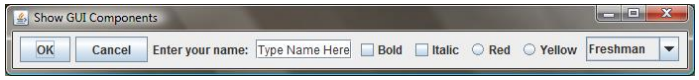
Chapter 9
Objects and Classes

Liang, Introduction to Java Programming, Eighth Edition, (c) 2011 Pearson Education, Inc. All rights reserved. 0132130807

Motivations

- After learning the preceding chapters, you are capable of solving many programming problems using selections, loops, methods, and arrays.
- However, these Java features are not sufficient for developing graphical user interfaces and large scale software systems.
- Suppose you want to develop a graphical user interface as shown below. How do you program it?



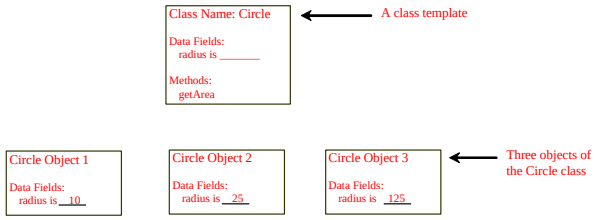
Liang, Introduction to Java Programming, Eighth Edition, (c) 2011 Pearson Education, Inc. All rights reserved. 0132130807

OO Programming Concepts

- Object-oriented programming (OOP) involves programming using objects.
- An *object* represents an entity in the real world that can be distinctly identified.
- For example, a student, a desk, a circle, a button, and even a loan can all be viewed as objects.
- An object has a unique identity, state, and behaviors.
- The *state* of an object consists of a set of *data fields* (also known as *properties*) with their current values.
- The *behavior* of an object is defined by a set of methods.

Liang, Introduction to Java Programming, Eighth Edition, (c) 2011 Pearson Education, Inc. All rights reserved. 0132130807

Objects



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

Liang, Introduction to Java Programming, Eighth Edition, (c) 2011 Pearson Education, Inc. All rights reserved. 0132130807

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.
- Additionally, a class provides a special type of methods, known as constructors, which are invoked to construct objects from the class.

Classes

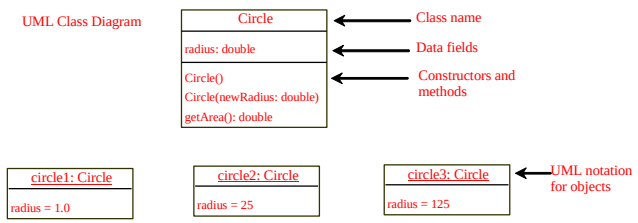
```
class Circle {
  /** The radius of this circle */
  double radius = 1.0;

  /** 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

UML Class Diagram



Constructors

```
Circle() {
}
```

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

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

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.

Creating Objects Using Constructors

```
new ClassName();
```

Example:

```
new Circle();

new Circle(5.0);
```


Default Constructor

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

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;
```

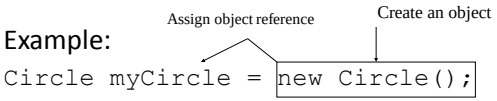
Example:

```
Circle myCircle;
```

Declaring/Creating Objects in a Single Step

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

Example:



Accessing Objects

- Referencing the object's data:

  `objectRefVar.data`

  e.g., `myCircle.radius`

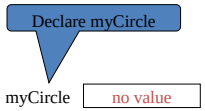
- Invoking the object's method:

  `objectRefVar.methodName(arguments)`

  e.g., `myCircle.getArea()`

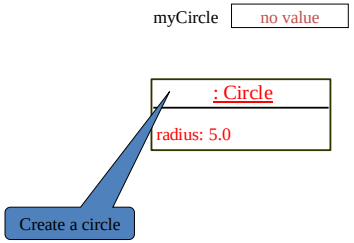
Trace Code

```
Circle myCircle = new Circle(5.0);
SCircle yourCircle = new Circle();
yourCircle.radius = 100;
```



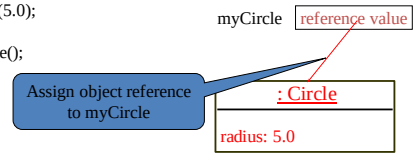
Trace Code, cont.

```
Circle myCircle = new Circle(5.0);
Circle yourCircle = new Circle();
yourCircle.radius = 100;
```



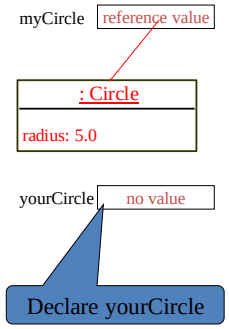
Trace Code, cont.

```
Circle myCircle = new Circle(5.0);
Circle yourCircle = new Circle();
yourCircle.radius = 100;
```



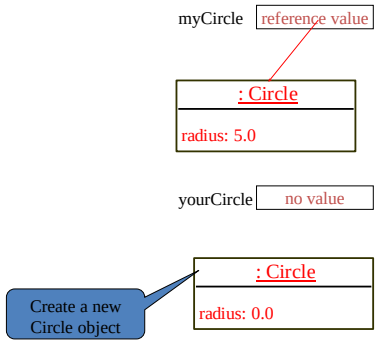
Trace Code, cont.

```
Circle myCircle = new Circle(5.0);
Circle yourCircle = new Circle();
yourCircle.radius = 100;
```



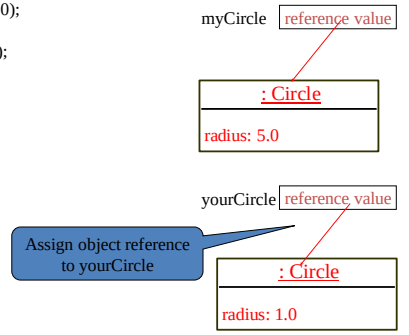
Trace Code, cont.

```
Circle myCircle = new Circle(5.0);
Circle yourCircle = new Circle();
yourCircle.radius = 100;
```



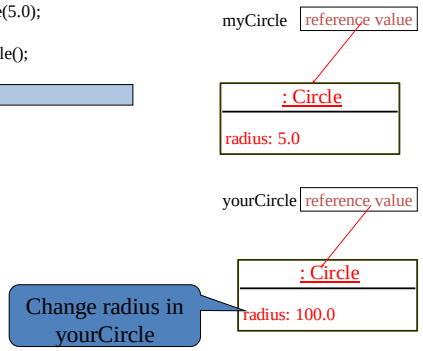
Trace Code, cont.

```
Circle myCircle = new Circle(5.0);
Circle yourCircle = new Circle();
yourCircle.radius = 100;
```



Trace Code, cont.

```
Circle myCircle = new Circle(5.0);

Circle yourCircle = new Circle();

yourCircle.radius = 100;
```



Caution

Recall that you use

Math.methodName(arguments) (e.g., Math.pow(3, 2.5))

to invoke a method in the Math class. Can you invoke getArea() using Circle1.getArea()? The answer is no. All the methods used before this chapter are static methods, which are defined using the static keyword. However, getArea() is non-static. It must be invoked from an object using

objectRefVar.methodName(arguments) (e.g., myCircle.getArea()).

More explanations will be given in the section on "Static Variables, Constants, and Methods."

Reference Data Fields

The data fields can be of reference types. For example, the following Student class contains a data field name of the String type.

```
public class Student {
  String name; // name has default value null
  int age; // age has default value 0
  boolean isScienceMajor; // isScienceMajor has default value false
  char gender; // c has default value '\u0000'
}
```


The null Value

If a data field of a reference type does not reference any object, the data field holds a special literal value, null.

Default Value for a Data Field

- The default value of a data field is
  - null for a reference type,
  - 0 for a numeric type,
  - false for a boolean type, and
  - '\u0000' for a char type.
- However, Java assigns no default value to a local variable inside a method.

```
public class Test {
  public static void main(String[] args) {
    Student student = new Student();
    System.out.println("name? " + student.name);
    System.out.println("age? " + student.age);
    System.out.println("isScienceMajor? " + student.isScienceMajor);
    System.out.println("gender? " + student.gender);
  }
}
```


Example

Java assigns no default value to a local variable inside a method.

```
public class Test {
  public static void main(String[] args) {
    int x; // x has no default value
    String y; // y has no default value
    System.out.println("x is " + x);
    System.out.println("y is " + y);
  }
}
```

Compilation error: variables not initialized

Differences between Variables of Primitive Data Types and Object Types