CS 172
Lab 2

1. Log in to Linux. Organize your directories (folders) and files so that you have something like this:

   - 🗁 your home folder
     - 🗁 CS172
       - 🗁 Lab1
         - 🗎 Circle.java
         - 🗎 Hypotenuse.java
       - 🗁 Lab2
         - 🗎
       - 🗁 Lab3
         - 🗎
       - etc.

   On the Linux desktop, you'll find the File System icon on the left side of the screen. This is an app that lets you organize your files and folders, similar to Windows "Computer".

2. Programs that you create for this lab assignment should go into the Lab2 folder. When you save a file, make sure that you know **<u>where</u>** you are putting it.

<u>Part A - Java Application - Errors.java</u>

- Check your NMSU email. You should have a message about Lab 2. Download and save Errors.java to your Lab2 folder.

- Open Errors.java in JGrasp. Fix the spelling, capitalization and punctuation errors.

- Compile the program. Debug if you have syntax errors. Then, run the program.

- Add the following header comments to the top of your program.

```
// CS 172
// lab number
// Written by:  your name
// date
```

- Edit, save, recompile and run the program again as needed to make sure that the output is correct.

<u>Part B - Java Applet Pie.java and drawapie.html</u>

Creating a Java Applet

The Java coordinate system is discussed in Section 2.7 & 2.9 of the text. Remember that, in Java, the upper left-hand corner of the window is the point (0,0).

The X axis goes across the window, and the Y axis goes down the window. So the bigger the X value, the farther a point is to the right. The bigger the Y value, the farther it is down. There are no negative X or Y values in the Java coordinate system.

1. Write an applet named Pie.java that displays a pie chart (overall shape should be a circle) with 8 equal slices, each slice with a different color. You may use the built-in Java Colors or you may create your own custom colors. Draw a black border (circle) around the pie and draw black lines between the slices.

2. Under the pie, draw a black, filled rectangle. Inside the rectangle, draw your name in a large, bold font. Make sure that the color of the text contrasts well with the black background. You may use the following code as an example:

   ```
   g.setFont (new Font("sansserif",Font.BOLD,32));
   g.drawString ("This is a test", 0, 300);
   ```

   Include header comments as shown above. Also, include inline comments to explain what is happening in your program.

3. Create a file called drawapie.html in the same folder as Pie.java. JGrasp users should click the File Menu, then New, then Plain Text. Insert the following 4 lines and save the file.

   ```
   <html>
   <applet code="Pie.class" width=600 height=400>
   </applet>
   </html>
   ```

4. Compile the java program. If you have syntax errors, fix them and recompile.

5. View the applet. In JGrasp, this is the same button as running an application. To view an applet, you have to do it through the html file.

   *** If you're using something other than JGrasp, you may need to type this command in the Gnome Terminal window.

   ```
   appletviewer drawapie.html
   ```

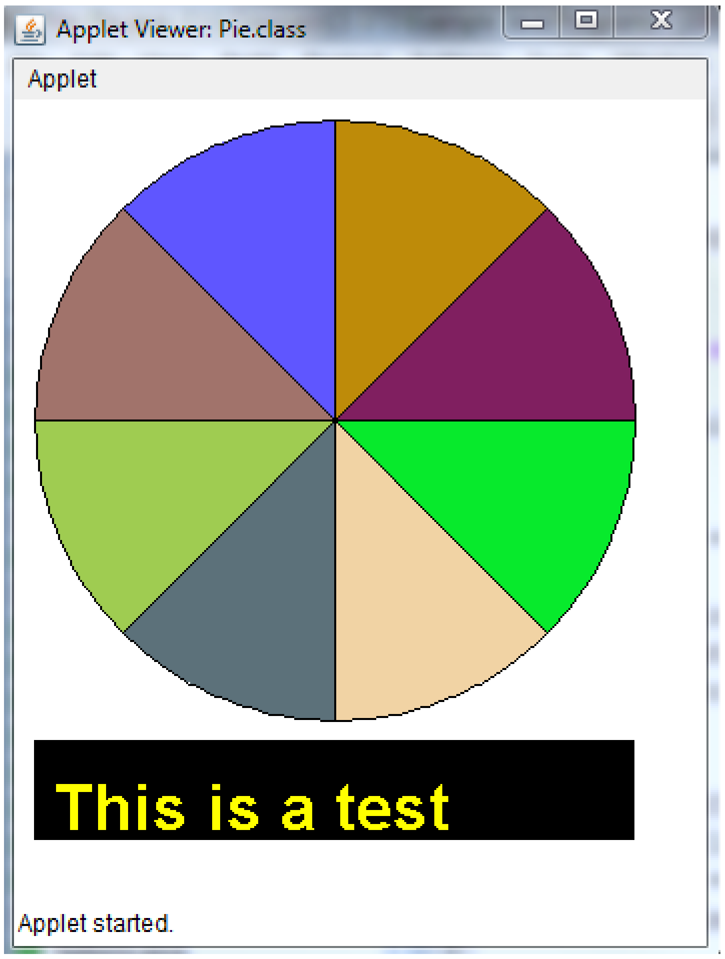
6. When you are finished looking at the web page, close the appletviewer window. You cannot make changes to the program until the appletviewer window is closed.

   Note to JGrasp users: JGrasp will create a temporary html file for you when you click the "run" button. You still need to create and submit the html file in step 3 in order to get full credit for this lab.

7. Save again, compile, debug if needed, view the applet... repeat until it looks right.

   Note: Every time you make a change to the program…no matter how small…you have to recompile it.

Here's what my pie looked like:



End of lab 2 - submit Errors.java, Pie.java and drawapie.html on Canvas