

Revision Exercise 6

(1) State whether each of the following is *true* or *false*. If *false*, explain why.

- a. When **string** objects are compared with **==**, the result is **true** if the **strings** contain the same values.
- b. A **string** can be modified after it is created.

(2) For each of the following, write a single statement that performs the indicated task.

- a. Compare the string in **s1** to the string in **s2** for equality of contents.
- b. Append the string **s2** to the string **s1** using **+=**.
- c. Determine the length of the string in **s1**.

(3) Fill in the blanks in each of the following:

- a. Class members are accessed via the _____ operator in conjunction with a reference to an object of the class.
- b. Members of a class specified as _____ are accessible only to methods of the class.
- c. A _____ is a special method used to initialize the instance variables of a class.
- d. A _____ method is used to assign values to **private** instance variables of a class.
- e. Methods of a class are normally made _____ and instance variables of a class are normally made _____.
- f. A _____ method is used to retrieve values of **private** data of a class.
- g. The keyword _____ introduces a class definition.
- h. Members of a class specified as _____ are accessible anywhere an object of the class is in scope.
- i. The _____ operator dynamically allocates memory for an object of a specified type and returns a to that type.
- j. A _____ instance variable represents class-wide information.
- k. The keyword _____ specifies that an object or variable is not modifiable after it is initialized.
- l. A method declared **static** cannot access _____ class members.

(4) Create a class called **Complex** for performing arithmetic with complex numbers. Write a driver program to test your class.

Complex numbers have the form

$$\text{realPart} + \text{imaginaryPart} * i$$

where i is

$$\sqrt{-1}$$

Use floating-point variables to represent the private data of the class. Provide a constructor method that enables an object of this class to be initialized when it is declared. Provide a no-argument constructor with default values in case no initializers are provided. Provide **public** methods for each of the following:

- a. Addition of two **Complex** numbers: The real parts are added together and the imaginary parts are added together.
- b. Subtraction of two **Complex** numbers: The real part of the right operand is subtracted from the real part of the left operand and the imaginary part of the right operand is subtracted from the imaginary part of the left operand.
- c. Printing **Complex** numbers in the form **(a, b)**, where **a** is the real part and **b** is the imaginary part.

(5) Create a class **Rectangle**. The class has attributes **length** and **width**, each of which defaults to 1. It has methods that calculate the **perimeter** and the **area** of the rectangle. It has *set* and *get* methods for both **length** and **width**. The *set* methods should verify that **length** and **width** are each floating-point numbers larger than 0.0 and less than 20.0.

Answer

(1)

- a. False. **String** objects that are compared with operator **==** are actually compared to determine if they are the same object in memory
- b. False. **String** objects are constant and cannot be modified after they are created. **StringBuffer** objects can be modified after they are created.

(2)

- a. **s1.equals(s2)**
- b. **s1 += s2;**
- c. **s1.length()**

(3) a) dot (.). b) **private**. c) constructor. d) set. e) **public, private**. f) get. g) **class**. h) **public**. i) **new**, reference. j) **static**. k) **final**. l) non-**static**.

(4)

```
// Complex.java
// Definition of class Complex
public class Complex {
    private double real;
    private double imaginary;
    // Initialize both parts to 0
    public Complex() { this( 0.0, 0.0 ); }
    // Initialize real part to r and imaginary part to 0
    public Complex( double r ) { this( r, 0.0 ); }
    // Initialize real part to r and imaginary part to i
    public Complex( double r, double i )
    {
        real = r;
        imaginary = i;
    }
    // Add two Complex numbers
    public Complex add( Complex right )
    {
        return new Complex( real + right.real,
                             imaginary + right.imaginary );
    }
    // Subtract two Complex numbers
    public Complex subtract( Complex right )
    {
        return new Complex( real - right.real,
                             imaginary - right.imaginary );
    }
    // Return String representation of a Complex number
    public String toString()
```

```

        { return "(" + real + ", " + imaginary + ")"; }
    }

```

// Test the Complex number class

```

public class ComplexTest {
    public static void main( String args[] )
    {
        Complex a, b;
        a = new Complex( 9.9, 7.7 );
        b = new Complex( 1.2, 3.1 );
        String result = "a = " + a;
        result += "\nb = " + b;
        result += "\na + b = " + a.add( b );
        result += "\na - b = " + a.subtract( b );
        System.out.println(result);
    }
}

```

(5)

// MyRectangle.java

// Definition of class MyRectangle

```

public class MyRectangle {
    private double length, width;
    public MyRectangle() { this( 1.0, 1.0 ); }
    public MyRectangle( double l, double w )
    {
        setLength( l );
        setWidth( w );
    }
    public void setLength( double len )
    { length = ( len >= 0.0 && len <= 20.0 ? len : 1.0 ); }
    public void setWidth( double w )
    { width = ( w >= 0 && w <= 20.0 ? w : 1.0 ); }
    public double getLength() { return length; }
    public double getWidth() { return width; }
    public double perimeter() { return 2 * length + 2 * width; }
    public double area() { return length * width; }
}
// Definition of class RectangleTest

```

```

import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;
public class RectangleTest extends Applet
    implements ActionListener {
    private Label prompt1, prompt2;
    private TextField input1, input2;
    private MyRectangle r;
    public void init()
    {
        prompt1 = new Label( "Length:" );
        prompt2 = new Label( "Width:" );
        input1 = new TextField( 10 );
        input2 = new TextField( 10 );
        input2.addActionListener( this );
        add( prompt1 );
        add( input1 );
        add( prompt2 );
        add( input2 );
        r = new MyRectangle();
    }
    public void paint( Graphics g )
    {
        g.drawString( "Length = " + r.getLength(), 25, 75 );
        g.drawString( "Width = " + r.getWidth(), 25, 90 );
        g.drawString( "Perimeter = " + r.perimeter(), 25, 105 );
        g.drawString( "Area = " + r.area(), 25, 120 );
    }
    public void actionPerformed( ActionEvent e )
    {
        double d1, d2;
        d1 = Double.parseDouble( input1.getText() );
        d2 = Double.parseDouble( input2.getText() );
        r.setLength( d1 );
        r.setWidth( d2 );
        repaint();
    }
}

```