Name:_____ Section: _____CM: _____

CSSE 220---Object-Oriented Software Development

Exam 1 – Graphics Part, January, 2021

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 others uses or accessing websites other than those mentioned above are not allowed.

Instructions. You must disable Microsoft Lync, IM, email, and other such communication programs before beginning the graphics part of the exam. Any communication with anyone other than the instructor or a TA during the exam may result in a failing grade for the course.

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.

To pass this part of the exam you must complete:

BOTH Part 1 AND Part 2
as well as EITHER
Part 3 OR Part 4

If you complete ALL 4 parts you may receive possible bonus credit for other parts of the exam

In Java syntax… if P1, P2, P3, P4 are booleans and true means pass, false means fail:

```
boolean didIPassTheGraphicsPart = (P1 && P2) && (P3 || P4)
```

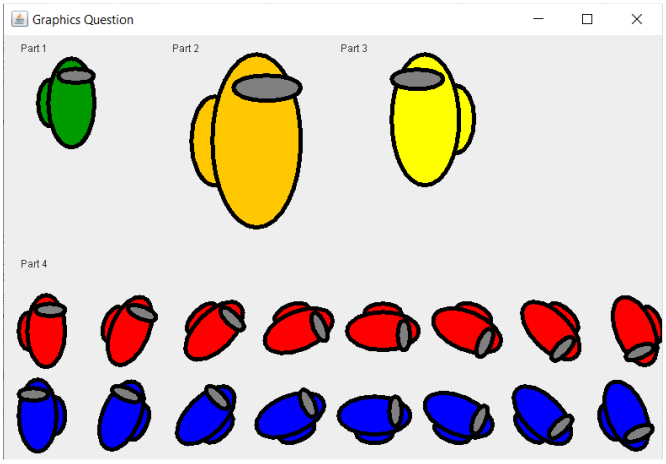
Submit all modified files via Moodle.

Problem Description

Graphics Problem (Must complete 1,2,3 OR 1,2,4)

- Read over all these instructions carefully
- Make sure you understand completely what functionality you have 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 also email your instructor FOLLOWING the termination of the exam about the confusion you had

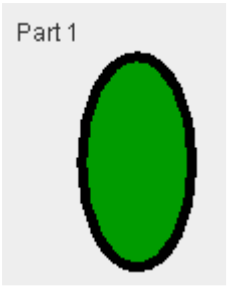
Finished Version



Part 1 (Required to pass) Default Astronaut (add backpack and visor)

Astronaut's suit:

Run the *main* method found in the *AstronautViewer* class.
When constructed with no parameters, an Astronaut object should draw looking like the picture to the right.

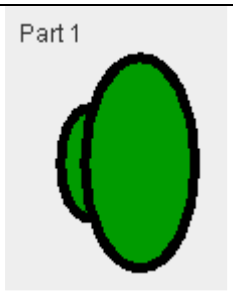


The Astronaut's center is (DEFAULT_CENTER_X, DEFAULT_CENTER_Y)
The Astronaut's height is DEFAULT_HEIGHT
The Astronaut's width is DEFAULT_HEIGHT / HEIGHT_TO_WIDTH_RATIO
The default color for an Astronaut is DEFAULT_COLOR (which is predefined)

```
new Color(0,155,0)
```

Astronaut's backpack:

Modify class Astronaut's *drawAstronaut* method so that it draws a backpack as shown in the picture to the right.

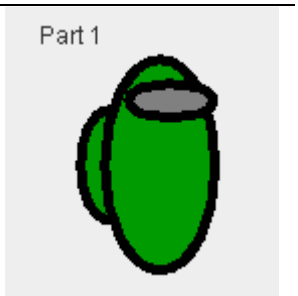


The ovals for the backpack and suit must be drawn in the correct order to match the appearance at the right. The backpack's width and height are ½ of the size of the suit. The backpack is centered vertically, and is aligned to be centered on the left side of the bounding rectangle of the suit. The backpack's color is the same as the suit.

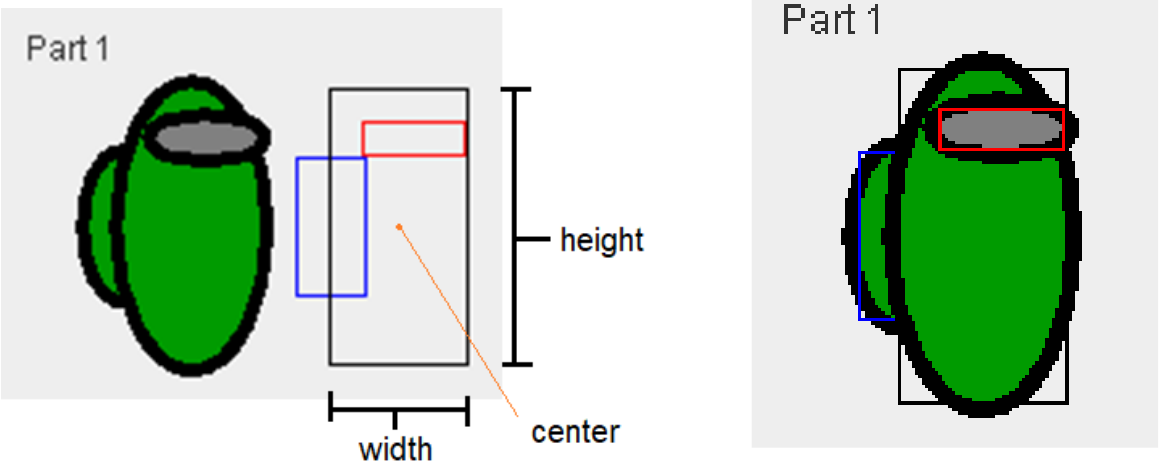
See next page for more detailed visual guide to the layout of the ovals.

Astronaut's Visor:

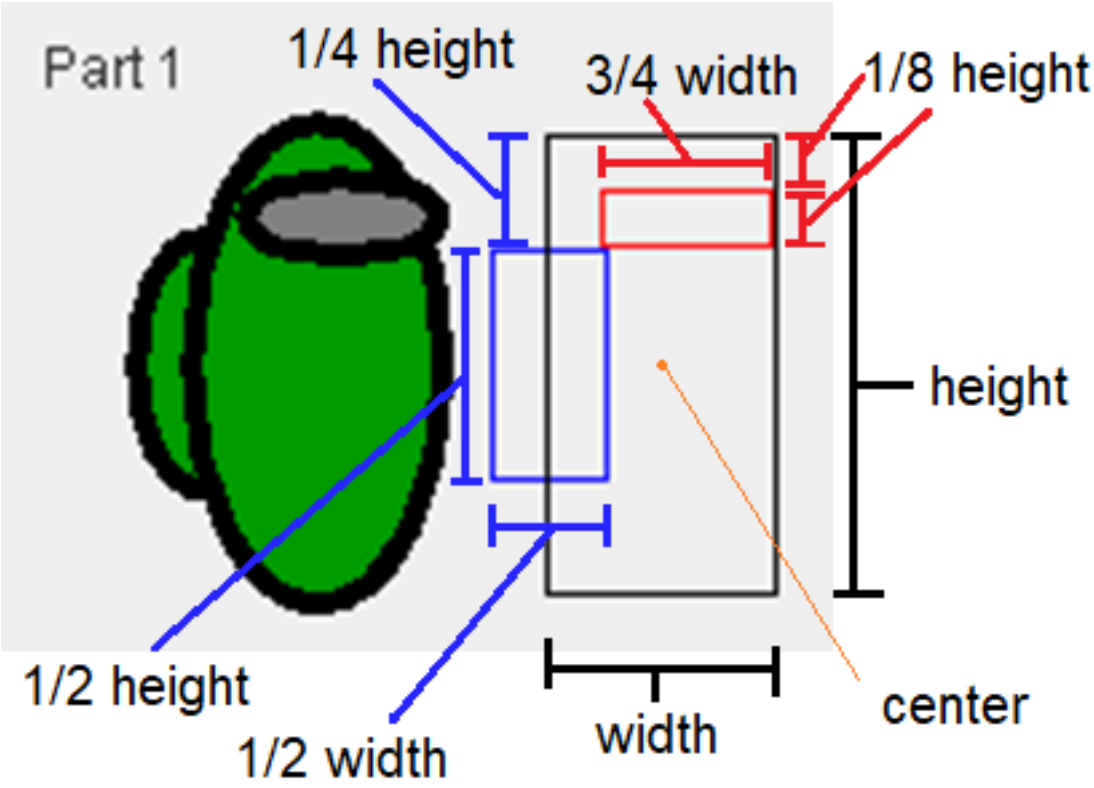
Modify class Astronaut's *drawAstronaut* method so that it draws a visor on top of the suit as shown in the picture to the right. The color of the visor should always be `Color.GRAY`. The width of the visor is ¾ the width of the suit and its bounding rectangle should line up with the left side of the suit's bounding box. The height of the visor is 1/8 the height of the suit and it should start 1/8 of the height LOWER than the top of the suit's bounding box.



See next page for more detailed visual guide to the layout of the ovals.



Supplementary figures zoomed in on Astronaut's bounding rectangles for:
the suit (black), backpack (blue), and visor (red)



Visual depiction of the dimensions of bounding boxes and relationship between boxes.

Part 2 (Required to pass) Different Astronauts – You can do this part even if you did not complete Part 1

Uncomment the *Part 2* code in *AstronautComponent* to begin this part

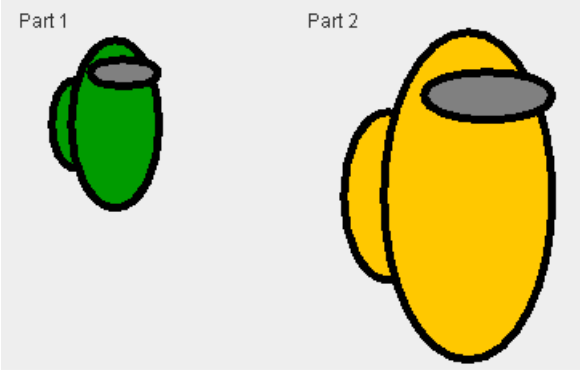
Add Constructors:
Add a new parameterized constructor for the Astronaut class that takes 4 parameters:

1. *centerX* – the Astronaut's x value for its center position (center of suit's bounding rectangle)
2. *centerY* – the Astronaut's y value for its center position (center of suit's bounding rectangle)
3. *height* – the height of the Astronaut (height of suit, also used to calculate the width (width= ½ height)
4. *color* – Astronaut's fill color (suit and backpack)

This ***may*** require you to:

- declare new instance variables (i.e., fields) in the Astronaut class
- make changes to the Astronaut's *drawAstronaut* method
- create a 0-parameter constructor (using the DEFAULT values found at the top of Astronaut class) in order to ensure Part 1 code continues to work correctly

When finished, the *Part 1* and *Part 2* part of your screen should look like the following:



Part 3 (To pass you must complete EITHER this part OR Part 4) Orienting Astronaut - You CANNOT do this part unless you added AT LEAST the backpack OR visor from Part 1

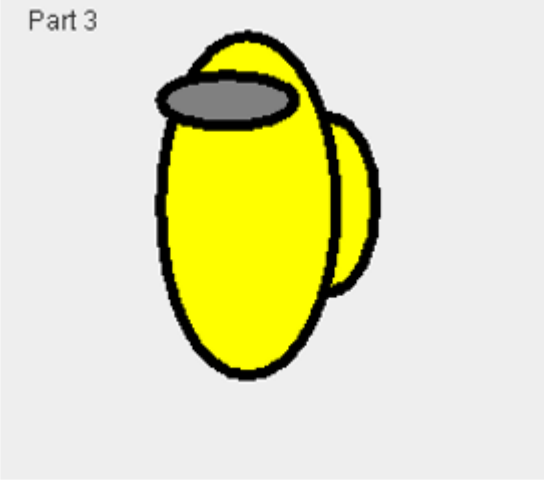
Uncomment the *Part 3* code in *AstronautComponent* to begin this part

Add Orientation:
Add method *setOrientation*(*String orientation*) that takes a String as an input. If the String is "LEFT" then you should have the Astronaut appear to be facing the to the left (visor aligned to the left, backpack aligned to the right), if the String is "RIGHT" then the Astronaut should have the same appearance as described in Part 1-2. Thus, an Astronaut will only appear differently than shown in Parts 1-2 IF and ONLY IF the orientation is set to be "LEFT". (In other words, the default orientation is "RIGHT")

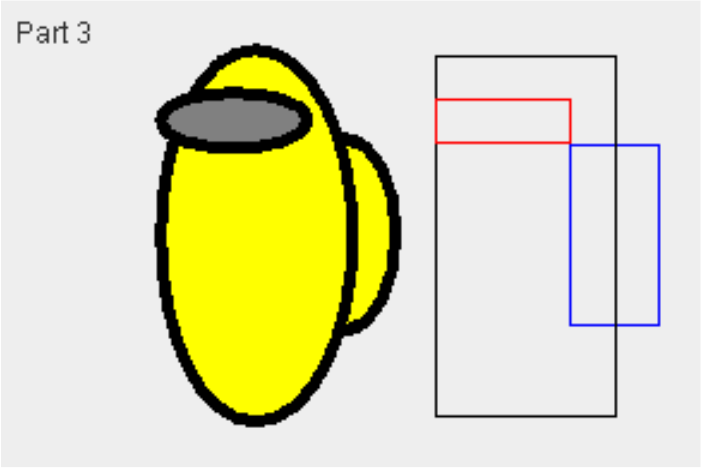
In addition to creating the method mention above, this ***may*** require you to:

- declare new instance variables (i.e., fields) in the Astronaut class
- make changes to the Astronaut's's *drawAstronaut* method

When finished, the *Part 3* part of your screen should look like the following:



For reference, below is a supplemental diagram showing the outline of the suit, backpack, and visor for a "LEFT" aligned astronaut. The center of the suit is UNCHANGED by switching orientations.



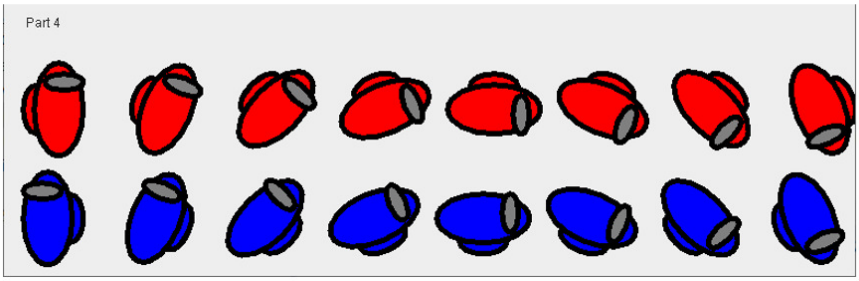
Part 4 (To pass you must complete EITHER this part OR Part 3) Create a Spinning Astronaut - You CAN do this even if you did not complete Parts 3

Uncomment the *Part 4* code in *AstronautComponent* to begin this part

Add Rotation:
Add method *setRotationFactor*(*radians*) that takes an input value in radians
This ***may*** require you to:

- declare new instance variables (i.e., fields) in the Astronaut class
- make changes to the Astronaut's *drawAstronaut* method

When finished, *Part 4* of your screen should look like the following:



A Final High Resolution image of Solution for easy comparison: