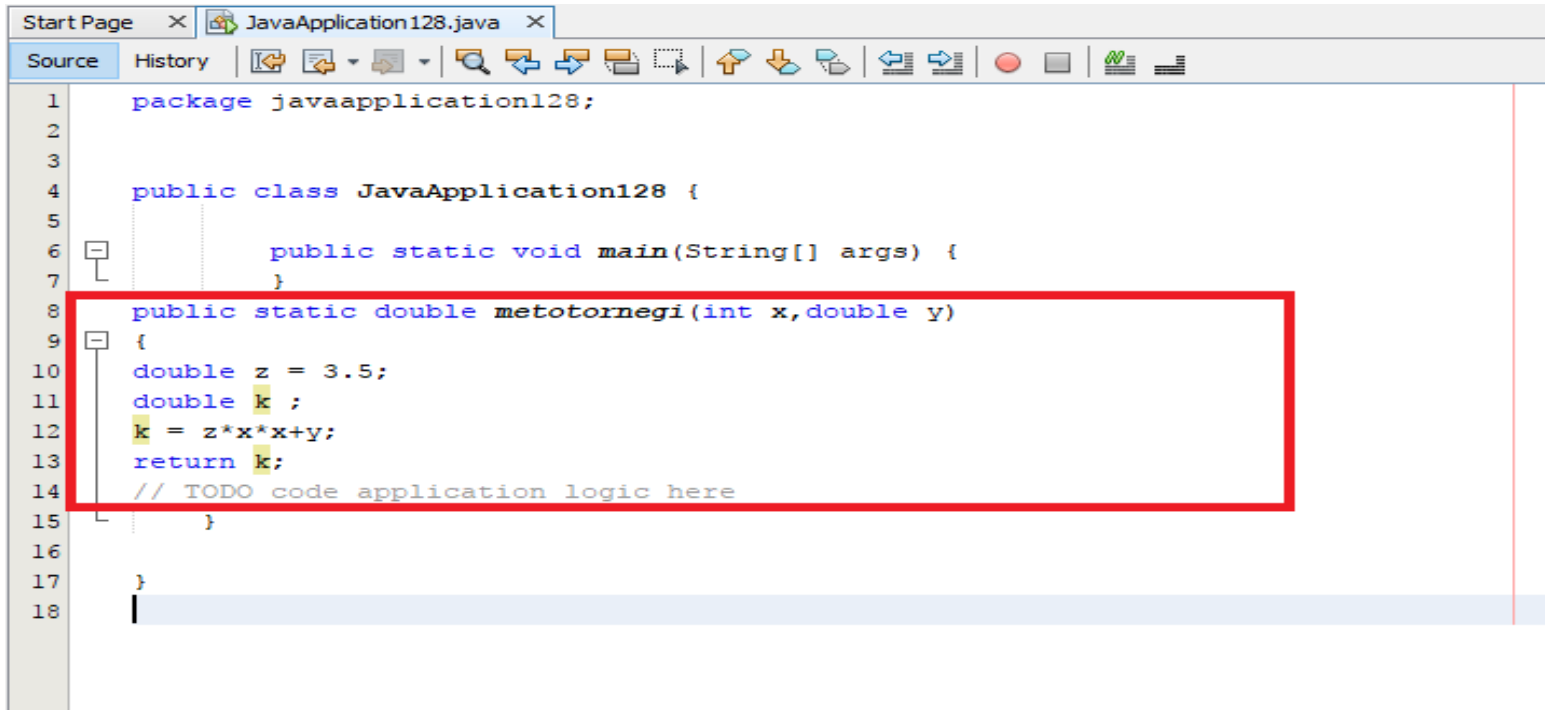
METOTLAR (METHODS):

- Metotlar java programlarının ana parçalarıdır.
- Metotlar sınıfların(class) içinde yer alan küçük program parçacıklarıdır.
- Metotların çoğunda değişken parametreler metotlar ve sınıflar arasında iletişimi sağlarlar.
- Her metotun kendine özgü değişkenleri de vardır.
- Metot yapısının ana sebebi programları modüler hale getirmektir.
- Aynı zamanda aynı program parçacığının tekrarlanmasını önlemeyi de sağlar.
- Her metot çağrıldığı proğram parçacığına belli bir değişkeni aktarabilir.
- Metotların tanımlarında aktardıkları değişken türü tanımlanır.
- Eğer metot hiçbir değişken aktarmıyorsa void sözcüğüyle tanımlanır.

- Metotların genel tanımı aşağıdaki gibidir. Parantez içindeki terimler kullanılmıyabilir.

Genel Metot tanımı

(public) (static) sınıf değişken türü sınıf ismi(sınıf değişken girdi listesi)
{
sınıf içinde geçerli degişken tanımları
Metotun ana gövdesi
return **metot çıktı değişkeni**
}



- ✓ Bu metotdaki x ve y gerçek (double) değişkenleri metotun girdi değişkenleridir.
- ✓ k ve z değişkenleri metotun yerel değişkenleridir ve bu metot dışında tanımları yoktur.
- ✓ Metot **k** değişkeninin değerini çıktı olarak metotun dışına aktarmaktadır.

ÖRNEK 1:

Start Page | MainForm.java | Person.java | Personel.java | Person.java | JavaApplication23

Source | History

```
4   public class JavaApplication27 {
5
6
7       public static void main(String[] args) {
8         tam();
9         eksik();
10        yarim();
11      }
12       public static void tam(){
13           System.out.println("fghfghfghf");
14       }
15      public static void yarim(){
16          double y=20.15;
17          System.out.println(y);
18      } public static void eksik(){
19          int x=20;
20          System.out.println(x);
21  }
22  }
23
24
```

javaapplication27.JavaApplication27 > eksik >

Output - JavaApplication27 (run)

```
run:
fghfghfghf
20
20.15
BUILD SUCCESSFUL (total time: 1 second)
```

Start Page | NewJFrame.java | Exampl1.java | JavaApplication6.java | JavaApplication7.java | JavaApp

Source | History

```
1
2   package javaapplication8;
3
4
5   public class JavaApplication8 {
6
7
8       public static void main(String[] args) {
9           ilk ();
10          son ();
11          first();
12          last();
13      }
14     public static void ilk() {
15          System.out.print("Caner ");
16      } public static void son() {
17          System.out.println("Koç ");
18      }public static void first() {
19          System.out.println(" kars ");
20      }public static void last() {
21          System.out.println(" türkiye ");
22      }
23  }
24
```

javaapplication8.JavaApplication8 >

Output - JavaApplication8 (run)

```
run:
Caner Koç
 kars
 türkiye
BUILD SUCCESSFUL (total time: 0 seconds)
```

ÖRNEK 2:

Start Page | JavaApplication129.java

Source | History

```
1
2    package javaapplication129;
3
4
5    public class JavaApplication129 {
6
7        public static void main(String[] args) {
8
9            ornek ();
10
11           // TODO code application logic here
12       }
13       public static double ornek (){
14           int x=5;
15           int y=10;
16           double z;
17           z=x*y+10;
18           System.out.println("z:"+z);
19           return z;
20
21       }
22   }
23
```

javaapplication129.JavaApplication129

Output - JavaApplication129 (run)

```
run:
z:60.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

ÖRNEK 3:

- Metot oluşturarak dışarıdan 2 ondalıklı değer ürettirin ve çarpımlarını yazdırın.

ÖRNEK 3:

Start Page | MainForm.java | Person.java | Personel.java | Person.java | JavaApplication23.java

Source | History

```
1
2   package javaapplication27;
3   import java.util.Scanner;
4
5   public class JavaApplication27 {
6
7
8       public static void main(String[] args) {
9        times();
10          }
11      public static void times(){
12       double a; double b;
13          Scanner deger= new Scanner(System.in);
14          Scanner deger2 = new Scanner(System.in);
15          System.out.println("a değeri gir: ");
16          a = deger.nextDouble();
17          System.out.println("b değeri gir: ");
18          b = deger2.nextDouble();
19          System.out.println(a*b);
20      }
21  }
```

javaapplication27.JavaApplication27 > times

Output - JavaApplication27 (run)

```
run:
a değeri gir:
23,2
b değeri gir:
56,4
1308.48
BUILD SUCCESSFUL (total time: 8 seconds)
```

NESNE (OBJECT) TANIMI VE METOTLARDA KULLANIMI

Dışarıdan değer çağırma için kullanılan nesne kalıbı:

Scanner input = new Scanner(System.in);

Scanner sınıfından(class) input nesnesini tanımlar.
Nesne(object) tanımı yaparken aynı metotlarda olduğu gibi değişken veya nesneleri girdi olarak tanımlanabilir.

Sınıf (class)_ismi nesne(object)_ismi;

nesne(object)_ismi = new Sınıf (class)_ismi(değişken veya nesne girdi listesi)

Sınıf(class)_ismi nesne(object_ismi = new Sınıf (class) ismi(değişken veya nesne girdi listesi)

FOR LOOP :

Yazılan programlarda bazen belli komut parçalarının birçok kez çalışması gerekebilir. Yani bazı komut parçalarının, belli şartlar altında tekrar tekrar islenmesi gerekebilir.

Döngü deyimleri, bahsedilen komut parçalarının belirtilen şartlar gerçekleştikçe tekrar tekrar islenmesini sağlar.

Java'da üç tür döngü vardir. Bunlar çogu programlama dilinde olan; "for", "do-while" ve "while" döngüleridir. Bu döngülerden for ve while döngüsünde, koşul sınaması döngüye girmeden önce yapılırken, do-while döngüsünde koşul sınaması bir kod parçası çalıştıktan sonra yapılır.

FOR: Şart kontrolü while yapısında olduğu gibi döngüye girmeden yapılır. Bu döngü yapısının diğerlerinden farklı olarak başlangıç değeri ve döngü sayacına sahip olmasıdır. Genel yapısı:

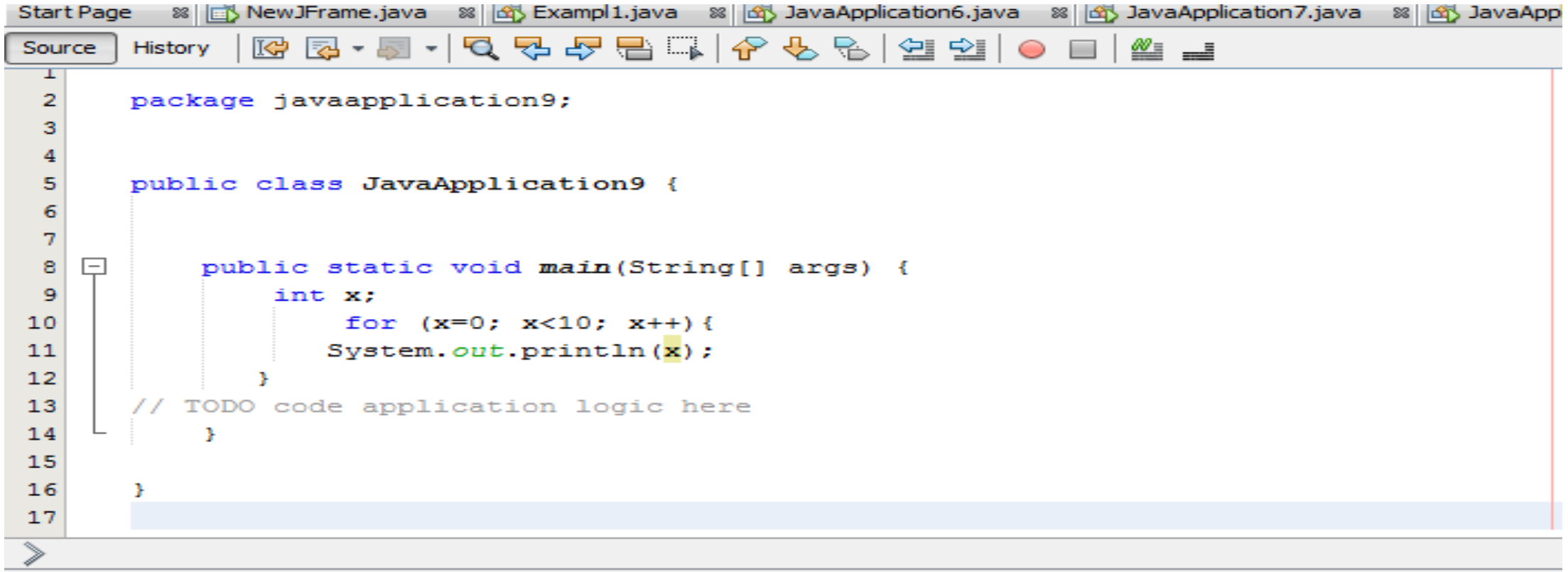
```
for (başlangıç; şart; artım)
 {
 ...
 döngüdeki komutlar;
 ...
 }
```

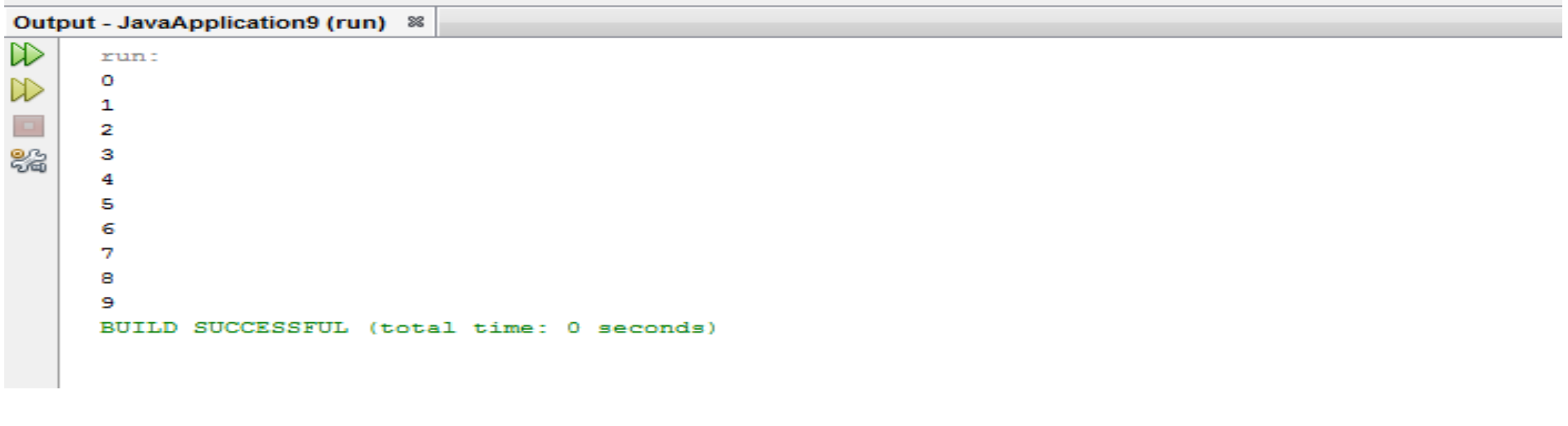
• başlangıç: Döngüye ilk defa girildiğinde burada belirtilen komut çalıştırılır. (Örn: i=1) Döngünün diğer adımlarında bu işlem tekrarlanmaz.
• şart: Döngünün tekrarı veya sonlandırılacağı buradaki şarta bağlıdır. Buradaki şart doğru olduğu sürece döngü tekrarlanır. (Örn: i<N)
• artım: Döngünün her çalıştırılması sonucu döngü değişkeni artımı burada belirtilir. (Örn: i=i+1) Bir döngü adımı bitince, bir sonraki döngüye geçilip geçilmeyeceği için şarta bakılmadan önce artım işlemi gerçekleştirilir. (önce artım → sonra şart kontrolü)

ÖRNEK : 4

- 0’den 10 a kadar ve 10 dan 0’a kadar teker teker ve çifter çifter yazacak döngüyü oluşturunuz.

```
for (int a=10; a>0; a-=2){
```


Start Page
NewJFrame.java
Exampl1.java
JavaApplication6.java
JavaApplication7.java
JavaApp
Source
History
1 package javaapplication9;
2
3 public class JavaApplication9 {
4
5
6 public static void main(String[] args) {
7 int x;
8 for (x=0; x<10; x++){
9 System.out.println(x);
10 }
11 // TODO code application logic here
12 }
13
14 }
15
16


Output - JavaApplication9 (run)
run:
0
1
2
3
4
5
6
7
8
9
BUILD SUCCESSFUL (total time: 0 seconds)

ÖRNEK 5 : birden 10 a kadar sayıları yazdıran for döngüsü:

Start Page | JavaApplication129.java

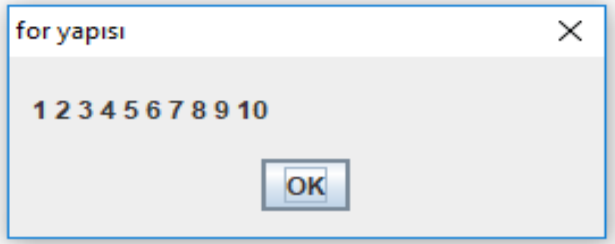
Source | History

```
1
2   package javaapplication129;
3
4
5   import javax.swing.*;
6
7   public class JavaApplication129 {
8
9       public static void main(String[] args) {
10        String s="";
11  for(int saydirici=1;saydirici <= 10; saydirici++){
12        s=s+saydirici+" ";
13  }
14  JOptionPane.showMessageDialog(null,s,
15  "for yapısı",JOptionPane.PLAIN_MESSAGE);
16
17      }
18  }
19
20
21
```

javaapplication129.JavaApplication129 > main

Output - JavaApplication129 (run)

```
run:
```



Array [ ]:

Double array [ ] = { };

Source | History

```
1
2   package array;
3
4
5   public class Array {
6
7       public static void main(String[] args) {
8
9           int hedef []= {0,3,5,4,9,78,32,-35};
10
11
12          System.out.println(hedef [3]);
13  // TODO code application loghic here
14      }
15
16  }
17
```

WHILE …. DÖNGÜSÜ

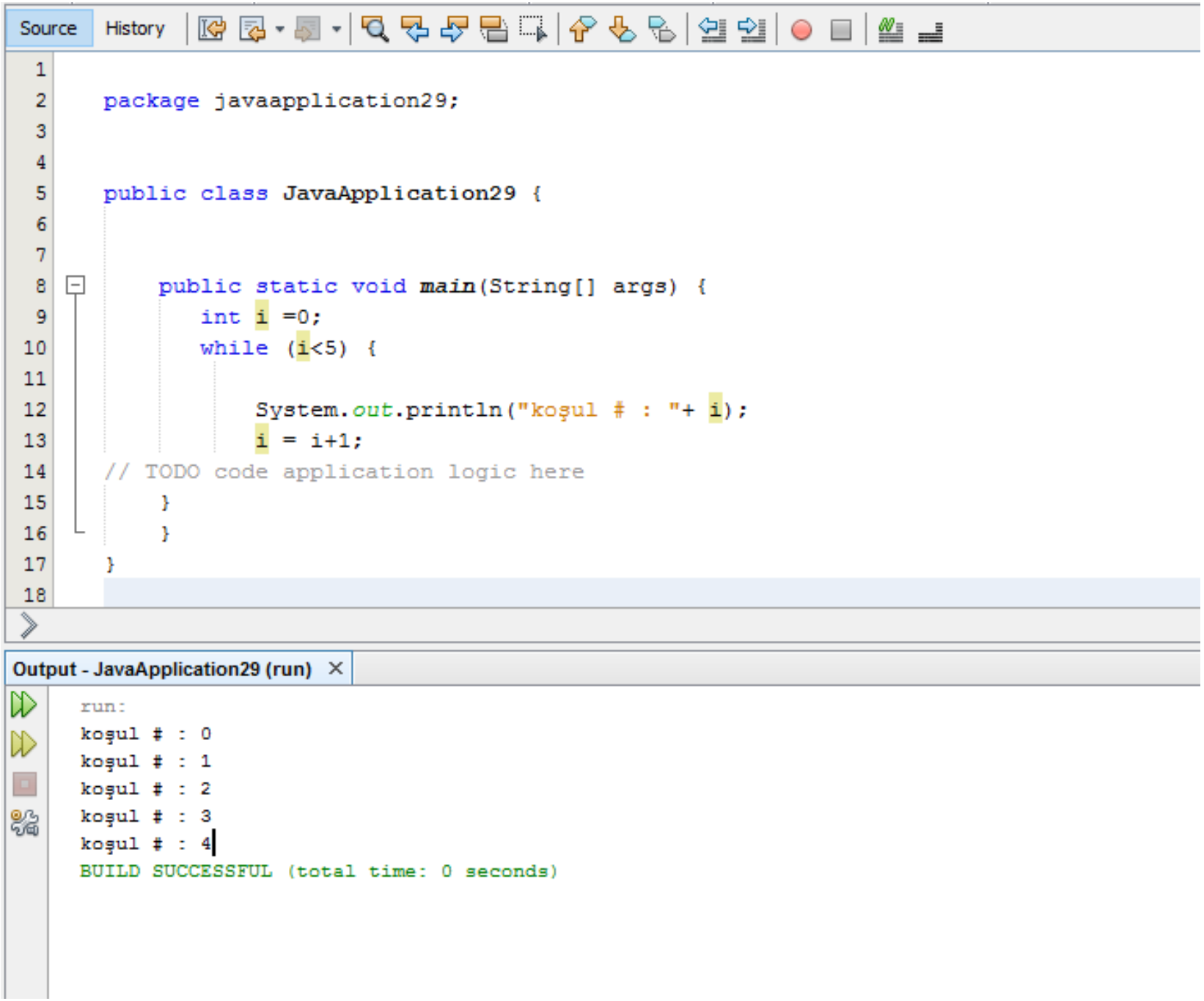
While (durum- conditions) {

Kod- Statements

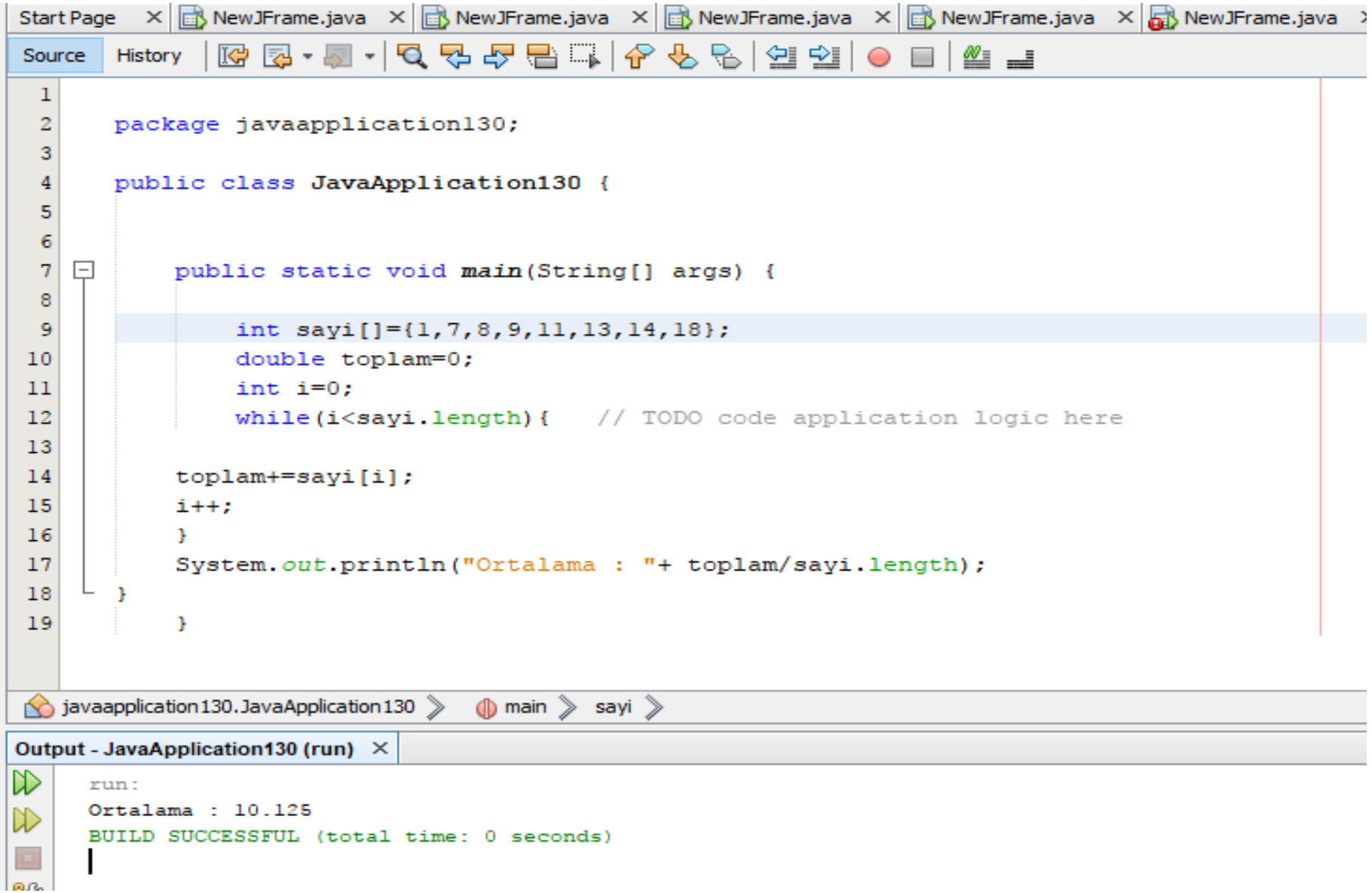
}

- While en çok kullanılan tekrarlama yapılarından biridir.
- ( ) içindeki boolean terim true (doğru) olduğu sürece yeniden işlemi döndürür. Eğer programda boolean işlemini sonuçlandıracak bir ifade yoksa sonsuza dek veya program başka bir yöntemle durdurulana dek devam eder.

ÖRNEK 5: 0 dan başlayıp 5. koşula kadar
yazdıracak kodları oluşturunuz.



ÖRNEK 6: WHILE döngüsü ile ortalama hesaplama.



ÖRNEK 6: WHILE döngüsü ile iki boyutlu değişkeni toplama.