```java
package EKPL.Chapter4;

/**
 * Created by Sheldon on 11/3/2016.
 */
public class R4_6 {
  public static void main(String[] args) {
    double x = 2.5;
    double y = -1.5;
    int m = 18;
    int n = 4;
    String s = "Hello";
    String t = "World";

    double a = x + n * y - (x+n)  * y;
    System.out.println(a);
    double e = Math.sqrt(Math.sqrt(n));
    System.out.println(e);
    String what = s.substring(s.length()/2, s.length());
    System.out.println(what);
  }
}

```

```java
package EKPL.Chapter4.Fuel;

import javax.swing.*;

/**
 * Created by Sheldon on 10/31/2016.
 * A program that calculates the fuel economy of miles per gallon
 */
// Pseudocodes
// user input : fuel in tank
//              mpg (distance(miles) can be reached per 1 gallon)
//              price per gallon
// output     : cost per 100 miles
//              distance can be reached by the fuel in tank
// miles per gallon = distance / fuelconsumed ~ user input
// gallons per mile = 1 / mpg
// gallons per 100 miles = gallons per mile * 100
// cost per 100 miles = gallonsper100miles * price
// distance can reached by fuel in tank = miles per gallon * gallons

public class FuelEfficiency {
  private double _fuelAmount; //holds the fuel amount in the tank
  private double _milesPerGallon; //holds the distance reached per 1 gallon of fuel
  private double _pricePerGallon; //holds the price per 1 gallon of fuel
  private double _gallonsPerMiles; //holds the fuel needed per specified miles
  private double _cost; //holds the cost per specified miles
  private double _distance; //holds the distance can be reached with remaining fuel

  /**
   * Constructs the fuel economy simulator object with
   * specified fuel reamining, mpg rate, and price per gallon of fuel
   *
   * @param fuelAmount the remaining fuel amount in tank
   * @param mpg        the distance in miles can be reached per 1 gallon
   * @param price      the price per 1 gallon of fuel
   */
  public FuelEfficiency(double fuelAmount, double mpg, double price) {
    _fuelAmount = fuelAmount;
    _milesPerGallon = mpg;
    _pricePerGallon = price;
  }

  /**
   * Calculates the fuel amount(gallons) needed per specified miles
   *
   * @param distance the specified distance(miles)
   */
  public void gallonsPerMiles(double distance) {
    final double GALLON_PER_MILE = (1 / _milesPerGallon);
    _gallonsPerMiles = GALLON_PER_MILE * distance;
  }

  /**
   * Calculates the fuel amount(gallons) needed per 100 miles
   */
  public void gallonsPerHundredMiles() {
    final double DISTANCE = 100.0;
    final double GALLON_PER_MILE = (1 / _milesPerGallon);
    _gallonsPerMiles = GALLON_PER_MILE * DISTANCE;
  }

  /**
```

```
   * Calculates the cost per miles
   */
  public void costPerMiles() {
    _cost = _gallonsPerMiles * _pricePerGallon;
  }

  /**
   * Calculates the distance reached with remaining fuel
   */
  public void distance() {
    _distance = _fuelAmount * _milesPerGallon;
  }

  /**
   * Get the remaining fuel amount
   *
   * @return the fuel amount
   */
  public double getFuelAmount() {
    return _fuelAmount;
  }

  /**
   * Get the amount of fuel(gallons) needed per specified miles
   *
   * @return the gallons of fuel
   */
  public double getGallons() {
    return _gallonsPerMiles;
  }

  /**
   * Get the cost per specified miles
   *
   * @return the cost
   */
  public double getCost() {
    return _cost;
  }

  /**
   * Get the distance reached with remaining fuel
   *
   * @return the distance
   */
  public double getDistance() {
    return _distance;
  }
}
```

```java
package EKPL.Chapter4.Fuel;

import javax.swing.*;

/**
 * Created by Sheldon on 11/1/2016.
 * E4.10
 * A program that simulates the fuel economy of miles per gallon
 */

public class FuelEfficiencyTester {
  public static void main(String[] args) {
    //Prompt user data
    String strFuelAmount = JOptionPane.showInputDialog("How many gallons of fuel in your tank?");
    String strMilesPerGallon = JOptionPane.showInputDialog("Input the fuel efficiency (mpg)");
    String strPrice = JOptionPane.showInputDialog("Input the price per gallon of fuel ($)");

    double fuelAmount = Double.parseDouble(strFuelAmount);
    double mpg = Double.parseDouble(strMilesPerGallon);
    double price = Double.parseDouble(strPrice);

    //Construct the simulator object and calculate the functions
    FuelEfficiency myVehicle = new FuelEfficiency(fuelAmount, mpg, price);
    myVehicle.gallonsPerHundredMiles();
    myVehicle.costPerMiles();
    myVehicle.distance();

    double gallons = myVehicle.getGallons();
    double cost = myVehicle.getCost();
    double distance = myVehicle.getDistance();

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

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

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

```java
package EKPL.Chapter4.Tiles;

/**
 * Created by Sheldon on 11/2/2016.
 */
public class Tiles {
  public static void main(String[] args) {
    int totalWidth = 100;
    int tileWidth = 5;
    int numberOfPairs = (totalWidth - tileWidth) / (2 * tileWidth);
    int numberOfTiles = 1 + 2 * (int) numberOfPairs;
    double eachEndGap = (double) (totalWidth - numberOfTiles * tileWidth) / 2;
    System.out.printf("Total Width %10s%4s%7s%n", ": ", totalWidth, "inches");
    System.out.printf("Tile width %11s%4s%7s%n", ": ", tileWidth, "inches");
    System.out.printf("Number of Pairs %6s%4s%6s%n", ": ", numberOfPairs, "pairs");
    System.out.printf("Number of Tiles %6s%4s%6s%n", ": ", numberOfTiles, "tiles");
    System.out.printf("Gap each end %9s%4s%7s%n", ": ", eachEndGap, "inches");

  }
}

```

```java
package EKPL.Chapter4.Balloon;

/**
 * Created by Sheldon on 11/1/2016.
 * E4.23
 * A program that calculates the balloon volume after
 * certain amount air is loaded
 */
public class Balloon {
  private double _volume; //holds the added volume
  private double _surfaceArea;  //holds the surface area
  private double _radius; //holds the radius

  /**
   * Construct a balloon object without parameters
   */
  public Balloon() {
  }

  /**
   * Construct a balloon object with specified volume
   *
   * @param volume the specified volume
   */
  public Balloon(double volume) {
    _volume = volume;
  }

  /**
   * Calculate the current volume after certain amount of air is loaded
   *
```

```java
     * @param amount the air volume
     */
    public void addAir(double amount) {
      _volume = _volume + amount;
    }

    /**
     * Calculate the balloon radius
     */
    public void calculateRadius() {
      final double MULTIPLIER = 3;
      final double DIVIDER = 4;
      double r = (_volume * MULTIPLIER) / (DIVIDER * Math.PI);
      _radius = Math.cbrt(r);
    }

    /**
     * Calculate the balloon surface area
     */
    public void calculateSurfaceArea() {
      final double MULTIPLIER = 4.0;
      _surfaceArea = MULTIPLIER * Math.PI * _radius * _radius;
    }

    /**
     * Get the current balloon volume
     *
     * @return the balloon volume
     */
    public double getVolume() {
      return _volume;
```

```
  }

  /**
   * Get the current balloon surface area
   *
   * @return the suface area
   */
  public double getSurfaceArea() {
    return _surfaceArea;
  }

  /**
   * Get the current balloon radius
   *
   * @return the radius
   */
  public double getRadius() {
    return _radius;
  }
}
```

```java
package EKPL.Chapter4.Balloon;

import javax.swing.*;
import java.text.DecimalFormat;

/**
 * Created by Sheldon on 11/1/2016.
 * E4.23
 * A program that simulates the balloon properties calculation
 */
public class BalloonTester {
  public static void main(String[] args) {
    System.out.printf("%60s%n%n", "A program that simulates the balloon properties calculation");
    //Prompt the user to add certain amount of air
    double airVolume = Double.parseDouble(JOptionPane.showInputDialog("Pump your balloon! (volume cm^3)"));

    //Construct a balloon object
    Balloon theBalloon = new Balloon();

    //Calculate balloon properties
    theBalloon.addAir(airVolume);
    theBalloon.calculateRadius();
    theBalloon.calculateSurfaceArea();

    double volume = theBalloon.getVolume();
    double radius = Double.parseDouble(new DecimalFormat(".##").format(theBalloon.getRadius()));
    double surfaceArea = Double.parseDouble(new DecimalFormat(".##").format(theBalloon.getSurfaceArea()));

    //Display the result
    System.out.printf("Current balloon volume %11s%,8.2f%4s%n", "(V)= ", volume, " cm3");
    System.out.printf("Current balloon radius %11s%,8.2f%3s%n", "(r)= ", radius, " cm");
    System.out.printf("Current balloon surface area %4s%,8.2f%4s%n", "(A)= ", surfaceArea, " cm2");
  }
}
```

```java
package EKPL.Chapter4.Triangle;

import javax.swing.*;
import java.awt.*;
import java.text.DecimalFormat;

/**
 * Created by Sheldon on 11/1/2016.
 */
public class TriangleTester {
  public static void main(String[] args) {
    //Display opening string
    String str1 = "A program that calculates triangle properties";
    String str2 = "with given points";
    System.out.printf("%52s%n%37s%n%n", str1, str2);
    System.out.printf("Point n  : (%3s)%n", "x,y");

    //Prompt the user to input the coordinates
    String strPoint1 = JOptionPane.showInputDialog("Please input first point (x1,y1)");
    System.out.printf("Point 1  : (%5s)%n", strPoint1);
    String strPoint2 = JOptionPane.showInputDialog("Please input second point (x2,y2)");
    System.out.printf("Point 2  : (%5s)%n", strPoint2);
    String strPoint3 = JOptionPane.showInputDialog("Please input third point (x3,y3)");
    System.out.printf("Point 3  : (%5s)%n%n", strPoint3);
    int strPointLength1 = strPoint1.length();
    int strPointLength2 = strPoint2.length();
    int strPointLength3 = strPoint3.length();

    //Extract the ordinates
    double x1 = Double.parseDouble(strPoint1.substring(0, strPoint1.indexOf(",")));
    double y1 = Double.parseDouble(strPoint1.substring(strPoint1.indexOf(",") + 1, strPointLength1));
    double x2 = Double.parseDouble(strPoint2.substring(0, strPoint2.indexOf(",")));
    double y2 = Double.parseDouble(strPoint2.substring(strPoint2.indexOf(",") + 1, strPointLength2));
    double x3 = Double.parseDouble(strPoint3.substring(0, strPoint3.indexOf(",")));
    double y3 = Double.parseDouble(strPoint3.substring(strPoint3.indexOf(",") + 1, strPointLength3));

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

    //Do the calculation
    theTriangle.constructPoints();
    theTriangle.calculateSidesLength();
    theTriangle.calculateAngles();
    theTriangle.calculatePerimeter();
    theTriangle.calculateArea();

    //Get the calculation result
    double sideA = Double.parseDouble(new DecimalFormat(".#").format(theTriangle.getSideLengthA()));
    double sideB = Double.parseDouble(new DecimalFormat(".#").format(theTriangle.getSideLengthB()));
    double sideC = Double.parseDouble(new DecimalFormat(".#").format(theTriangle.getSideLengthC()));
    double angleA = Double.parseDouble(new DecimalFormat(".#").format(theTriangle.getAngleA()));
    double angleB = Double.parseDouble(new DecimalFormat(".#").format(theTriangle.getAngleB()));
    double angleC = Double.parseDouble(new DecimalFormat(".#").format(theTriangle.getAngleC()));
    double perimeter = Double.parseDouble(new DecimalFormat(".#").format(theTriangle.getPerimeter()));
    double area = Double.parseDouble(new DecimalFormat(".#").format(theTriangle.getArea()));

    //Display the result
    String symbols = "+--------+";
    String symbols1 = "---------+";
    String symbols2 = "+============================+";
    String symbols3 = "=============================+";
    System.out.printf("%38s%n", "Triangle Properties");
    System.out.printf("%s%30s%n", symbols2, symbols3);
    System.out.printf("%s%19s%10s%21s%9s%n",
            "|", "Side Length", "|", "Corner Angle", "|");
    System.out.printf("%s%17s%12s%19s%11s%n",
            "|", "(units)", "|", "(degrees)", "|");
    System.out.printf("%s%10s%10s%10s%10s%10s%n",
            symbols, symbols1, symbols1, symbols1, symbols1, symbols1);
    System.out.printf("%s%4s%5s%5s%5s%5s%5s%5s%5s%5s%5s%5s%5s %n",
            "|", "a", "|", "b", "|", "c", "|", "A", "|", "B", "|", "C", "|");
    System.out.printf("%s%10s%10s%10s%10s%10s%n",
            symbols, symbols1, symbols1, symbols1, symbols1, symbols1);
    System.out.printf("%s%6s%3s%7s%3s%7s%3s%6s%4s%6s%4s%6s%4s %n",
            "|", sideA, "|", sideB, "|", sideC, "|", angleA, "|", angleB, "|", angleC, "|");
    System.out.printf("%s%30s%n", symbols2, symbols3);
    System.out.printf("Perimeter %5s%,10.1f%6s%n", ":", perimeter, "units");
    System.out.printf("Area %10s%,10.1f%13s%n", ":", area, "square units");

    //Sketch the triangle in new window
    JFrame theFrame = new JFrame();
    theFrame.setSize(270, 250);
    theFrame.setTitle("Triangle Sketch");
    theFrame.getContentPane().setBackground(Color.WHITE);
    theFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    theFrame.setVisible(true);
    theFrame.add(theTriangle);
  }
}
```

```java
package EKPL.Chapter4.Triangle;

import javax.swing.*;
import java.awt.*;
import java.awt.geom.Line2D;
import java.awt.geom.Point2D;

/**
 * Created by Sheldon on 11/1/2016.
 * P4.3
 * A program that calculates triangle properties with
 * given points
 */
public class TriangleComponent extends JComponent {
  //Declare the triangle attributes
  private Point2D.Double point1, point2, point3;
  private double xOrdinate1, yOrdinate1,
          xOrdinate2, yOrdinate2,
          xOrdinate3, yOrdinate3;
  private double sideLengthA, sideLengthB, sideLengthC;
  private double angleA, angleB, angleC;
  private double perimeter;
  private double area;

  /**
   * Construct TriangleComponent class with specified ordinates
   * @param x1 the first ordinate x
   * @param y1 the first ordinate y
   * @param x2 the second ordinate x
   * @param y2 the second ordinate y
   * @param x3 the third ordinate x
   * @param y3 the third ordinate y
   */
  public TriangleComponent(double x1, double y1, double x2, double y2, double x3, double y3) {
    xOrdinate1 = x1;
    yOrdinate1 = y1;
    xOrdinate2 = x2;
    yOrdinate2 = y2;
    xOrdinate3 = x3;
    yOrdinate3 = y3;
  }

  /**
   * Construct coordinate of points from given ordinates
   */
  public void constructPoints() {
    point1 = new Point2D.Double(xOrdinate1, yOrdinate1); //(x1, y1)
    point2 = new Point2D.Double(xOrdinate2, yOrdinate2); //(x2, y2)
    point3 = new Point2D.Double(xOrdinate3, yOrdinate3); //(x3, y3)
  }

  /**
   * Calculate sides length by measuring the distance of points
   */
  public void calculateSidesLength() {
    sideLengthA = point1.distance(point2); //point A is side a opposite
    sideLengthB = point2.distance(point3); //point B is side b opposite
    sideLengthC = point1.distance(point3); //point C is side c opposite
  }

  /**
   * Calculate each point angle in degrees
   */
  public void calculateAngles() {
    final int ORDER = 2;
    double arccosineA = ((Math.pow(sideLengthB, ORDER) + Math.pow(sideLengthC, ORDER)) -
            Math.pow(sideLengthA, ORDER)) / (ORDER * sideLengthB * sideLengthC);
    double arccosineB = ((Math.pow(sideLengthC, ORDER) + Math.pow(sideLengthA, ORDER)) -
            Math.pow(sideLengthB, ORDER)) / (ORDER * sideLengthA * sideLengthC);
    double arccosineC = ((Math.pow(sideLengthA, ORDER) + Math.pow(sideLengthB, ORDER)) -
            Math.pow(sideLengthC, ORDER)) / (ORDER * sideLengthA * sideLengthB);

    //Convert the value to degrees
    angleA = Math.toDegrees(Math.acos(arccosineA));
    angleB = Math.toDegrees(Math.acos(arccosineB));
    angleC = Math.toDegrees(Math.acos(arccosineC));
  }

  /**
   * Calculate triangle perimeter
   */
  public void calculatePerimeter() {
    perimeter = sideLengthA + sideLengthB + sideLengthC;
  }

  /**
   * Sketch the triangle shape
   * @param g the painter
   */
  public void paintComponent(Graphics g) {
    Graphics2D painter = (Graphics2D) g;

    //Construct triangle sides
    Line2D.Double sideA = new Line2D.Double(point1, point2);
    Line2D.Double sideB = new Line2D.Double(point2, point3);
    Line2D.Double sideC = new Line2D.Double(point1, point3);

    //Draw the sides
    painter.draw(sideA);
    painter.draw(sideB);
    painter.draw(sideC);

    //Get the mid-point of each side
    float xMidPointA = (float) (xOrdinate1 + xOrdinate2) / 2;
    float yMidPointA = (float) (yOrdinate1 + yOrdinate2) / 2;
    float xMidPointB = (float) (xOrdinate2 + xOrdinate3) / 2;
    float yMidPointB = (float) (yOrdinate2 + yOrdinate3) / 2;
    float xMidPointC = (float) (xOrdinate1 + xOrdinate3) / 2;
    float yMidPointC = (float) (yOrdinate1 + yOrdinate3) / 2;

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

    painter.setFont(new Font("angleFont", Font.BOLD, 12));
    painter.drawString("A", (float) xOrdinate3, (float) yOrdinate3);
    painter.drawString("B", (float) xOrdinate1, (float) yOrdinate1);
    painter.drawString("C", (float) xOrdinate2, (float) yOrdinate2);
  }

  /**
   * Calculate the area of triangle using Heron's formula
   * S is the semi-perimeter of the triangle
   */
  public void calculateArea() {
    final double S = ((sideLengthA + sideLengthB + sideLengthC) / 2);
    area = Math.sqrt((S * (S - sideLengthA) * (S - sideLengthB) * (S - sideLengthC)));
  }

  /**
   * Get point 1 coordinate
   * @return the coordinate
   */
  public Point2D.Double getPoint1() {
    return point1;
  }

  /**
   * Get point 2 coordinate
   * @return the coordinate
   */
  public Point2D.Double getPoint2() {
    return point2;
  }

  /**
   * Get point 3 coordinate
   * @return the coordinate
   */
  public Point2D.Double getPoint3() {
    return point3;
  }

```

```java
/**
 * Get side A length
 * @return the side length
 */
public double getSideLengthA() {
  return sideLengthA;
}

/**
 * Get side B length
 * @return the side length
 */
public double getSideLengthB() {
  return sideLengthB;
}

/**
 * Get side C length
 * @return the side length
 */
public double getSideLengthC() {
  return sideLengthC;
}

/**
 * Get point A angle
 * @return the angle
 */
public double getAngleA() {
  return angleA;
}
```

```java

  /**
   * Get point B angle
   * @return the angle
   */
  public double getAngleB() {
    return angleB;
  }

  /**
   * Get point C angle
   * @return the angle
   */
  public double getAngleC() {
    return angleC;
  }

  /**
   * Get the triangle perimeter
   * @return the perimeter
   */
  public double getPerimeter() {
    return perimeter;
  }

  /**
   * Get the triangle area
   * @return the area
   */
  public double getArea() {
    return area;
    }
}
```

```java
package EKPL.Chapter4.PlainString;

import javax.swing.*;

/**
 * Created by Sheldon on 11/1/2016.
 * E4.12
 * A program that extracts the numbers without comma(,)
 */
public class PlainString {
  public static void main(String[] args) {
    System.out.printf("%60s%n%n", "A program that extracts the numbers without comma(,)");
    //Prompt the user to input a number
    String theInput = JOptionPane.showInputDialog("Please input an integer between 1,000 - 999,999:");
    int inputLength = theInput.length();

    //Extract the numbers by elliminating the ','
    String firstExtract = theInput.substring(0, theInput.indexOf(","));
    String secondExtract = theInput.substring(theInput.indexOf(",") + 1, inputLength);
    String extractedInput = firstExtract + secondExtract;

    //Display the result
    System.out.printf("%35s%5s%10s%n", "User's input integer", ": ", theInput);
    System.out.printf("%32s%8s%10s%n", "Extracted integer", ": ", extractedInput);
  }
}

```

```java
package EKPL.Chapter4.PlainString;

import javax.swing.*;

/**
 * Created by Sheldon on 11/2/2016.
 */
public class PlainStringBasic {
  public static void main(String[] args) {
    String theInput = JOptionPane.showInputDialog("Please input an integer between 1,000 - 999,999:");
    int inputLength = theInput.length();

    String firstExtract = theInput.substring(0, inputLength - 4);
    String secondExtract = theInput.substring(inputLength - 3, inputLength);

    System.out.println(inputLength);
    System.out.println(firstExtract);
    System.out.println(secondExtract);
  }
}

```

```java
package EKPL.Chapter4.LargeLetters;

/**
 * Created by Sheldon on 11/1/2016.
 * E4.17
 * A program that prints a string horizontally or vertically
 */
public class LargeName {
  public static void main(String[] args) {
    //Constuct the letters
    final String LETTER_D = "****\n" +
            "*    *\n" +
            "*     *\n" +
            "*     *\n" +
            "*     *\n" +
            "*    *\n" +
            "****\n";
    final String LETTER_I = "*******\n" +
            "   *\n" +
            "   *\n" +
            "   *\n" +
            "   *\n" +
            "   *\n" +
            "*******\n";
    final String LETTER_M = "****     ****\n" +
            "*   *   *   *\n" +
            "*    * *    *\n" +
            "*     *     *\n" +
            "*           *\n" +
            "*           *\n" +
            "*           *\n";
    final String LETTER_A = "      **\n" +
            "     *  *\n" +
            "    *    *\n" +
            "   ********\n" +
            "  *        *\n" +
            " *          *\n" +
            "*            *\n";

    final String LETTER_S = "  ***********\n" +
            " *\n" +
            "*\n" +
            "  ***********\n" +
            "             *\n" +
            "           *\n" +
            "***********";
    //Print a string vertically
    System.out.println(LETTER_D);
    System.out.println(LETTER_I);
    System.out.println(LETTER_M);
    System.out.println(LETTER_A);
    System.out.println(LETTER_S);

    //Print a string horizontally
    System.out.println("****    *******  ****     ****        **           ***********\n" +
            "*    *     *     *   *   *   *       *  *         *\n" +
            "*     *    *     *    * *    *      *    *       *\n" +
            "*     *    *     *     *     *     ********        ***********\n" +
            "*     *    *     *           *    *        *                  *\n" +
            "*    *     *     *           *   *          *               *\n" +
            "****    *******  *           *  *            *  ***********");
  }
```

```
}

```

```java
package EKPL.Chapter4.DepositGrowth;

import javax.swing.*;
import java.text.DecimalFormat;

/**
 * Created by Sheldon on 11/2/2016.
 * A program that simulates the growth of deposit balance
 */
public class DepositGrowthSimulator {
  private double balance;
  private double rate;

  /**
   * Construct the account object
   *
   * @param initialBalance the initial balance
   * @param annualRate     the annual interest rate
   */
  public DepositGrowthSimulator(double initialBalance, double annualRate) {
    balance = initialBalance;
    rate = annualRate;
  }

  /**
   * Calculate the balance growth
   */
  public void simulateGrowth() {
    final double MONTHLY_RATE = (rate / 12.0) / 100.0;
    balance = balance + (balance * MONTHLY_RATE);
  }

  /**
   * Get the current balance
   *
   * @return the balance
   */
  public double getBalance() {
    return balance;
  }


  public static void main(String[] args) {
    //Prompt the user to input data
    System.out.printf("%45s%n%n", "A program that simulates deposit growth");
    double initialBalance = Double.parseDouble(JOptionPane.showInputDialog("Please input" +
            " your account inital balance ($):"));
    double annualRate = Double.parseDouble(JOptionPane.showInputDialog("Please input" +
            " the annual interest rate (%)"));
    int nMonth = Integer.parseInt(JOptionPane.showInputDialog("Please input time period of months"));

    //Construct the account object to simulate
    DepositGrowthSimulator myAccount = new DepositGrowthSimulator(initialBalance, annualRate);

    //Display tha simulation result
    System.out.printf("%29s%5s%.2f%n", "Your account balance", ": $", initialBalance);
    System.out.printf("%29s%4s%.1f%s%n", "Annual interest rate", ": ", annualRate, "%");

    //Simulate the balance growth in specified period of time
    for (int i = 0; i < nMonth; i++) {
      myAccount.simulateGrowth();
      double balance = Double.parseDouble(new DecimalFormat(".##").format(myAccount.getBalance()));
```

```
        System.out.printf("%20s%s%13s%,.2f%n", "After month", (i + 1), ": $", balance);
      }
    }
}
```

```java
package EKPL.Chapter4.TicTacToeGrid;

/**
 * Created by Sheldon on 11/1/2016.
 * E4.14
 * A program that builds the tic-tac-toe grid
 */
public class TicTacToeGrid {
  public static void main(String[] args) {
    //Construct the patterns
    final String COMB_SHAPE = "+--+--+--+\r\n|  |  |  |";
    final String BOTTOM_LINE = "+--+--+--+";

    //Print the combined patterns
    System.out.println(COMB_SHAPE);
    System.out.println(COMB_SHAPE);
    System.out.println(COMB_SHAPE);
    System.out.println(BOTTOM_LINE);
  }
}

```

```java
package EKPL.Chapter4.CircularObject;

import javax.swing.*;
import java.awt.*;
import java.awt.geom.Ellipse2D;
import java.awt.geom.Line2D;

/**
 * Created by Sheldon on 10/31/2016.
 * E4.7
 * A program that prompts the user for a radius then
 * calculate the area and circumference of a circle and
 * the volume and surface area of a sphere with the radius prompted
 */
public class CircularObject extends JComponent {
  private double radius;  //holds the radius
  private double area;  //holds the circle area
  private double circumference; //holds the circle circumference
  private double volume;  //holds the sphere volume
  private double surfaceArea; //holds the sphere surface area

  /**
   * Constructs a circular object with radius r
   *
   * @param r the radius
   */
  public CircularObject(double r) {
    radius = r;
  }

  /**
```

```java
   * Calculates the circular object area
   */
  public void calculateArea() {
    area = Math.PI * radius * radius;
  }

  /**
   * Calculates the circular object circumference
   */
  public void calculateCircumference() {
    final double MULTIPLIER = 2.0;
    circumference = MULTIPLIER * Math.PI * radius;
  }

  /**
   * Calculates the circular object volume
   */
  public void calculateVolume() {
    final double MULTIPLIER = 4.0 / 3.0;
    volume = MULTIPLIER * Math.PI * radius * radius * radius;
  }

  /**
   * Calculates the circular object surface area
   */
  public void calculateSurfaceArea() {
    final double MULTIPLIER = 4.0;
    surfaceArea = MULTIPLIER * Math.PI * radius * radius;
  }

  /**
```

```java
     * Function that sketches the circular object
     *
     * @param g the painter
     */
   public void paintComponent(Graphics g) {
     Graphics2D painter = (Graphics2D) g;

     // Construct the circular object shapes
     final double INITIAL_POSITION = 50.0;
     Ellipse2D.Double circle = new Ellipse2D.Double(INITIAL_POSITION, INITIAL_POSITION, radius, radius);
     Ellipse2D.Double sphere = new Ellipse2D.Double(INITIAL_POSITION * 2 + radius, INITIAL_POSITION, radius, radius);
     Ellipse2D.Double ellipse = new Ellipse2D.Double(INITIAL_POSITION * 2 + radius, INITIAL_POSITION + (radius / 2.5),
             radius, radius / 5);

     // Sketch the circular object shapes
     painter.draw(circle);
     painter.draw(sphere);
     painter.setColor(Color.gray);
     painter.draw(ellipse);

     // Construct and sketch the circle radius
     painter.setColor(Color.RED);
     painter.setFont(new Font("Annotation", Font.ITALIC, 14));
     Line2D.Double circleRadius = new Line2D.Double(INITIAL_POSITION + (radius / 2), INITIAL_POSITION + (radius / 2),
             INITIAL_POSITION + radius, INITIAL_POSITION + (radius / 2));
     painter.draw(circleRadius);
     painter.drawString("r = " + radius, (float) (INITIAL_POSITION + (radius * 0.6)),
             (float) (INITIAL_POSITION + (radius / 2)));
```

```

    // Construct and sketch the sphere radius
    Line2D.Double sphereRadius = new Line2D.Double(INITIAL_POSITION * 2 + (radius * 1.5),
            INITIAL_POSITION + (radius / 2), INITIAL_POSITION * 2 + (radius * 2), INITIAL_POSITION + (radius
/ 2));
    painter.draw(sphereRadius);
    painter.drawString("r = " + radius, (float) (INITIAL_POSITION * 2 + (radius * 1.6)),
            (float) (INITIAL_POSITION + (radius / 2)));

    // Add the radius values
    painter.setColor(Color.BLACK);
    painter.setFont(new Font("Annotation1", Font.BOLD, 14));
    painter.drawString("Circle", (float) (INITIAL_POSITION + radius * 0.415),
            (float) (INITIAL_POSITION + radius + 20));
    painter.drawString("Sphere", (float) (INITIAL_POSITION * 2 + (radius * 1.405)),
            (float) (INITIAL_POSITION + radius + 20));
  }

  /**
   * Get the circle object area
   *
   * @return the area
   */
  public double getArea() {
    return area;
  }

  /**
   * Get the circle object circumference
   *
   * @return the circumference
```

```
     */
    public double getCircumference() {
      return circumference;
    }

    /**
     * Get the circle object volume
     *
     * @return the volume
     */
    public double getVolume() {
      return volume;
    }

    /**
     * Get the circle object surface area
     *
     * @return the surface area
     */
    public double getSurfaceArea() {
      return surfaceArea;
    }
}
```

```java
package EKPL.Chapter4.CircularObject;

import javax.swing.*;
import java.awt.*;

/**
 * Created by Sheldon on 10/31/2016.
 * E4.7
 * A program that simulates the calculation of circular object
 * properties by radius prompted
 * Sketches the shape of circular object
 */
public class CircularObjectMain {
  public static void main(String[] args) {

    // Prompt the user to input the radius
    String inputRadius = JOptionPane.showInputDialog("Set the object's radius (units)");
    double radius = Double.parseDouble(inputRadius);
    System.out.printf("Circular object radius %13s%,14.2f%6s%n", "(r) =", radius, "units");

    // Construct the circular object
    CircularObject theCircularObject = new CircularObject(radius);

    // Calculate the circular object properties
    theCircularObject.calculateCircumference();
    theCircularObject.calculateArea();
    theCircularObject.calculateSurfaceArea();
    theCircularObject.calculateVolume();

    double circumference = theCircularObject.getCircumference();
    double area = theCircularObject.getArea();
    double surfaceArea = theCircularObject.getSurfaceArea();
    double volume = theCircularObject.getVolume();

    // Display the results
    System.out.printf("Circular object circumference %6s%,14.2f%6s%n", "(C) =", circumference, "units");
    System.out.printf("Circular object area %15s%,14.2f%13s%n", "(A) =", area, "square units");
    System.out.printf("Circular object surface area %7s%,14.2f%13s%n", "(A) =", surfaceArea, "square units");
    System.out.printf("Circular object volume %13s%,14.2f%11s%n", "(V) =", volume, "cube units");

    // Display the sketch in a new window
    JFrame theFrame = new JFrame();
    theFrame.setSize(600, 400);
    theFrame.getContentPane().setBackground(Color.WHITE);
    theFrame.setTitle("Circular Object");
    theFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    theFrame.setVisible(true);
    theFrame.add(theCircularObject);
  }
}
```