Jul 28 21:44 2004 ShapeInt.java Page 1

```
package drawableShapes;

/** ShapeInt.java
 *  @author Koffman & Wolfgang
 * */

public interface ShapeInt {
  double computeArea();

  double computePerimeter();

  void readShapeData();
}
```

Jul 29 07:44 2004 Rectangle.java Page 1

```
package drawableShapes;
import javax.swing.*;

/** Represents a rectangle.
    Implements ShapeInt.
    @author Koffman & Wolfgang
 */

public class Rectangle
    implements ShapeInt {

  // Data Fields
  /** The width of the rectangle. */
  private int width = 0;
  /** The height of the rectangle. */
  private int height = 0;

  // Constructors
  /** Constructs a rectangle of the specified size.
      @param wid the width
      @param hei the height
   */

  /**** BEGIN EXERCISE ****/
  public Rectangle() {
    width = 0;
    height = 0;
  }

  /**** END EXERCISE ****/

  public Rectangle(int wid, int hei) {
    width = wid;
    height = hei;
  }

  // Methods
  /** Get the width.
      @return The width
   */
  public int getWidth() {
    return width;
  }

  /** Get the height.
      @return The height
   */
  public int getHeight() {
    return height;
  }

  /** Compute the area.
      @return The area of the rectangle
   */
  public double computeArea() {
    return height * width;
```

```
  }

  /** Compute the perimeter.
      @return the perimeter of the rectangle
   */
  public double computePerimeter() {
    return 2 * (height + width);
  }

  /** Read the parameters of the rectangle. */
  public void readShapeData() {
    String widthStr = JOptionPane.showInputDialog
        ("Enter the width of the Rectangle");
    width = Integer.parseInt(widthStr);
    String heightStr = JOptionPane.showInputDialog
        ("Enter the height of the Rectangle");
    height = Integer.parseInt(heightStr);
  }

  /** Create a string representaion of the rectangle.
      @return A string representation of the rectangle
   */
  public String toString() {
    return "Rectangle: width is " + width + ", height is " + height;
  }

}
```



```
package drawableShapes;

import javax.swing.*;

/** Circle.java
    Represents a circle.
    Implements ShapeInt.
    @author Koffman and Wolfgang
 */
public class Circle
    implements ShapeInt {

// *** BEGIN EXERCISE ***
  private int radius;

  public Circle() {

  }

  public Circle( int rad ) {
    radius = rad;
  }

  public void setRadius( int rad ) {
    radius = rad;
  }

  public int getRadius() {
    return radius;
  }

  public double computeArea() {
    return radius * radius * Math.PI;
  }

  public double computePerimeter() {
    return 2 * radius * Math.PI;
  }

  public String toString() {
    return "Circle radius is " + radius;
  }

  public void readShapeData() {

  }

// *** END EXERCISE ***

}
```

```
package drawableShapes;
import javax.swing.*;

/**** BEGIN EXERCISE ****/
/** RtTriangle.java
    Represents a right triangle.
    Implements ShapeInt.
    @author Koffman and Wolfgang
 */
public class RtTriangle
    implements ShapeInt {

  // Data Fields
  /** The base of the triangle */
  private int base = 0;
  /** The height of the triangle */
  private int height = 0;

  // Constructors
  /** Construct a default right triangle
   */
  public RtTriangle() {}

  /** Construct a right triangle of the specified size
      @param bas the base
      @param hei the height
   */
  public RtTriangle(int bas, int hei) {
    base = bas;
    height = hei;
  }

  // Mehods
  /** Get the base
      @return the base
   */
  public int getBase() {
    return base;
  }

  /** Get the height
      @return the height
   */
  public int getHeight() {
    return height;
  }

  /** Compute the area
      @return the area of the right triangle
   */
  public double computeArea() {
    return 0.5 * height * base;
  }

  /** Compute the perimeter
      @return the preimeter of the right triangle
```



```
   */
  public double computePerimeter() {
    double hypotenuse = Math.sqrt(height * height
                                  + base * base);
    return hypotenuse + base + height;
  }

  /** Read the parameters of the right triangle */
  public void readShapeData() {
    String baseStr = JOptionPane.showInputDialog
        ("Enter the base of the Right Triangle");
    base = Integer.parseInt(baseStr);
    String heightStr = JOptionPane.showInputDialog
        ("Enter the height of the Right Triangle");
    height = Integer.parseInt(heightStr);
  }

  /** Create a string representation of the right triangle
      @return a string representation of the right triangle
   */
  public String toString() {
    return "Right Triangle: base is " + base + ", height is "
        + height;
  }

}
/**** END EXERCISE ****/
```

Jul 28 21:44 2004 DrawableInt.java Page 1

```java
package drawableShapes;

import java.awt.*;

/** DrawableInt.java
 * @author Koffman & Wolfgang
 * */

public interface DrawableInt {
  void drawMe(Graphics g);

  void setBorderColor(Color border);

  void setInteriorColor(Color inter);

  void setPos(Point p);
}
```

Jul 29 07:53 2004 Drawable.java Page 1

```java
package drawableShapes;
import java.awt.*;

/** This class encapsulates the common attributes of drawable objects.
 *  @author Koffman & Wolfgang
 */
public class Drawable {
  // Data Fields
  /** The position of the origin of the shape. */
  protected Point pos = new Point(0, 0);
  /** The border color. */
  protected Color borderColor = Color.black;
  /** The interior color. */
  protected Color interiorColor = Color.white;

  // Constructors
  /** Construct a Drawable with the specified parameters.
        @param poi The position of the origin
        @param border The border color
        @param inter The interior color
   */
  public Drawable(Point poi, Color border, Color inter) {
    pos = poi;
    borderColor = border;
    interiorColor = inter;
  }

  /** Create a string representation of this class.
        @return A string representation of this class
   */
  public String toString() {
    return "\nx coordinate is " + pos.x
        + ", y coordinate is " + pos.y
        + "\nborder color is " + borderColor
        + "\ninterior color is " + interiorColor;
  }
}
```

Jul 28 21:44 2004 DrawableShape.java Page 1

```java
package drawableShapes;
import java.awt.*;

/** DrawableShape.java
 * @author Koffman & Wolfgang
 * */
public abstract class DrawableShape
    extends Drawable
    implements ShapeInt, DrawableInt {
  // Data Fields
  /** Reference to the shape object. */
  protected ShapeInt theShape;

  // Constructor
  /** Construct a DrawableShape with the given parameters.
        @param poi The position of the origin
        @param border The border color
        @param inter The interior color
        @param aShape The shape
   */
  public DrawableShape(Point poi, Color border, Color inter,
                       ShapeInt aShape) {
    super(poi, border, inter);
    theShape = aShape;
  }

  // Delegated Methods
  /** Compute the area.
        @return The area
   */
  public double computeArea() {
    return theShape.computeArea();
  }

  /** Compute the perimeter.
        @return The perimeter
   */
  public double computePerimeter() {
    return theShape.computePerimeter();
  }

  /** Read the size parameters from the user. */
  public void readShapeData() {
    theShape.readShapeData();
  }
}
```

Sep 16 06:42 2005 DrawableRectangle.java Page 1

```java
package drawableShapes;
import java.awt.*;

/** Represents a drawable rectangle.
    Implements ShapeInt and DrawableInt.
    @author Koffman & Wolfgang
 */

public class DrawableRectangle
    extends DrawableShape {

  // Constructors
  /** Construct a rectangle of the given size, position, and color.
        @param wid The width
        @param hei The height
        @param poi The coordinates of the upper left corner
        @param border The border color
        @param inter The interior color
   */
  public DrawableRectangle(int wid, int hei,
                           Point poi, Color border, Color inter) {
    super(poi, border, inter, new Rectangle(wid, hei));
  }

  /**** BEGIN EXERCISE ****/
  public void setBorderColor( Color bor ) {}

  public void setInteriorColor( Color inter ) {}

  public void setPos( Point p ) {}
  /**** END EXERCISE ****/


  /** Draw the rectangle.
        @param g The graphics context (screen)
   */
  public void drawMe(Graphics g) {
    g.setColor(interiorColor);
    Rectangle rectangle = (Rectangle) theShape;
    g.fillRect(pos.x, pos.y,
               rectangle.getWidth(), rectangle.getHeight());
    g.setColor(borderColor);
    g.drawRect(pos.x, pos.y,
               rectangle.getWidth(), rectangle.getHeight());
  }

  public String toString() {
    return "Drawable " + theShape + super.toString();
  }
}
```

Sep 16 06:42 2005 DrawableCircle.java Page 1

```
package drawableShapes;
import java.awt.*;

/** Represents a drawable circle
    Implements ShapeInt and DrawableInt.
    @author Koffman & Wolfgang
 */
public class DrawableCircle
    extends DrawableShape {

  // Constructors
  /** Construct a DrawableCircle of the given radius,
        position, and color.
        @param rad The radius
        @param poi The coordinates of the center
        @param border The background color
        @param inter The interior color
   */
  public DrawableCircle(int rad,
                        Point poi, Color border, Color inter) {
    super(poi, border, inter, new Circle(rad));
  }

  // Methods
  /** Draw the circle.
        @param g The graphics context (screen)
   */
  public void drawMe(Graphics g) {
    g.setColor(interiorColor);
    Circle circle = (Circle) theShape;
    g.fillOval(pos.x - circle.getRadius(),
               pos.y - circle.getRadius(),
               2 * circle.getRadius(), 2 * circle.getRadius());
    g.setColor(borderColor);
    g.drawOval(pos.x - circle.getRadius(),
               pos.y - circle.getRadius(),
               2 * circle.getRadius(), 2 * circle.getRadius());
  }

  /**** BEGIN EXERCISE ****/
  public void setBorderColor( Color bor ) {}

  public void setInteriorColor( Color inter ) {}

  public void setPos( Point p ) {}
  /**** END EXERCISE ****/

  /** Return a string representation of the circle.
        @return A string representation
   */
  public String toString() {
    return "Drawable " + theShape + super.toString();
  }
}
```

Sep 16 06:42 2005 DrawableTriangle.java Page 1

```
package drawableShapes;
import java.awt.*;

/** Represents a drawable triangle.
    Implements ShapeInt and DrawableInt.
    @author Koffman & Wolfgang
 */
public class DrawableTriangle
    extends DrawableShape {

  // Constructor
  /** Construct a DrawableTriangle with the given color
        and position.
        @param bas The base
        @param hei The height
        @param poi The location of the lower left corner
        @param border The border color
        @param inter The interior color
   */
  public DrawableTriangle(int bas, int hei,
                          Point poi, Color border, Color inter) {
    super(poi, border, inter, new RtTriangle(bas, hei));
  }

  // Methods

  /**** BEGIN EXERCISE ****/
  public void setBorderColor( Color bor ) {}

  public void setInteriorColor( Color inter ) {}

  public void setPos( Point p ) {}

  /**** END EXERCISE ****/

  /** Draw the triangle using the given graphics context.
        @param g The graphics context
   */
  public void drawMe(Graphics g) {
    Polygon rtTri = new Polygon();
    RtTriangle triangle = (RtTriangle) theShape;
    rtTri.addPoint(pos.x, pos.y);
    rtTri.addPoint(pos.x, pos.y - triangle.getHeight());
    rtTri.addPoint(pos.x + triangle.getBase(), pos.y);
    g.setColor(interiorColor);
    g.fillPolygon(rtTri);
    g.setColor(borderColor);
    g.drawPolygon(rtTri);
  }

  /** Return a string representation of the triangle.
        @return A string representation of the triangle
   */
  public String toString() {
    return "Drawable " + theShape + super.toString();
  }
```

```
}
```



```
package drawableShapes;
import java.awt.*;
import javax.swing.*;

/** Draws a collection of drawable objects stored in an array.
    Uses DrawableRectangle, DrawableCircle,
    DrawableTriangle, and awt.
    @author Koffman & Wolfgang
 */
public class TestDrawFigures
    extends JPanel {

  /** An array of DrawableShapes. */
  Object[] drawableShapes;

  /** Create some drawable shapes and compute their areas and
      perimeters.
   */
  private TestDrawFigures() {
    Object[] drawableShapes = {
        new DrawableCircle(50, new Point(50, 50),
                           Color.BLUE, Color.GREEN),
        new DrawableRectangle(100, 200, new Point(100, 100),
                              Color.RED, Color.YELLOW),
        new DrawableTriangle(50, 100, new Point(250, 300),
                             Color.BLACK, Color.RED),
        new DrawableRectangle(60, 50, new Point(300, 200),
                              Color.ORANGE, Color.GRAY),
        new DrawableCircle(0, new Point(0, 0),
                           Color.BLACK, Color.WHITE),
        new DrawableRectangle(0, 0, new Point(0, 0),
                              Color.BLACK, Color.WHITE),
        new DrawableTriangle(0, 0, new Point(0, 0),
                             Color.BLACK, Color.WHITE),
        new DrawableCircle( -100, new Point(300, 350),
                           Color.BLACK, Color.WHITE),
        new DrawableRectangle( -50, 50, new Point(50, 400),
                              Color.GREEN, Color.RED)
    };

    // Save a reference to this array of objects.
    this.drawableShapes = drawableShapes;

    // Display three attributes of the drawable objects.
    for (int i = 0; i < drawableShapes.length; i++) {
      System.out.println("\n" + drawableShapes[i]);
      System.out.println
          ("Area: "
           + ( (ShapeInt) drawableShapes[i]).computeArea()
           + ", perimeter: "
           + ( (ShapeInt) drawableShapes[i]).computePerimeter());
    }
  }

  /** Main method. Create a frame and a TestDrawFigures object
      and place the TestDrawFigures into the frame.
```

```
      @param args The command line arguments \227 not used
   */
  public static void main(String[] args) {
    // Create a frame to display the figures.
    JFrame frame = new JFrame("Test Draw Figures");
    frame.setSize(400, 450);
    // Create a collection of figures in the content pane.
    frame.setContentPane(new TestDrawFigures());
    // Display the frame and draw the figures.
    frame.show();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }

  /** Paint the drawable shapes.
      @param g The graphics environment
   */
  public void paintComponent(Graphics g) {
    // Paint the background.
    super.paintComponent(g);

    // Draw the drawable objects.
    for (int i = 0; i < drawableShapes.length; i++) {
      ( (DrawableInt) drawableShapes[i]).drawMe(g);
    }
  }
}
```



```
package drawableShapes;
import javax.swing.JOptionPane;

/** Computes the area and perimeter of selected figures.
    @author Koffman & Wolfgang
 */
public class ComputeAreaAndPerim {

  /** The main program performs the following steps.
        1. It asks the user for the type of figure.
        2. It asks the user for the characteristics of that figure.
        3. It computes the perimeter.
        4. It computes the area.
        5. It displays the result.
        @param args The command line arguments -- not used
   */
  public static void main(String args[]) {
    ShapeInt myShape;
    double perimeter;
    double area;
    myShape = getShape(); // Ask for figure type
    myShape.readShapeData(); // Read the shape data
    perimeter = myShape.computePerimeter(); // Compute perimeter
    area = myShape.computeArea(); // Compute the area
    displayResult(area, perimeter); // Display the result
    System.exit(0); // Exit the program
  }

  /** Ask the user for the type of figure.
        @return An instance of the selected shape
   */
  public static ShapeInt getShape() {
    String figType = JOptionPane.showInputDialog(
        "Enter C for circle\nEnter R for Rectangle\n"
        + "Enter T for rightTriangle");
    if (figType.equalsIgnoreCase("c"))
      return new Circle();
    else if (figType.equalsIgnoreCase("r"))
      return new Rectangle();
    else if (figType.equalsIgnoreCase("t"))
      return new RtTriangle();
    else
      return null;
  }

  /** Display the result of the computation.
        @param area The area of the figure
        @param perim The perimeter of the figures
   */
  private static void displayResult(double area, double perim) {
    JOptionPane.showMessageDialog(null, "The area is " + area
                                  + "\nThe perimeter is " + perim);
  }
}
```