|  |  |
| --- | --- |
| **AP Computer Science** | **Lab14a Assignment** |
| **The Polymorphic Shapes**  **Graphics Program** | **80, 90 and 100 Point Versions** |
| **Assignment Purpose:**  The purpose of this program is to become comfortable with interfaces, implementing concrete classes and polymorphic methods. | |

Write a program which uses an **interface** and 4 *concrete, implementing* classes to generate an output graphics display of four shapes: a square, a triangle, an octagon and a circle.

The **Shape** interface contains one, two or three abstract, polymorphic methods, called **drawShape**, **displayName** and **displayNumSides**.

You are provided with the **Shape** interface and a partial testing program. The program requires four concrete classes that will implement the interface. You will also need to complete the testing program so that it will display the shapes, names and messages using polymorphism.

Six starting files are provided in the **Lab14ast** folder. The first file is **Shape.java**. This interface is complete and does not require any additions. The comments will need to be removed for the 80-point and the 100-point version.

|  |
| --- |
| **Shape.java Provided Student File** |
| // Shape.java // This is the student starting version.  import java.awt.\*;  public abstract interface Shape {  public abstract void drawShape(Graphics g);   // public abstract void displayName(Graphics g);   // public abstract void displayNumSides(Graphics g); } |

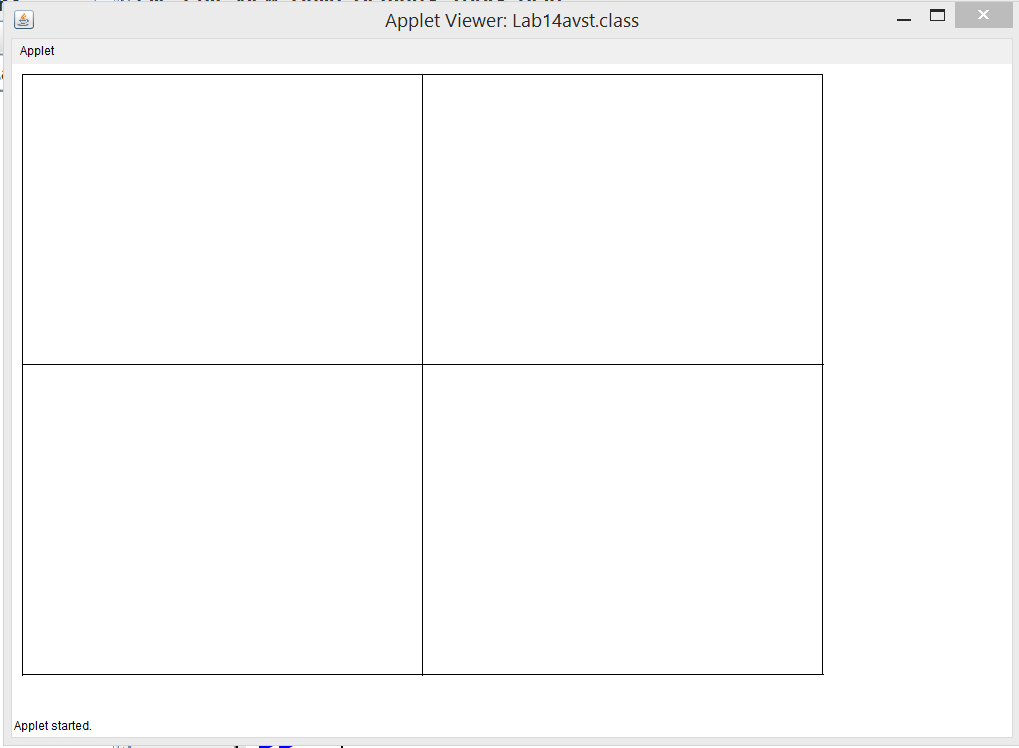
There are also four files for each one of the concrete classes that will implement the **Shape** interface. The classes are called: **Shape1Square, Shape2Triangle, Shape3Octagon** and **Shape4Circle**. As you see below, very little is finished in the classes. Each class may need to have proper data attributes, a constructor and the polymorphic methods implemented depending on the point version.

|  |
| --- |
| **Shape1Square.java Provided Student File There are also three more similar files:**  **Shape2Triangle.java, Shape3Octogan.java and Shape4Circle.java** |
| // Shape1Square.java // Lab14a // Student starting version  import java.awt.\*;  public class Shape1Square implements Shape {  public void drawShape(Graphics g)  {  } } |

The last file tests the program and displays the shapes. You need to declare a dynamic array of shapes, add objects to the array and display the shapes using polymorphism. The current testing program does compile and will display a grid of four cells where the shapes will be displayed.

|  |
| --- |
| **Lab14ast,java.java Provided Student File** |
| // Lab14avst.java // Lab14a // Student starting version  import java.awt.\*; import java.applet.\*;  public class Lab14avst extends Applet {  public void paint(Graphics g)  {  drawGrid(g);  }   public void drawGrid(Graphics g)  {  g.drawRect(10,10,800,600);  g.drawLine(10,300,810,300);  g.drawLine(410,10,410,610);  } } |

The student starting files do compile and execute. Right now they will only display a grid of four cells for each one of the four shapes.

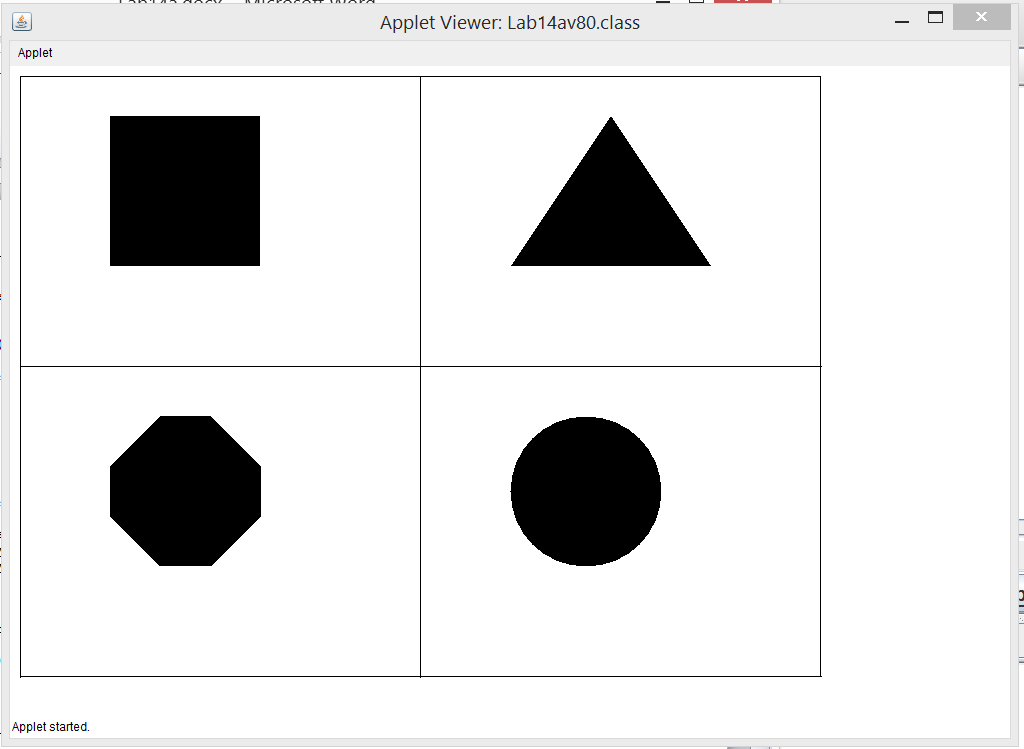


The **Shape** interface contains these 3 **abstract** methods in the table below. Two of the methods are commented out for the 80-point version. Make sure to remove the comments for the 90-point and 100-point versions.

|  |  |
| --- | --- |
| **drawShape** | Draws the square, triangle, octagon or circle on the screen |
| **displayName** | Displays the title at the top of the cell |
| **displayNumSides** | Displays the message at the bottom of the cell which indicate how many sides the shape has. |

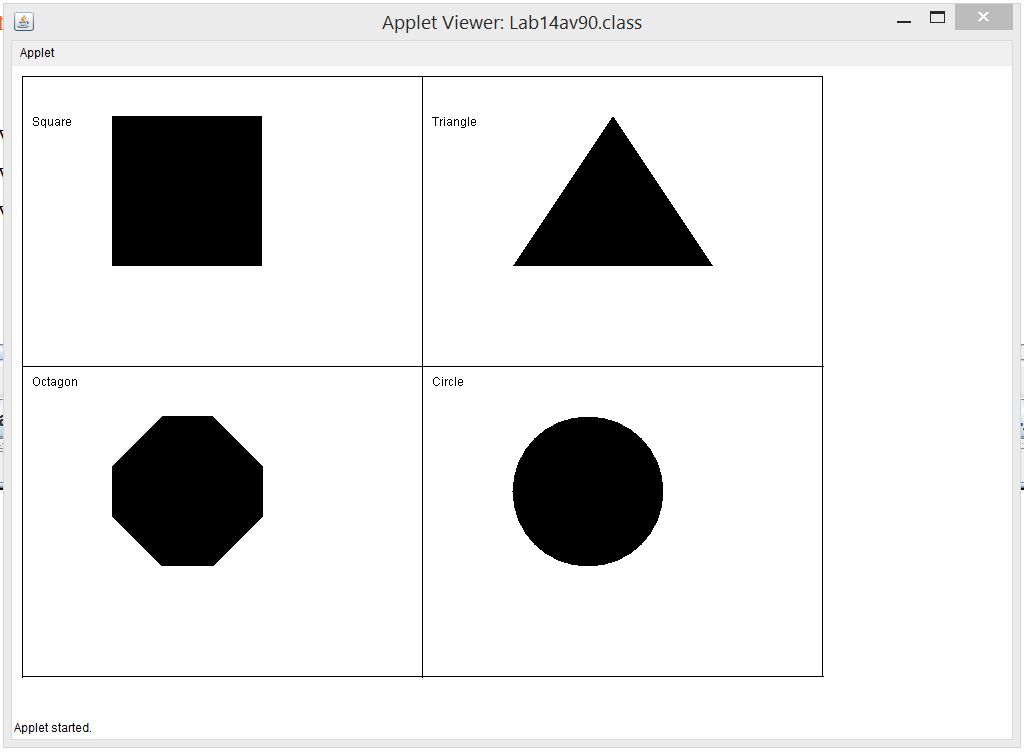
**80 Point Version Specifics and Output**

The 80-Point version requires implementing each one of the four concrete **Shape** classes. A constructor is not needed, nor are there any data attributes. This version completes each individual concrete class by implementing the **drawShape** method only. The geometric shapes are roughly drawn in the center of each cell as the output display shows below. The testing program needs to create an array of **Shape** objects and use this array to display each object.



**90 Point Version Specifics**

The 90-point version implements all four of the concrete **Shape** classes. Data attributes are needed for the title of each shape and the (x,y) coordinate position of the title. A constructor is required to initialize the data attributes. This version implements the **drawShape** and the **drawName** methods. The shape title are displayed in the top-left areas of the cell, shown below. The testing program needs to create an array of **Shape** objects and use this array to display each object and the tile of each shape.



**100 Point Version Specifics**

The 100-point version implements all four of the concrete **Shape** classes. Data attributes are needed for the title of each shape, the (x,y) coordinate position of the title, message describing the shape and the (x,y) coordinate position of the message. A constructor is required to initialize the data attributes. This version implements the **drawShape**, the **drawName** and the **displayNumSides** methods. The shape titles are displayed in the top-left areas of the cell and the shape message is displayed below each shape as shown below. The testing program needs to create an array of **Shape** objects and use this array to display each object, the tile of each shape and the message for each shape.

