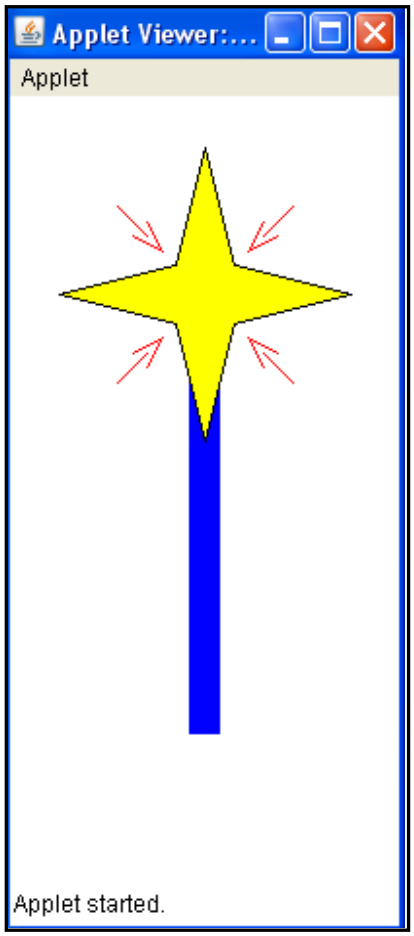
CIT 149: Java I
Chapter 7 Lab2

In this lab we will create programming project # 13 on page 567. When run the applet will display as:



Let's get started.

1. Open a new document in Textpad and save the program as MagicWand.java.
2. Type the code that import all classes in the swing and awt packages.
3. Type the class header that extends the JApplet class.

4. Type the following which will construct three integers that will be used to create the polygons that create the wand:

```
public static final int XBASE = 100;
public static final int YBASE = 100;
public static final int POINT = 75;
```

- Notice that the integers are listed as final, which means that their values cannot be altered.

5. Our first method is the drawStar() which will draw the star. Type:

```
public static void drawStar(Graphics canvas)
{
   int xValue[] = new int[9];
   int yValue[] = new int[9];

   xValue[0] = XBASE;
   yValue[0] = YBASE - POINT;
   xValue[1] = XBASE+POINT/5;
   yValue[1] = YBASE - POINT/5;
   xValue[2] = XBASE+POINT;
   yValue[2] = YBASE;
   xValue[3] = XBASE+POINT/5;
   yValue[3] = YBASE + POINT/5;
   xValue[4] = XBASE;
   yValue[4] = YBASE + POINT;
   xValue[5] = XBASE-POINT/5;
   yValue[5] = YBASE + POINT/5;
   xValue[6] = XBASE-POINT;
   yValue[6] = YBASE;
   xValue[7] = XBASE-POINT/5;
   yValue[7] = YBASE - POINT/5;
   xValue[8] = XBASE;
   yValue[8] = YBASE - POINT;
```

```
    canvas.setColor(Color.YELLOW);
    canvas.fillPolygon(xValue, yValue, 9);
    canvas.setColor(Color.BLACK);
    canvas.drawPolyline(xValue, yValue, 9);
}
```

- Two arrays are declared and each element in the arrays are given values.
- The color is set to yellow.
- The fillPolygon() method will create a solid polygon. This polygon will have 9 points designated by the two arrays. The ninth point is the point that closes the polgon. Review pages 520-522 on the fillPolgyon() and drawPolyline() methods.
- The color is now set to black, which will allow the line around the polygon to be black for a nice contrast.

6. The drawWand() method will draw the handle. Type:

```
public static void drawWand(Graphics canvas)
{

    int xValue[] = new int[5];
    int yValue[] = new int[5];

    xValue[0] = XBASE-POINT/9;
    yValue[0] = YBASE;
    xValue[1] = XBASE+POINT/9;
    yValue[1] = YBASE;
    xValue[2] = XBASE+POINT/9;
    yValue[2] = YBASE + 3*POINT;
    xValue[3] = XBASE-POINT/9;
    yValue[3] = YBASE + 3*POINT;
    xValue[4] = XBASE-POINT/9;
    yValue[4] = YBASE;
```

```
    canvas.setColor(Color.BLUE);
    canvas.fillPolygon(xValue, yValue, 5);
}
```

This is very similar to the code used to draw the star. Only 5 points are used to draw the wand though.

7. The drawAura() method, draws the lines pointing towards the star. Type:

```
public static void drawAura(Graphics canvas)
{

    int xValue[] = new int[3];
    int yValue[] = new int[3];

    xValue[0] = XBASE+4*POINT/10;
    yValue[0] = YBASE + 5*POINT/10;
    xValue[1] = XBASE+3*POINT/10;
    yValue[1] = YBASE + 3*POINT/10;
    xValue[2] = XBASE+5*POINT/10;
    yValue[2] = YBASE + 4*POINT/10;

    canvas.setColor(Color.RED);
    canvas.drawPolyline(xValue, yValue, 3);
    canvas.drawLine(XBASE+4*POINT/10, YBASE+4*POINT/10,
        XBASE+6*POINT/10, YBASE+6*POINT/10);

    xValue[0] = XBASE-4*POINT/10;
    yValue[0] = YBASE + 5*POINT/10;
    xValue[1] = XBASE-3*POINT/10;
    yValue[1] = YBASE + 3*POINT/10;
    xValue[2] = XBASE-5*POINT/10;
    yValue[2] = YBASE + 4*POINT/10;
```

```
    canvas.drawPolyline(xValue, yValue, 3);
    canvas.drawLine(XBASE-4*POINT/10, YBASE+4*POINT/10,
        XBASE-6*POINT/10, YBASE+6*POINT/10);

    xValue[0] = XBASE+4*POINT/10;
    yValue[0] = YBASE - 5*POINT/10;
    xValue[1] = XBASE+3*POINT/10;
    yValue[1] = YBASE - 3*POINT/10;
    xValue[2] = XBASE+5*POINT/10;
    yValue[2] = YBASE - 4*POINT/10;

    canvas.drawPolyline(xValue, yValue, 3);
    canvas.drawLine(XBASE+4*POINT/10, YBASE-4*POINT/10,
        XBASE+6*POINT/10, YBASE-6*POINT/10);

    xValue[0] = XBASE-4*POINT/10;
    yValue[0] = YBASE - 5*POINT/10;
    xValue[1] = XBASE-3*POINT/10;
    yValue[1] = YBASE - 3*POINT/10;
    xValue[2] = XBASE-5*POINT/10;
    yValue[2] = YBASE - 4*POINT/10;

    canvas.drawPolyline(xValue, yValue, 3);
    canvas.drawLine(XBASE-4*POINT/10, YBASE-4*POINT/10,
XBASE-
        6*POINT/10, YBASE-6*POINT/10);
}
```

- This is pretty simple to follow. A great deal simply changed previous values so the lines will be drawn in separate locations.

8. Our final method is the paint() method which does the actual drawing on the screen. Type:

```
public void paint(Graphics canvas)
```

```
{
    setSize(200,400);
    drawWand(canvas);
    drawAura(canvas);
    drawStar(canvas);
}
```

- The paint() method always has the same argument.
    - the canvas size is set
    - each of the other methods are invoked so they can do their drawing of the different parts of the wand.

9. Close the class and compile the program. Fix any errors if necessary.
10. Next create your html document and set the width to 200 and height to 400. Name the document MagicWand.html.
11. Next I want you to rename this class as MagicGUI.java and convert it to a GUI application. Try doing this on your own.
    - In converting this to a GUI application you do not have to specify the setSize() method since this is already specified in the paint() method.
    - Remember that since the paint method is painting things on the frame, the constructor method will not have any code within it.
    - The star may be a little high on the screen. This is easily remedied by changing the value of the YBASE constant to 125 instead of 100.
12. Compress the following files into a single zip or rar file and submit to the appropriate drop box:

    MagicWand.java
    MagicWand.class
    MagicWand.html
    MagicGUI.java
    MagicGUI.class