Object-Oriented Design
COMP 2000
Spring 2024

Lab 8 Supplement 1 (version 2)

First, some early guidance:

- You should look at the code in class `Game` and try to understand what it is doing as best you can. Ask questions! However, you should NOT edit this class in any way.

- If you look at the code for `Polygon` you will see that it makes significant use of `Point2D.Double` from the `java.awt.geom` package. You should definitely look up the API for this and read about this class so you understand what it represents and how it works.

- `Asteroids` is a subclass of `Game`. You will be adding instance (state) variables, modifying its constructor and `paintComponent()` method, and possibly adding additional utility methods.

- The `main()` method calls the constructor for `Asteroids`, which calls the method `paint()` which calls the method paintComponent(). There are some details about how the constructor also sets up a timer to call `repaint()`, which in turn calls `paint()`, which in turn calls `paintComponent()` which could be important for debugging, so they are explained in the next paragraph, but at a high level, you can think of what is happening in the code as:

```
Asteroids a = new Asteroids();
a.paint(...)
a.paint(...)
a.paint(...)
a.paint(...)
.
.
.
```

- This implicit loop of calling paint executes 30 times every second, allowing each call to paint to serve as a single frame in an animated sequence. This is what gives you the ability to create ships and asteroids, etc., that move over time.

- This paragraph will explain the details of what is actually happening when the Timer set up in the constructor calls `repaint()`, which is a method defined by Java in the class `Component`, which is the superclass of `Container`, which is the superclass of `JComponent`, which is the superclass of `JPanel`, which is the superclass of `Game`. So the `paintComponent` method of `Asteroids` will serve as the effective 'main method' of your game: this is where all the action occurs.

Now we are ready to outline the first steps:

- The first thing you should do is create the class Ship that is a subclass of Polygon.
  - It is ok to import individual classes from `java.awt` as you need them, but do not import `java.awt.*`
  - You will need a constructor that has the same `position` and `rotation` parameters in its signature as `Polygon`'s constructor. It should call `super()` as its first instruction. This means that it is up to the ship to specify it's own shape and pass this to `super()`. One easy way to do this is to have a private static method that returns the list of points that represents the ship. In this method you should instantiate and populate an array of points with the coordinates that you want to use to represent the shape of the ship.
  - You will also need a `paint()` method that takes a single `Graphics brush` parameter. This is the method that will display the ship on the canvas.
  - You will also need an `update()` method that is parameterless and returns `void`. You should leave this method with an empty body (just a pair of curly braces) for now. We will implement this method in supplement #2.
  - Before you draw the ship, make sure the paint method is set up correctly by using the `drawLine` method in `Graphics` to draw a single line between the coordinates (100,200) and (300,400). To do this, type `brush.drawLine(100,200,300,400);` in the `paint` method in `Ship`. Once you see that it works you can comment out this line.
  - Drawing the ship will require two steps. The Polygon class has been provided with a method `fill()` method that allows you to draw this interior of a polygon. It also has an `outline()` method that will draw the boundary of the polygon. If you notice, these methods cast the brush to a `Graphics2D` brush and then calls `Graphics2D` methods to do the drawing. It would be a good idea to look up the API for `Graphics2D` and study it.
- In order to test your ship class you will need to modify `Asteroids` to act as a client of `Ship`.
  - Declare a `Ship` instance variable.
  - In the constructor for `Asteroid`, build a ship object referenced by this variable. The initial position of the ship should be centered on the screen (so a point with coordinates (400, 300), but don't hard code these numbers in; rather, figure out how you can get the window dimensions divide those in half), and should be oriented facing east.
  - In the `paintComponents()` method for Asteroid, call the `paint()` method on the ship.

All work must be done individually. Never look at someone else's code. Please refer to the course policies if you have any questions about academic integrity. If you have trouble with the assignment, I am always available for assistance.