

READ ALL INSTRUCTIONS BEFORE STARTING YOUR SOLUTION.

Topics: analysis, structured, top-down **design**, expressions, variables, loops, class constants, formatting, Graphics, interactive programming (user input), selection (`if`), iteration (`for`), variables, methods, programming style, integer issues and basic analytic geometry.

Specifications:

1. You must follow all the style requirements in [Java_Program_Style_Requirements_GradingComponents.pdf](#) to earn full credit. That document describes the scoring process used and some design hints.
2. ****You must use the graphics package stored in the file DrawingPanel.java. You must put a copy of this file in the folder with your code and compile it before compiling your code.**
3. Your program must draw
 - a. in a window of the size selected by the user,
 - b. using the alternating colors selected by the user,
 - c. 5 of the **solid** (filled) shapes selected by the user.
4. In the console window, announce the name of the program and the author (you).
5. Prompt the user for the following items in **exactly the order listed**:
 - I. the width for the window (a square drawing panel).
 - II. 2 colors: **with separate prompts before each color**
 - III. then a shape – a **circle or a polygon** (these must be the only options at this point).
6. If the user chooses a circle, draw 5 solid, concentric circles, like a bulls eye, using the chosen colors. The outer circle should be tangent to the edges of the window. The radii of the circles should increase linearly, for example: 3, 6, 9, 12, 15.
7. **Only after the user chooses a polygon**, ask the user to choose between a square and a triangle. Use this nested decision tree only if the user has chosen a polygon, not a circle.
8. If the user asks for a square, draw 5 solid, concentric squares – a square bulls eye? The outer square should reach to within a pixel or few of the edges of the window. The dimensions of the squares should increase linearly (for example: 4, 8, 12, 16, 20).
9. If the user chooses a triangle, draw 5 solid, concentric isosceles triangles.
Alert: the text book does NOT tell you how to draw a closed, fill-able triangle.
You can create such a triangle as a polygon object:



```
Polygon myTriangle = new Polygon();
myTriangle.addPoint(x1, y1);
myTriangle.addPoint(x2, y2);
myTriangle.addPoint(x3, y3);
```

```
g.fillPolygon(myTriangle); // will fill the figure.
```

The base of the outer triangle should reach to within a pixel or few of the bottom of the window. Its top vertex should almost touch the middle of the top of the window. The dimensions of the triangles should increase linearly.

Triangles do not look equilateral due to the crt aspect ratio – **do not try to fix this**.

10. Your solution should let the user choose any 2 colors available in the set of 13 color constants that are built into the Color class--**EXCEPT BLACK**. Make it as easy for the user as you can to tell you the color -- within the limits of a text-based UI. Use ifs (and whiles). Do NOT ask the user to specify the RGB components of a color. **Your code will have to convert from the user text string into an object of type Color** .

11. Display the **name of the shape and your name**, for example, Triangle – Ed Hyde, across your drawing, in **BLACK**, **approximately** centered vertically. First study the drawstring examples in the book near p. 202. The steps are (1) decide what text you want to display—a String object, (2) create a new Font object, (3) decide the (x,y) coordinates for the start of the base of the text, (4) ask the Graphics object to draw that String. **Alert:** there is no information about centering the text in your book. For this assignment, use half the height of the panel as the y-coordinate; use 0 as the x-coordinate (left-aligned). For the font-size, experiment: try 2 or 3 different font sizes and see how they look. Then write a formula to relate the font size to the panel size. Remember to cast your result to int. Is there a lower limit? – by that I mean, is there a panel size so small that the computed font size won't work? Experiment! Then **include that limit into your instructions to the user – and check it in your code (bullet-proofing)**. Warning: There are no guarantees: Java assumes 72 dpi screen resolution but Windows uses 96 dpi or 120 dpi depending on the font size setting in the display properties (user's choice). It need not be beautiful, but must work on your system.

THIS is **NOT** required **and NOT** advised, but after you meet all other requirements and want to be more exact, see <http://www.leepoint.net/notes-java/GUI-appearance/fonts/18font.html>

This program will **include console and graphical output**. As before, user input will come from the console. As before, the console output should have a blank line at the beginning and end of the program, with the name of the program and your name at the top, and a blank line after that.

Your user interface should present just a few choices at a time – and only the ones that are appropriate at that time, based on the user's previous choice. Assume that the user understands the terms *height, width, polygon, circle, square, and triangle*. Remember that the user will not know how your program works, so **your prompts must tell the user what choices are appropriate** at each point in the program. Do that in a user-friendly way.

What if the user enters an unworkable response? A number that's too large or too small? A double instead of an integer? A String that doesn't match? Does case matter? Using the techniques covered so far, your program should try to avoid a crash. Convert or cast where appropriate, or ignore letter case, or provide a default value, or re-prompt or end with an appropriate (user friendly) message.

Use a **combination of planning and incremental refinement**. Your main method must clearly indicate the decomposition and overall flow of your program by calling a suitable set of well named methods (NOT head, body, trailer). Put prompts and error checking in methods that return values. Remember that methods should be short and focused. For designs use flowcharts or hierarchy chart or pseudo-code.

Test all choice options for each input. Print a screen capture of the drawing panel when the user has chosen a window of 400 pixels (wide x high), colors GRAY & WHITE, a triangle. The Graphics class expects `int(eger)s` so you may need to cast some calculated values before calling methods of that class. A Graphics object named `g` responds to `g.drawPolygon(p)` and `g.fillPolygon(p)` if you have constructed a Polygon named `p`.