Description:

For this week's assignment, I created a Rock Paper Scissors game. I used separation of concerns to divide responsibility between the GUI (RPSGUI), Interfaces (IController), Game Logic (GameLogic), and Main Game Functionality (RPSGame). I didn't think the player needed their own class since there is only one and they don't have a lot of functionality. At the beginning of my game, the player can choose to play in the console or the GUI.

If they play in the console the game will loop until they have either pressed 'q' for Quit or until they have played 4 games. At that point the game will display their score (games won) and ask if they want to play again.

If the player chooses to play with a GUI, the GUI will pop up and the player can select a button to choose if they want to play rock, paper or scissors. Because the player must select a button before the game can run, I modified the do while loop in runGame() to accommodate this. In the loop, the game loops and waits until the player presses a button to continue. The game then runs through all 4 plays and displays the score on the GUI.

For my computer winning strategies, I created a GamePlayFactory in which a random strategy is selected. One strategy is choosing a random bet (rock, paper, or scissors), the other strategy is to choose whatever bet will win the game. This way the player has a chance to win but most likely will lose or tie with the computer.

All buttons share the same action listener and the GUI uses the Builder Pattern. I don't have any repeated code, only code that is commented out depending on if the player chooses GUI or console.

Note:

Screenshots, use case and the use-case diagram are attached below. Beginning of class files are highlighted in yellow.

```
package RPSGame;

import java.util.Scanner;

import javax.swing.JFrame;

public class RPSGame {

	public static void main(String[] args) {
		GameController controller = new GameController();
	}

}
class GameView implements IView
```

```
{
	Scanner sc = new Scanner(System.in);
	char input;
	private static char[] matches = new char[] {'r', 'p', 's', 'q', 'c', 'g', 't', 'f'};

	public void display(String message) {
		System.out.println(message);

	}

	@Override
	public Character getInput() {
		boolean isCorrectInput = false;
		do {
	input = sc.next().charAt(0);
	input = Character.toLowerCase(input);
	for(int i = 0; i < matches.length; i++){
		if (input == matches[i]) {
			return new Character(input);
		}
	}
	System.out.print("Please respond with an expected character:  ");
      } while (!isCorrectInput);
		return null;
	}

}

class GameController implements IController
{
	GameView view = new GameView();
	Move move;
	static Character input;
	static String myResult = "";

	static String compMsg = "";
	static String myMsg = "";
	static String msg = "";

	static boolean setMove;
```

```
public GameController()
{
	init();
}

public static String RockBtn()
{
	myResult = "rock";
	setMove();
	return myResult;
}
public static String PaperBtn()
{
	myResult = "paper";
	setMove();
	return myResult;
}
public static String ScissorsBtn()
{
	myResult = "scissors";
	setMove();
	return myResult;
}

public static String setMove()
{
	String myMove = "";
	boolean isCorrect = false;

	do
	{
		input = Character.toLowerCase(input);

		if(input == 'r' || myResult == "rock")
		{
			myMove = "rock";
		}

		else if(input == 'p' || myResult == "paper")
```

```
                {
                        myMove = "paper";
                }

                else if(input == 's' || myResult == "scissors")
                {
                        myMove = "scissors";
                }

                else if(input == 'q')
                {
                        System.out.println("Game Over");
                }

                else
                {
                        System.out.println("Enter valid input");
                }

                return myMove;

        }
        while(!isCorrect);

}

@Override
public void runGame() {
        String myMove;
        String compMove;

        int numRounds = 1;
        int score = 0;

        //for gui
        do
        {
                if(myResult == "")
                {
                        setMove();
```

```
                }
                if(myResult != "")
                {
                        do
                        {
                                myMove = myResult;
                                move = GamePlayFactory.getMoveType();
                                compMove = move.getMove();
                                view.display("Computer Played: " + compMove);

                                compMsg = "Computer Played: " + compMove;
                                myMsg = "Your Bet: " + myMove;



                                if(compMove.equals("rock") &&
myMove.equals("paper"))
                                {
                                        view.display("You Win! " + numRounds + "
round(s) played out of 4");
                                        msg = "You Win! " + numRounds + " round(s)
played out of 4";
                                        score++;
                                }
                                else if(compMove.equals("scissors") &&
myMove.equals("rock"))
                                {
                                        view.display("You Win! " + numRounds + "
round(s) played out of 4");
                                        msg = "You Win! " + numRounds + " round(s)
played out of 4";
                                        score++;
                                }
                                else if(compMove.equals("paper") &&
myMove.equals("scissors"))
                                {
                                        view.display("You Win! " + numRounds + "
round(s) played out of 4");
                                        msg = "You Win! " + numRounds + " round(s)
played out of 4";
```

```
                                        score++;
                                }
                                else if(compMove == myMove)
                                {
                                        view.display("It's a Tie! " + numRounds + "
round(s) played out of 4");
                                        msg = "It's a Tie! " + numRounds + " round(s)
played out of 4";
                                }
                                else
                                {
                                        view.display("You Lost: " + numRounds + "
round(s) played out of 4");
                                        msg = "You Lost: " + numRounds + " round(s)
played out of 4";
                                }
                                numRounds++;
                        }
                        while(numRounds < 5);
                        view.display("Total Score: " + score + " out of 4 games. Play
again? (t/f)");
                        msg = "Total Score: " + score + " out of 4 games.";

                        numRounds = 1;
                        break;
                }

        }while(setMove == false);

        //for console
        /*
        do
        {
                do
                {
                        view.display("\n" +  "Enter your bet: Rock (r), Paper(p), Scissors
(s) or Quit (q)?" );
                        input = view.getInput();
                        myMove = setMove();
                        //myMove = myResult;
```

```
                        move = GamePlayFactory.getMoveType();
                        compMove = move.getMove();
                        view.display("Computer Played: " + compMove);

                        compMsg = "Computer Played: " + compMove;
                        myMsg = "Your Bet: " + myMove;



                        if(compMove.equals("rock") && myMove.equals("paper"))
                        {
                                view.display("You Win! " + numRounds + " round(s)
played out of 4");
                                msg = "You Win! " + numRounds + " round(s) played out
of 4";
                                score++;
                        }
                        else if(compMove.equals("scissors") && myMove.equals("rock"))
                        {
                                view.display("You Win! " + numRounds + " round(s)
played out of 4");
                                msg = "You Win! " + numRounds + " round(s) played out
of 4";
                                score++;
                        }
                        else if(compMove.equals("paper") &&
myMove.equals("scissors"))
                        {
                                view.display("You Win! " + numRounds + " round(s)
played out of 4");
                                msg = "You Win! " + numRounds + " round(s) played out
of 4";
                                score++;
                        }
                        else if(compMove == myMove)
                        {
                                view.display("It's a Tie! " + numRounds + " round(s)
played out of 4");
                                msg = "It's a Tie! " + numRounds + " round(s) played out
of 4";
```

```
                        }
                        else
                        {
                                view.display("You Lost: " + numRounds + " round(s)
played out of 4");
                                msg = "You Lost: " + numRounds + " round(s) played out
of 4";
                        }
                        numRounds++;
                }
                while(numRounds < 5);
                view.display("Total Score: " + score + " out of 4 games. Play again?
(t/f)");
                msg = "Total Score: " + score + " out of 4 games.";

                numRounds = 1;

        }
        while((char) view.getInput() == 't');
        */


        //view.display("Game Over! Thanks for Playing!");
    }

    @Override
    public void init() {
        view.display("This program lets you play Rock, Paper, Scissors" + "\n" + "Choose
to play "
                        + "console or GUI? (c/g)");
        IFrame guiGame = new RPSGUI();
        input = view.getInput();
        boolean isCorrect = false;

        do
        {
                input = Character.toLowerCase(input);
                if(input == 'c')
                {
                        runGame();
```

```
                    isCorrect = true;
                }

                else if(input == 'g')
                {
                    guiGame.buildCanvas();
                    guiGame.buildButtonPanel();

                    JFrame f = guiGame.getFrame();
                    f.setVisible(true);

                    runGame();
                    isCorrect = true;

                }
                else
                {
                    System.out.println("Enter valid input");
                }

                input = view.getInput();

            }
            while(!isCorrect);
        }

}
```

```
package RPSGame;

import java.awt.*;

import javax.swing.*;

public class RPSGUI implements IFrame {
	JPanel pnl, canvas;

	@Override
	public void buildButtonPanel() {
		pnl = new JPanel();

		ImageIcon r = new
ImageIcon("C:\\\\Users\\\\prosi\\\\OneDrive\\\\Pictures\\\\Rock_001.png");
		ImageIcon p = new
ImageIcon("C:\\Users\\prosi\\OneDrive\\Pictures\\paper.png");
		ImageIcon s = new
ImageIcon("C:\\Users\\prosi\\OneDrive\\Pictures\\scissors_002.png");

		Image rImage = r.getImage();
		Image pImage = p.getImage();
		Image sImage = s.getImage();

		Image rResize = rImage.getScaledInstance(200, 200, Image.SCALE_SMOOTH);
		Image pResize = pImage.getScaledInstance(200, 200,
Image.SCALE_SMOOTH);
		Image sResize = sImage.getScaledInstance(200, 200, Image.SCALE_SMOOTH);

		r = new ImageIcon(rResize);
		p = new ImageIcon(pResize);
		s = new ImageIcon(sResize);

		JButton rock = new JButton(r);
		JButton paper = new JButton(p);
		JButton scissors = new JButton(s);

		GridLayout layout = new GridLayout();
		layout.setHgap(40);
```

```
        pnl.setLayout(layout);
        pnl.setBorder(BorderFactory.createEmptyBorder(40, 40, 40, 40));

        pnl.add(rock);
        pnl.add(paper);
        pnl.add(scissors);

        ActionListener listener = new ActionListener()
        {
                public void actionPerformed(ActionEvent e)
                {
                        if(e.getSource() == rock)
                        {
                                GameController.RockBtn();
                                canvas.repaint();
                        }
                        else if(e.getSource() == paper)
                        {
                                GameController.PaperBtn();
                        }
                        else if(e.getSource() == scissors)
                        {
                                GameController.ScissorsBtn();
                        }
                }
        };

        rock.addActionListener(listener);
        paper.addActionListener(listener);
        scissors.addActionListener(listener);

}

@Override
public void buildCanvas() {
        class Canvas extends JPanel
        {

                public void paintComponent(Graphics g)
                {
```

```
                        super.paintComponent(g);
                        g.setFont(new Font("TimesRoman", Font.PLAIN, 24));
                        g.drawString(GameController.compMsg, 200, 50);
                        g.drawString(GameController.myMsg, 700, 50);
                        g.drawString(GameController.msg, 350, 200);
                    }
                }
                canvas = new Canvas();
        }

        @Override
        public JFrame getFrame() {
                JFrame frame = new JFrame("Game");

                frame.add(pnl, "North");
                frame.add(canvas, "Center");
                frame.setSize(1200, 800);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                return frame;
        }

}
```

```
package RPSGame;

import java.util.Random;

class PlayRandom implements Move
{
        public String getMove()
        {
                String compMove = "";
                int rand = (int)(Math.random() * 3);
                if(rand == 0)
                {
                        compMove = "rock";
```

```
                }
                else if(rand == 1)
                {
                        compMove = "paper";
                }
                else
                {
                        compMove = "scissors";
                }

                return compMove;
        }
}

class PlayStrategy implements Move
{
        public String getMove()
        {
                String compMove = "";

                if(GameController.setMove().equals("rock"))
                {
                        compMove = "paper";
                }
                else if(GameController.setMove().equals("paper"))
                {
                        compMove = "scissors";
                }
                else
                {
                        compMove = "rock";
                }
                return compMove;
        }
}

class GamePlayFactory
{
        public static Move getMoveType()
        {
```

```
                Random rnd = new Random();
                int select = 1 + rnd.nextInt(2);
                Move move = null;

                switch(select)
                {
                case 1: move = new PlayRandom();
                        break;
                case 2: move = new PlayStrategy();
                }

                return move;

        }
}
```

```
package RPSGame;

import javax.swing.JFrame;

public interface IController {
        void runGame();
        void init();
}

interface IView
{
        void display(String message);
        <T> T getInput();
}

interface IFrame
{
        void buildButtonPanel();
        void buildCanvas();
        JFrame getFrame();
}
```

```
interface Move
{
	public String getMove();
}
```

RockPaperScissors - RockPaperScissors/src/RPSGame/RPSGame.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Expl... | RPSGUI.java | RPSGame.java | IController.java | GameLogic.java | Task List | Outline

RockPaperScissors

```
90              myMove = "rock";
91          }
92
93          else if(input == 'p' || myResult == "paper")
94          {
95              myMove = "paper";
```

Problems | Javadoc | Declaration | Console

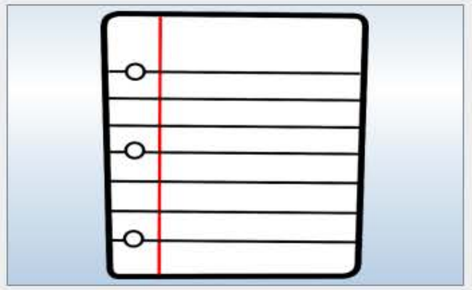
RPSGame [Java Application] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe (Apr 21, 2021, 5:26:20 PM)

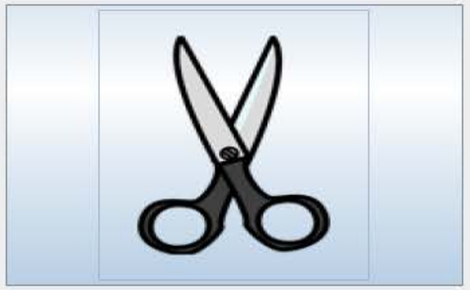
```
This program lets you play Rock, Paper, Scissors
Choose to play console or GUI? (c/g)
c

Enter your bet: Rock (r), Paper(p), Scissors (s) or Quit (q)?
r
Computer Played: paper
You Lost: 1 round(s) played out of 4

Enter your bet: Rock (r), Paper(p), Scissors (s) or Quit (q)?
p
Computer Played: scissors
You Lost: 2 round(s) played out of 4

Enter your bet: Rock (r), Paper(p), Scissors (s) or Quit (q)?
p
Computer Played: scissors
You Lost: 3 round(s) played out of 4

Enter your bet: Rock (r), Paper(p), Scissors (s) or Quit (q)?
s
Computer Played: rock
You Lost: 4 round(s) played out of 4
Total Score: 0 out of 4 games. Play again? (t/f)
t

Enter your bet: Rock (r), Paper(p), Scissors (s) or Quit (q)?
r
Computer Played: paper
You Lost: 1 round(s) played out of 4

Enter your bet: Rock (r), Paper(p), Scissors (s) or Quit (q)?
q
Game Over
Game Over
Game Over
Computer Played: rock
You Lost: 2 round(s) played out of 4

Enter your bet: Rock (r), Paper(p), Scissors (s) or Quit (q)?
```

Type here to search

5:26 PM
4/21/2021

Game





Computer Played: scissors

Your Bet: paper

Total Score: 0 out of 4 games.

```
Enter valid input
Enter valid input
Enter valid input
Computer Played: scissors
You Lost: 1 round(s) played out of 4
Computer Played: paper
It's a Tie! 2 round(s) played out of 4
Computer Played: scissors
You Lost: 3 round(s) played out of 4
Computer Played: scissors
You Lost: 4 round(s) played out of 4
Total Score: 0 out of 4 games. Play again? (t/f)
```

Task List

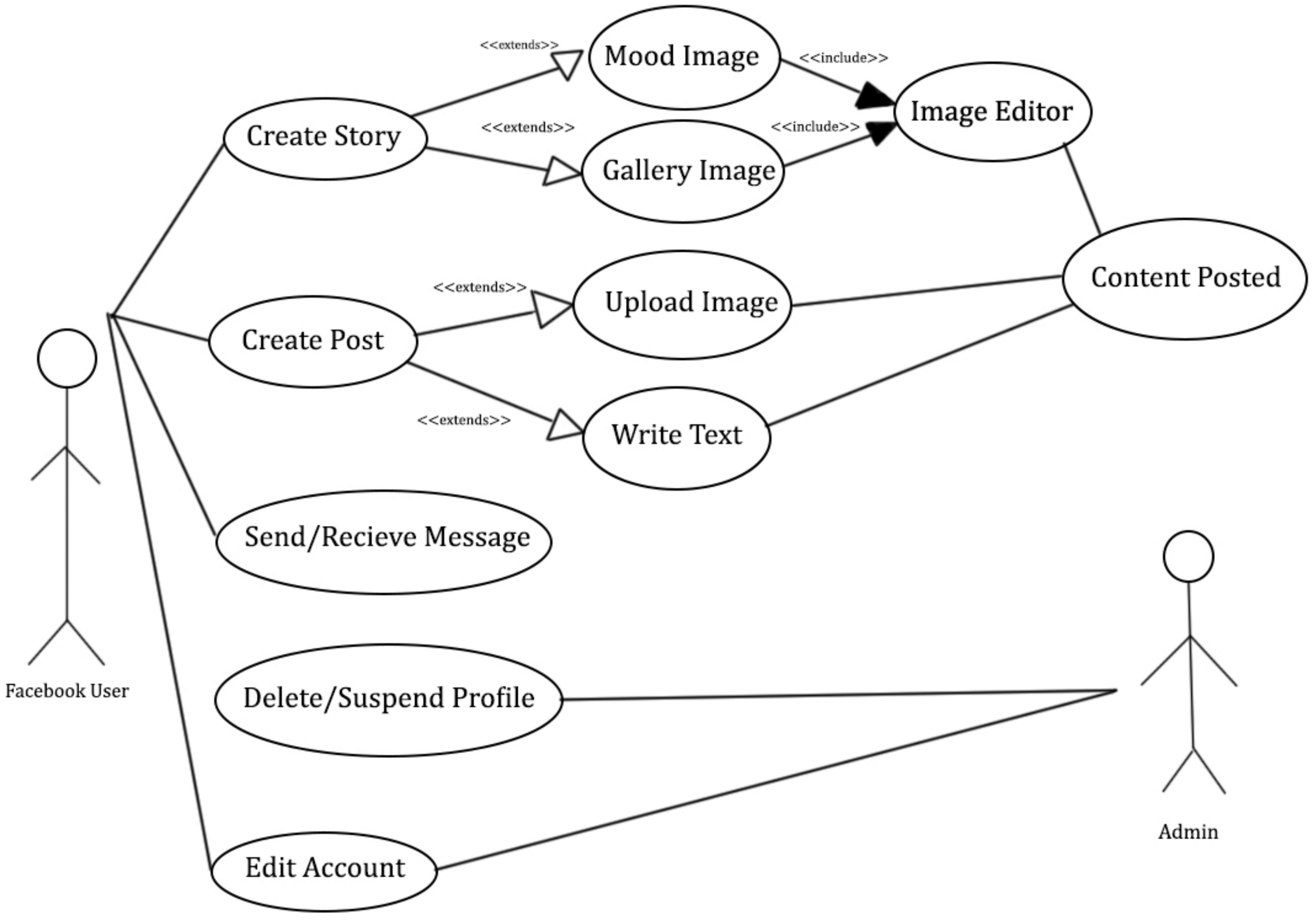
Outline

- input : Character
- myResult : String
- compMsg : String
- myMsg : String
- msg : String
- setMove : boolean
- GameController()
- RockBtn() : String
- PaperBtn() : String
- ScissorsBtn() : String
- setMove() : String
- runGame() : void
- init() : void

Type here to search

6:53 PM
4/21/2021

ID:	1234
Title:	Facebook Create Story
Description:	Facebook user opens Facebook in a browser or an app. Then he/she adds his/her own content to their Facebook story to share with friends for 24 hours.
Primary Actor:	Facebook User
Preconditions:	User has a Facebook account User has access to Facebook on an app or browser User is logged into their Facebook
Postconditions:	Facebook User has shared content added to their 'story'
Main Success Scenario:	1. User selects "Create Story" on the Facebook homepage 2. System displays a variety of features to add (Text, Music, Boomerang, Mood, Selfie, and Poll) as well as images from the users gallery 3. User selects an image from their gallery 4. System displays the enlarged image, a share to story button and functions to edit the image 5. User selects "Share to Story" 6. System adds the image to the Users story and shares it on Facebook, use case ends.
Extensions:	3a. User selects 'Mood' from the features available - 3a1. System displays multiple GIFs that encapsulate a 'mood' - 3a2. User selects a GIF and the use case continues to completion 4a. User edits their image by selecting the function 'sticker' - 4a1. System displays a menu of 'stickers' the User can choose from - 4a2. User selects a 'Heart' sticker from the menu - 4a3. System displays a heart image in the center of the Users gallery image, use case continues to completion
Frequency of Use:	Decided by the User
Status:	Content posted
Owner:	Facebook User


<<extends>>
Mood Image
<<include>>
Image Editor
Create Story
<<extends>>
Gallery Image
<<include>>
Content Posted
<<extends>>
Upload Image
Create Post
<<extends>>
Write Text
Send/Recieve Message
Delete/Suspend Profile
Edit Account
Facebook User
Admin