

Revision Exercise 4

(1) Answer each of the following:

- a. Program modules in Java are called _____ and _____.
- b. A method is invoked with a _____.
- c. A variable known only within the method in which it is defined is called a _____.
- d. The _____ statement in a called method can be used to pass the value of an expression back to the calling method.
- e. The keyword _____ is used in a method header to indicate that a method does not return a value.
- f. The _____ of an identifier is the portion of the program in which the identifier can be used.
- g. The three ways to return control from a called method to a caller are _____, _____ and _____.
- h. The _____ method is invoked once when an applet begins execution.
- i. The _____ method is used to produce random numbers.
- j. The _____ method is invoked each time the user of a browser revisits the HTML page on which an applet resides.
- k. The _____ method is invoked to draw on an applet.
- l. Variables declared in a block or in a method's parameter list are of _____ duration.
- m. The _____ method invokes the applet's **update** method, which in turn invokes the applet's **paint** method.
- n. The _____ method is invoked for an applet each time the user of a browser leaves an HTML page on which the applet resides.
- o. A method that calls itself either directly or indirectly is a _____ method.
- p. A recursive method typically has two components: one that provides a means for the recursion to terminate by testing for a _____ case and one that expresses the problem as a recursive call for a slightly simpler problem than the original call.
- q. In Java, it is possible to have various methods with the same name that each operate on different types and/or numbers of arguments. This is called method _____.
- r. The _____ qualifier is used to declare read-only variables.

(2) For the following program, state the scope (either class scope or block scope) of each of the following elements.

- a. The variable **x**.
- b. The variable **y**.
- c. The method **cube**.
- d. The method **paint**.
- e. The variable **yPos**.

```
public class CubeTest extends Applet {
    int x;

    public void paint( Graphics g )
    {
        int yPos = 25;
        for ( x = 1; x <= 10; x++ ) {
            g.drawString( cube( x ), 25, yPos );
            yPos += 15;
        }
    }
    public int cube( int y )
    {
        return y * y * y;
    }
}
```

(3) Give the method header for each of the following methods.

- a. Method **hypotenuse**, which takes two double-precision, floating-point arguments **side1** and **side2** and returns a double-precision, floating-point result.
- b. Method **smallest**, which takes three integers, **x**, **y**, **z** and returns an integer.
- c. Method **instructions**, which does not take any arguments and does not return a value. [*Note:* Such methods are commonly used to display instructions to a user.]
- d. Method **intToFloat**, which takes an integer argument, **number** and returns a floating-point result.

(4) Find the error in each of the following program segments and explain how the error can be corrected:

- a.

```
int g() {  
    System.out.println( "Inside method g" );  
    int h() {  
        System.out.println( "Inside method h" );  
    }  
}
```
- b.

```
int sum( int x, int y ) {  
    int result;  
    result = x + y;  
}
```
- c.

```
int sum( int n ) {  
    if ( n == 0 )  
        return 0;  
    else  
        n + sum( n - 1 );  
}
```
- d.

```
void f( float a ); {  
    float a;  
    System.out.println( a );  
}
```
- e.

```
void product() {  
    int a = 6, b = 5, c = 4, result;  
    result = a * b * c;  
    System.out.println( "Result is " + result );  
    return result;  
}
```

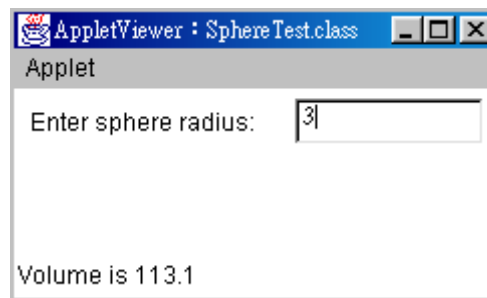
(5) Write a complete Java applet to prompt the user for the **double** radius of a sphere and call method **sphereVolume** to calculate and display the volume of that sphere using the assignment

```
volume = ( 4.0 / 3.0 ) * Math.PI * Math.pow( radius, 3 )
```

The user should input the radius through a **TextField**. A string can be converted to a **double** value as follows (assume **e** is the **ActionEvent** object passed to the **actionPerformed** method when the user presses the *Enter* key in the **TextField**):

```
// Create a Double object using the text field value.
Double val = Double.parseDouble( e.getActionCommand() );
```

Note that `e.getActionCommand()` (when executed in `actionPerformed`) returns the string the user typed in the `TextField` before pressing the *Enter* key in that `TextField`.



(6) What is the value of **x** after each of the following statements is performed?

- a. `x = Math.abs(7.5);`
- b. `x = Math.floor(7.5);`
- c. `x = Math.abs(0.0);`
- d. `x = Math.ceil(0.0);`
- e. `x = Math.abs(-6.4);`
- f. `x = Math.ceil(-6.4);`
- g. `x = Math.ceil(-Math.abs(-8 + Math.floor(-5.5)));`

(7) Write statements that assign random integers to the variable *n* in the following ranges:

- a. $1 \leq n \leq 2$
- b. $1 \leq n \leq 100$
- c. $0 \leq n \leq 9$
- d. $1000 \leq n \leq 1112$
- e. $-1 \leq n \leq 1$
- f. $-3 \leq n \leq 11$

(8) Define a method **hypotenuse** that calculates the length of the hypotenuse of a right triangle when the other two sides are given. The method should take two arguments of type **double** and return the hypotenuse as a **double**. Incorporate this method into an applet that reads integer values for **side1** and **side2** from `TextFields` and performs the calculation with the **hypotenuse** method. Determine the length of the hypotenuse for each of the following triangles. [Note: Register for event handling on only the second

TextField. The user should interact with the program by typing numbers in both **TextFields** and pressing *Enter* in the second **TextField**.]

| Triangle | Side 1 | Side 2 |
|----------|--------|--------|
| 1 | 3.0 | 4.0 |
| 2 | 5.0 | 12.0 |
| 3 | 8.0 | 15.0 |

(9) Write an applet that inputs integers (one at a time) and passes them one at a time to method **isEven**, which uses the modulus operator to determine if an integer is even. The method should take an integer argument and return **true** if the integer is even and **false** otherwise. Use sentinel-controlled looping and an input dialog.

(10) Write a method **squareOfAsterisks** that displays a solid square of asterisks whose side is specified in integer parameter **side**. For example, if **side** is **4**, the method displays

```
****
****
****
****
```

Incorporate this method into an applet that reads an integer value for **side** from the user at the keyboard and performs the drawing with the **squareOfAsterisks** method. Note that this method should be called from the applet's **paint** method and should be passed the **Graphics** object from **paint**.

(11) Write program segments that accomplish each of the following:

- Calculate the integer part of the quotient when integer **a** is divided by integer **b**.
- Calculate the integer remainder when integer **a** is divided by integer **b**.
- Use the program pieces developed in a) and b) to write a method **displayDigits** that receives an integer between **1** and **99999** and prints it as a series of digits, each pair of which is separated by two spaces. For example, the integer **4562** should be printed as
4 5 6 2.
- Incorporate the method developed in c) into an applet that inputs an integer from an input dialog and invokes **displayDigits** by passing the method

the integer entered. Display the results in a message dialog.

(12) Implement the following integer methods:

- a. Method **celsius** returns the Celsius equivalent of a Fahrenheit temperature using the calculation

$$C = 5.0 / 9.0 * (F - 32);$$

- b. Method **fahrenheit** returns the Fahrenheit equivalent of a Celsius temperature.

$$F = 9.0 / 5.0 * C + 32;$$

- b. Use these methods to write an applet that enables the user to enter either a Fahrenheit temperature and display the Celsius equivalent or enter a Celsius temperature and display the Fahrenheit equivalent.

[*Note:* This applet will require that two **TextField** objects that have registered action events. When **actionPerformed** is invoked, the **ActionEvent** parameter has method **getSource()** to determine the GUI component with which the user interacted. Your **actionPerformed** method should contain an **if/else** structure of the following form:

```
if ( e.getSource() == input1 ) {  
    // process input1 interaction here  
}  
else { // e.getSource() == input2  
    // process input2 interaction here  
}
```

where **input1** and **input2** are **TextField** references.]

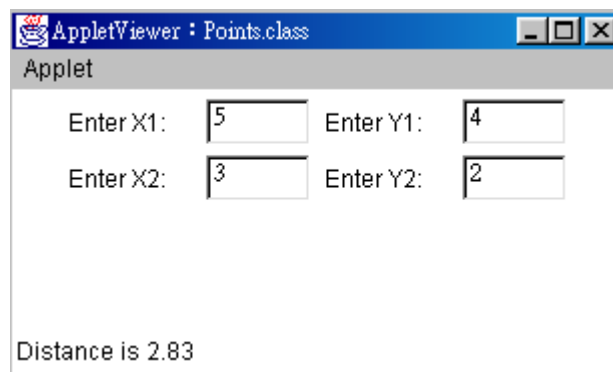
(13) An integer number is said to be a *perfect number* if its factors, including 1 (but not the number itself), sum to the number. For example, 6 is a perfect number because $6 = 1 + 2 + 3$. Write a method **perfect** that determines if parameter **number** is a perfect number. Use this method in an applet that determines and displays all the perfect numbers between 1 and 1000. Print the factors of each perfect number to confirm that the number is indeed perfect. Challenge the computing power of your computer by testing numbers much larger than 1000.

(14) Write a method that takes an integer value and returns the number with its digits reversed. For example, given the number 7631, the method should return 1367. Incorporate the method into an applet that reads a value from the user. Display the result of the method in the status bar.

(15) Write a method **qualityPoints** that inputs a student's average and returns 4 if a student's average is 90—100, 3 if the average is 80—89, 2 if the average is 70—79, 1 if the average is 60—69 and 0 if the average is lower than 60. Incorporate the method into an applet that reads a value from the user. Display the result of the method in the status bar.

(16) Write an applet that simulates coin tossing. Let the program toss the coin each time the user presses the "**Toss**" button. Count the number of times each side of the coin appears. Display the results. The program should call a separate method **flip** that takes no arguments and returns **false** for tails and **true** for heads. [Note: If the program realistically simulates the coin tossing, each side of the coin should appear approximately half the time.]

(17) Write method **distance**, which calculates the distance between two points (x1, y1) and (x2, y2). All numbers and return values should be of type **double**. Incorporate this method into an applet that enables the user to enter the coordinates of the points.



(18) Write an applet that uses a method **circleArea** to prompt the user for the radius of a circle and to calculate and print the area of that circle.

Answer:

(1) a) Methods and classes. b) Method call. c) Local variable. d) **return**. e) **void**. f) Scope. g) **return**; or **return expression**; or encountering the closing right brace of a method. h) **init**. i) **Math.random**. j) **start**. k) **paint**. l) Automatic. m) **repaint**. n) **stop**. o) Recursive. p) Base. q) Overloading. r) **final**.

(2) a) Class scope. b) Block scope. c) Class scope. d) Class scope. e) Block scope.

(3)

- a. `double hypotenuse(double side1, double side2)`
- b. `int smallest(int x, int y, int z)`
- c. `void instructions()`
- d. `float intToFloat(int number)`

(4)

- a. Error: Method **h** is defined in method **g**.
Correction: Move the definition of **h** out of the definition of **g**.
- b. Error: The method is supposed to return an integer, but does not.
Correction: Delete variable **result** and place the following statement in the method:
`return x + y;`
or add the following statement at the end of the method body:
`return result;`
- c. Error: The result of `n + sum(n - 1)` is not returned by this recursive method, resulting in a syntax error.
Correction: Rewrite the statement in the **else** clause as
`return n + sum(n - 1);`
- d. Error: The semicolon after the right parenthesis that encloses the parameter list and redefining the parameter **a** in the method definition are each incorrect.
Correction: Delete the semicolon after the right parenthesis of the parameter list and delete the declaration `float a;`.
- e. Error: The method returns a value when it is not supposed to.
Correction: Change the return type to **int**.

(5)

```
// SphereTest.java
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
public class SphereTest extends Applet
    implements ActionListener {
    Label prompt;
    TextField input;
    public void init()
```



```

{
    prompt = new Label( "Enter sphere radius: " );
    input = new TextField( 10 );
    input.addActionListener( this );
    add( prompt );
    add( input );
}
public void actionPerformed( ActionEvent e )
{
    double radius =
        Double.parseDouble( e.getActionCommand() );
    showStatus( "Volume is " + Math.round(sphereVolume( radius ) * 100) /
100.00 );
}
public double sphereVolume( double radius )
{
    double volume =
        ( 4.0 / 3.0 ) * Math.PI * Math.pow( radius, 3 );
    return volume;
}
}
(6)

```

- a. **ANS: 7.5**
- b. **ANS: 7.5**
- c. **ANS: 0.0**
- d. **ANS: 0.0**
- e. **ANS: 6.4**
- f. **ANS: -6.0**
- g. **ANS: -14.0**

(7)

- a. **ANS: $n = (\text{int}) (1 + \text{Math.random()} * 2);$**
- b. **ANS: $n = (\text{int}) (1 + \text{Math.random()} * 100);$**
- c. **ANS: $n = (\text{int}) (\text{Math.random()} * 10);$**
- d. **ANS: $n = (\text{int}) (1000 + \text{Math.random()} * 113);$**
- e. **ANS: $n = (\text{int}) (-1 + \text{Math.random()} * 3);$**
- f. **ANS: $n = (\text{int}) (-3 + \text{Math.random()} * 15);$**

(8)

```
// Triangle.java
// Program calculates the hypotenuse of
// a right triangle.
import java.applet.Applet;
import java.awt.event.*;
import java.awt.*;

public class Triangle extends Applet
    implements ActionListener {
    TextField sideInput, side2Input;
    Label sidePrompt, sidePrompt2;
    public void init()
    {
        sideInput = new TextField( 4 );
        side2Input = new TextField( 4 );
        side2Input.addActionListener( this );
        sidePrompt = new Label( "Enter side 1: " );
        sidePrompt2 = new Label( "Enter side 2: " );
        add( sidePrompt );
        add( sideInput );
        add( sidePrompt2 );
        add( side2Input );
    }
    public void actionPerformed( ActionEvent e )
    {
        double side1, side2;
        side1 = Double.parseDouble( side2Input.getText() );
        side2 = Double.parseDouble( sideInput.getText() );
        double h = hypotenuse( side1, side2 );
        showStatus( "Hypotenuse is : " + h );
    }
    public double hypotenuse( double s1, double s2 )
    {
        double hypotSquared = Math.pow( s1, 2 ) + Math.pow( s2, 2 );
        return Math.sqrt( hypotSquared );
    }
}
```

(9)

```
// EvenOdd.java
// Determines if a number is odd or even
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class EvenOdd extends Applet
    implements ActionListener{
    private TextField input;
    private Label prompt;
    public void init()
    {
        input = new TextField( 4 );
        input.addActionListener( this );
        prompt = new Label( "Enter number: " );
        add( prompt );
        add( input );
    }
    public void actionPerformed((ActionEvent e) )
    {
        int number = Integer.parseInt( input.getText() );
        String result = "";
        if ( isEven( number ) == true )
            result = number + " is even";
        else
            result = number + " is odd ";
        showStatus( result );
    }
    public boolean isEven( int num )
    {
        if ( num % 2 == 0 )
            return true;
        return false;
    }
}
```

(10)

```
// Square.java
// Program draws a square of asterisks
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class Square extends Applet implements ActionListener{
    int size;
    Label prompt;
    TextField input;
    public void init()
    {
        prompt = new Label("Enter square size:" );
        input = new TextField(4);
        input.addActionListener(this);
        add(prompt);
        add(input);
    }
    public void actionPerformed(ActionEvent e) {
        size = Integer.parseInt( input.getText());
        repaint();
    }
    public void squareOfAsterisks( Graphics g )
    {
        int y = 50, x = 5;
        for ( int a = 1; a <= size * size; a++ ) {
            g.drawString( "*", x += 5, y );
            if ( a % size == 0 ) {
                y += 10;
                x = 5;
            }
        }
    }
    public void paint( Graphics g )
    {
        squareOfAsterisks( g );
    }
}
```

(11)

```
// Digits.java
// Program separates a four digit number
// into its individual digits.
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class Digits extends Applet
    implements ActionListener {
    TextField input;
    Label prompt, display;
    public void init()
    {
        input = new TextField( 6 );
        input.addActionListener( this );
        display = new Label();
        prompt = new Label( "Enter a five digit number:" );
        add( prompt );
        add( input );
        add( display );
    }
    public void actionPerformed( ActionEvent e )
    {
        int number = Integer.parseInt( input.getText() );
        if ( number >= 1 && number <= 99999 ) {
            showStatus( "" );
            displayDigits( number );
        }
        else {
            showStatus( "Invalid input" );
            display.setText( "" );
        }
    }
    // Part A
    public int quotient( int a, int b )
    {
        return a / b;
    }
}
```

```

// Part B
public int remainder( int a, int b )
{
    return a % b;
}

// Part C
public void displayDigits( int number )
{
    int divisor = 10000, digit, x = 5, y = 80;
    String s = "";
    while ( divisor >= 1 ) {
        digit = quotient( number, divisor );
        s += digit + " ";
        number = remainder( number, divisor );
        divisor = quotient( divisor, 10 );
    }
    display.setText( s );
}
}

```

(12)

```

// Convert.java
// Program converts Fahrenheit to Celcius
// and vice versa.
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class Convert extends Applet
    implements ActionListener {
    TextField cInput, fInput;
    Label cLabel, fLabel;
    public void init()
    {
        cInput = new TextField( 4 );
        fInput = new TextField( 4 );
        cInput.addActionListener( this );
        fInput.addActionListener( this );
        cLabel = new Label( "Celcius:" );
        fLabel = new Label( "Fahrenheit:" );
    }
}

```

```

        add( cLabel );
        add( cInput );
        add( fLabel );
        add( fInput );
    }
    public void actionPerformed((ActionEvent e)
    {
        if ( e.getSource() == cInput ) {
            int c = Integer.parseInt( cInput.getText() );
            fInput.setText( String.valueOf( celcius( c ) ) );
            showStatus( "Celcius to Fahrenheit" );
        }
        else if ( e.getSource() == fInput ) {
            int f = Integer.parseInt( fInput.getText() );
            cInput.setText( String.valueOf( fahrenheit( f ) ) );
            showStatus( "Fahrenheit to Celcius" );
        }
    }
}

public int celcius( int cTemp )
{
    return ( ( int ) ( 9.0 / 5.0 * cTemp + 32 ) );
}

public int fahrenheit( int fTemp )
{
    return ( ( int ) ( 5.0 / 9.0 * ( fTemp - 32 ) ) );
}
}

```

(13)

```

// PerfectNum.java
// Program determines if a number
// is a "perfect" number
import java.applet.Applet;
import java.awt.*;
public class PerfectNum extends Applet {
    String output = "";
    public void init()
    {
        int y = 5;
    }
}

```

```

        for ( int a = 2; a <= 1000; a++ )
            if ( perfect( a ) == true )
                output += "\n" + a + " is perfect.";
    }
    public void paint(Graphics g){
        g.drawString(output , 25 , 40);
    }
    public boolean perfect( int value )
    {
        int factorSum = 1;
        for ( int b = 2; b <= value / 2; b++ )
            if ( value % b == 0 )
                factorSum += b;
        if ( factorSum == value )
            return true;
        return false;
    }
}
(14)
// Reverse.java
// Program takes a four digit number
// and prints out its digits reversed
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class Reverse extends Applet
    implements ActionListener {
    TextField input;
    Label prompt;
    int number;
    public void init()
    {
        input = new TextField( 6 );
        input.addActionListener( this );
        prompt = new Label( "Enter a four digit number: " );
        add( prompt );
        add( input );
    }
}

```



```

public void actionPerformed((ActionEvent e)
{
    number = Integer.parseInt( input.getText() );
    reverseDigits();
}
public void reverseDigits()
{
    int digit1 = 0, digit2 = 0, digit3 = 0,
        digit4 = 0, factor = 1000, value = 0;
    while ( factor >= 1 ) {
        int temp = number / factor;
        switch ( factor ) {
            case 1000:
                digit4 = temp;
                break;
            case 100:
                digit3 = temp * 10;
                break;
            case 10:
                digit2 = temp * 100;
                break;
            case 1:
                digit1 = temp * 1000;
                break;
        }
        number %= factor;
        factor /= 10;
    }
    if ( digit1 == 0 ) // special case when last digit initially is 0
        showStatus( String.valueOf( 0 ) + String.valueOf( digit2 / 100 )
            + String.valueOf( digit3 / 10 ) +
            String.valueOf( digit4 ) );
    else
        showStatus( String.valueOf(digit1 + digit2 + digit3 + digit4) );
}
}

```

(15)

```
// Average.java
// Program displays a number
// representing the student's average
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class Average extends Applet
    implements ActionListener {
    TextField input;
    Label prompt;
    public void init()
    {
        input = new TextField( 4 );
        input.addActionListener( this );
        prompt = new Label( "Enter average:" );
        add( prompt );
        add( input );
    }
    public void actionPerformed( ActionEvent e )
    {
        int number = Integer.parseInt( input.getText() );
        if ( number >= 0 && number <= 100 )
            showStatus( "Point is: " + qualityPoints( number ) );
        else
            showStatus( "Invalid input." );
    }
    public int qualityPoints( int grade )
    {
        if ( grade >= 90 )
            return 4;
        else if ( grade >= 80 )
            return 3;
        else if ( grade >= 70 )
            return 2;
        else if ( grade >= 60 )
            return 1;
        else
```

```

        return 0;
    }
}
(16)
// Coin.java
// Program simulates tossing a coin.
import java.applet.Applet;
import java.awt.event.*;
import java.awt.*;
public class Coin extends Applet
    implements ActionListener {
    int heads, tails;
    Button b;
    public void init()
    {
        b = new Button( "Toss" );
        b.addActionListener( this );
        add( b );
    }
    public void actionPerformed( ActionEvent e )
    {
        if ( flip() == true )
            ++heads;
        else
            ++tails;
        showStatus( "Heads: " + heads + "      Tails: " + tails );
    }
    public boolean flip()
    {
        if ( ( int ) ( Math.random() * 2 ) == 1 )
            return true;
        else
            return false;
    }
}

```

(17)

```
// Points.java
// Program calculates the distance
// between two points
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class Points extends Applet
    implements ActionListener {
    TextField x1Input, x2Input, y1Input, y2Input;
    Label labelX1, labelX2, labelY1, labelY2;
    double theDistance;
    boolean enabled;
    public void init()
    {
        x1Input = new TextField( 4 );
        x2Input = new TextField( 4 );
        y1Input = new TextField( 4 );
        y2Input = new TextField( 4 );
        y2Input.addActionListener( this );
        labelX1 = new Label( "Enter X1: " );
        labelY1 = new Label( "Enter Y1: " );
        labelX2 = new Label( "Enter X2: " );
        labelY2 = new Label( "Enter Y2: " );
        add( labelX1 );
        add( x1Input );
        add( labelY1 );
        add( y1Input );
        add( labelX2 );
        add( x2Input );
        add( labelY2 );
        add( y2Input );
    }
    public void actionPerformed( ActionEvent e )
    {
        enabled = false;
        double tempX1, tempX2, tempY1, tempY2;
        tempX1 = Double.parseDouble( x1Input.getText() );
```

```

tempY1 = Double.parseDouble( y1Input.getText() );
tempX2 = Double.parseDouble( x2Input.getText() );
tempY2 = Double.parseDouble( y2Input.getText() );
theDistance = distance( tempX1, tempY1, tempX2, tempY2 );
showStatus( "Distance is " + Math.round(theDistance * 100) / 100.00 );
}
public double distance( double x1, double y1, double x2, double y2 )
{
    double d;
    enabled = true;
    d = Math.sqrt( Math.pow( ( x1 - x2 ), 2 ) +
                  Math.pow( ( y1 - y2 ), 2 ) );
    return d;
}
}
(18)

```

```

// Circle.java
// Program calculates the area of a circle
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class Circle extends Applet
    implements ActionListener {
    TextField input;
    Label instruction;
    public void init()
    {
        input = new TextField( 4 );
        input.addActionListener( this );
        instruction = new Label( "Enter the radius: " );
        add( instruction );
        add( input );
    }
    public void actionPerformed((ActionEvent e)
    {
        showStatus( "" );
        int r = Integer.parseInt( input.getText() );
        circleArea( r );
    }
}

```

```
    }  
    public void circleArea( int radius )  
    {  
        showStatus( "Area is " + Math.round(radius * radius * Math.PI * 100) /  
100.00);  
    }  
}
```