

Name: _____ Section: _____ CM: _____

CSSE 220---Object-Oriented Software Development

Exam 2 – Graphics Part, February, 2021

Allowed Resources on Part 2. 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.

Submit all modified files via Moodle.

Problem Description

Part C3: Graphics Problem (pass/no-pass)

You must pass Parts 1, 2, and **either 3 or 4** for this assessment (bonus if you must complete Parts 1-5)

- 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 any confusion
- Open **exam2_graphics_animated_gif.gif**

file to see an animated demo of the final app →

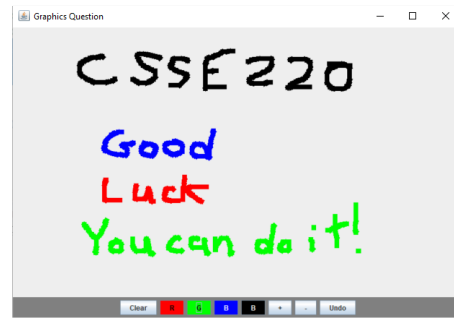
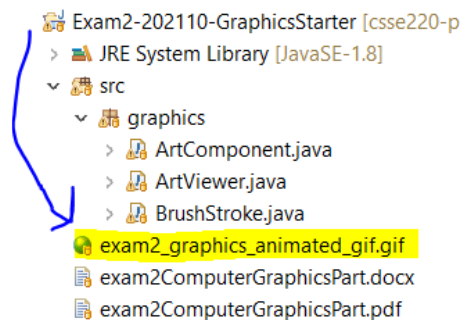


Figure 1 – Finished Version



Part 0 – Run the App as is

The `main` operation can be found in `ArtViewer.java`

When you run this app, it will look like Figure 2 (below). The starting text is there for a quick reminder of what needs to be done and the single, thick, blue line is to provide you with example code. You can comment out the code for both of these or leave it as is.

Part 1 – REQUIRED TO PASS: Allow user to draw on the screen

Locate “TODO Part 1” in the starter file `ArtViewer.java` and follow its instructions.

This part will have you setup the `BrushStroke` class and use listeners so that when a user drags the mouse a thin, black line is draw on the screen. Each time the mouse is dragged it should use the previous location of the mouse and the current one to create a `BrushStroke` and add it to a list of `BrushStrokes` (`ArtComponent`) that are drawn.

- Figure 2 shows the app prior to any modifications
- Figure 3 shows the app after being used when you have completed Part 1

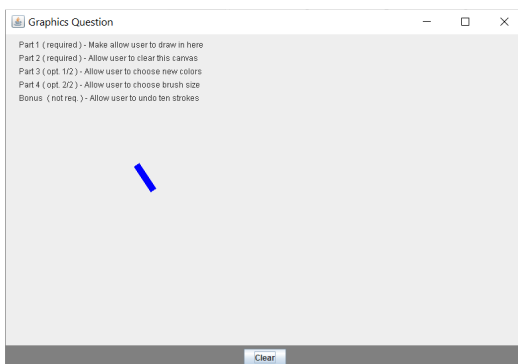


Figure 2

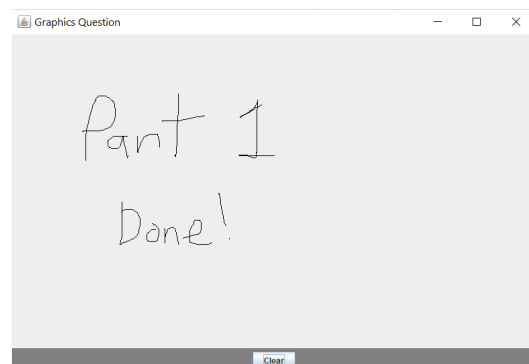


Figure 3

JavaDocs for MouseMotionListener

Method Summary

Modifier and Type	Method and Description
void	mouseDragged (MouseEvent e) Invoked when a mouse button is pressed on a component and then dragged.
void	mouseMoved (MouseEvent e) Invoked when the mouse cursor has been moved onto a component but no buttons have been pushed.

Method Detail

mouseDragged

```
void mouseDragged(MouseEvent e)
```

Invoked when a mouse button is pressed on a component and then dragged. MOUSE_DRAGGED events will continue to be delivered to the component where the drag originated until the mouse button is released (regardless of whether the mouse position is within the bounds of the component).

Due to platform-dependent Drag&Drop implementations, MOUSE_DRAGGED events may not be delivered during a native Drag&Drop operation.


mouseMoved

```
void mouseMoved(MouseEvent e)
```

Invoked when the mouse cursor has been moved onto a component but no buttons have been pushed.

Graphics Question

Part 1 (required) - Make allow user to draw in here
Part 2 (required) - Allow user to clear this canvas
Part 3 (opt. 1/2) - Allow user to choose new colors
Part 4 (opt. 2/2) - Allow user to choose brush size
Bonus (not req.) - Allow user to undo ten strokes



ArtViewer (1) [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\java.exe
Mouse moved at: 271, 460
Mouse moved at: 270, 459
Mouse dragged at: 270, 458
Mouse dragged at: 271, 458

Figure 4
Starter code provides console logs

As you can see `MouseMotionListener` is an interface very similar to `MouseListener`. In order to allow a user to click and drag their mouse like in a drawing/painting program, you will need to make use of this interface. The default starting code is setup to use a `MouseMotionListener` so you can see how to attach it and trigger code from events (Statements will appear in the console while moving or dragging the mouse as shown in the screenshot in **Figure 4**). Feel free to use/modify that code to help you complete your program.

In order to function as desired, when the user releases the mouse button (a `MouseListener` event) the most recent location of the mouse should be reset in some way so that on the next `mouseDragged` event a `BrushStroke` will **NOT** result in a long extended single `BrushStroke` from that potentially distant last location. Instead, it is OK to wait until two `mouseDragged` events occur before adding the next `BrushStroke` (the `BrushStroke` would connect the two locations of these `mouseDragged` events).

(It would also be OK to set the previous location of the mouse on a mouse pressed event (a `MouseListener` event) so that on the first `mouseDragged` event a `BrushStroke` will be added, however, this is unnecessary and would not make a large visible difference.)

Part 2 – REQUIRED TO PASS: Make the “Clear” Button work

Locate “TODO Part 2” in the starter file *ArtViewer.java* and follow its instructions.

This part will have you make the “Clear” button remove ALL BrushStrokes from the screen.

When you are finished a user will be able to clear the current drawing with a single button click.

In Figure 5, a user has cleared the screen.

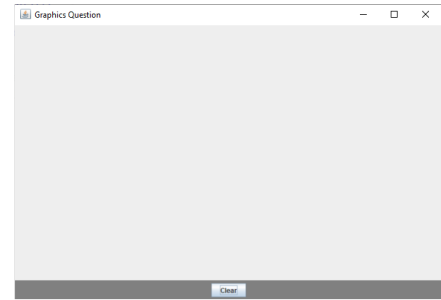


Figure 5

Part 3 – Option 1 of 2 to Pass: Add Buttons for Selecting Color

Locate “TODO Part3” in the starter file *ArtViewer.java* and follow its instructions.

This part will have you add four buttons along with listeners. When any of the buttons are clicked by a user, a different color will be selected which will then be used to draw BrushStrokes until a different color is selected. The background and foreground color for each button should match Figure 6 below.

- Figure 6 shows a drawing using all the colors

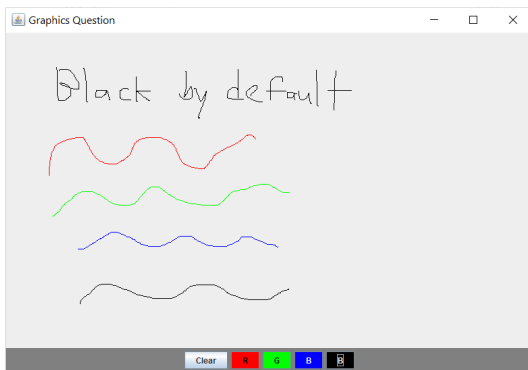


Figure 6

Button	Text	Foreground	Background
Red	R	Black	Red
Green	G	Black	Green
Blue	B	White	Blue
Black	B	White	Black

Part 4 – Option 2 of 2 to Pass: Add Buttons for Selecting Brush Size

Locate “TODO Part4” in the starter file *ArtViewer.java* and follow its instructions.

This part will have you add two buttons along with listeners. One button with a “+” as its text will INCREASE the size of the BrushStrokes (+1) drawn by the user. The opposite button with a “-” as its text will DECREASE the size of the BrushStrokes drawn by the user, but NOT below 1.

- Figure 7 shows a drawing using different stroke sizes (print the current stroke size to the console to easily see this information)

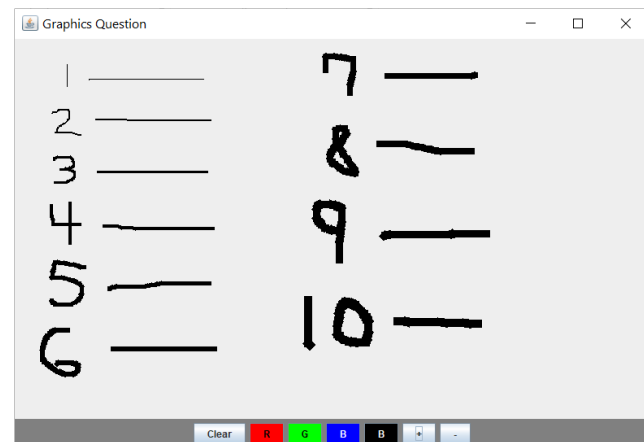


Figure 7

Part 5 – Bonus: Undo Ten BrushStrokes

Locate “TODO Part5” in the starter file *ArtViewer.java* and follow its instructions. In case a user makes an error when drawing we want an easy way to undo drawing. However, each individual BrushStroke is VERY small, so we want to remove ten with each click. Be sure not to throw an errors when you remove the final set of BrushStrokes (e..g there are only 5 on the screen, you should remove all but not throw an error)

- Figure 8 and 9 shows a drawing (8) and the same drawing after a user has pressed the Undo button (9).

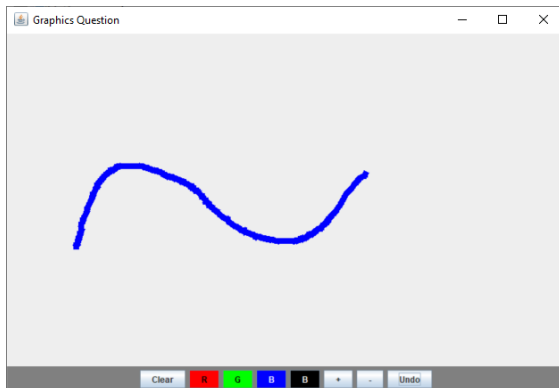


Figure 8

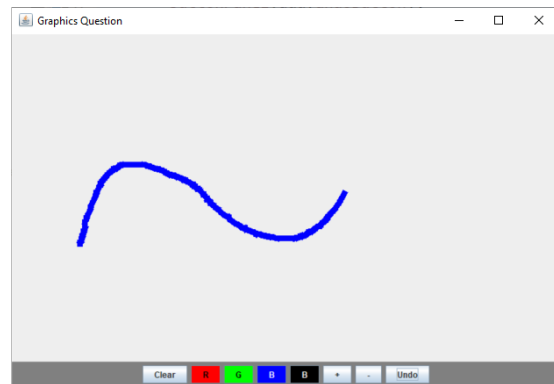


Figure 9

If you liked this problem feel free to draw a picture and take a screenshot to upload with your solution. ☺