ITSC 1212 – Assignment 2 – Pictures and Turtles!

The main idea behind this assignment is to use a turtle to programmatically draw on top of your own headshot.

Part 1 – Outline your face.
Part 2 – Add some features.
Part 3 – Add some captions.

Part 1 – Outline your face (10%)
In this part of the assignment, your job is to write a program called FaceOutline.java that will do the following:

- load in an image called headshot.jpg
- display the image as the background in a world
- create a turtle in the world
- make the turtle draw an outline around the face in the picture in thick red ink

Look at the Lab programs provided as a guide for how to create a new program. In order to do this assignment, you'll want to use the pictureObj.explore() method in the interactions pane in order to find appropriate coordinates on the picture to do the outlining. If you can't remember, see Lab 6 for details on how to use the explore() method.

Important: include the code that is in Appendix 1 at the beginning of your program so that the picture can be chosen with the file chooser or passed in as a command line parameter.

Part 2 – Add some features (40%)
In this part of the assignment, your job is to add **two** methods to the Turtle.java program that add features. Pick two of the following (using the exact names below):

- drawEyeglasses
- drawMustache
- drawGoatee
- drawEarrings
- drawBowtie
- drawHalo
- drawHorns

Your methods should take two location parameters for where the feature should be drawn, as well as a color parameter for the color the feature should be drawn with. Parameters must be in that order (x, y, color). Each method should meet the following conditions:

> **Pre-conditions**: assume turtle exists and a picture is in the world, assume turtle is facing up, with penDown.
> **Post-conditions**: the feature has been drawn over the picture in the specified color and the turtle ends at the bottom-right area of the feature,

facing right, with penUp. The feature is drawn with the bottom-left corner of the feature area at the x and y location passed in.
Return value: void

Once you have created the two methods in Turtle.java, test them out. Create a copy of your FaceOutline.java program and save it as FaceFeatures.java. Keep the face outline code, and add calls to your two new feature methods, each with a different color. You will need to write code to place your turtle in the right position, and orientation in order to make use of your feature methods. Again, you will probably want to use the pictureObj.explore() method to help you figure out where to place your facial features. You will be graded on putting the facial features in appropriate places (ie. a mustache up in the air above your head will lose points).

Important: include the code that is in Appendix 1 at the beginning of your program so that the picture can be chosen with the file chooser or passed in as a command line parameter.

Part 3 – Add some captions (30%)

In Assignment 1, you created some letters using the turtles. We have collated the letters into an alphabet and created the file TurtleAlphabet.java. Do not make changes to this file. This file gives you the ability to draw any letter of the alphabet in any color, bolded or not, and either small or large. Download TurtleAlphabet.java and save it in the same folder as Turtle.java. The letter methods in the TurtleAlphabet class are all static methods, so you don't need to create a TurtleAlphabet object, but you do need to pass in a reference to your turtle. You can call any of the letter methods using the following:

```
TurtleAlphabet.drawABoldSize(Turtle t, Color c, boolean bold, int size)
TurtleAlphabet.drawBBoldSize(Turtle t, Color c, boolean bold, int size)
…
TurtleAlphabet.drawZBoldSize(Turtle t, Color c, boolean bold, int size)
```

You should first experiment with calling the letter methods in the Interaction pane. Once you get a feel for how the letters work, your job will be to add some words or captions to your headshot image.

Create a new file (copied from FaceFeatures.java) called FaceCaptions.java. Keep the outline and facial features in the code. Add two new methods to this file, each of which will draw a different word or phrase (at least five letters in each phrase). These methods should be called drawPhrase1() and drawPhrase2(). These methods will have to be static methods, since they are called from the Main() method in FaceCaptions.java. Main methods are always static, and static methods can only call other static methods in the same class. Because these methods are static, you will need to pass your turtle object in to these methods.

Your methods should take a turtle parameter and meet the following conditions:

Pre-conditions: the turtle exists in the world, the turtle is facing up, with penDown and is at the bottom-left corner of the area where the feature will be drawn.
Post-conditions: the turtle has drawn a phrase or word with each letter in a specified (possibly different) color and the turtle ends at the bottom-right area of the phrase, facing right, with penUp
Return value: void

Finally, you will need to add some code to your Main() method to call these two new methods that you have created. You will also need to maneuver the turtle around so that the phrases are drawn in the areas where you want them to be drawn.

Important: include the code that is in Appendix 1 at the beginning of your program so that the picture can be chosen with the file chooser or passed in as a command line parameter.

Coding Style – (15%)
This grade is awarded for proper coding style. This includes:

- Appropriate method, variable, field, object and class names
- Proper indentation
- Good commenting (explains what code is doing)
- Well-organized, elegant solution

What to submit:

- FaceOutline.java
- FaceFeatures.java
- FaceCaptions.java
- Turtle.java
- headshot.jpg (the image you used for this assignment)

Appendix 1

You **must** have this code at the beginning of **each of your three program file main methods**, so that you can either choose your headshot picture, or a picture can be passed in as a command line parameter when the code is run (important for grading):

```
String filename;

if (args.length > 0) {
  // got a filename passed into program as a runtime parameter
  filename = args[0];
  System.out.println("Filename passed in: " + filename);

} else {
  // ask user for a picture
  filename = FileChooser.pickAFile();
  System.out.println("User picked file: " + filename);
}

// use the filename to create the picture object
 Picture pic = new Picture(filename);
```

Rubric:

Part 1 - Face Outlining	No picture imported, no outlining ***0points***	Picture imported, but no outlining ***3points***	Picture imported, some outlining done ***7points***	Picture imported, excellent outline of face. ***10points***
Part 2 - Facial Feature 1	No facial feature method added ***0points***	Facial feature method added, but no parameters ***5points***	Facial feature added with color parameter and location parameters, but facial feature is not well-placed ***10points***	Facial feature added with color and location parameters, well-placed. ***15points***
Part 2 - Facial Feature 1 Quality	No facial feature added. ***0points***	Facial feature is barely identifiable. ***2points***	Facial feature looks good. ***4points***	Facial feature looks great. ***5points***
Part 2 - Facial Feature 2	No facial feature method added ***0points***	Facial feature method added, but no parameters ***5points***	Facial feature added with color parameter and location parameters, but facial feature is not well-placed. ***10points***	Facial feature added with color and location parameters, well-placed. ***15points***
Part 2 - Facial Feature 2 Quality	No facial feature added. ***0points***	Facial feature is barely identifiable. ***0points***	Facial feature looks good. ***0points***	Facial feature looks great. ***5points***
Part 3 - Caption Phrase 1	No caption phrase method added. ***0points***	5+ letter caption drawn, but not using a method. ***5points***	5+ letter caption drawn using a static method, looks good. ***10points***	5+ letter caption drawn using a static method, looks great. ***15points***
Part 3 - Caption Phrase 2	No caption phrase method added. ***0points***	5+ letter caption drawn, but not using a method. ***5points***	5+ letter caption drawn using a static method, looks good. ***10points***	5+ letter caption drawn using a static method, looks great. ***15points***
Coding Style	Poor indentation, bad naming style, poor or no commenting ***0points***	Mediocre style, some okay commenting, indentation and naming, but lots that is problematic.	style, some okay commenting, indentation and naming, but lots that is problematic. 5points Good	Excellent style - appropriate commenting, good naming conventions, excellent indentation, clear

		5points	style, but solution is not as clear or organized as it could be ***10points***	and organized solution ***15points***
Excellence (most people will get zero, we will give at most 10% of the class these final five points, which demonstrate excellence). This is purely at the discretion of the TAs and Professors.	Normal effort ***0points***	Normal effort ***0points***	Normal effort ***0points***	Excellence - for effort above and beyond the assignment requirements. ***5points***