CSSE 220---Object-Oriented Software Development

# Final Exam -- Graphics, February, 2021

**Allowed Resources for this Part**

Open book, open notes, and open computer. Limited network access. You may use the network only to access your own files, the course Moodle and Piazza sites (but obviously don't post on Piazza) and web pages, the textbook's site, Oracle's Java website, and Logan Library's online books.

**Do not use non--approved websites like search engines (Google) or any website other than those listed above.**

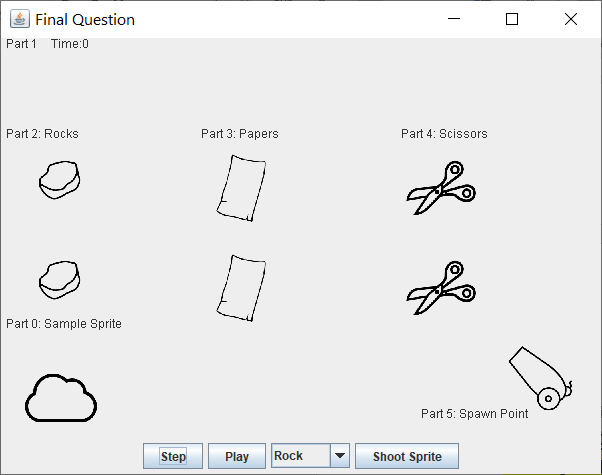
**Instructions**

You must disable Microsoft Lync, IM, email, and other such communication programs before beginning this 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. There are several different small methods to write, so you can get a lot of partial credit by getting some of them to work. 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 you can earn (a small amount of) partial credit.

Submit your modified source code via Moodle. **Be sure to check your Moodle submission to ensure you submitted the right version of the files before leaving.**

 **Graphics Question**

**Required to Pass:**

5 out of the 6 parts (1-6)

Before you begin this problem, we suggest you watch

an animation demonstrating the final working version.

**You can see an sped up animated gif of the**

**solution in the file: final\_question.gif**

which is located in the same folder as this Word file.

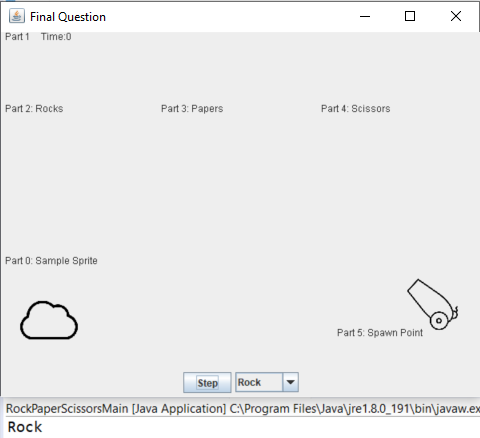
**Part 0:** (not graded) Examine Starting Code.

In this problem you start with a Sprite class that

allows you to specify the starting location, dimensions, velocity, and filename of an image to be used to draw an image to the screen. This problem is designed to test your understanding of the kind of coding required for the final project. You are of course welcome to look at code from your final project. For your convenience, there are already a couple of Sprites (cloud and cannon) are already added to a list which is drawn to the screen (Part 0).

***Also note, for sake of simplicity the instance variables in Sprite are defined with protected accessibility, so that you can easily access them from the subclasses without using getters/setters.***

*What the program and console looks like when run with the starting code:*



**Part 1:** (required to pass) Count the steps/ticks of the running program.

For Part 1 you should figure out a way to report how many times the “Step” button has been clicked and display that on the RPSComponent. Currently there is a string drawn which simply says, “Time: 0” which is not updated when the step button is pressed. Add whatever variables, methods, etc. so that when the Step button is pressed that the number of steps/ticks so far will be reported. In addition, the provided cloud sprite should move with each click of the Step button.

**Part 2:** (required to pass) Add a Play/Pause Button and Make Sprite Move

Add an additional JButton to the right of the Step button, which has text, “Play” and when pressed causes the step/tick/time counting (drawing the string= Time: \_) and the Sprites to start to move every 20 milliseconds (constant named DELAY is provided). After the first press, the text on the button should change to be “Pause”. When pressed a second time, the Sprites should stop moving and the step counting should stop as well. Following the second press, the button should say, “Play” again and then continue on in this fashion alternating between Play-Pause-Play-Pause etc.

**Part 3:** Rocks, Paper, Scissors – Basic Classes

For this part you need to create three new classes, Rock, Paper, Scissors which each provide different functionality, however, they will also require functionality provided in the Sprite class. There is a clear design choice of what you should apply to this in this scenario.

The following table will provide a summary of the information you need for this part:

|  |  |  |
| --- | --- | --- |
|  | Image filename | Width and Height |
| Rock | img/rock.png | 80 |
| Paper | img/paper.png | 80 |
| Scissors | img/scissors.png | 80 |

After creating these, you should be able to uncomment the code in RPSComponent.java (look for Part 3) and afterwards, when run, you should be able to see the different Sprites bouncing around the screen (no collisions between different objects yet).

**Part 4:** Rocks, Paper, Scissors – Self Collisions -

**Note: You CANNOT use type predicated code in your solution to pass this part.**

Now increase the functionality of these classes so that the following occurs:

1. When Rocks collide with Rocks they **BOTH change their vertical velocity**, but not their horizontal velocity.
2. When Scissors collide with Scissors they **BOTH change their horizontal velocity**, but not their vertical velocity.

|  |  |
| --- | --- |
|  | Collisions of two instances of the same class |
| Rock | Both Rocks’ dy (yVelocity) flips from positive to negative and vice versa |
| Paper | No change |
| Scissors | Both Scissors’ dx (xVelocity) flips from positive to negative and vice versa |

**Part 5:** Rocks, Paper, Scissors – Destructive Collisions

**Note: You CANNOT use type predicated code in your solution to pass this part.**

For this part you will need to make it so that each Rock/Paper/Scissor when colliding with each other will act appropriately. The rules are Rock wins vs. Scissors, Scissors wins vs. Paper, Paper wins vs. Rock.

In each case you MUST do TWO things to pass this part:

1. Print to the console ( System.out.println() ) the “**quoted text”** in the table below ONCE for each collision
2. Remove the appropriate Rock/Paper/Scissors from the running simulator.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Rock | Paper | Scissor |
| Rock | Both Rocks’ dy flips  nothing removed | “**Rock Cover By Paper**”  Rock removed | “**Rock Smashed By Scissors**”  Scissors removed |
| Paper | “**Rock Cover By Paper**”  Rock removed | Nothing happen | “**Paper Cut By Scissors**”  Paper removed |
| Scissors | “**Rock Smashed By Scissors**”  Scissors removed | “**Paper Cut By Scissors**”  Paper removed | Both Scissors’ dx flips  nothing removed |

**Part 6:** Sprite Cannon

RockPaperScissorMain provides you with this code to help allow a user to select a type of Sprite to be created:

// Starter Code

String[] choices = {"Rock", "Paper", "Scissors" };

**final** JComboBox<String> spriteSelector = **new** JComboBox<String>(choices);

String selectedItem = (String)spriteSelector.getSelectedItem();

System.***out***.println( selectedItem );

// End Starter Code

Feel free to use this code and create a new button with the text “Shoot Cannon” which when pressed should create a new instance of the selected class which should be placed at the location and velocity as defined in the constants in RockPaperScissorsMain.java

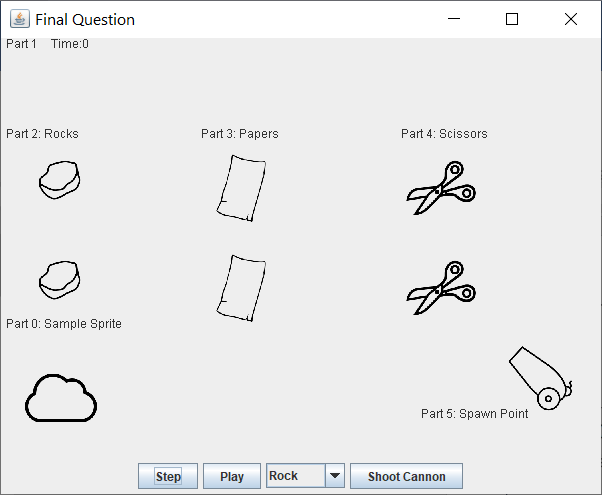
Starting Location: (SPAWN\_X, SPAWN\_Y)

Starting Velocity: (SPAWN\_DX, SPAWN\_DY)

***These newly added sprites should then be drawn, move, and interact with other objects just like the starting ones.***

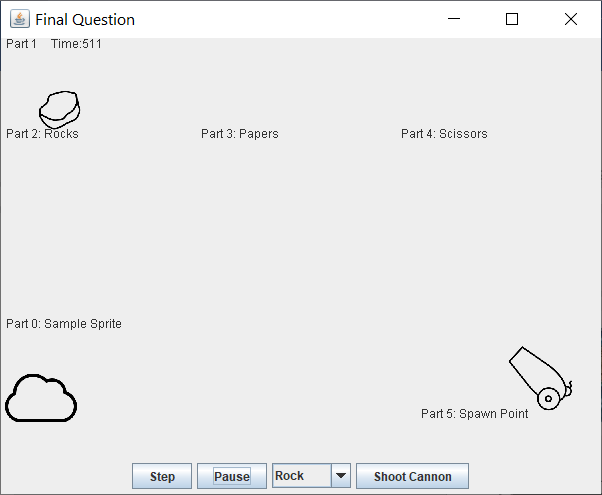
Final Code Screenshots on next Page:

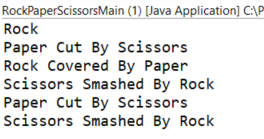
**Final Code before running:**



**Final Code after running with console output:**

*These collisions should happen in this order and eventually a single Rock remains (plus the cloud).*





You can watch an animated gif in the eclipse project folder here which shows a fully completed solution:

***Please note: for faster viewing this was run with a DELAY=1 instead of DELAY=20, so it will move faster than in your program if you have the DELAY set to be 20***

