```
/**
 * Abstract strategy to compute the area of a geometrical shape.
 * @author Dung X. Nguyen
 *
 * Copyright 1999 by Dung X. Nguyen - All rights reserved.
 *
 * Modifications by Alan L. Cox.
 */
public abstract class AShape
{
  /**
   * For concrete subclasses to provide code to compute the area.
   * @return this Shape's area.
   */
  public abstract double getArea();
}
```

```
/**
 * Concrete strategy to compute the area of a circular shape: knows
 * its own radius and how to compute its area.
 *
 * @author Dung X. Nguyen
 *
 * Copyright 1999 by Dung X. Nguyen - All rights reserved.
 *
 * Modifications by Alan L. Cox.
 */
public class Circle extends AShape
{
   private double _radius;

  /**
   * Initializes this Circle with a given radius.
   * @param radius the radius, >= 0.
   * @exception IllegalArgumentException if param < 0.
   */
  public Circle(double radius)
  {
    if (radius < 0) {
      throw new IllegalArgumentException("Circle.Circle(radius): radius < 0");
    }
    _radius = radius;
  }

  /**
   * @returns this Circle’s area.
   */
  public double getArea()
  {
    return Math.PI * _radius * _radius;
  }

  /**
   * @returns a String describing a Circle and its radius.
   */
  public String toString()
  {
    return "Circle (radius = " + _radius + ")";
  }
}
```

```
/**
 * The best little pizza house in Texas.
 *
 * @author Dung X. Nguyen
 *
 * Copyright 1999 by Dung X. Nguyen - All rights reserved.
 *
 * Modifications by Alan L. Cox.
 */
public class Pizza
{
  private double _price;

  /**
   * Knows how to compute its area.
   */
  private AShape _shape;

  /**
   * Initializes this Pizza to a given price and a given concrete shape.
   * @param price selling price, >= 0.
   * @param shape a concrete shape, != null.
   * @exception IllegalArgumentException, if params do not satisfy preconditions.
   */
  public Pizza(double price, AShape shape)
  {
    if (price < 0 || shape == null) {
      throw new IllegalArgumentException("Pizza.Pizza(price, shape): price < 0) or shape ==
null");
    }
    _price = price;
    _shape = shape;
  }

  /**
   * @return this Pizza's price in US$.
   */
  public double getPrice()
  {
    return _price;
  }

  /**
   * Delegates to the strategy to compute the area.
   * @return this Pizza's area.
   */
  public double getArea()
  {
    return _shape.getArea();
  }

  /**
   * Overrides the method inherited from the Object class, and uses
   * the strategy to obtain part of the string.
   * @return a String representation of this Pizza.
   */
  public String toString()
  {
    return ("Pizza Special: " + _shape.toString() + " All for $" + _price);
  }
}
```

```
/**
 * Simple test code for Pizza.
 *
 * @author Dung X. Nguyen
 *
 * Copyright 1999 by Dung X. Nguyen - All rights reserved.
 *
 * Modifications by Alan L. Cox.
 *
 * @dependency Pizza uses
 * @dependency Rectangle uses
 * @dependency Circle uses
 * @dependency AShape uses
 */
public class PizzaClient
{
  /**
   * Compares the price/area ratios of a round pizza and a rectangular pizza.
   * Illustrates how to wait for the user's input from the keyboard.
   */
  public static void main(String[] args)
  {
    Pizza cirPizza = new Pizza(4.69, new Circle(2.5));
    System.out.println(cirPizza);

    Pizza rectPizza = new Pizza(4.49, new Rectangle(6, 4));
    System.out.println(rectPizza);

    System.out.print("Round Pizza is a better deal than Rectangular Pizza: ");
    System.out.println((cirPizza.getPrice() / cirPizza.getArea() <
                        rectPizza.getPrice() / rectPizza.getArea()));

    /**
     * Wait for user to press enter:
     */
    try {
      System.out.print("Press enter to quit...");
      System.in.read();
    }
    catch (Exception e) {
    }
  }
}
```

```
/**
 * Concrete strategy to compute the area of a rectangular shape: knows
 * its own width and height and how to compute its area.
 *
 * @author Dung X. Nguyen
 *
 * Copyright 1999 by Dung X. Nguyen - All rights reserved.
 *
 * Modifications by Alan L. Cox.
 */
public class Rectangle extends AShape
{
  private double _height;
  private double _width;

  /**
   * Initializes this Rectangle width a given width and a given height
   * @param width width of this Rectangle, >= 0.
   * @param height height of this Rectangle, >= 0.
   * @exception IllegalArgumentException, if params do not satisfy preconditions.
   */
  public Rectangle(double width, double height)
  {
    if (width < 0 || height < 0) {
      throw new IllegalArgumentException("Rectangle.Rectangle(width, height): width < 0) or
height < 0.");
    }
    _height = height;
    _width  = width;
  }

  /**
   * @returns this Rectangle's area.
   */
  public double getArea()
  {
    return _height * _width;
  }

  /**
   * @returns a String describing a Rectangle with its width and height.
   */
  public String toString()
  {
    return "Rectangle (width = " + _width + ", heigth = " + _height + ")";
  }
}
```