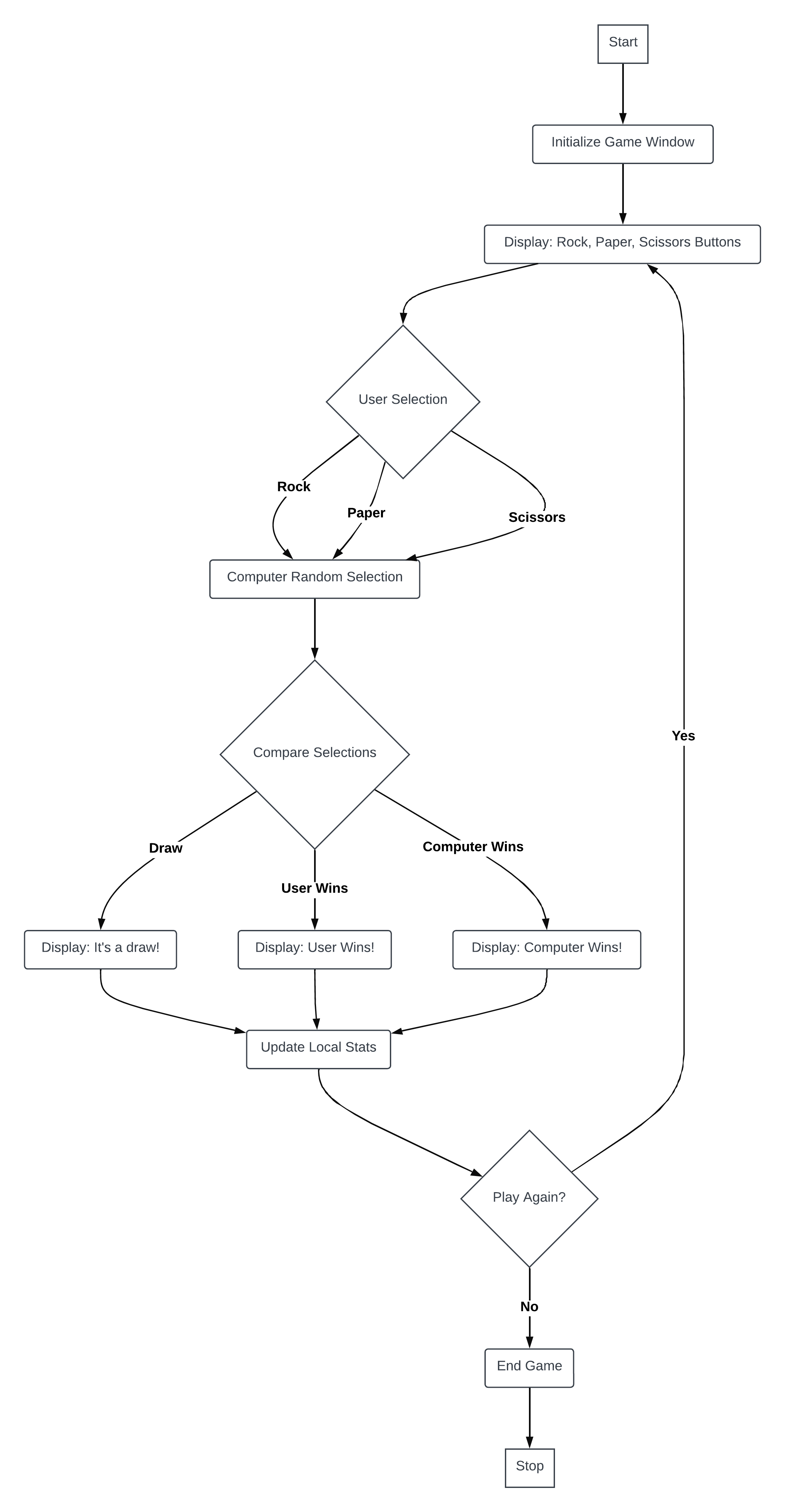


Student ID: Dikshith Reddy Macherla

Student Name: 0789055

UML Activity Diagram: Export your UML Activity Diagram as a PNG or a JPEG and copy and paste it here.



Program Testing: Screenshot based functionality testing with a short description of each test. You won’t have very many test cases (it is a simple game with only a button click as user input).

Test Case 1: User Selects Rock

Objective: Verify the game correctly handles the scenario when the user selects "Rock."

Expected Result: The game randomly selects Rock, Paper, or Scissors for the computer. The result is displayed in a popup, indicating whether the user won, lost, or drew. The scores are updated accordingly.

Screenshot:

A screenshot of a computer

Description automatically generated

Test Case 2: User Selects Paper

Objective: Ensure the game accurately processes the user's selection of "Paper."

Expected Result: The computer's choice is shown, along with the game's outcome based on the rules (Paper beats Rock, loses to Scissors, and draws with Paper). Scores are updated to reflect the result.

Screenshot:

A screenshot of a computer screen

Description automatically generated

Test Case 3: User Selects Scissors

Objective: Test the game's response to the user choosing "Scissors."

Expected Result: The computer's selection is displayed, and the game announces if the user won, lost, or drew the round, based on the rules (Scissors beats Paper, loses to Rock). The score is updated to reflect the new totals.

Screenshot:

A screenshot of a computer screen

Description automatically generated

Test Case 4: Tallying Scores

Objective: Confirm that the game correctly tallies wins, losses, and draws over multiple rounds.

Expected Result: The score accurately reflects the cumulative outcome of all rounds played.

Screenshot:

A close-up of a button

Description automatically generated

Copy and paste your code below:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.util.Random;

public class Main {

    // Enum for the choices

    private enum Choice {

        ROCK, PAPER, SCISSORS

    }

    // Scores

    private int wins = 0;

    private int losses = 0;

    private int draws = 0;

    // GUI Components

    private JFrame frame;

    private JLabel scoreLabel;

    public Main() {

        // Setup the main window

        frame = new JFrame("Rock Paper Scissors Game");

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setLayout(new FlowLayout());

        JButton rockButton = new JButton("Rock");

        JButton paperButton = new JButton("Paper");

        JButton scissorsButton = new JButton("Scissors");

        rockButton.addActionListener(e -> playGame(Choice.ROCK));

        paperButton.addActionListener(e -> playGame(Choice.PAPER));

        scissorsButton.addActionListener(e -> playGame(Choice.SCISSORS));

        scoreLabel = new JLabel("Wins: 0 Losses: 0 Draws: 0");

        frame.add(rockButton);

        frame.add(paperButton);

        frame.add(scissorsButton);

        frame.add(scoreLabel);

        frame.pack();

        frame.setVisible(true);

    }

    private void playGame(Choice userChoice) {

        Choice computerChoice = getComputerChoice();

        String message;

        if (userChoice == computerChoice) {

            draws++;

            message = "It's a Draw! Both chose " + userChoice;

        } else if ((userChoice == Choice.ROCK && computerChoice == Choice.SCISSORS) ||

                   (userChoice == Choice.PAPER && computerChoice == Choice.ROCK) ||

                   (userChoice == Choice.SCISSORS && computerChoice == Choice.PAPER)) {

            wins++;

            message = "You Win! " + userChoice + " beats " + computerChoice;

        } else {

            losses++;

            message = "You Lose! " + computerChoice + " beats " + userChoice;

        }

        updateScore();

        JOptionPane.showMessageDialog(frame, message);

    }

    private Choice getComputerChoice() {

        Random random = new Random();

        return Choice.values()[random.nextInt(Choice.values().length)];

    }

    private void updateScore() {

        scoreLabel.setText("Wins: " + wins + " Losses: " + losses + " Draws: " + draws);

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(Main::new);

    }

}

Prove using a screenshot (or more than one screenshot) that you created a feature branch named adding\_funcitonality, and that you merged that feature branch with your main branch successfully.

A screenshot of a computer program

Description automatically generated