

```
1 package EKPL.Chapter4;
2
3 /**
4 * Created by Sheldon on 11/3/2016.
5 */
6 public class R4_6 {
7     public static void main(String[] args) {
8         double x = 2.5;
9         double y = -1.5;
10        int m = 18;
11        int n = 4;
12        String s = "Hello";
13        String t = "World";
14
15        double a = x + n * y - (x+n) * y;
16        System.out.println(a);
17        double e = Math.sqrt(Math.sqrt(n));
18        System.out.println(e);
19        String what = s.substring(s.length() / 2, s.length());
20        System.out.println(what);
21    }
22}
23
```

```

1 package EKPL.Chapter4.Fuel;
2
3 import javax.swing.*;
4
5 /**
6 * Created by Sheldon on 10/31/2016.
7 * A program that calculates the fuel economy of miles per gallon
8 */
9 // Pseudocodes
10 // user input : fuel in tank
11 // mpg (distance(miles) can be reached per 1 gallon)
12 // price per gallon
13 // output : cost per 100 miles
14 // distance can be reached by the fuel in tank
15 // miles per gallon = distance / fuelconsumed ~ user input
16 // gallons per mile = 1 / mpg
17 // gallons per 100 miles = gallons per mile * 100
18 // cost per 100 miles = gallonsper100miles * price
19 // distance can reached by fuel in tank = miles per gallon * gallons
20
21 public class FuelEfficiency {
22     private double _fuelAmount; //holds the fuel amount in the tank
23     private double _milesPerGallon; //holds the distance reached per 1 gallon of fuel
24     private double _pricePerGallon; //holds the price per 1 gallon of fuel
25     private double _gallonsPerMiles; //holds the fuel needed per specified miles
26     private double _cost; //holds the cost per specified miles
27     private double _distance; //holds the distance can be reached with remaining fuel
28
29 /**
30 * Constructs the fuel economy simulator object with
31 * specified fuel remaining, mpg rate, and price per gallon of fuel

```

```

32   *
33   * param fuelAmount the remaining fuel amount in tank
34   * param mpg the distance in miles can be reached per 1 gallon
35   * param price the price per 1 gallon of fuel
36   */
37   public FuelEfficiency(double fuelAmount, double mpg, double price) {
38     fuelAmount = fuelAmount;
39     _milesPerGallon = mpg;
40     _pricePerGallon = price;
41   }
42
43   /**
44   * Calculates the fuel amount(gallons) needed per specified miles
45   */
46   * param distance the specified distance(miles)
47   */
48   public void gallonsPerMiles(double distance) {
49     final double GALLON_PER_MILE = (1 / _milesPerGallon);
50     _gallonsPerMiles = GALLON_PER_MILE * distance;
51   }
52
53   /**
54   * Calculates the fuel amount(gallons) needed per 100 miles
55   */
56   public void gallonsPerHundredMiles() {
57     final double DISTANCE = 100.0;
58     final double GALLON_PER_MILE = (1 / _milesPerGallon);
59     _gallonsPerMiles = GALLON_PER_MILE * DISTANCE;
60   }
61
62   /**

```

```
63 * Calculates the cost per miles
64 */
65 public void costPerMiles() {
66     _cost = _gallonsPerMiles * _pricePerGallon;
67 }
68
69 /**
70 * Calculates the distance reached with remaining fuel
71 */
72 public void distance() {
73     _distance = _fuelAmount * _milesPerGallon;
74 }
75
76 /**
77 * Get the remaining fuel amount
78 */
79 * return the fuel amount
80 */
81 public double getFuelAmount() {
82     return _fuelAmount;
83 }
84
85 /**
86 * Get the amount of fuel(gallons) needed per specified miles
87 */
88 * return the gallons of fuel
89 */
90 public double getGallons() {
91     return _gallonsPerMiles;
92 }
93
```

```
94 /**
95 * Get the cost per specified miles
96 *
97 * @return the cost
98 */
99 public double getCost() {
100     return _cost;
101 }
102 /**
103 * Get the distance reached with remaining fuel
104 *
105 * @return the distance
106 */
107 /**
108 public double getDistance() {
109     return _distance;
110 }
111 }
```

```
1 package EKPL.Chapter4.Fuel;
2
3 import javax.swing.*;
4
5 /**
6 * Created by Sheldon on 11/1/2016.
7 * E4.10
8 * A program that simulates the fuel economy of miles per gallon
9 */
10
11 public class FuelEfficiencyTester {
12     public static void main(String[] args) {
13         //Prompt user data
14         String strFuelAmount = JOptionPane.showInputDialog("How many gallons of fuel in your tank?");
15         String strMilesPerGallon = JOptionPane.showInputDialog("Input the fuel efficiency (mpg)");
16         String strPrice = JOptionPane.showInputDialog("Input the price per gallon of fuel ($)");
17
18         double fuelAmount = Double.parseDouble(strFuelAmount);
19         double mpg = Double.parseDouble(strMilesPerGallon);
20         double price = Double.parseDouble(strPrice);
21
22         //Construct the simulator object and calculate the functions
23         FuelEfficiency myVehicle = new FuelEfficiency(fuelAmount, mpg, price);
24         myVehicle.gallonsPerHundredMiles();
25         myVehicle.costPerMiles();
26         myVehicle.distance();
27
28         double gallons = myVehicle.getGallons();
29         double cost = myVehicle.getCost();
30         double distance = myVehicle.getDistance();
31 }
```

```
32 //Display the result
33 System.out.println("Within 100 miles, your vehicle needs " + gallons + " gallons of fuel;\n" +
34 "it will cost you $" + cost +"\n");
35 System.out.println("With " + fuelAmount + " gallons of fuel in your tank,\n" +
36 "you can reach " + distance + " miles.");

37
38 // if distance is specified:
39 // String strDistance = JOptionPane.showInputDialog("Input the price per gallon of fuel ($) ");
40 // double theDistance = Double.parseDouble(strDistance);
41 // myVehicle.gallonsPerMiles(theDistance); //here's the difference
42 // myVehicle.costPerMiles();
43 // myVehicle.distance();

44
45 // System.out.println("Within 100 miles, your vehicle needs " + gallons + " gallons of fuel");
46 // System.out.println("Within 100 miles, it will cost $" + cost);
47 // System.out.println("With " + fuelAmount + " gallons of fuel in your tank,\n" +
48 // "you can reach " + distance + " miles");
49 }
50 }
```

```
1 package EKPL.Chapter4.Tiles;
2
3 /**
4 * Created by Sheldon on 11/2/2016.
5 */
6 public class Tiles {
7     public static void main(String[] args) {
8         int totalWidth = 100;
9         int tileWidth = 5;
10        int numberOfTiles = (totalWidth - tileSize) / (2 * tileSize);
11        int numberOfPairs = 1 + 2 * (int) numberOfPairs;
12        double eachEndGap = (double) (totalWidth - numberOfTiles * tileSize) / 2;
13        System.out.printf("Total Width %s%n", ":", " ", "inches");
14        System.out.printf("Tile width %s%n", ":", " ", "inches");
15        System.out.printf("Number of Pairs %s%n", ":", " ", "pairs");
16        System.out.printf("Number of Tiles %s%n", ":", " ", "tiles");
17        System.out.printf("Gap each end %s%n", ":", " ", "inches");
18
19    }
20}
21
```

```
1 package EKPL.Chapter4.Balloon;
2
3 /**
4 * Created by Sheldon on 11/1/2016.
5 * E4.23
6 * A program that calculates the balloon volume after
7 * certain amount air is loaded
8 */
9 public class Balloon {
10     private double _volume; //holds the added volume
11     private double _surfaceArea; //holds the surface area
12     private double _radius; //holds the radius
13
14     /**
15      * Construct a balloon object without parameters
16     */
17     public Balloon() {
18
19
20     /**
21      * Construct a balloon object with specified volume
22     */
23     * param volume the specified volume
24     */
25     public Balloon(double volume) {
26         _volume = volume;
27     }
28
29     /**
30      * Calculate the current volume after certain amount of air is loaded
31     *
```

```
32     * param amount the air volume
33     */
34     public void addAir(double amount) {
35         _volume = _volume + amount;
36     }
37
38     /**
39      * Calculate the balloon radius
40     */
41     public void calculateRadius() {
42         final double MULTIPLIER = 3;
43         final double DIVIDER = 4;
44         double r = (_volume * MULTIPLIER) / (DIVIDER * Math.PI);
45         _radius = Math.cbrt(r);
46     }
47
48     /**
49      * Calculate the balloon surface area
50     */
51     public void calculateSurfaceArea() {
52         final double MULTIPLIER = 4.0;
53         _surfaceArea = MULTIPLIER * Math.PI * _radius * _radius;
54     }
55
56     /**
57      * Get the current balloon volume
58     */
59     * return the balloon volume
60     */
61     public double getVolume() {
62         return _volume;
```

```
63 }
64 /**
65 * Get the current balloon surface area
66 *
67 * @return the surface area
68 */
69 public double getSurfaceArea () {
70     return _surfaceArea;
71 }
72 */
73 /**
74 * Get the current balloon radius
75 *
76 * @return the radius
77 */
78 public double getRadius () {
79     return _radius;
80 }
81 }
82 }
```

```

1 package EKPL.Chapter4.Balloon;
2
3 import javax.swing.*;
4 import java.text.DecimalFormat;
5
6 /**
7 * Created by Sheldon on 11/1/2016.
8 * E4.23
9 * A program that simulates the balloon properties calculation
10 */
11 public class BalloonTester {
12     public static void main(String[] args) {
13         System.out.printf("%60s%n", "A program that simulates the balloon properties calculation");
14         //Prompt the user to add certain amount of air
15         double airVolume = Double.parseDouble(JOptionPane.showInputDialog("Pump your balloon! (volume cm^3)"));
16
17         //Construct a balloon object
18         Balloon theBalloon = new Balloon();
19
20         //Calculate balloon properties
21         theBalloon.addAir(airVolume);
22         theBalloon.calculateRadius();
23         theBalloon.calculateSurfaceArea();
24
25         double volume = theBalloon.getVolume();
26         double radius = Double.parseDouble(new DecimalFormat("#").format(theBalloon.getRadius()));
27         double surfaceArea = Double.parseDouble(new DecimalFormat("#").format(theBalloon.getSurfaceArea()));
28
29         //Display the result
30         System.out.printf("Current balloon volume %11s%, 8.2f%4s%n", "(V) = ", volume, " cm^3");
31         System.out.printf("Current balloon radius %11s%, 8.2f%3s%n", "(r) = ", radius, " cm");

```

```
32     System.out.printf("Current balloon surface area %4.2f\n", 8.2f*4s%n);
33 }
34 }
```

```

1 package EKPL.Chapter4.Triangle;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.text.DecimalFormat;
6
7 /**
8 * Created by Sheldon on 11/1/2016.
9 */
10 public class TriangleTester {
11     public static void main(String[] args) {
12         //Display opening string
13         String str1 = "A program that calculates triangle properties";
14         String str2 = "with given points";
15         System.out.printf("%52s%n%37s%n", str1, str2);
16         System.out.printf("Point n : (%3s)%n , %x, %y");
17
18         //Prompt the user to input the coordinates
19         String strPoint1 = JOptionPane.showInputDialog("Please input first point (x1,y1)");
20         System.out.printf("Point 1 : (%5s)%n", strPoint1);
21         String strPoint2 = JOptionPane.showInputDialog("Please input second point (x2,y2)");
22         System.out.printf("Point 2 : (%5s)%n", strPoint2);
23         String strPoint3 = JOptionPane.showInputDialog("Please input third point (x3,y3)");
24         System.out.printf("Point 3 : (%5s)%n", strPoint3);
25         int strPointLength1 = strPoint1.length();
26         int strPointLength2 = strPoint2.length();
27         int strPointLength3 = strPoint3.length();
28
29         //Extract the ordinates
30         double x1 = Double.parseDouble(strPoint1.substring(0, strPoint1.indexOf(",") ));
31         double y1 = Double.parseDouble(strPoint1.substring(strPoint1.indexOf(",") + 1, strPointLength1));

```

```

32     double x2 = Double.parseDouble(strPoint2.substring(0, strPoint2.indexOf(" , "))); ;
33     double y2 = Double.parseDouble(strPoint2.substring(strPoint2.indexOf(" , ") + 1, strPointLength2)); ;
34     double x3 = Double.parseDouble(strPoint3.substring(0, strPoint3.indexOf(" , "))); ;
35     double y3 = Double.parseDouble(strPoint3.substring(strPoint3.indexOf(" , ") + 1, strPointLength3)); ;

36
37     //Construct the triangle object
38     //double x1 = 50, y1 = 50, x2 = 50, y2 = 140, x3 = 170, y3 = 140; //debugging
39     TriangleComponent theTriangle = new TriangleComponent(x1, y1, x2, y2, x3, y3); ;

40
41     //Do the calculation
42     theTriangle.constructPoints();
43     theTriangle.calculateSidesLength();
44     theTriangle.calculateAngles();
45     theTriangle.calculatePerimeter();
46     theTriangle.calculateArea();

47
48     //Get the calculation result
49     double sideA = Double.parseDouble(new DecimalFormat(" . #").format(theTriangle.getSideLengthA())); ;
50     double sideB = Double.parseDouble(new DecimalFormat(" . #").format(theTriangle.getSideLengthB())); ;
51     double sideC = Double.parseDouble(new DecimalFormat(" . #").format(theTriangle.getSideLengthC())); ;
52     double angleA = Double.parseDouble(new DecimalFormat(" . #").format(theTriangle.getAngleA())); ;
53     double angleB = Double.parseDouble(new DecimalFormat(" . #").format(theTriangle.getAngleB())); ;
54     double angleC = Double.parseDouble(new DecimalFormat(" . #").format(theTriangle.getAngleC())); ;
55     double perimeter = Double.parseDouble(new DecimalFormat(" . #").format(theTriangle.getPerimeter())); ;
56     double area = Double.parseDouble(new DecimalFormat(" . #").format(theTriangle.getArea())); ;

57
58     //Display the result
59     String symbols = "-----+";
60     String symbols1 = "-----+";
61     String symbols2 = "=====+";
62     String symbols3 = "=====+";

```



```
1 package EKPL.Chapter4.Triangle;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.geom.Line2D;
6 import java.awt.geom.Point2D;
7
8 /**
9 * Created by Sheldon on 11/1/2016.
10 * P4.3
11 * A program that calculates triangle properties with
12 * given points
13 */
14 public class TriangleComponent extends JComponent {
15     //Declare the triangle attributes
16     private Point2D.Double point1, point2, point3;
17     private double xOrdinate1, yOrdinate1,
18             xOrdinate2, yOrdinate2,
19             xOrdinate3, yOrdinate3;
20     private double sideLengthA, sideLengthB, sideLengthC;
21     private double angleA, angleB, angleC;
22     private double perimeter;
23     private double area;
24
25 /**
26 * Construct TriangleComponent class with specified coordinates
27 * @param x1 the first ordinate x
28 * @param y1 the first ordinate y
29 * @param x2 the second ordinate x
30 * @param y2 the second ordinate y
31 * @param x3 the third ordinate x
```

```

32   * param y3 the third ordinate y
33   */
34   public TriangleComponent(double x1, double y1, double x2, double y2, double x3, double y3) {
35     xordinate1 = x1;
36     yordinate1 = y1;
37     xordinate2 = x2;
38     yordinate2 = y2;
39     xordinate3 = x3;
40     yordinate3 = y3;
41   }
42
43   /**
44   * Construct coordinate of points from given ordinates
45   */
46   public void constructPoints() {
47     point1 = new Point2D.Double(xordinate1, yordinate1); // (x1, y1)
48     point2 = new Point2D.Double(xordinate2, yordinate2); // (x2, y2)
49     point3 = new Point2D.Double(xordinate3, yordinate3); // (x3, y3)
50   }
51
52   /**
53   * Calculate sides length by measuring the distance of points
54   */
55   public void calculateSidesLength() {
56     sideLengthA = point1.distance(point2); // point A is side a opposite
57     sideLengthB = point2.distance(point3); // point B is side b opposite
58     sideLengthC = point1.distance(point3); // point C is side c opposite
59   }
60
61   /**
62   * Calculate each point angle in degrees

```

```

63  /*
64   public void calculateAngles() {
65     final int ORDER = 2;
66     double arccosineA = ( (Math.pow(sideLengthB, ORDER) + Math.pow(sideLengthC, ORDER)) -
67                           Math.pow(sideLengthA, ORDER)) / (ORDER * sideLengthB * sideLengthC);
68     double arccosineB = ( (Math.pow(sideLengthA, ORDER) + Math.pow(sideLengthC, ORDER)) -
69                           Math.pow(sideLengthB, ORDER)) / (ORDER * sideLengthA * sideLengthC);
70     double arccosineC = ( (Math.pow(sideLengthA, ORDER) + Math.pow(sideLengthB, ORDER)) -
71                           Math.pow(sideLengthC, ORDER)) / (ORDER * sideLengthA * sideLengthB);
72
73     //Convert the value to degrees
74     angleA = Math.toDegrees(Math.acos(arccosineA));
75     angleB = Math.toDegrees(Math.acos(arccosineB));
76     angleC = Math.toDegrees(Math.acos(arccosineC));
77   }
78
79   /**
80    * Calculate triangle perimeter
81   */
82   public void calculatePerimeter() {
83     perimeter = sideLengthA + sideLengthB + sideLengthC;
84   }
85
86   /**
87    * Sketch the triangle shape
88    * param g the painter
89   */
90   public void paintComponent(Graphics g) {
91     Graphics2D painter = (Graphics2D) g;
92
93     //Construct triangle sides

```

```

94 Line2D.Double sideA = new Line2D.Double(point1, point2);
95 Line2D.Double sideB = new Line2D.Double(point2, point3);
96 Line2D.Double sideC = new Line2D.Double(point1, point3);

97 //Draw the sides
98 painter.draw(sideA);
99 painter.draw(sideB);
100 painter.draw(sideC);

101
102 //Get the mid-point of each side
103 float xMidPointA = (float) (xOrdinate1 + xOrdinate2) / 2;
104 float yMidPointA = (float) (yOrdinate1 + yOrdinate2) / 2;
105 float xMidPointB = (float) (xOrdinate2 + xOrdinate3) / 2;
106 float yMidPointB = (float) (yOrdinate2 + yOrdinate3) / 2;
107 float xMidPointC = (float) (xOrdinate1 + xOrdinate3) / 2;
108 float yMidPointC = (float) (yOrdinate1 + yOrdinate3) / 2;

109
110 //Draw annotations
111 painter.setFont(new Font("sideFont", Font.ITALIC, 12));
112 painter.drawString("a", xMidPointA, yMidPointA);
113 painter.drawString("b", xMidPointB, yMidPointB);
114 painter.drawString("c", xMidPointC, yMidPointC);

115
116
117 painter.setFont(new Font("angleFont", Font.BOLD, 12));
118 painter.drawString("A", (float) xOrdinate3, (float) yOrdinate3);
119 painter.drawString("B", (float) xOrdinate1, (float) yOrdinate1);
120 painter.drawString("C", (float) xOrdinate2, (float) yOrdinate2);

121
122 /**
123 * Calculate the area of triangle using Heron's formula
124 */

```

```
125 * S is the semi-perimeter of the triangle
126 */
127 public void calculateArea() {
128     final double S = ((sideLengthA + sideLengthB + sideLengthC) / 2);
129     area = Math.sqrt((S * (S - sideLengthA) * (S - sideLengthB) * (S - sideLengthC)));
130 }
131 /**
132 * Get point 1 coordinate
133 * @return the coordinate
134 */
135 public Point2D.Double getPoint1() {
136     return point1;
137 }
138 /**
139 * Get point 2 coordinate
140 * @return the coordinate
141 */
142 public Point2D.Double getPoint2() {
143     return point2;
144 }
145 /**
146 * Get point 3 coordinate
147 * @return the coordinate
148 */
149 public Point2D.Double getPoint3() {
150     return point3;
151 }
152 }
```

```
156  /**
157   * Get side A length
158   * @return the side length
159   */
160  public double getSideLengthA() {
161      return sideLengthA;
162  }
163
164  /**
165   * Get side B length
166   * @return the side length
167   */
168  public double getSideLengthB() {
169      return sideLengthB;
170  }
171
172  /**
173   * Get side C length
174   * @return the side length
175   */
176  public double getSideLengthC() {
177      return sideLengthC;
178  }
179
180  /**
181   * Get point A angle
182   * @return the angle
183   */
184  public double getAngleA() {
185      return angleA;
186  }
```

```
187  /**
188   * Get point B angle
189   * @return the angle
190   */
191
192 public double getAngleB() {
193     return angleB;
194 }
195
196 /**
197  * Get point C angle
198  * @return the angle
199 */
200 public double getAngleC() {
201     return angleC;
202 }
203
204 /**
205  * Get the triangle perimeter
206  * @return the perimeter
207 */
208 public double getPerimeter() {
209     return perimeter;
210 }
211
212 /**
213  * Get the triangle area
214  * @return the area
215 */
216 public double getArea() {
217     return area;
218 }
```

218 }
219 }

```
1 package EKPL.Chapter4.PlainString;
2
3 import javax.swing.*;
4
5 /**
6 * Created by Sheldon on 11/1/2016.
7 * E4.12
8 * A program that extracts the numbers without comma (, )
9 */
10 public class PlainString {
11     public static void main(String[] args) {
12         System.out.printf("%60s%n", "A program that extracts the numbers without comma (, )");
13         //Prompt the user to input a number
14         String theInput = JOptionPane.showInputDialog("Please input an integer between 1,000 - 999,999:");
15         int inputLength = theInput.length();
16
17         //Extract the numbers by eliminating the ','
18         String firstExtract = theInput.substring(0, theInput.indexOf(","));
19         String secondExtract = theInput.substring(theInput.indexOf(",") + 1, inputLength);
20         String extractedInput = firstExtract + secondExtract;
21
22         //Display the result
23         System.out.printf("%35s%10s%n", "User's input integer", ": ", theInput);
24         System.out.printf("%32s%8s%10s%n", "Extracted integer", ": ", "extractedInput");
25     }
26 }
27
```

```
1 package EKPL.Chapter4.PlainString;
2
3 import javax.swing.*;
4
5 /**
6 * Created by Sheldon on 11/2/2016.
7 */
8 public class PlainStringBasic {
9     public static void main(String[] args) {
10         String theInput = JOptionPane.showInputDialog("Please input an integer between 1,000 - 999,999:");
11         int inputLength = theInput.length();
12
13         String firstExtract = theInput.substring(0, inputLength - 4);
14         String secondExtract = theInput.substring(inputLength - 3, inputLength);
15
16         System.out.println(inputLength);
17         System.out.println(firstExtract);
18         System.out.println(secondExtract);
19     }
20 }
21
```

```

1 package EKPL.Chapter4.LargeLetters;
2
3 /**
4 * Created by Sheldon on 11/1/2016.
5 * E4.17
6 * A program that prints a string horizontally or vertically
7 */
8
9 public class LargeName {
10     //Construct the letters
11     final String LETTER_D = "*****\n" +
12         "*\n" +
13         "*\n" +
14         "*\n" +
15         "*\n" +
16         "*\n" +
17         "*****\n";
18     final String LETTER_I = "*****\n" +
19         "*\n" +
20         "*\n" +
21         "*\n" +
22         "*\n" +
23         "*\n" +
24         "*****\n";
25     final String LETTER_M = "*****\n" +
26         "*\n" +
27         "*\n" +
28         "*\n" +
29         "*\n" +
30         "*\n" +
31         "*\n";

```


63 }
64

```
1 package EKPL.Chapter4.DepositGrowth;  
2  
3 import javax.swing.*;  
4 import java.text.DecimalFormat;  
5  
6 /**  
7 * Created by Sheldon on 11/2/2016.  
8 * A program that simulates the growth of deposit balance  
9 */  
10 public class DepositGrowthSimulator {  
11     private double balance;  
12     private double rate;  
13  
14     /**  
15      * Construct the account object  
16      */  
17     * param initialBalance the initial balance  
18     * param annualRate the annual interest rate  
19     */  
20     public DepositGrowthSimulator(double initialBalance, double annualRate) {  
21         balance = initialBalance;  
22         rate = annualRate;  
23     }  
24  
25     /**  
26      * Calculate the balance growth  
27      */  
28     public void simulateGrowth() {  
29         final double MONTHLY_RATE = (rate / 12.0) / 100.0;  
30         balance = balance + (balance * MONTHLY_RATE);  
31     }  
}
```

```

32
33     /**
34      * Get the current balance
35      *
36      * return the balance
37      */
38     public double getBalance () {
39         return balance;
40     }
41
42     public static void main (String[] args) {
43         //Prompt the user to input data
44         System.out.printf ("%45s%n", "A program that simulates deposit growth");
45         double initialBalance = Double.parseDouble ( JOptionPane.showInputDialog ("Please input" +
46             " your account initial balance ($ : ") );
47         double annualRate = Double.parseDouble ( JOptionPane.showInputDialog ("Please input" +
48             " the annual interest rate (%)) );
49         int nMonth = Integer.parseInt ( JOptionPane.showInputDialog ("Please input time period of months"));
50
51         //Construct the account object to simulate
52         DepositGrowthSimulator myAccount = new DepositGrowthSimulator (initialBalance, annualRate);
53
54         //Display tha simulation result
55         System.out.printf ("%29s%5%.2f%n", "Your account balance", ": $", initialBalance);
56         System.out.printf ("%29s%4s%.1f%%n", "Annual interest rate", ": ", annualRate, "%");
57
58         //Simulate the balance growth in specified period of time
59         for (int i = 0; i < nMonth; i++) {
60             myAccount.simulateGrowth ();
61             double balance = Double.parseDouble (new DecimalFormat ("#.##").format (myAccount.getBalance ()) );
62

```

```
63     System.out.printf("20$%s$13$%,2f$%n", i + 1) ;  
64 }  
65 }  
66 }
```

```
1 package EKPL.Chapter4.TicTacToeGrid;
2
3 /**
4 * Created by Sheldon on 11/1/2016.
5 * E4.14
6 * A program that builds the tic-tac-toe grid
7 */
8 public class TicTacToeGrid {
9     public static void main(String[] args) {
10         //Construct the patterns
11         final String COMB_SHAPE = "+---+---+\r\n| | | |";
12         final String BOTTOM_LINE = "+---+---+---+";
13
14         //Print the combined patterns
15         System.out.println(COMB_SHAPE);
16         System.out.println(COMB_SHAPE);
17         System.out.println(COMB_SHAPE);
18         System.out.println(BOTTOM_LINE);
19     }
20 }
21
```

```
1 package EKPL.Chapter4.CircularObject;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.geom.Ellipse2D;
6 import java.awt.geom.Line2D;
7
8 /**
9 * Created by Sheldon on 10/31/2016.
10 * E4.7
11 * A program that prompts the user for a radius then
12 * calculate the area and circumference of a circle and
13 * the volume and surface area of a sphere with the radius prompted
14 */
15 public class CircularObject extends JComponent {
16     private double radius; //holds the radius
17     private double area; //holds the circle area
18     private double circumference; //holds the circle circumference
19     private double volume; //holds the sphere volume
20     private double surfaceArea; //holds the sphere surface area
21
22 /**
23 * Constructs a circular object with radius r
24 *
25 * param r the radius
26 */
27 public CircularObject(double r) {
28     radius = r;
29 }
30
31 */
```

```
32  * Calculates the circular object area
33  */
34  public void calculateArea() {
35      area = Math.PI * radius * radius;
36  }
37
38  /**
39   * Calculates the circular object circumference
40   */
41  public void calculateCircumference() {
42      final double MULTIPLIER = 2.0;
43      circumference = MULTIPLIER * Math.PI * radius;
44  }
45
46  /**
47   * Calculates the circular object volume
48   */
49  public void calculateVolume() {
50      final double MULTIPLIER = 4.0 / 3.0;
51      volume = MULTIPLIER * Math.PI * radius * radius * radius;
52  }
53
54  /**
55   * Calculates the circular object surface area
56   */
57  public void calculateSurfaceArea() {
58      final double MULTIPLIER = 4.0;
59      surfaceArea = MULTIPLIER * Math.PI * radius * radius;
60  }
61
62  /**
```

```

63 * Function that sketches the circular object
64 *
65 * @param g the painter
66 */
67 public void paintComponent(Graphics g) {
68     Graphics2D painter = (Graphics2D) g;
69
70     // Construct the circular object shapes
71     final double INITIAL_POSITION = 50.0;
72     Ellipse2D.Double circle = new Ellipse2D.Double(INITIAL_POSITION, INITIAL_POSITION, radius, radius);
73     Ellipse2D.Double sphere = new Ellipse2D.Double(INITIAL_POSITION * 2 + radius, INITIAL_POSITION, radius,
74     radius);
75     Ellipse2D.Double ellipse = new Ellipse2D.Double(INITIAL_POSITION * 2 + radius, INITIAL_POSITION + (radius /
76     2.5),
77     radius, radius / 5);
78
79     // Sketch the circular object shapes
80     painter.draw(circle);
81     painter.draw(sphere);
82     painter.setColor(Color.gray);
83     painter.draw(ellipse);
84
85     // Construct and sketch the circle radius
86     Line2D.Double circleRadius = new Line2D.Double(INITIAL_POSITION + (radius / 2), INITIAL_POSITION + (radius /
87     2),
88     INITIAL_POSITION + radius, INITIAL_POSITION + (radius / 2));
89     painter.drawString("r = " + radius, (float) (INITIAL_POSITION + (radius * 0.6)), (float)
90     (INITIAL_POSITION + (radius / 2)));

```

```

91 // Construct and sketch the sphere radius
92 Line2D.Double sphereRadius = new Line2D.Double(INITIAL_POSITION * 2 + (radius * 1.5) ,
93 INITIAL_POSITION + (radius / 2) , INITIAL_POSITION * 2 + (radius * 2) , INITIAL_POSITION + (radius /
94 2));
95 painter.draw(sphereRadius);
96 painter.drawString("r = " + radius, (float) (INITIAL_POSITION * 2 + (radius * 1.6)) ,
97 (float) (INITIAL_POSITION + (radius / 2)));
98
99 // Add the radius values
100 painter.setColor(Color.BLACK);
101 painter.setFont(new Font("Annotation1", Font.BOLD, 14));
102 painter.drawString("Circle", (float) (INITIAL_POSITION + radius * 0.415),
103 (float) (INITIAL_POSITION + radius + 20));
104 painter.drawString("Sphere", (float) (INITIAL_POSITION * 2 + (radius * 1.405)),
105 (float) (INITIAL_POSITION + radius + 20));
106
107 /**
108 * Get the circle object area
109 *
110 * @return the area
111 */
112 public double getArea() {
113     return area;
114 }
115 }
116 /**
117 * Get the circle object circumference
118 *
119 * @return the circumference
120 */

```

```
121  */
122  public double getCircumference () {
123      return circumference;
124  }
125
126 /**
127 * Get the circle object volume
128 *
129 * @return the volume
130 */
131 public double getVolume () {
132     return volume;
133 }
134
135 /**
136 * Get the circle object surface area
137 *
138 * @return the surface area
139 */
140 public double getSurfaceArea () {
141     return surfaceArea;
142 }
143 }
```

```
1 package EKPL.Chapter4.CircularObject;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 /**
7 * Created by Sheldon on 10/31/2016.
8 * E4.7
9 * A program that simulates the calculation of circular object
10 * properties by radius prompted
11 * Sketches the shape of circular object
12 */
13 public class CircularObjectMain {
14     public static void main(String[] args) {
15         // Prompt the user to input the radius
16         String inputRadius = JOptionPane.showInputDialog("Set the object's radius (units)");
17         double radius = Double.parseDouble(inputRadius);
18         System.out.printf("Circular object radius %13s%, 14.2f%6s%n", "r", radius, "units");
19
20         // Construct the circular object
21         CircularObject theCircularObject = new CircularObject(radius);
22
23         // Calculate the circular object properties
24         theCircularObject.calculateCircumference();
25         theCircularObject.calculateArea();
26         theCircularObject.calculateSurfaceArea();
27         theCircularObject.calculateVolume();
28
29         double circumference = theCircularObject.getCircumference();
30         double area = theCircularObject.getArea();
31     }
32 }
```

```
32     double surfaceArea = theCircularObject.getSurfaceArea();  
33     double volume = theCircularObject.getVolume();  
34  
35     // Display the results  
36     System.out.printf("Circular object circumference %6s%, 14.2f%6s%n", " (C) =\"", circumference, "units");  
37     System.out.printf("Circular object area %15s%, 14.2f%13s%n", " (A) =\"", area, "square units");  
38     System.out.printf("Circular object surface area %7s%, 14.2f%13s%n", " (A) =\"", surfaceArea, "square units");  
39     System.out.printf("Circular object volume %13s%, 14.2f%11s%n", " (V) =\"", volume, "cube units");  
40  
41     // Display the sketch in a new window  
42     JFrame theFrame = new JFrame();  
43     theFrame.setSize(600, 400);  
44     theFrame.getContentPane().setBackground(Color.WHITE);  
45     theFrame.setTitle("Circular Object");  
46     theFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
47     theFrame.setVisible(true);  
48     theFrame.add(theCircularObject);  
49 }  
50 }
```