



**CMSC 113: Computer Science I**  
**Warmup Homework**  
**due on Gradescope by the beginning of class on September 20, 2017**

In all homework assignments and projects, you will be writing Java code to write a program (or *applet*) according to the specification given in the assignment. Your program must meet all the requirements described in the assignment, but you are welcome to add extra features and embellish on the description here. Always check the syllabus page for a link to a page with demos of the assignments; most assignments will have these demos available.

One particular note: names are important. When I give you the name of an applet or compound object (i.e., class), please use that name exactly, including capitalization. If you don't, Gradescope will complain when you submit.

This assignment is broken into several parts. Finish one completely before going onto the next; they are in order of difficulty.

**Part I**

1. Start up Eclipse.
2. When you are asked what workspace to open, choose to open the workspace you set up as part of Lab #1. (If you're on a different computer now, you may need to repeat the steps of Lab #1.) The workspace should be named *cs113* and is likely in your *Documents* or *My Documents* folder.
3. Create a new project by clicking the down arrow next to the "New" button  and choose *Java Project*.
4. Name the project *Warmup* and click *Next*.
5. Click the *Libraries* tab.
6. Use the *Add External JARs...* button to add the *acm.jar* library. (It's in your *cs113* folder.)
7. Click *Finish*.
8. Create a new class with the  button.
9. Name the class *Richie*.
10. Click *Finish*.

You should now have a file *Richie.java* with some Java code in it. Edit this code so that the program draws Richie, the owl. Your depiction must include a face with eyes, ears, and a beak, at the minimum. The elements must be properly centered. In other words, the left eye must be the same distance from the center as the right eye, the beak must come out from the center, etc. You may find it helpful to work this out on paper before writing the code.

Beyond these basics, feel free to embellish as you desire!

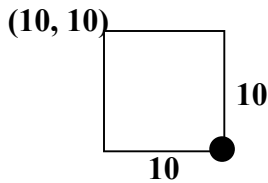
### Exercises:

Write the answers to the following questions in a comment in your file.

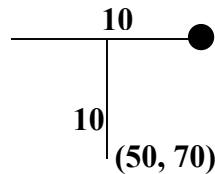
- We learned in class how to draw a rectangle. How would we draw a square?
- How big is the applet window before you resize it? You will need to do some experimentation to figure this out.
- What happens when you draw outside the applet?
- What lines of code would make a rectangle that is 20 pixels wide by 30 pixels tall with its top-left corner at coordinate (40, 40)?
- What lines of code would make that same rectangle with its *lower-right* corner at (40, 40)?

For the following 2 shapes, what are the coordinates of the big dot, given what is drawn in the diagram?

f.



g.



### Part II

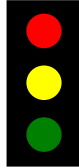
- Following a procedure like you did for *Richie*, write an applet *Dots* (use the same *Warmup* project that you did for the first part; that is, start at step 8, above) that draws a dot centered wherever the user clicks and another one centered wherever the mouse currently is. Draw the line that connects these dots. The dots should move; do not draw new ones.
- Write an applet *Tree* that draws a tree where the mouse clicked. The mouse arrow should be pointing to the center of the bottom of the trunk. The tree must consist of a filled-in green triangle and a filled-in brown rectangle (forming the bottom of the trunk). It should look like this:



This will require using `GPolygon`. To learn about `GPolygon`, visit the ACM reference website at <http://jtf.acm.org/javadoc/student/index.html> Look at the *acm.graphics* section to find `GPolygon`. Your program should move one tree around, not draw new ones.

### Part III


1. Write a compound object *TrafficLight* that represents a correctly-colored traffic light (red on top, then yellow, and green on bottom). The hotspot of the traffic light must be at the center of the green light. Then, write an applet *OneLight* using your compound object: It should start with the traffic light in the center. Every time the user clicks, the traffic light moves so that the green light is centered on the click.



2. Write another applet *ManyLights*: It starts empty. When the mouse moves on the applet, an image of a traffic light follows underneath the mouse, with the mouse pointing at the center of the green light. When the user clicks, an image of the traffic light is left at the location of the click.

### Part IV

Create a text file in your *Warmup* project:

1. Click the down-arrow next to the "New" button  and choose *File*.
2. Name the file *reflections.txt*. (A *.txt* file is a file for plain text, without any formatting.)
3. Put your name in the file and answer the following questions:
  1. Did you complete all parts of the assignment?
  2. If not, which parts were you unable to complete and why?
  3. How long did this assignment take you?
  4. What was the most challenging part?
  5. Do you have any other questions or experiences to share?

**Hand in:** Submit your code on Gradescope.

1. Select the *Warmup* project within Eclipse.
2. Right-click and choose to *Export...*
3. The *Export* dialog box appears. Under *General*, choose *Archive File* and click *Next*.
4. Make sure your *Warmup* project is selected in the list at the top left of the window and that *Save in zip format* and *Create directory structure for files* are selected toward the bottom.
5. *Browse...* to find a good spot to save the file that will be exported. Name the file *Warmup.zip*.
6. Click *Finish*. Eclipse will create a *Warmup.zip* file as directed.
7. Log into Gradescope and submit your *Warmup.zip*.