Classes

- *Classes* are constructs that define objects of the same type.
- A Java class uses variables to define data fields and methods to define behaviors.
- A class provides a special type of methods, known as constructors, which are invoked to construct objects from the class.

Examples

```
class Person{

    // Data fields
    int age;
    String firstName;
    String lastName;
    ...

    // Behaviors
    void speak(){
    }
    void listen(){
    }
    ......

}
```

Variables to define data fields

Methods to define behaviors

Check Point

1) The relationship between a class and an object is best described as
A) objects are the instance data of classes
B) objects and classes are the same thing
C) classes are programs while objects are variables
D) classes are instances of objects
E) objects are instances of classes

2) The behavior of an object is defined by the object's
A) constructor
B) instance data
C) methods
D) visibility modifiers
E) all of the above

Check Point

Types in Java are divided into two categories. The primitive types are `boolean`, `byte`, `char`, `short`, `int`, `long`, `float` and `double`. All other types are _____ types.

A) static

B) reference

C) declared

D) source

Objects

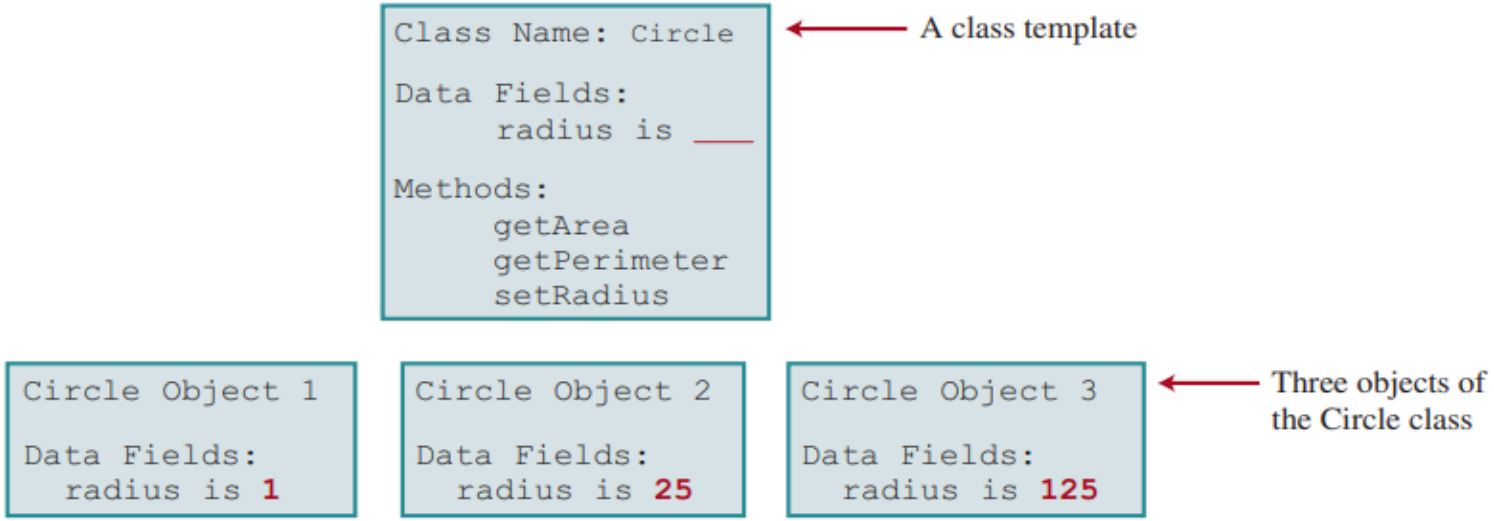


FIGURE 9.2 A class is a template for creating objects.

An object has both a state and behavior. The state defines the object, and the behavior defines what the object does.

Classes

```
class Circle {
  /** The radius of this circle */
  double radius = 1.0;

  /** Construct a circle object */
  Circle() {
  }

  /** Construct a circle object */
  Circle(double newRadius) {
    radius = newRadius;
  }

  /** Return the area of this circle */
  double getArea() {
    return radius * radius * 3.14159;
  }
}
```

Data field

Constructors

Method

Constructors

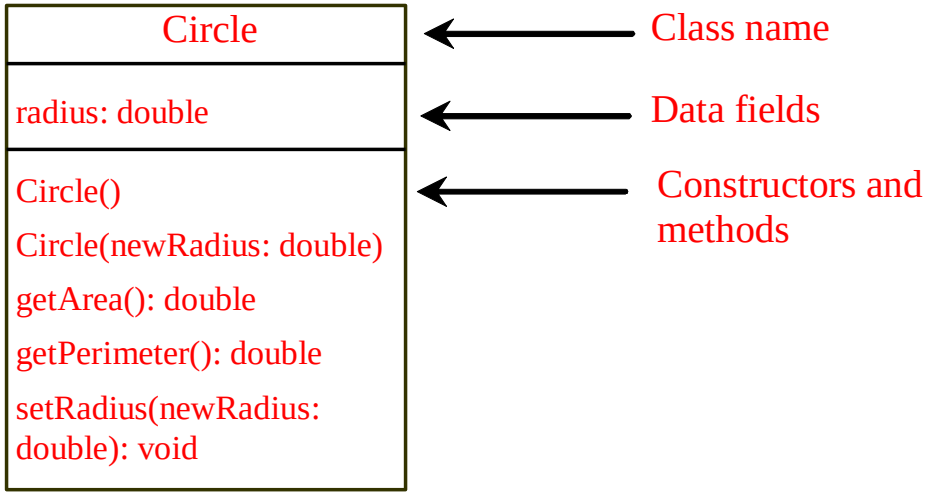
Constructors are a special kind of methods that are invoked to construct objects.

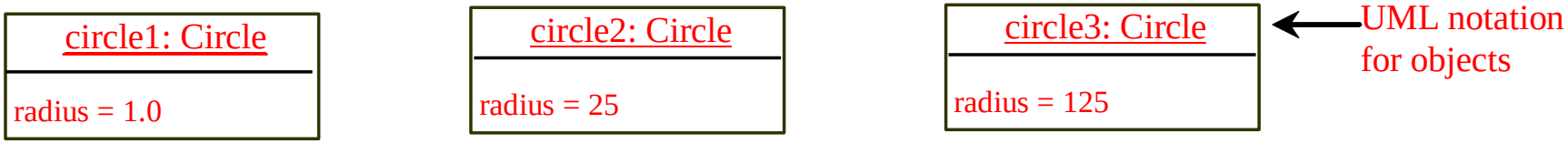
```
Circle() {          //no-arg constructor.

}

Circle(double newRadius) {
  radius = newRadius;
}
```

UML Class Diagram

Unified Modeling Language (UML) notation

UML Class Diagram





Example: Defining Classes and Creating Objects

The + sign indicates a public modifier.

TV	
`channel: int`	The current channel (1 to 120) of this TV.
`volumeLevel: int`	The current volume level (1 to 7) of this TV.
`on: boolean`	Indicates whether this TV is on/off.
`+TV()`	Constructs a default TV object.
`+turnOn(): void`	Turns on this TV.
`+turnOff(): void`	Turns off this TV.
`+setChannel(newChannel: int): void`	Sets a new channel for this TV.
`+setVolume(newVolumeLevel: int): void`	Sets a new volume level for this TV.
`+channelUp(): void`	Increases the channel number by 1.
`+channelDown(): void`	Decreases the channel number by 1.
`+volumeUp(): void`	Increases the volume level by 1.
`+volumeDown(): void`	Decreases the volume level by 1.

Creating Objects Using Constructors

```
new ClassName();//create object of type ClassName
```

Example:

```
new Circle();//create object of type Circle
```

```
new Circle(5.0);
```

Creating Objects

```
class Rectangle {
  double width;
  double height;

  Rectangle() {
    width = 4.0;
    height = 5.3;
  }

  double area() {
    return width * height;
  }
}
```

```
class Rectangle {
  double width;
  double height;

  Rectangle(double w, double h) {
    width = w;
    height = h;
  }

  double area() {
    return width * height;
  }
}
```

Rectangle rectObj = new Rectangle();

Rectangle rectObj = new Rectangle(4.0,5.2);

Creating Objects

```
class Rectangle {
  double width;
  double height;

 Rectangle() {
   width = 1.0;
   height = 5.4;
  }

  Rectangle(double w, double h) {
   width = w;
   height = h;
  }

  double area() {
   return width * height;
  }
}
```

```
Rectangle rectObj = new Rectangle();
```

```
Rectangle rectObj2 = new Rectangle(4.0,5.0);
```

rectObj
width=1.0 height=5.4

rectObj2
width=4.0 height=5.0

Constructors, cont.

- A constructor with no parameters is referred to as a *no-arg constructor*.
- Constructors must have the same name as the class itself.
- Constructors do not have a return type—not even void.
- Constructors are invoked using the new operator when an object is created.
- Constructors play the role of initializing objects(such as initializing the data fields of objects).

Default Constructor

A class may be defined without constructors. In this case, a **public no-arg constructor with an empty body is implicitly defined in the class**. This constructor, called *a default constructor*, is provided automatically ***only if no constructors*** *are explicitly defined in the class.*

Default Constructor

```
class Rectangle {
  double width;
  double height;

  double area() {
    return width * height;
  }

}
```

```
Rectangle mybox1 = new Rectangle();
```

Correct for both

No-arg constructor

```
class Rectangle {
  double width;
  double height;

  Rectangle() {
    width = 10;
    height = 10;
  }

  double area() {
    return width * height;
  }
}
```

Declaring Object Reference Variables

To reference an object, assign the object to a reference variable.

To declare a reference variable, use the syntax:

```
ClassName objectRefVar;
```

Example: **Circle** `object;`

Circle `myCircle;`

* We use **class** keyword to **declare a class**

Example: `class` Circle;

Declaring/Creating Objects in a Single Step

```
ClassName objectRefVar = new ClassName();
```

Example:

Assign object reference

Create an object

```
Circle myCircle = new Circle();
```