Assignment 5: Class Design and More GUI

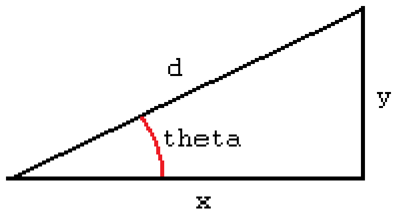
Due Friday October 2, 2009 11:55PM

For this assignment, you'll be making a program with two frames, one that displays a turtle, and another that controls the turtle's movement. **Please read all of the instructions before beginning.**

5.1: Turtle Class

The `Turtle` class will represent a turtle in our panel. Specs:

1. The class should be named `Turtle`
2. Instance variables to store the following, with the appropriate visibility / protection:
    a. The direction it's facing (as a `double`)
    b. Its x and y location
    c. An `ImageIcon` for its image
3. A constructor that takes in an initial x, y location and an initial direction (in degrees) for the turtle to face. The constructor should also make the `ImageIcon` from the provided turtle.gif file.
4. A method called "`turn`" that takes in (as a `double`) the number of degrees to turn the turtle and turns the turtle the number of degrees specified.
5. A method called "`move`" that takes in (as an `int`) the distance to move the turtle and moves the turtle the distance specified in the direction it is facing. Recall from trigonometry:

    The distance x is equal to d * cosine(theta) and the distance y is equal to d * sine(theta)

    Also, theta (in degrees) = theta (in radians) * 180 / pi

    theta (in radians) = theta (in degrees) * pi / 180

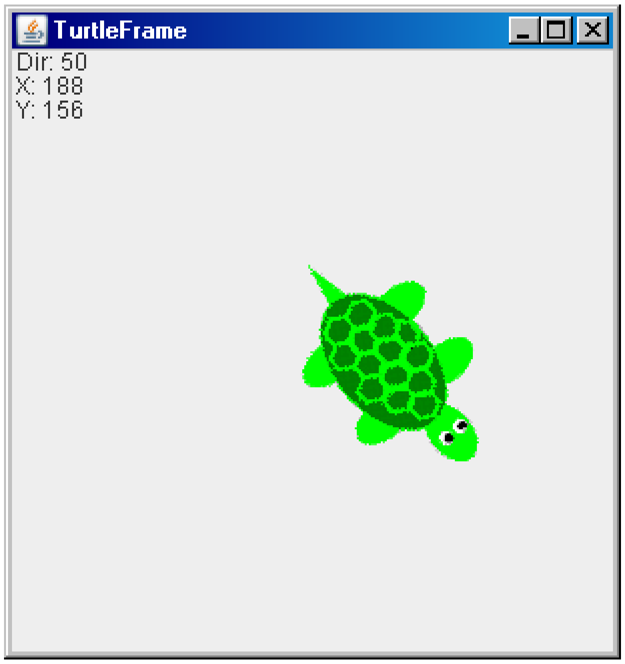
6. Getters for the turtle's x and y location, and its direction, using proper Java naming conventions for getters / setters.
7. A method called "`draw`" that takes in a `Graphics` object on which to draw the turtle. To achieve the rotation affect shown below in 5.2, you may use the method given below:

```java
import java.awt.geom.AffineTransform;

/**
 * This method will draw an image centered on the specified X,Y location,
 * rotated by the specified angle (in radians)
 *
 * @param g The graphics object on which to draw
 * @param icon The icon to draw
 * @param x The X location to draw the image.
 *          The center of the image is used as the anchor.
 * @param y The Y location to draw the image.
 *          The center of the image is used as the anchor.
 * @param rotation The amount to rotate the image. Positive is a clockwise
 *                 rotation. This should be in radians.
 */
public void drawImage(Graphics g, ImageIcon icon,
                      int x, int y, double rotation)
{
      Graphics2D g2 = (Graphics2D)g;
      AffineTransform af = new AffineTransform();
      af.translate(x - icon.getIconWidth() / 2,
                   y - icon.getIconHeight() / 2);
      af.rotate(rotation, icon.getIconWidth() / 2, icon.getIconHeight() / 2);
      g2.drawImage(icon.getImage(), af, null);
}
```

5.2: TurtlePanel Class

The `TurtlePanel` class will be responsible for drawing the turtle. For example:



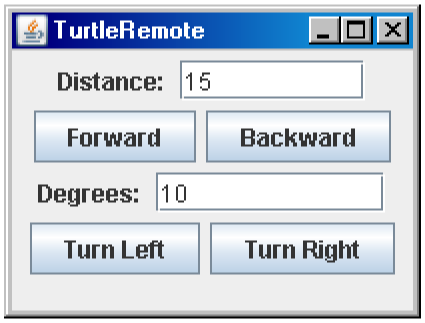
```
Dir: 50
X: 188
Y: 156
```

Specs:

1. The class should be named `TurtlePanel`, and be a subclass of `JPanel`.
2. It should have an instance variable to store a `Turtle` object.
3. It should have a constructor that takes in a `Turtle` object, and stores that `Turtle` object. The preferred size of the panel should also be set here (to something reasonable. The above example is 300x300).
4. It should have a `paintComponent` method that takes in a `Graphics` object. The turtle should be drawn on this `Graphics` object using its draw method. The turtle's direction and x,y location should also be drawn, as shown in the above example.

5.3: TurtleRemotePanel Class

The `TurtleRemotePanel` will be a panel with buttons and text fields to control the turtle. For example:



Specs:

1. The class should be named `TurtleRemotePanel`, and be a subclass of `JPanel`.
2. It should have instance variables to store:
   a. The distance text field
   b. The forward and backward buttons
   c. The degrees text field
   d. The turn left and turn right buttons
   e. The Turtle object to control
   f. The `TurtlePanel` object that is responsible for drawing the turtle
3. It should have a constructor that:
   a. takes in the `Turtle` object to control, and the `TurtlePanel` object that is responsible for drawing the turtle, and stores them to the appropriate instance variables.
   b. creates the text fields and buttons, and add them to the panel.
   c. adds the appropriate action listeners to the buttons (see step 4).
   d. sets preferred size of the panel, so that the UI elements wrap around nicely as above.
4. It should have at least one private inner class that implements `ActionListener` to handle clicking the buttons.
   a. When the forward button is clicked, it should move the turtle forward by the distance specified in the distance text field.
   b. When the backward button is clicked, it should move the turtle backward by the distance specified in the distance text field.
   c. When the “turn left” button is clicked, it should rotate the turtle counter-clockwise by the number of degrees specified in the degrees text field
   d. When the “turn right” button is clicked, it should rotate the turtle clockwise by the number of degrees specified in the degrees text field.
   e. The repaint method of the turtle panel should be called after any button clicks, so that the panel redraws where the turtle is.
   f. Hint: recall that the `ActionEvent` object passed to the `actionPerformed` method has the method `getSource()` that will return which object triggered the event.

5.4: TurtleMain Class

The TurtleMain class will be the class that glues the program together. Specs:

1. It should be named `TurtleMain`.
2. It should have a main method that:
    a. creates a `Turtle` object with some initial location and some initial direction.
    b. creates a `TutlePanel` with the above Turtle object.
    c. creates a `TurtleRemotePanel` with the above Turtle object and `TurtlePanel` object.
    d. puts the above `TurtlePanel` and `TurtleRemotePanel` objects into `JFrames`, sets the default close operation, packs the `JFrames`, and makes them visible.

Turn-in Procedure

Turn in the following files on T-Square. Double-check that you have *submitted* and not just saved as draft (if the submission is successful, you will receive an e-mail from T-Square within a few minutes). Also, be sure all of your files compile and run.

- Turtle.java
- TurtlePanel.java
- TurtleRemotePanel.java
- TurtleMain.java
- Any other files needed to run your submission

All .java files should have a brief descriptive javadoc comment.

Don't forget your collaboration statement. You should include a statement with every homework you submit, even if you worked alone.