// --------------------------------

 // ----- ENTER YOUR CODE HERE -----

 // --------------------------------

 // --------------------------------

 // --------- END USER CODE --------

 // --------------------------------

/\*\*

 Simple demo program that tests the stub methods of the Figure, Triangle,

 and Rectangle classes.

 \*/

public class FigureDemo {

 public static void main(String[] args) {

 Figure f1 = new Figure(7, 7);

 Figure t1 = new Triangle(5, 5, 5, 10);

 Figure r1 = new Rectangle(15, 15, 5, 10);

 System.out.println("Testing the draw() methods:");

 f1.draw();

 t1.draw();

 r1.draw();

 System.out.println();

 System.out.println("Testing the center() methods:");

 f1.center();

 t1.center();

 r1.center();

 System.out.println();

 System.out.println("Testing the erase() methods:");

 f1.erase();

 t1.erase();

 r1.erase();

 }

}

Consider a graphics system that has classes for various figures, say rectangles, boxes, triangles, circles and so on. For example, a rectangle might have data members height, width, and center point, while a box and circle might have only a center point and an edge length or radius respectively. In a well designed system these would be derived from a common class, Figure. You are to implement such a system.

The class Figure is the base class. You should add only Rectangle and Triangle classes derived from Figure. Each class has stubs for methods erase and draw. Each of these methods outputs a message telling the name of the class and what method has been called. Since these are just stubs, they do nothing more than output this message. The method center calls the erase and draw methods to erase and redraw the figure at the center. Since you have only stubs for erase and draw, center will not do any “centering” but will call the methods erase and draw which will allow you to see which versions of draw and center it calls. Also, add an output message in the method center that announces that center is being called. The methods should take no arguments. Also, define a demonstration program for your classes.

For a real example, you would have to replace the definition of each of these methods with code to do the actual drawing. You will be asked to do this in Programming Project 2.

In class Figure, you should define the following methods:

 public Figure(int centerX, int centerY)

 public void draw()

 public void erase()

 public void center()

 public int getCenterX()

 public void setCenterX(int centerX)

 public int getCenterY()

 public void setCenterY(int centerY)

 public String toString()

 public boolean equals(Object other)

In class Rectangle, you should define:

 public Rectangle(int centerX, int centerY, int width, int height)

 public void draw()

 public void erase()

 public int getWidth()

 public void setWidth(int width)

 public int getHeight()

 public void setHeight(int height)

 public String toString()

 public boolean equals(Object other)

In Triangle, you should define:

 public Triangle(int centerX, int centerY, int baseLength, int height)

 public void draw()

 public void erase()

 public int getBaseLength()

 public void setBaseLength(int baseLength)

 public int getHeight()

 public void setHeight(int height)

 public String toString()

 public boolean equals(Object other)

Since all Figures have a center point, the implementation of the center method in the Figure superclass could contain the logic for moving any Figure to the center of the display. It should therefore only be necessary to implement the center method in the Figure superclass, and not in any of the subclasses.

In order for your solution to compile and execute properly within CodeMate, you should *not* declare the Figure, Rectangle, and Triangle classes to be public. For example, instead of:

 public class Figure {

 // Implementation goes here...

 }

you should declare the class as:

 class Figure {

 // Implementation goes here...

 }

The constructors in your implementation of the Triangle and Rectangle subclasses should invoke the appropriate superclass constructor.