Name:_____ Section: _____CM: _____

CSSE 220---Object-Oriented Software Development

Exam 1 – Graphics Part, Spring 2023-24

Allowed Resources on this part. Open book, open notes, and computer. Limited network access. You may use the network only to access your own files, the course Moodle and the course web pages, the textbook's site, Oracle's Java website, and Logan Library's online books. You may only use a search engine (like Google) to search within Oracle's Java website - all other uses or accessing websites other than those mentioned above are not allowed.

Instructions. You must disable Microsoft Teams, IM, email, and other such communication programs when working on the graphics part of the exam. Any communication with anyone other than the instructor during the exam or violation of any other rules may result in a failing grade for the course.

Artificial Intelligence (AI) Generated Code Policy:
You are NOT allowed to use any kind of AI-assisted or generated technology including, but not limited to ChatGPT and GitHub Co-Pilot.

All the work you turn in has to be your own.
You must not use any forms of communication or cooperation.

The following are the ONLY items you are allowed to use:

Open book, open notes, and computer. Limited network access.
You may use the network only to access your own files
You may look at the CSSE220 Moodle web-site and any page linked directly from it (not more than one click/link away).
You may look at any of the Java documentation on the Oracle Website.
You may use the optional textbook's site and Logan Library's online books
You may only use a search engine (like Google) to search within Oracle's Java website - all other uses or accessing websites other than those mentioned above are not allowed.
You are allowed to use the normal auto-complete features built into Eclipse

You must actually get these problems working on your computer. Almost all of the credit for the problems will be for code that actually works. If you get every part working, comments are not required. If you do not get a method to work, comments may help me to understand enough so that you can earn (possibly a small amount of) partial credit.

Submit all modified files via GradeScope as directed by your instructor.

Grading Guidelines:

- Part 1 - 50% (draw default mouse)
  - 20% - Correctly draw the triangle for the shape of the head
  - 10% - Correctly draw the nose
  - 10% - Correctly draw the two eyes
  - 10% - Correctly draw the two ears
- Part 2 - 30% (location, size)
  - 10% - Add the ability to store and draw at different locations
  - 10% - Add the ability to store and draw with different sizes
  - 10% - Create and draw proper mouses in MouseComponent.java
- Part 3 - 20% (specify the rotation of a mouse)
  - 10% - Add the ability to set the rotation of mouses and draw them correctly (add new method)
  - 10% - Create and draw proper mouses in MouseComponent.java

Submission:

- Upload all modified .java files to the Exam1 Graphics Dropbox on the CSSE220 GradeScope site

Problem Description

Graphics Problem Top Level Instructions

- Read over all these instructions carefully
- Make sure you understand what functionality you are required to implement before you start coding
- If anything is unclear, simply do your best to follow the instructions and leave comments in your code describing your assumptions
- You can email your instructor about any confusion, but no communication without anyone else.

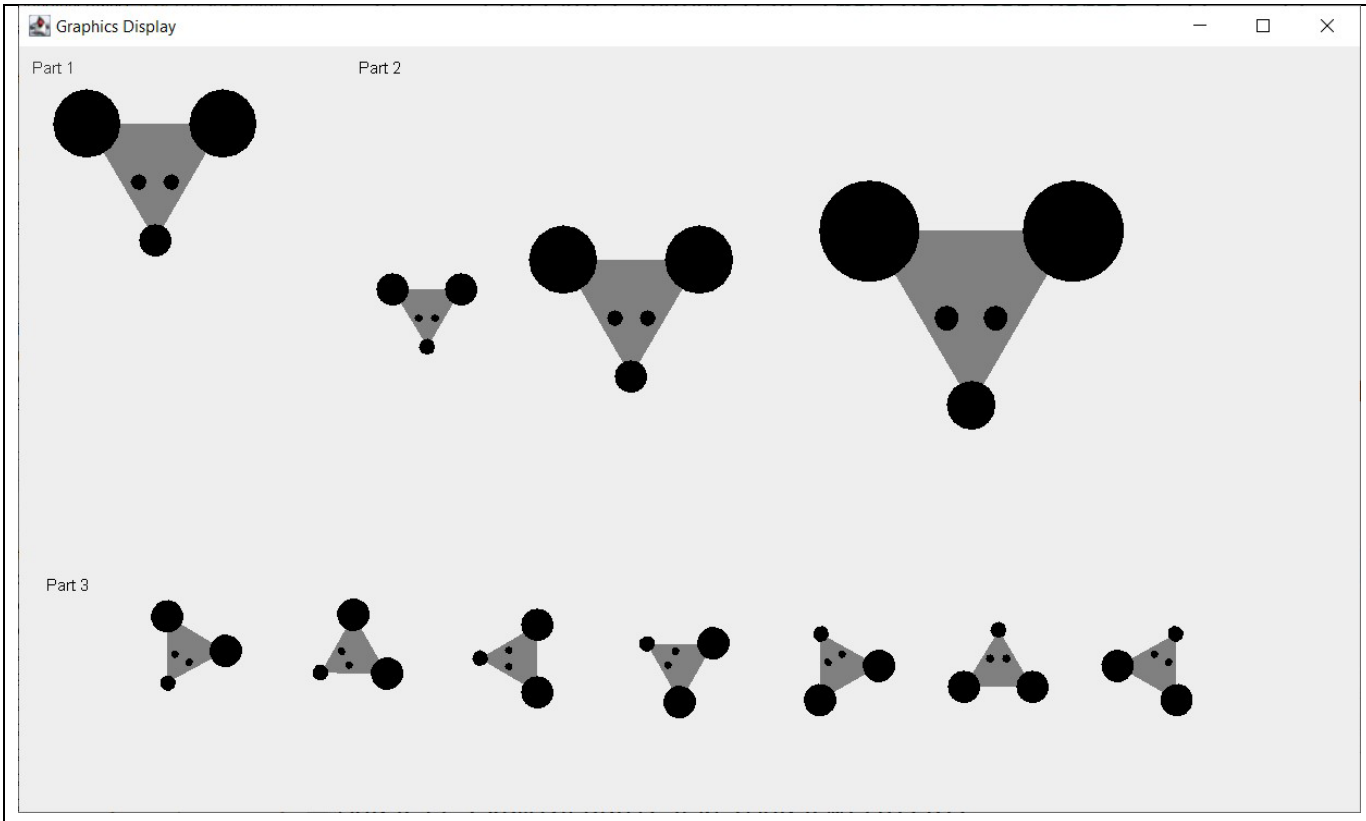

Figure 1 - Finished Version of this Assessment

Part 0: Default Code (starting image without modification)

Original Outline:

Run the *main* method found in the *MouseViewer* class.

You should see a rectangle drawn like the one to the right. This is an outline of the rectangle where your mouse's triangular head will be placed.

After you have established the correct location for your default Mouse, you should comment out the call to the method that draws this rectangle in MouseComponent.java.

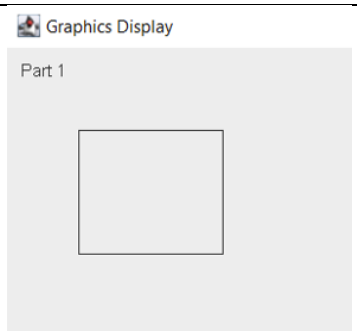

Figure 2 – Bounding Rectangle

Part 1 (50%): Default Mouse (drawing the default Mouse)

Mouse with all parts drawn:
You should add code to the *Mouse* class which will complete the drawing by adding the following parts as shown in Figures 3a and 3b

IMPORTANT NOTE: See the next page for calculations of specific dimensions and positions.

Circles

1. Draw a triangle for the head according to the constants provided in Mouse.java. If done properly, the triangle should fit exactly inside the example outline provided from Part 0. (See 3a to the right, drawn in HEAD_COLOR= gray)
2. Draw circles for the **ears** of the Mouse which are centered at the top left and right corners of the triangle.
   (Color will be EAR_COLOR = black)
3. Draw a circle for the **nose** of the Mouse which is centered at the bottom corner of the triangle.
   (Color will be NOSE_COLOR = black)
   Draw the **eyes** of the Mouse which are offset to the left and right of the center. (Color will be EYE_COLOR = black)
4. Verify the image you have looks like the one to the right.
5. After completing these parts, you can comment out the default rectangle drawing code. (See Figure 3b)

Eyes:
Each eye will be offset from the center of the head as shown on the next page. The offset is based on the eye diameter.

This part of the problem will involve the Mouse that gets drawn using the zero-parameter constructor. You should have this code drawn using instance variables for a Mouse object, otherwise, you will not be able to draw additional Mouses and cannot receive credit for later parts of the problem (Part 2, Part 3).

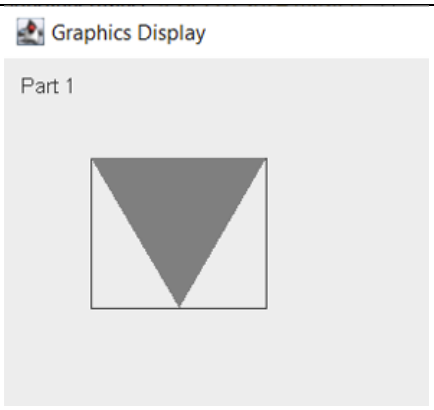

Figure 3a – Example Outline Overlay

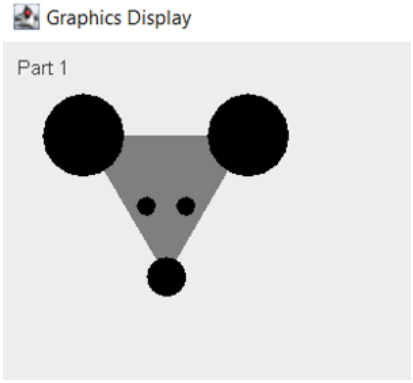

Figure 3b – Mouse (commenting out example code)

Helpful Layout

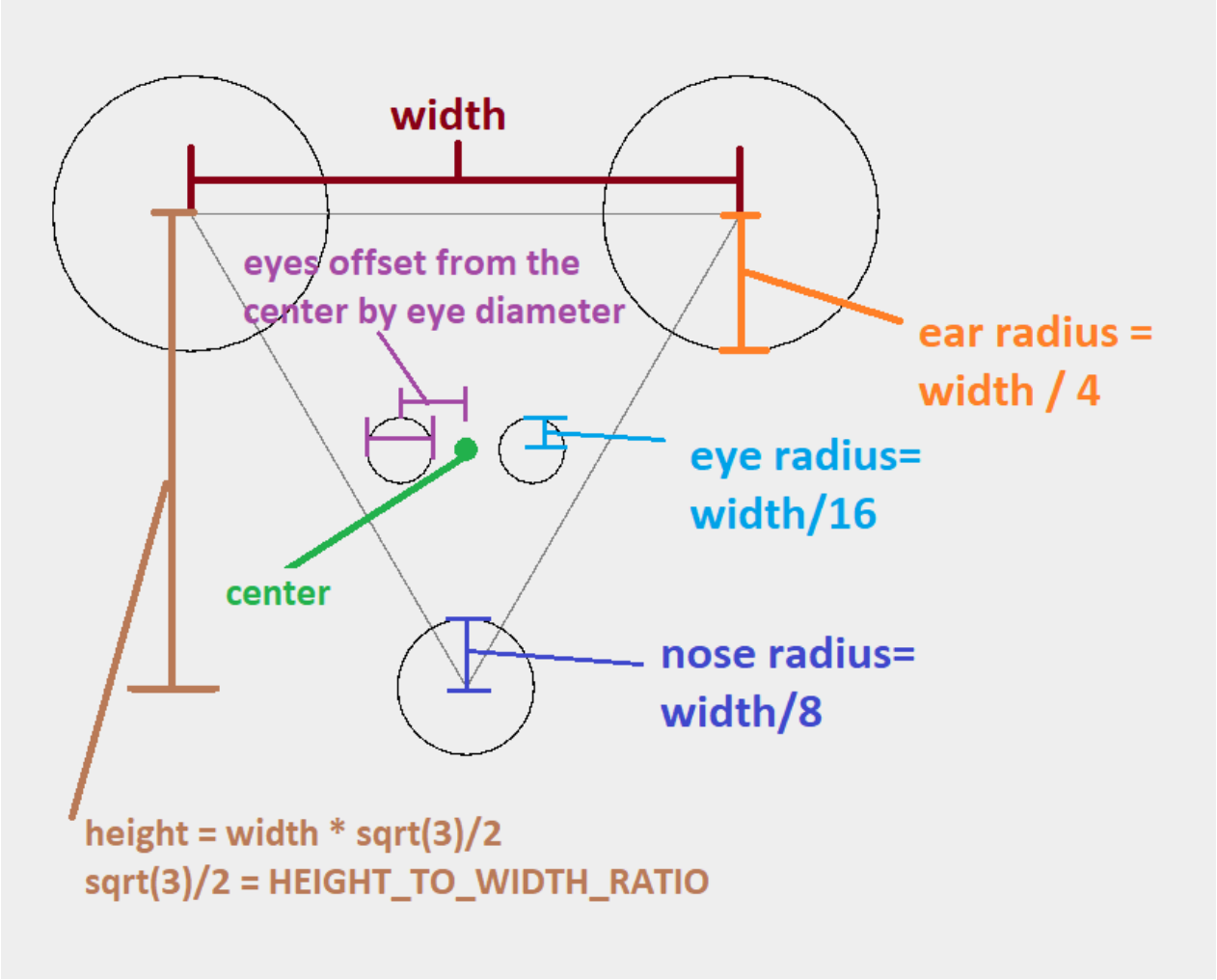

Figure 4 – Layout Overlay

The Mouse is made up of a triangle two ears, two eyes, and a nose as shown in Figure 4a. The Mouse triangle's center (vertically and horizontally centered), is to be treated as the center of the entire Mouse. The figure gives a means of calculating the dimensions and positions based on the width (length of a side of the triangle):

Part 2 (30%): Mouse - Add different locations and height

Sample images are shown to the right

- Figure 5 identifies all the different-sized and located drawn Mouses
- To create and draw these you will need to create a new 3-parameter constructor to allow you to specify these additional fields
- The parameters to include are: center X, center Y, and width.
- The rest of the Mouse should be scaled according to the provided width just as shown in Figure 4.
- There are comments in *MouseComponent.java* that will provide the necessary details for the locations and heights of the various objects to be drawn.

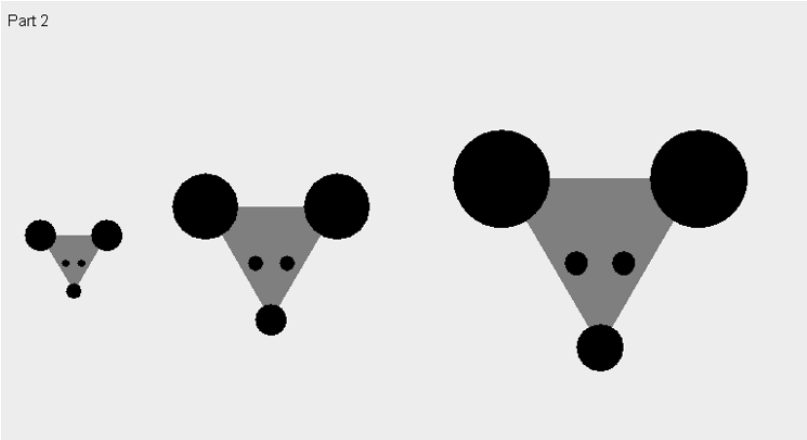

Figure 5 – Drawing Different Mouses

Part 3 (20%): Rotation of Mouse

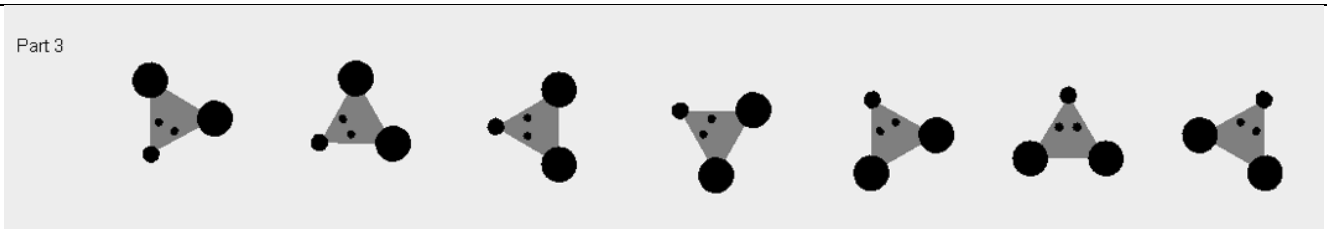

Figure 6 – Spinning Mouse!

Rotation:
Follow the instructions for *Part 3* found in *MouseComponent.java* to complete this part. When you are finished with Part 3, your app will draw 7 Mouses that will appear as shown above in Figure 6.

Hint: There are comments describing exactly how to adjust the position and rotation. You will want to use the loop that is described in the code to be able to produce this sequence of drawings. You are told to add methods to help you accomplish these tasks, be sure to read and follow the instructions carefully.

Hint: Remember to use the rotation method that we practiced in class and on the graphics homework assignment.

When finished, your app will draw the Mouse from Part 1, Part 2, and Part 3 shown in Figure 1 (also shown on last page).

Final Version after Completing All Parts