2. Ders: JAVA ile Nesne Yönelimli Programlama Nesneler ve sınıflar (Objects and classes)

Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği Bölümü

YMH112 Algoritma ve Programlama-II

Dr. Öğr. Üyesi Yaman Akbulut

JAVA ile Nesne Yönelimli Programlama

- http://www.kriptarium.com/algoritma.html (Yardımcı kaynak)

- JAVA ile Nesne Yönelimli Programlama (Ders: 1-10) video.

    - Ders 1: Nesne Yönelimli Programlamaya Giriş (izle)
    - Ders 2: Neden Nesne Yönelimli Programlama? (izle)
    - Ders 3: Nesne (Object) Kavramı (izle)
    - Ders 4: Sınıf (Class) Kavramı (izle)
    - Ders 5: Nesne ve Sınıf Kavramları Arasındaki İlişki (izle)
    - Ders 6: Kalıtım/Miras (Inheritance) Kavramı (izle)
    - Ders 7: Çok Biçimlilik (Polymorphism) (izle)
    - Ders 8: Soyutlama (Abstraction) (izle)
    - Ders 9: Kapsülleme (Encapsulation) (izle)
    - Ders 10: Prosedür Tabanlı Programlama ile OOP Karşılaştırması (izle)

Java Keywords

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

Kaynak: Introduction to JAVA Programming 10th Edition, Y. Daniel Liang

Nesneler ve Sınıflar (Objects and Classes)

Nesne yönelimli programlama sayesinde büyük ölçekli yazılımlar ve grafik kullanıcı ara yüzleri tasarlayabilmek mümkündür.

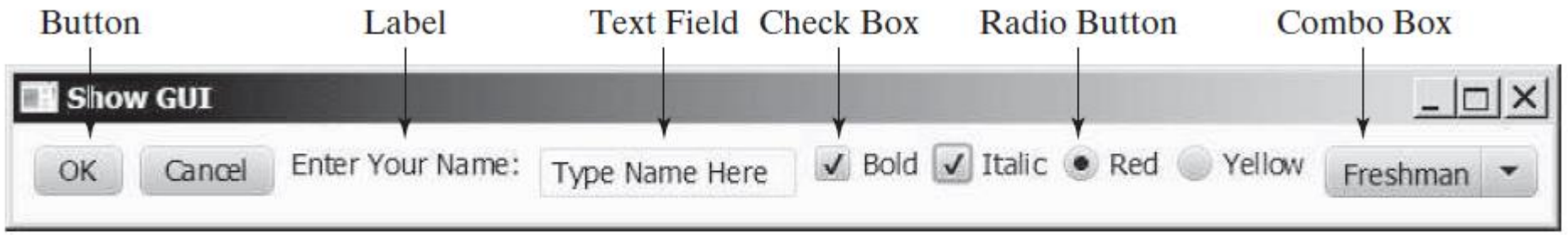


FIGURE 9.1 The GUI objects are created from classes.

Nesneler için Sınıf Tanımlama

Sınıf, bir nesnenin özelliklerini ve davranışlarını tanımlar.

Nesne yönelimli programlama nesneleri kullanarak programlamayı içerir.

Bir nesne, gerçek dünyada açıkça tanımlanabilen bir varlığı temsil eder.

Örnek olarak, bir öğrenci, bir masa, bir düğme, bir çember … bir nesnedir.

Bir nesnenin kendine has kimliği, durumu ve davranışı vardır.

Durum (Özellik veya Nitelik)

Bir nesnenin durumu (özellikleri veya nitelikleri) (properties or attributes), veri alanları ve mevcut değerlerle temsil edilir.

Çember bir nesnedir ve bu çemberi karakterize eden yarıçap veri alanı özelliğine sahiptir.

Dikdörtgen bir nesnedir. Dikdörtgen kendisini karakterize eden en ve boy özelliklerine sahiptir. En ve boy veri alanı şeklindedir ve değerleri vardır.

Davranış (Eylem)

Bir nesnenin davranışı (eylemleri) (actions), metotlarla temsil edilir.

Bir nesnenin metodunu çağırmak için nesneden eylem yapması istenir. Örnek olarak bir çember için getArea() ve getPerimeter() metotları tanımlanmış olsun.
Çember nesnesi getArea() ve getPerimeter() metotlarını çağırarak kendi alanının ve çevresinin değerini geri döndürür.

Ayrıca setRadius(yaricap) şeklinde bir metot tanımlanabilir. Çember bu metodu kullanarak kendi yaricap özelliğini değiştirebilir.

Nesne ve sınıf

Aynı türdeki nesneler, ortak bir sınıf kullanılarak tanımlanabilir.

Sınıf, bir nesnenin özelliklerinin (veri alanlarının) ve metotlarının (davranışının) ne olacağını tanımlayan bir şablon, plan veya sözleşmedir.

Bir nesne bir sınıfın örneğidir. Bir sınıfın birçok örneğini oluşturabilirsiniz. Örnek oluşturma, somutlaştırma olarak adlandırılır.

Nesne (object) ve örnek (instance) terimleri genellikle birbirinin yerine kullanılabilir.

Sınıflar ve nesneler arasındaki ilişki, bir kek tarifi ile kek arasındaki ilişkiye benzer: Tek bir tariften istediğiniz kadar kek yapabilirsiniz.

Nesne ve sınıf

Bir Java sınıfı, veri alanlarını tanımlamak için değişkenleri ve eylemleri tanımlamak için metotları kullanır.

Bir yapıcı (constructor) herhangi bir eylemi gerçekleştirebilir, ancak yapıcılar başlangıç eylemlerini gerçekleştirmek için tasarlanmıştır.

Bir nesnenin veri alanlarının başlangıç değer ataması buna örnek olarak verilebilir.

Circle sınıfı

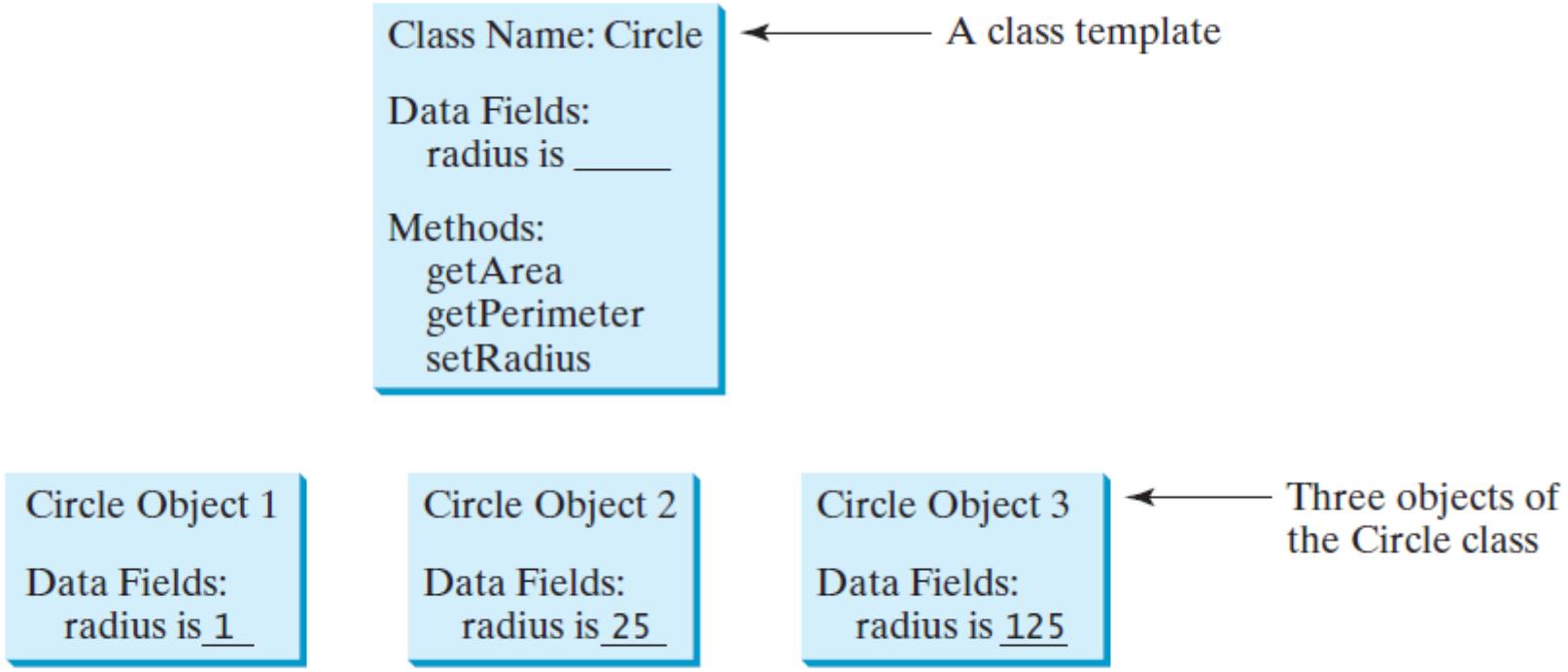


FIGURE 9.2 A class is a template for creating objects.

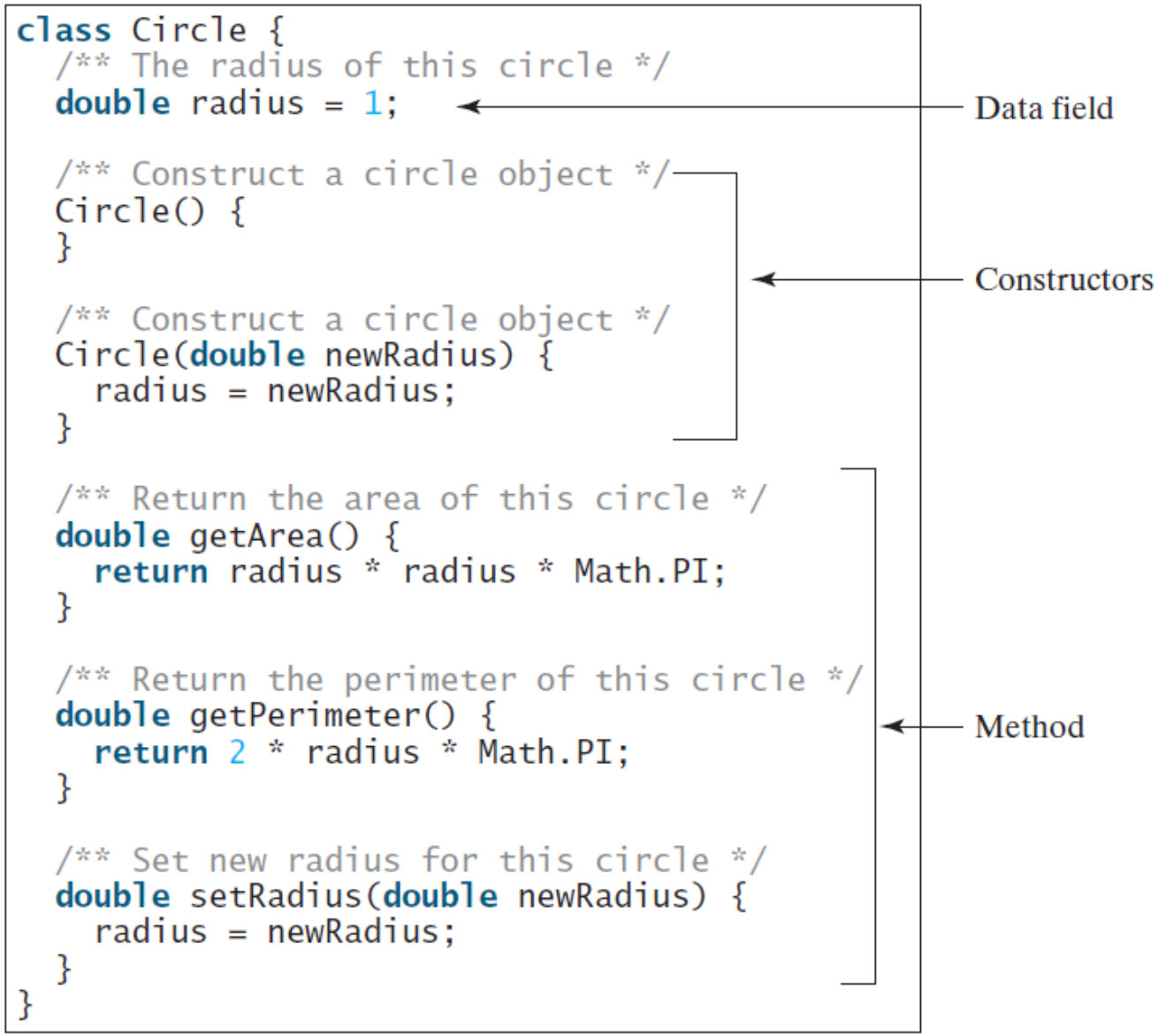


FIGURE 9.3 A class is a construct that defines objects of the same type.

UML Class Diagram

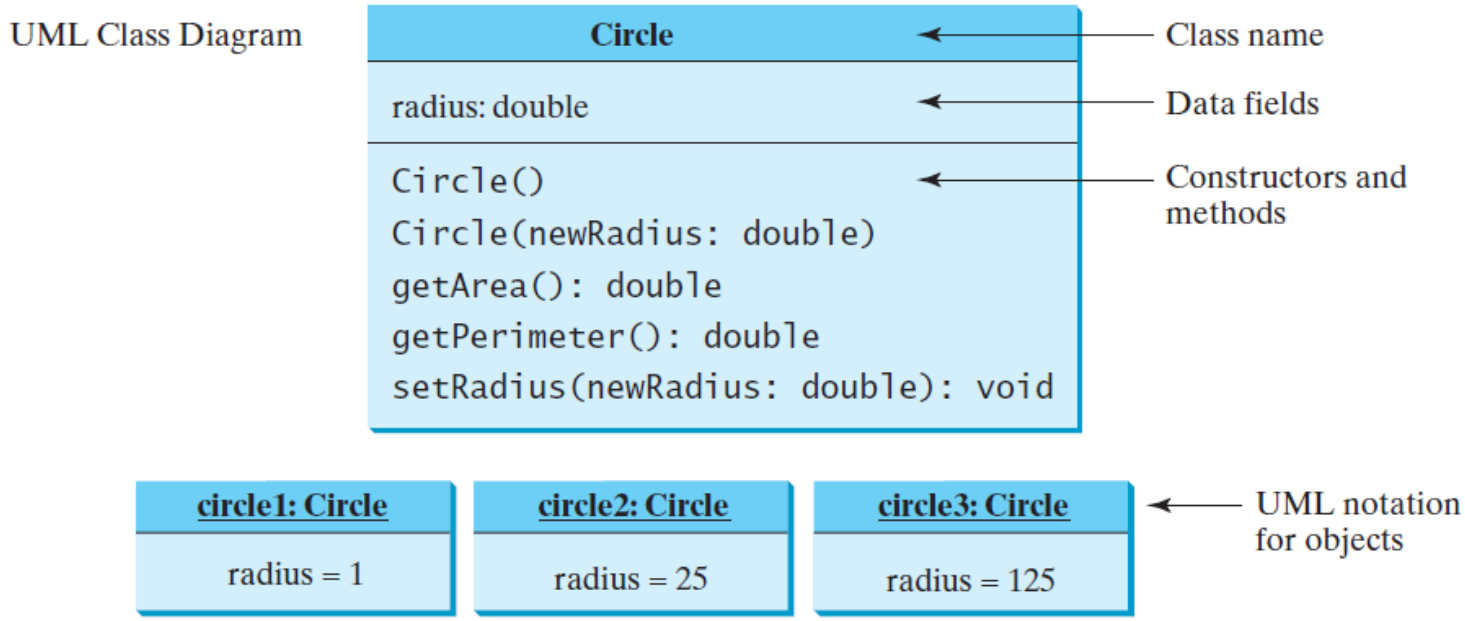


FIGURE 9.4 Classes and objects can be represented using UML notation.

the data field is denoted as

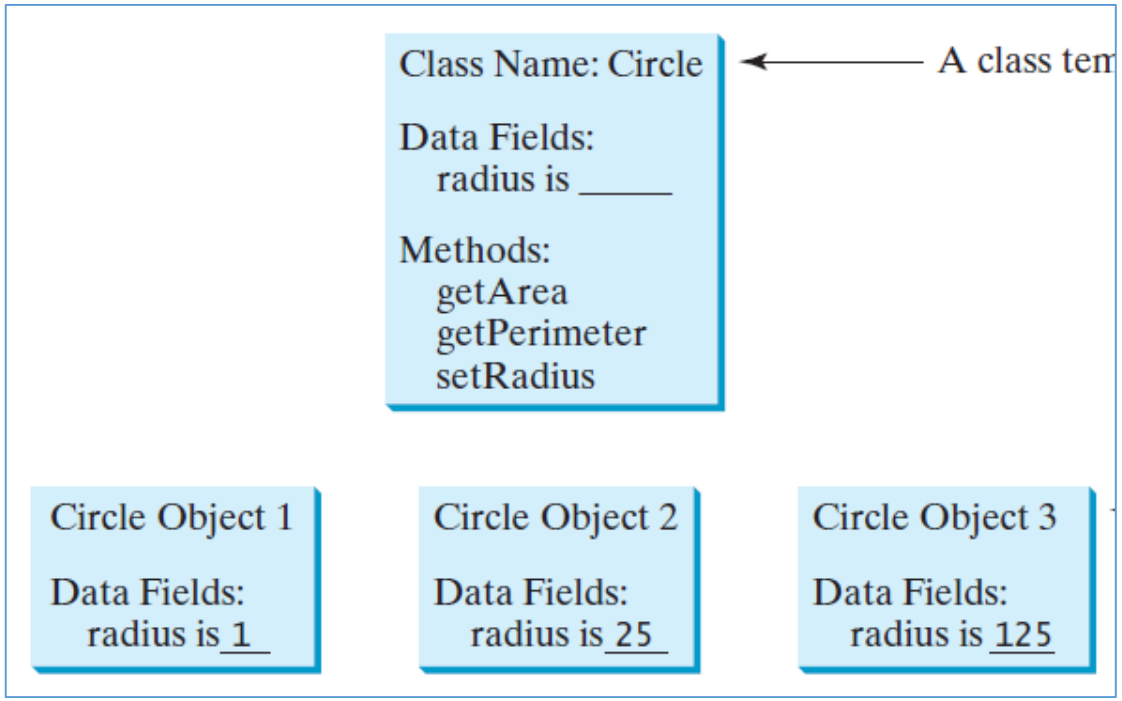
```
dataFieldName: dataFieldType
```

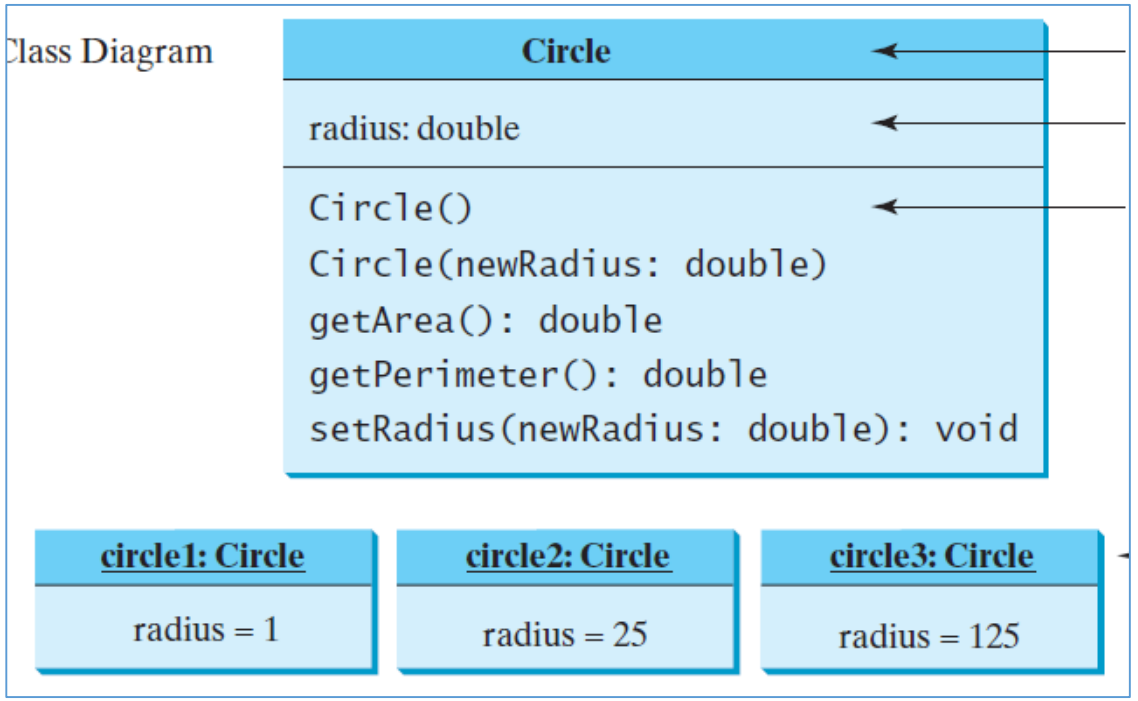
The constructor is denoted as

```
ClassName(parameterName: parameterType)
```

The method is denoted as

```
methodName(parameterName: parameterType): returnType
```





the data field is denoted as

```
dataFieldName: dataFieldType
```

The constructor is denoted as

```
ClassName(parameterName: parameterType)
```

The method is denoted as

```
methodName(parameterName: parameterType): returnType
```

Örnek 1: Sınıf tanımlama, nesne oluşturma

```
1  public class TestBasitCember {
2      /** Main metot */
3      public static void main(String[] args) {
4      // Yaricapi 1 olan cember olusturma
5      BasitCember cember1 = new BasitCember();
6      System.out.println("Yaricapi " + cember1.yaricap
7      + " olan cemberin alani: " + cember1.getAlan());
8
9      // Yaricapi 25 olan cember olusturma
10     BasitCember cember2 = new BasitCember(25);
11     System.out.println("Yaricapi " + cember2.yaricap
12     + " olan cemberin alani: " + cember2.getAlan());
13
14     // Yaricapi 125 olan cember olusturma
15     BasitCember cember3 = new BasitCember(125);
16     System.out.println("Yaricapi " + cember3.yaricap
17     + " olan cemberin alani: " + cember3.getAlan());
18
19     // Cember yaricapi degistirme
20     cember2.yaricap = 100; // veya cember2.setRadius(100)
21     System.out.println("Yaricapi " + cember2.yaricap
22     + " olan cemberin alani: " + cember2.getAlan());
23     }
24 }
25
```

```
C:\Program Files\Java\jdk-15.0.1\bin\yeni2>java TestBasitCember
Yaricapi 1.0 olan cemberin alani: 3.141592653589793
Yaricapi 25.0 olan cemberin alani: 1963.4954084936207
Yaricapi 125.0 olan cemberin alani: 49087.385212340516
Yaricapi 100.0 olan cemberin alani: 31415.926535897932
```

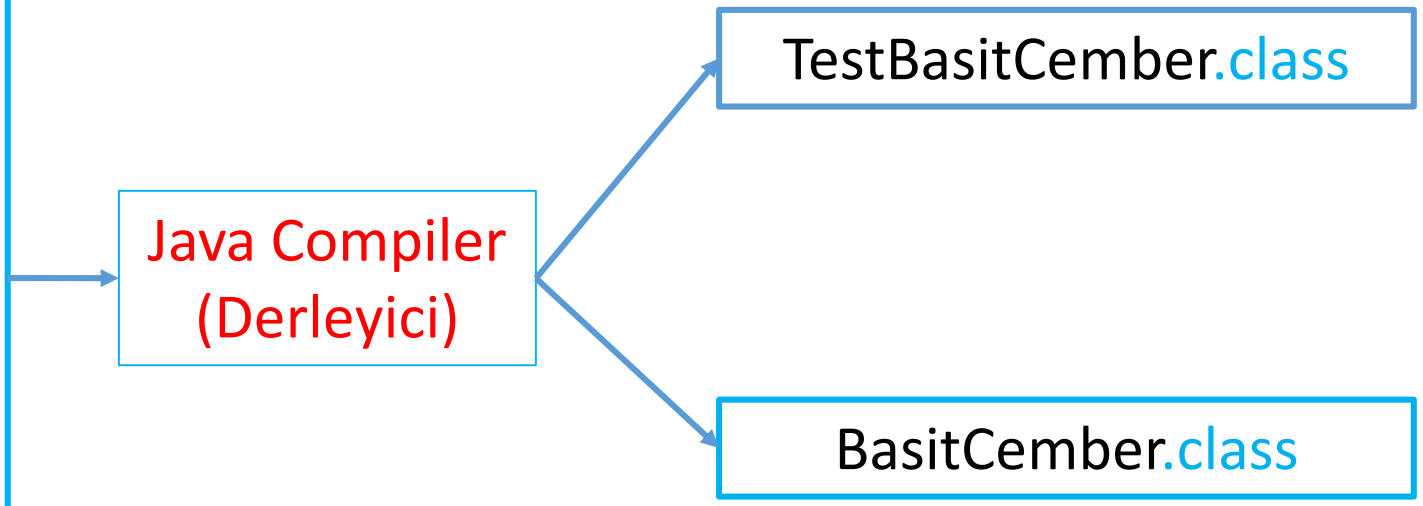
İki sınıfı tek bir dosyaya koyabiliriz, ancak dosyadaki yalnızca bir sınıf public class (genel sınıf) olabilir. Ayrıca, genel sınıf dosya adıyla aynı ada sahip olmalıdır.

Örnek 1: Sınıf tanımlama, nesne oluşturma

İki sınıfı tek bir dosyaya koyabiliriz, ancak dosyadaki yalnızca bir sınıf public class (genel sınıf) olabilir. Ayrıca, genel sınıf dosya adıyla aynı ada sahip olmalıdır.

TestBasitCember.java dosyasını derlediğimiz zaman TestBasitCember.class ve BasitCember.class adlarında iki sınıf dosyası oluşturulur (üretilir).

```
// Dosya TestBasitCember.java
public class TestBasitCember {
    ...
}

class BasitCember {
    ...
}
```



Örnek 1: Sınıf tanımlama, nesne oluşturma

```
26  // İki yapılandırıcı ile Cember sınıfı tanimalama
27  class BasitCember {
28      double yaricap;
29
30      /** Yaricapi 1 olan bir cember yapilandir */
31      BasitCember() {
32      yaricap = 1;
33      }
34
35      /** Yaricapi belirlenebilen bir cember yapilandir */
36      BasitCember(double yeniYaricap) {
37      yaricap = yeniYaricap;
38      }
39
40      /** Bu cemberin alanini geri döndür */
41      double getAlan() {
42      return yaricap * yaricap * Math.PI;
43      }
44
45      /** Bu cemberin cevresini geri döndür */
46      double getCevre() {
47      return 2 * yaricap * Math.PI;
48      }
49
50      /** Bu cember icin yeni bir yaricap ata */
51      void setYaricap(double yeniYaricap) {
52      yaricap = yeniYaricap;
53      }
54  }
```

Sınıflar, nesnelerin tanımlarıdır ve nesneler sınıflardan oluşturulur.

Örnek 1: Sınıf tanımlama, nesne oluşturma

```java
public class TestBasitCember {
    /** Main metot */
    public static void main(String[] args) {
    // Yaricapi 1 olan cember olusturma
    BasitCember cember1 = new BasitCember();
    System.out.println("Yaricapi " + cember1.yaricap
    + " olan cemberin alani: " + cember1.getAlan());

    // Yaricapi 25 olan cember olusturma
    BasitCember cember2 = new BasitCember(25);
    System.out.println("Yaricapi " + cember2.yaricap
    + " olan cemberin alani: " + cember2.getAlan());

    // Yaricapi 125 olan cember olusturma
    BasitCember cember3 = new BasitCember(125);
    System.out.println("Yaricapi " + cember3.yaricap
    + " olan cemberin alani: " + cember3.getAlan());

    // Cember yaricapi degistirme
    cember2.yaricap = 100; // veya cember2.setRadius(100)
    System.out.println("Yaricapi " + cember2.yaricap
    + " olan cemberin alani: " + cember2.getAlan());
    }
}
```

```
C:\Program Files\Java\jdk-15.0.1\bin\yeni2>java TestBasitCember
Yaricapi 1.0 olan cemberin alani: 3.141592653589793
Yaricapi 25.0 olan cemberin alani: 1963.4954084936207
Yaricapi 125.0 olan cemberin alani: 49087.385212340516
Yaricapi 100.0 olan cemberin alani: 31415.926535897932
```

```java
// İki yapılandırıcı ile Cember sınıfı tanimalama
class BasitCember {
    double yaricap;

    /** Yaricapi 1 olan bir cember yapilandir */
    BasitCember() {
    yaricap = 1;
    }

    /** Yaricapi belirlenebilen bir cember yapilandir */
    BasitCember(double yeniYaricap) {
    yaricap = yeniYaricap;
    }

    /** Bu cemberin alanini geri döndür */
    double getAlan() {
    return yaricap * yaricap * Math.PI;
    }

    /** Bu cemberin cevresini geri döndür */
    double getCevre() {
    return 2 * yaricap * Math.PI;
    }

    /** Bu cember icin yeni bir yaricap ata */
    void setYaricap(double yeniYaricap) {
    yaricap = yeniYaricap;
    }
}
```

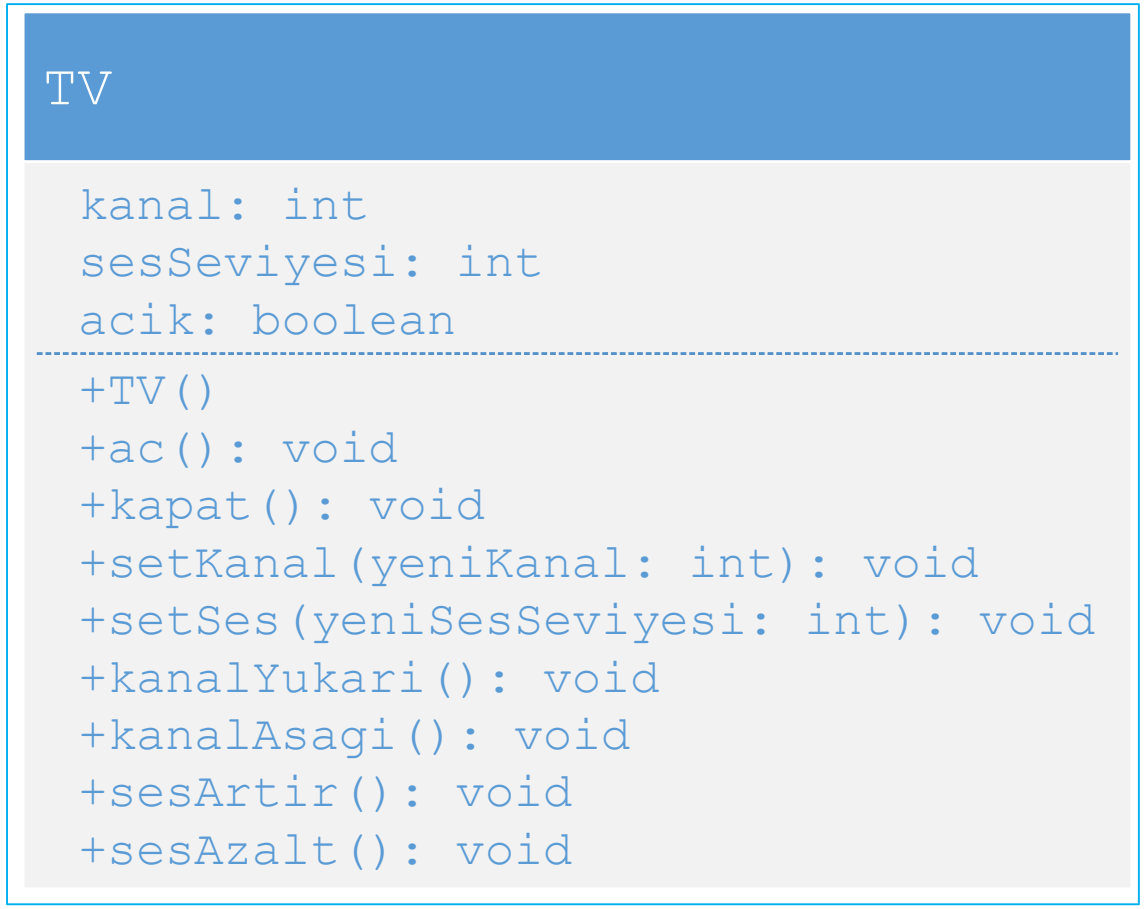
Örnek 2:

Java programları yazmanın birçok yolu vardır. Örnek 1'deki iki sınıfı aşağıda gösterildiği gibi tek bir sınıfta birleştirebiliriz.

```java
public class BasitCemberr {
    /** Main metot */
    public static void main(String[] args) {
    // Yaricapi 1 olan cember olusturma
    BasitCemberr cember1 = new BasitCemberr();
    System.out.println("Yaricapi " + cember1.yaricap
    + " olan cemberin alani: " + cember1.getAlan());

    // Yaricapi 25 olan cember olusturma
    BasitCemberr cember2 = new BasitCemberr(25);
    System.out.println("Yaricapi " + cember2.yaricap
    + " olan cemberin alani: " + cember2.getAlan());

    // Yaricapi 125 olan cember olusturma
    BasitCemberr cember3 = new BasitCemberr(125);
    System.out.println("Yaricapi " + cember3.yaricap
    + " olan cemberin alani: " + cember3.getAlan());

    // Cember yaricapi degistirme
    cember2.yaricap = 100; // veya cember2.setRadius(100)
    System.out.println("Yaricapi " + cember2.yaricap
    + " olan cemberin alani: " + cember2.getAlan());
    }

    double yaricap;

    /** Yaricapi 1 olan bir cember yapilandir */
    BasitCemberr() {
    yaricap = 1;
    }

    /** Yaricapi belirlenebilen bir cember yapilandir */
    BasitCemberr(double yeniYaricap) {
    yaricap = yeniYaricap;
    }

    /** Bu cemberin alanini geri döndür */
    double getAlan() {
    return yaricap * yaricap * Math.PI;
    }

    /** Bu cemberin cevresini geri döndür */
    double getCevre() {
    return 2 * yaricap * Math.PI;
    }

    /** Bu cember icin yeni bir yaricap ata */
    void setYaricap(double yeniYaricap) {
    yaricap = yeniYaricap;
    }
}
```

```
C:\Program Files\Java\jdk-15.0.1\bin\yeni2>java BasitCemberr
Yaricapi 1.0 olan cemberin alani: 3.141592653589793
Yaricapi 25.0 olan cemberin alani: 1963.4954084936207
Yaricapi 125.0 olan cemberin alani: 49087.385212340516
Yaricapi 100.0 olan cemberin alani: 31415.926535897932
```

Örnek 3: TV sınıfı TV kümesini modeller (UML)

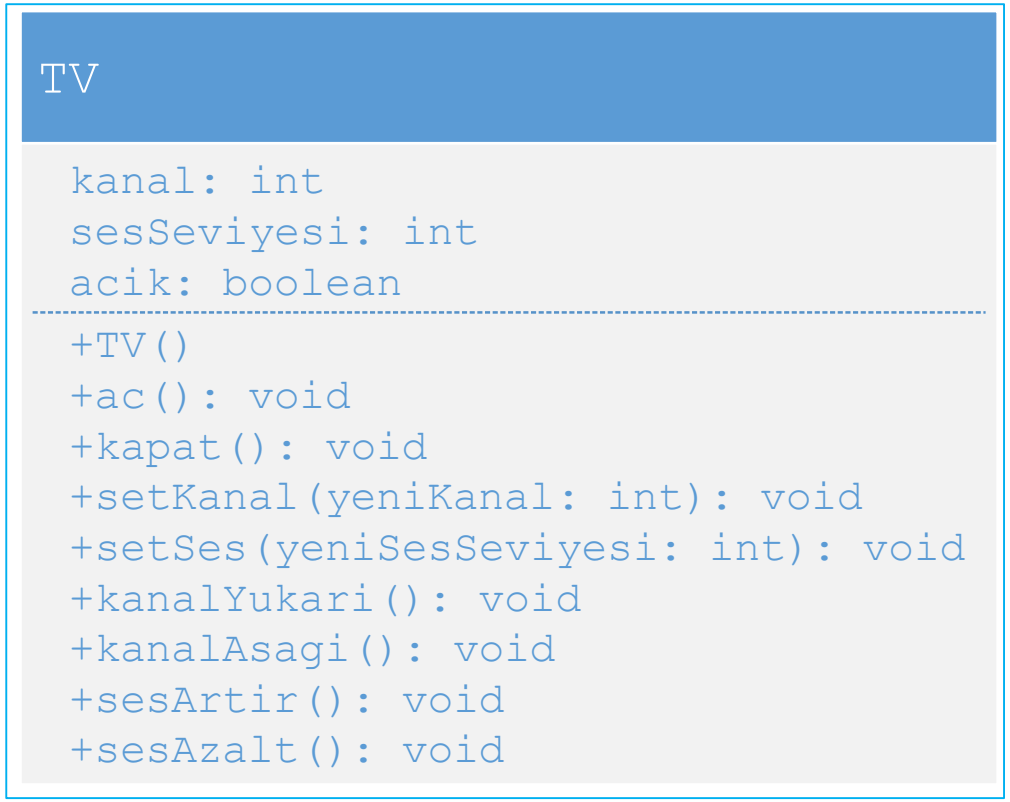


Bu TV'deki mevcut kanal (1-120 kanal).
Bu TV'deki mevcut ses seviyesi (1-7 seviye)
Bu TV'nin açık/kapalı olduğunu gösterir.

Varsayılan TV nesnesi oluştur.
Bu TV'yi aç.
Bu TV'yi kapat.
Bu TV için yeni kanal ata.
Bu TV için yeni ses seviyesi ata.
Kanal numarasını 1 artır.
Kanal numarasını 1 azalt.
Ses seviyesini 1 artır.
Ses seviyesini 1 azalt.

```java
public class TV {
    int kanal = 1; // Varsayılan kanal 1
    int sesSeviyesi = 1; // Varsayılan ses seviyesi 1
    boolean acik = false; // TV kapalı

    public TV() {
    }
    public void ac() {
    acik = true;
    }
    public void kapat() {
    acik = false;
    }
    public void setKanal(int yeniKanal) {
    if (acik && yeniKanal >= 1 && yeniKanal <= 120)
    kanal = yeniKanal;
    }
    public void setSes(int yeniSesSeviyesi) {
    if (acik && yeniSesSeviyesi >= 1 && yeniSesSeviyesi <= 7)
    sesSeviyesi = yeniSesSeviyesi;
    }
    public void kanalYukari() {
    if (acik && kanal < 120)
    kanal++;
    }
    public void kanalAsagi() {
    if (acik && kanal > 1)
    kanal--;
    }
    public void sesArtir() {
    if (acik && sesSeviyesi < 7)
    sesSeviyesi++;
    }
    public void sesAzalt() {
    if (acik && sesSeviyesi > 1)
    sesSeviyesi--;
    }
}
```

Örnek 3: TV sınıfı TV kümesini modeller (UML)



Örnek 3: TV sınıfını kullanan TestTV programı

```
1  public class TestTV {
2      public static void main(String[] args) {
3          TV tv1 = new TV();
4          tv1.ac();
5          tv1.setKanal(30);
6          tv1.setSes(3);
7
8          TV tv2 = new TV();
9          tv2.ac();
10         tv2.kanalYukari();
11         tv2.kanalYukari();
12         tv2.sesArtir();
13
14         System.out.println("tv1'nin kanali: " + tv1.kanal
15         + " ve ses seviyesi: " + tv1.sesSeviyesi);
16         System.out.println("tv2'nin kanali: " + tv2.kanal
17         + " ve ses seviyesi: " + tv2.sesSeviyesi);
18     }
19 }
```

```
C:\Program Files\Java\jdk-15.0.1\bin\yeni2>java TestTV
tv1'nin kanali: 30 ve ses seviyesi: 3
tv2'nin kanali: 3 ve ses seviyesi: 2
```

Yapıcı kullanarak nesneleri oluşturma

new operatörü (işleci) kullanarak bir nesne oluşturmak için bir yapıcı çağrılır.

Yapıcılar özel bir metot türüdür. 3 özelliği vardır:

1. Bir yapıcı, sınıfın kendisiyle aynı isme sahip olmalıdır.

2. Yapıcının dönüş türü yoktur (void bile yok).

3. Yapıcılar, bir nesne oluşturulduğunda new operatörü kullanılarak çağrılır.
   Yapıcılar, nesneleri başlatma rolünü oynar.

Yapıcı kullanarak nesneleri oluşturma

```
public void Cember() {
}
```

Yapıcı olabilmesi için void kullanmamalıyız.

```
new Cember()

new Cember(25)
```

Referans değişkenleri, referans tipleri

```
SinifAdi   nesneRefDegiskeni;

Cember benimCember;

benimCember = new Cember();
```

```
SinifAdi   nesneRefDegiskeni = new SinifAdi();

Cember benimCember = new Cember();
```

Nesnelere referans değişkenler aracılığıyla erişim

Bir nesnenin verilerine ve metotlarına, nesnenin referans değişkeni aracılığıyla nokta (.) operatörü ile erişilebilir.

```
nesneRefDeg.veriAlani;

nesneRefDeg.metot(argümanlar);
```

```
benimCember.yaricap;

benimCember.getAlan(20);

benimCember.getCevre(20);
```

anonymous object (anonim nesne)

```
new Cember();

veya

System.out.println("Alan: " + new Cember(5).getAlan());
```

`null` value (null değeri)

```
class Ogrenci {
        String isim; // isim varsayılan değeri null
        int yas; // yas varsayılan değeri 0
        boolean yetenekliMi; // yetenekliMi varsaılan değeri false
        char cinsiyet; // cinsiyet varsayılan değeri '\u0000'
}
```

`null` value (null değeri)

```
class Ogrenci {
        String isim;
        int yas;
        boolean yetenekliMi;
        char cinsiyet;
}

class TestOgrenci {
        public static void main(String[] args) {
                Ogrenci ogrenci = new Ogrenci();
                System.out.println("İsmi? " + ogrenci.isim);
                System.out.println("Yasi? " + ogrenci.yas);
                System.out.println("Yetenekli mi? " + ogrenci.yetenekliMi);
                System.out.println("Cinsiyeti? " + ogrenci.cinsiyet);
        }
}
```

Örnek 4:

```java
public class TestXY {
    public static void main(String[] args) {
        int x; // x'in varsayilan degeri yok
        String y; // y'nin varsayilan degeri yok
        System.out.println("x: " + x);
        System.out.println("y: " + y);
    }
}
```

```
C:\Program Files\Java\jdk-15.0.1\bin\yeni2>javac TestXY.java
TestXY.java:5: error: variable x might not have been initialized
                System.out.println("x: " + x);
                                           ^
TestXY.java:6: error: variable y might not have been initialized
                System.out.println("y: " + y);
                                           ^
2 errors
```

İlkel tip ve referans tipleri değişkenleri arasındaki fark

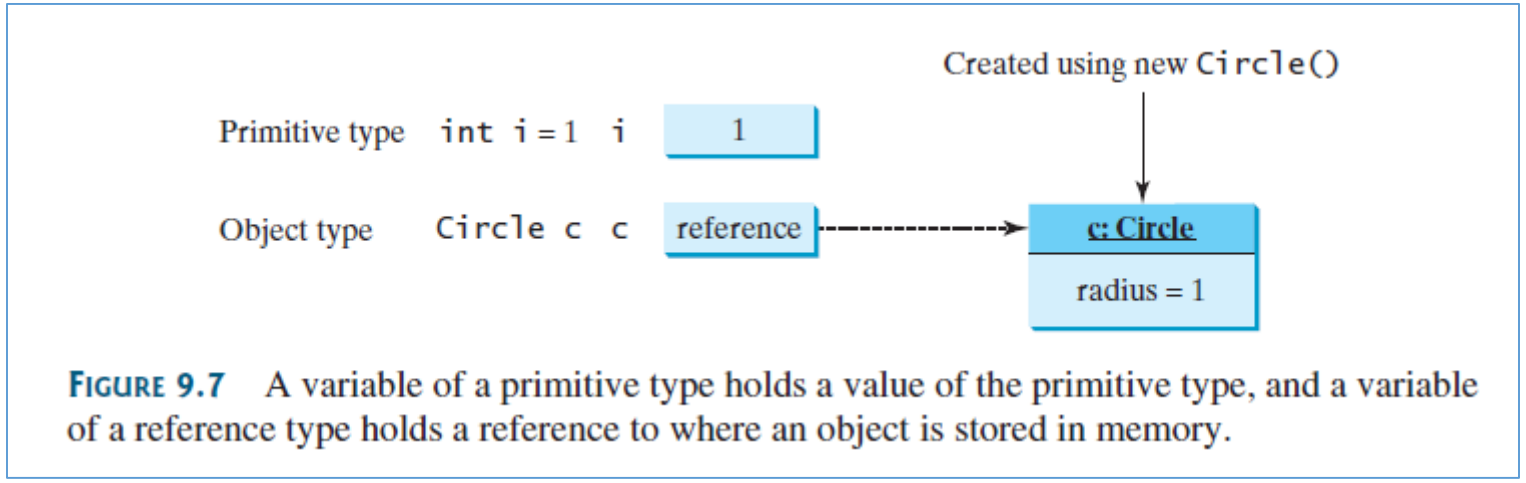


FIGURE 9.7 A variable of a primitive type holds a value of the primitive type, and a variable of a reference type holds a reference to where an object is stored in memory.

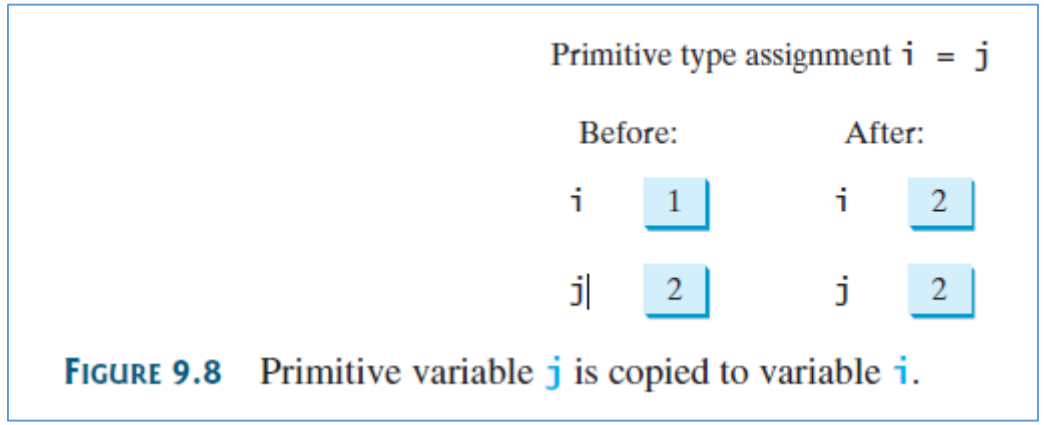


FIGURE 9.8 Primitive variable `j` is copied to variable `i`.

garbage collection

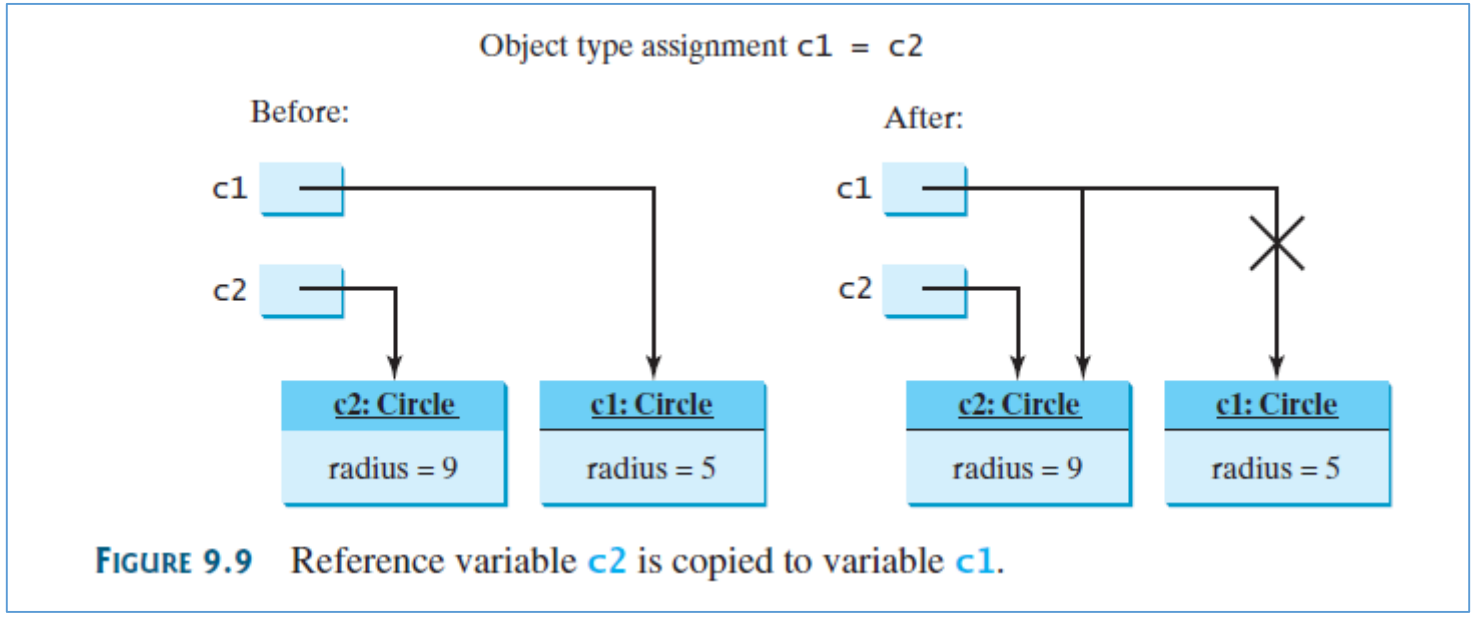


FIGURE 9.9 Reference variable c2 is copied to variable c1.

Ödev 1: Java kütüphanesi sınıfları: The Date class

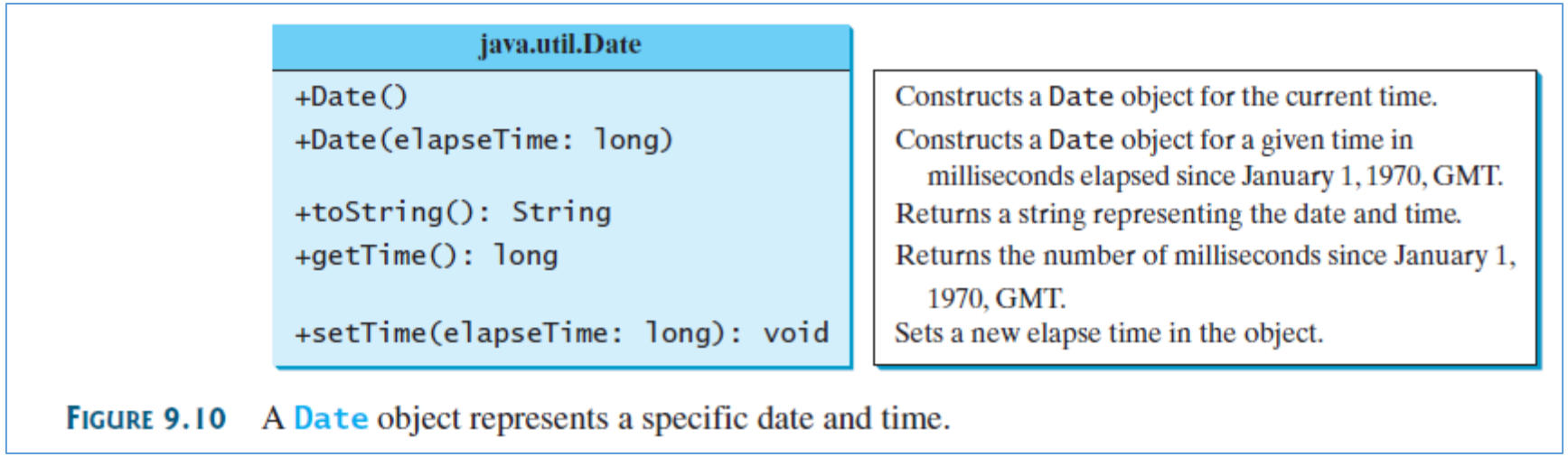


FIGURE 9.10 A `Date` object represents a specific date and time.

Ödev 1: Java kütüphanesi sınıfları: The Date class

```
java.util.Date date = new java.util.Date();
System.out.println("The elapsed time since Jan 1, 1970 is " +
  date.getTime() + " milliseconds");
System.out.println(date.toString());
```

```
The elapsed time since Jan 1, 1970 is 1324903419651 milliseconds
Mon Dec 26 07:43:39 EST 2011
```

Ödev 2: Java kütüphanesi sınıfları: The Random class

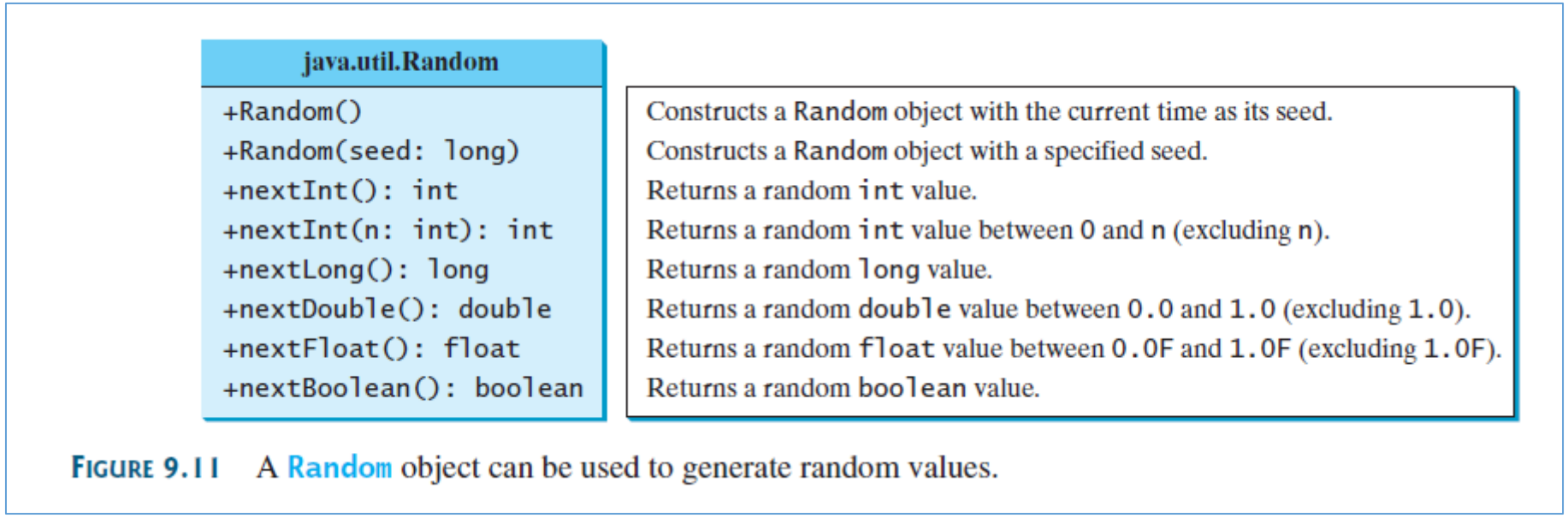


FIGURE 9.11 A Random object can be used to generate random values.

Ödev 2: Java kütüphanesi sınıfları: The Random class

```
Random random1 = new Random(3);
System.out.print("From random1: ");
for (int i = 0; i < 10; i++)
  System.out.print(random1.nextInt(1000) + " ");

Random random2 = new Random(3);
System.out.print("\nFrom random2: ");
for (int i = 0; i < 10; i++)
  System.out.print(random2.nextInt(1000) + " ");
```

java.util.Random
+Random()
+Random(seed: long)
+nextInt(): int
+nextInt(n: int): int
+nextLong(): long
+nextDouble(): double
+nextFloat(): float
+nextBoolean(): boolean

The code generates the same sequence of random `int` values:

```
From random1: 734 660 210 581 128 202 549 564 459 961
From random2: 734 660 210 581 128 202 549 564 459 961
```

Ödev 3: Java kütüphanesi sınıfları: The Point2D class

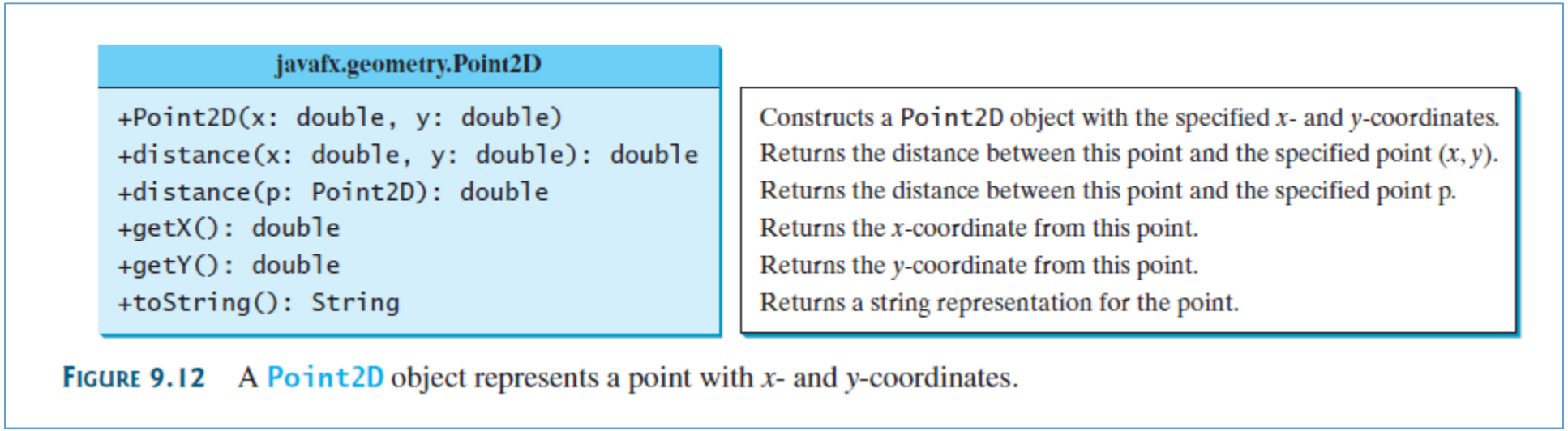


FIGURE 9.12 A `Point2D` object represents a point with *x*- and *y*-coordinates.

Ödev 3: Java kütüphanesi sınıfları: The Point2D class

```
 1  import java.util.Scanner;
 2  import javafx.geometry.Point2D;
 3
 4  public class TestPoint2D {
 5    public static void main(String[] args) {
 6      Scanner input = new Scanner(System.in);
 7
 8      System.out.print("Enter point1's x-, y-coordinates: ");
 9      double x1 = input.nextDouble();
10      double y1 = input.nextDouble();
11      System.out.print("Enter point2's x-, y-coordinates: ");
12      double x2 = input.nextDouble();
13      double y2 = input.nextDouble();
14
15      Point2D p1 = new Point2D(x1, y1);
16      Point2D p2 = new Point2D(x2, y2);
17      System.out.println("p1 is " + p1.toString());
18      System.out.println("p2 is " + p2.toString());
19      System.out.println("The distance between p1 and p2 is " +
20        p1.distance(p2));
21    }
22  }
```

javafx.geometry.Point2D

```
+Point2D(x: double, y: double)
+distance(x: double, y: double): double
+distance(p: Point2D): double
+getX(): double
+getY(): double
+toString(): String
```

```
Enter point1's x-, y-coordinates: 1.5 5.5 Enter
Enter point2's x-, y-coordinates: -5.3 -4.4 Enter
p1 is Point2D [x = 1.5, y = 5.5]
p2 is Point2D [x = -5.3, y = -4.4]
The distance between p1 and p2 is 12.010412149464313
```