Software Design Specification

for

Rock, Paper, Scissors, Saw Game

Version 1.0

Prepared by Casey Munga

The Red Howler

03/11/2020

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Casey Munga	03/15/2020	Initial Design	1.0

1. Introduction

1.1 Purpose

This document will describe the design for the Rock, Paper, Scissors, Saw Game as a new implementation.

1.2 Document Conventions

GUI – Graphical User Interface

1.3 Intended Audience and Reading Suggestions

This document is intended for the following stakeholders such as the project managers, users, developers, testers and document writers as well as the internal stakeholders.

1.4 Product Scope

This is a new implantation of the rock, paper, scissors, saw game. Initially there will be text based interface; however, there are plans for a GUI interface which is outside the current scope of this project. It will be implemented using Java Object Oriente approach.

1.4.1 Purpose

The purpose of this software is to be a stand-alone representation of the classic hand game of Rock , Paper, Scissors however with the addition of the Saw category. The object will be an AI representation of the computer representing a third player, in addition to two other human players.

1.4.2 High Level Flow

When the program starts the computer will prompt the user for the names of two human players, whose values will be used for the statistics and the prompts in the game.
The user will be presented a menu with the following selections

- Play the game
- Show the game results.
- Show statistics
- Exit

The Rules options selection will display the rules of the game. The Statistics option will display the aggregated wins, losses and ties for each player per round and per game. The Exit option will close the program.

When the user selects to play the game, each player will be able to choose either rock, paper scissors or saw (as the weapon against) his opponent, which in this case is the computer AI.
The user will be able to play three rounds and that will conclude a game.

At the conclusion of the round a winner will be selected and the user will be informed if he is the wins, ties or loses the game round. At the conclusion of the three rounds the user will be informed of the outcome of the game.

The user may opt to play multiple times and his statistics for rounds and games will be displayed, along with the top human winning score.

1.5 References

Documents that are of reference and to be used as guideline for design and implementation is the **SRS-RPSS-Game-1.doc** which is the software specification document.

1.6 System Overview

This system

2. Design Considerations

2.1 Assumptions and Dependencies

In order for this software to work as specified the following assumptions will be noted

Related software or hardware

- A usable system must be in place that is adequate to run the software
- The user must be familiar with a computer system

2.2 General Constraints

Software environment

The project must be written in Java and use the Object oriented approach.

2.3 Goals and Guidelines

The goal for this software is that it should be simple to implement and the written with an open ended design which would lend itself to expansion of weapons and increased players. It should follow the requirements in the reference documents and not go beyond the scope mentioned. The design UX design should be text based and the language simple enough to ensure a pleasant gaming experience. The output should be clear and informative.

2.4 Development Methods

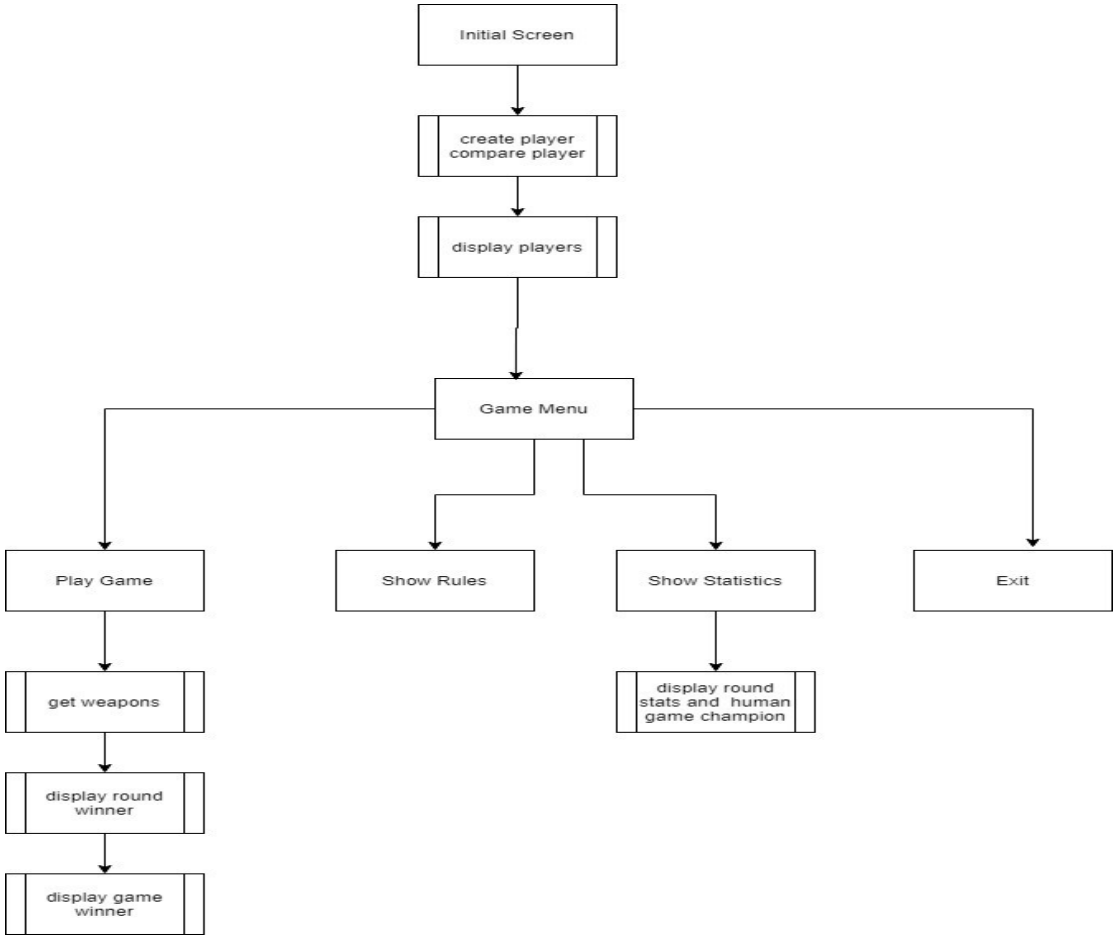
The software will be written with the Object Oriented methodology and in Java.

3. System Architecture

3.1 Architectural Strategies

The system will use Java and will employ a procedural system design to fully expose objects, methods and classes. The program will be written to be able to incorporate additional humans and other weapons for future installments should the need arise. Object Oriented methodology will be used.
The standard input will be a computer capable of running the software and the following peripherals of a keyboard, computer and monitor. This software will be a stand-alone system on a single PC. The game results will be held in memory at which time it will be considered the current game. All statistics and game output will be not be persisted when the Exit menu option is chosen.

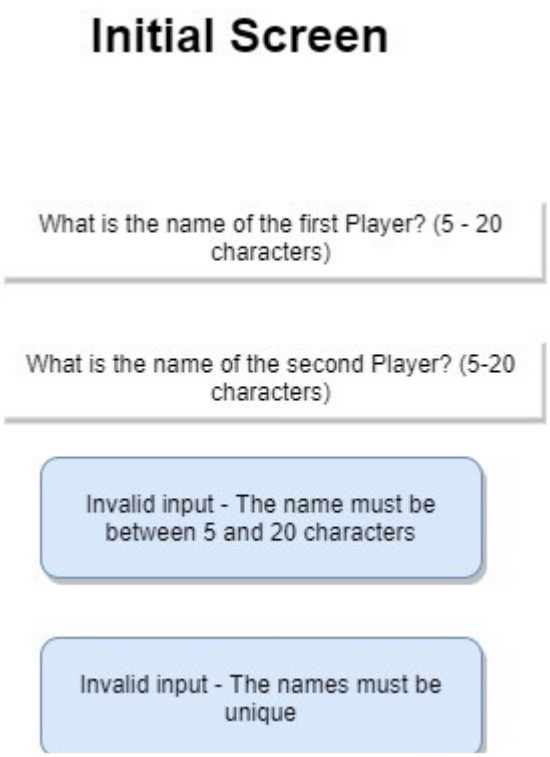
3.2 High Level Overview of System Architecture



4. Human Interface Design

4.1 Screen Images

Figure 1



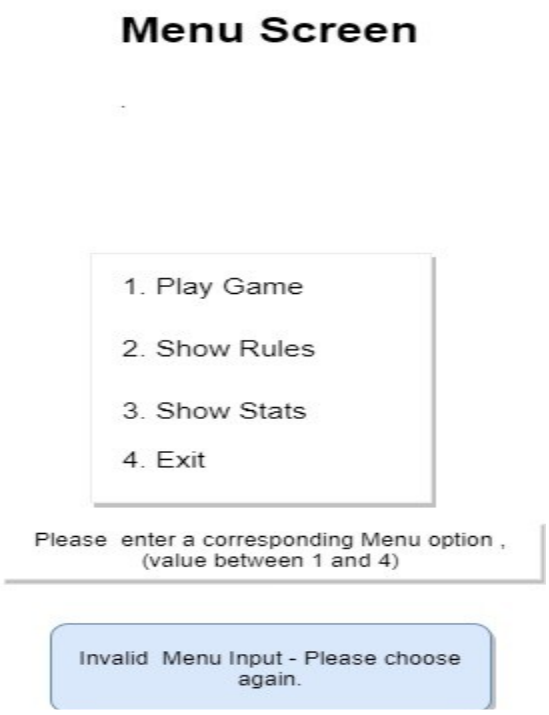

Menu Screen
1. Play Game
2. Show Rules
3. Show Stats
4. Exit
Please enter a corresponding Menu option , (value between 1 and 4)
Invalid Menu Input - Please choose again.

Figure 2

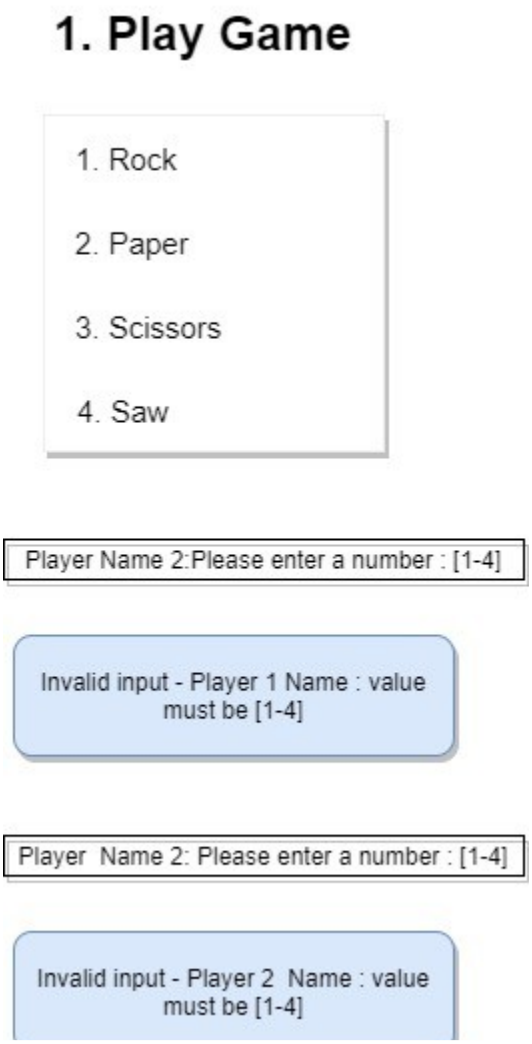
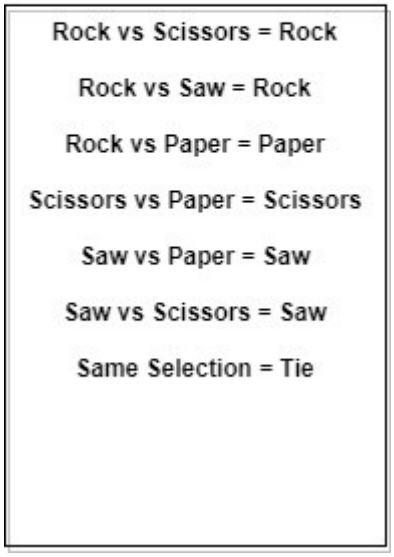

1. Play Game
1. Rock
2. Paper
3. Scissors
4. Saw
Player Name 2:Please enter a number : [1-4]
Invalid input - Player 1 Name : value must be [1-4]
Player Name 2: Please enter a number : [1-4]
Invalid input - Player 2 Name : value must be [1-4]

Figure 3

2. Game Rules



4. Exit

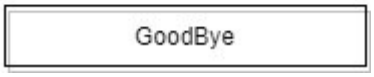
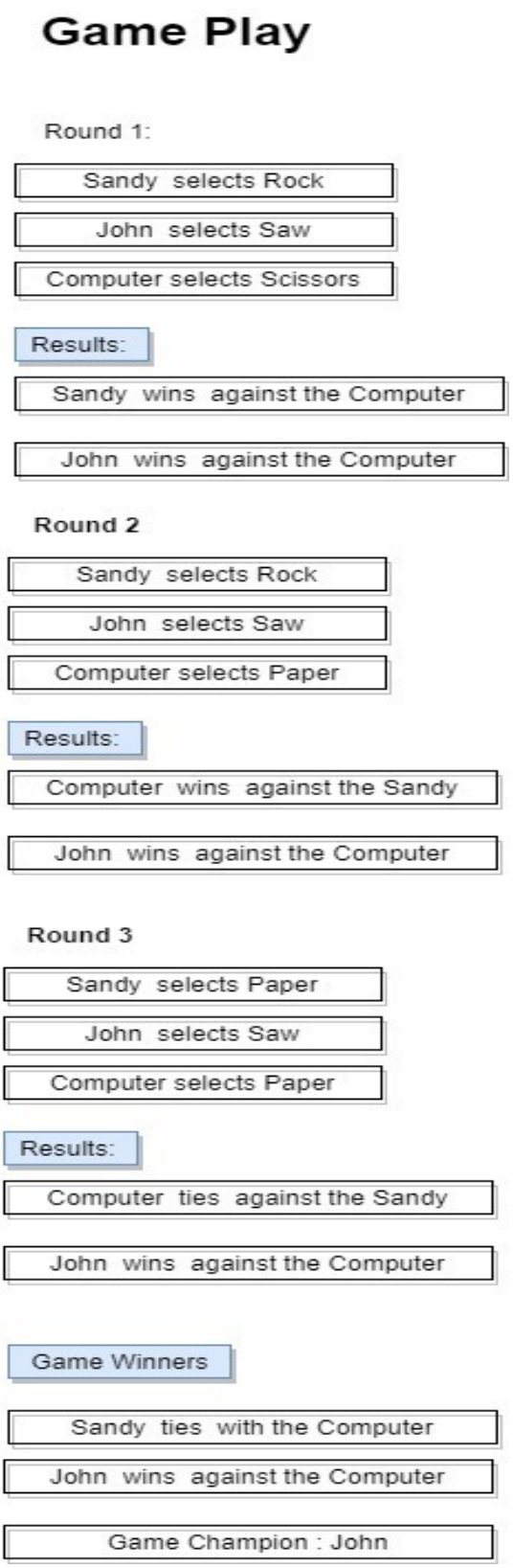


Figure 4


Game Play
Round 1:
Sandy selects Rock
John selects Saw
Computer selects Scissors
Results:
Sandy wins against the Computer
John wins against the Computer
Round 2
Sandy selects Rock
John selects Saw
Computer selects Paper
Results:
Computer wins against the Sandy
John wins against the Computer
Round 3
Sandy selects Paper
John selects Saw
Computer selects Paper
Results:
Computer ties against the Sandy
John wins against the Computer
Game Winners
Sandy ties with the Computer
John wins against the Computer
Game Champion : John

4.2 Screen Objects and Actions

Figure 1:

- Describes the input of the first player. If the player's name does not meet the criteria an error message will be displayed of invalid input and the player will be prompted to enter the name again.

- Describes the prompting and input of the second player. If the player's name does not meet the criteria an error message will be displayed of invalid input and the player will be prompted to enter the name again.

- The player's name will be validated for uniqueness and it fails an error message will be displayed and the second player will be prompted to enter the name again.

Figure 2:

- Describes the game menu options, when the player selects this option. The player must enter a number with in range [1..4] and the input will be validated. If it fails validation the player will be given an error message and prompted to input again.

Figure 3.

- Describes the game rules that will be displayed when player selects this option

Figure 4:

- Describes the conclusion of the game when the player selects this option. A message of Goodbye will be displayed.

Figure 5:

- Describes the game play that will occur and a sample of how the output will look as well as the implementation of the game rules and appropriate winners.

5. Detailed System Design

5.1 Data Structures

The major data types used will be classes, lists and list arrays.

5.2 Initial Screen

This screen will receive the initial input of the players and it is the starting point of the program.

5.2.1 Responsibilities

This module is responsible for

- Creating the player after receiving input from the player
- Validating the input.
- Displaying error messages derived from validation
- Calling the Game Menu module
-

5.2.2 Uses/Interactions

This module is the entry component for the game.

5.2.3 Constraints

- The program should be started and there should be at least one player to start the game.
- The maximum limit of human players is two.
- The player name must be more than four characters but less than twenty-one.
- The players names must be unique.
-

5.2.4 Composition

Function	Description
main	Receives input for and displays players names and calls the game menu
list createPlayers()	Receives input for the player's name and validates the name
boolean comparePlayerNames()	Compares the players names for uniqueness
displayPlayers()	Displays the player stats
Display gameMenu()	Displays the game menu on the screen

5.2.5 Resources

none

5.2.6 Processing

CreatePlayers():list

1. get Player Name
2. Loop until playerCount=2 and playerNames are unique
    1. call validatePlayerName if true increment playerCount and get another player
    2. check for two players, call CompareNames() if false continue loop
3. initialise player list with name and stats

ValidatePlayerName():boolean

1. check length of string for length > 4 and length < 21
2. return true if validated false if not validated

compareNames(string,string):boolean

1. check names for uniqueness. If they are not give error message
2. return true if validated false if not validated

displayPlayers(larray[][])

1. display players and Stats

5.3 Main Menu

This module will display the Game Menu and wait for input from the player

5.3.1 Responsibilities

This module is responsible for:

- displaying the game menu
- validating the input does not fall outside the menu options

5.3.2 Uses/Interactions

This module is called by the main program. It starts the game and gives access to the functionality of the program.

5.3.3 Constraints

- The input for the menu option must be an integer between 0 and 5
- A menu must be display with 4 options
- The first option must be labeled 1. PlayGame
- The second option must be labeled 2. Show Rules
- The third option must be labeled 3. Show Stats
- The fourth option must be labeled 4. Exit

5.3.4 Composition

Function	Description
generateMenu():int	Displays menu on screen and prompt for player input. Returns player menu choice
playGame(player,player,)	Initiates the game
showRules()	Displays the games won and lost and ties by the players
showStats	Displays the accumulative statistics per round and game for each player
exitGame()	Displays “GoodBye” Message .Ends the game,

5.3.5 Resources

none

5.3.6 Processing

…
generateMenu():None

1. Loop to display 4 menu choices
2. Call playGame()
3. Call showRules()
4. Call showStats()

5. Call exitGame()
6. Invalid Selection will generate an error message

displayRules():None

1. Display rules of game and how player can win.
2. Present player with a choice to return to the menu
3. Return to the menu.

showStats():None

1. Display games player won and lost.
2. Present player with choice to return to the menu
3. Return to the menu.

playRound():None

1. Player will play a single round of the game.
2. Present player with a choice to return to the menu after the game is completed.
3. Return to the menu.

exitGame():None
Displays Message: Goodbye

5.4 Play Game

This module will initiate and play the Rock, Paper, Scissors, Saw Game

5.4.1 Responsibilities

This module is responsible for:

- Prompting the player for his weapon (rock, paper, scissors,saw)
- Validation of input values[1..4]
- Using the correct player name when prompting the player.
- Choosing a random weapon for the AI
- Displaying the players selection and the winner of the round
- Restriction of 3 rounds per game.
- Displaying the winner of the game

5.4.2 Uses/Interactions

This module is called by the gameMenu module.

5.4.3 Constraints

Preconditions

- The player must have selected 1 on the game Menu.

Other constraints

- The weapon choice must be a numerical value in the range of [1..4]
- A game is restricted to 3 rounds
- A winner must be declared at the end of every round
- A winner must be declares at the end of every game

5.4.4 Composition

Function	Description
Getweapon():int	Prompts and validates the choice of weapon. Will randomize the computer weapon choice
displayRoundWinner(int,int,):int	Validates the rules and declares a winner(player,computer)
displayGameWinner(int,int,int))	Validates the rules and declares a game winner(Array[][])

5.4.5 Resources

none

5.4.6 Processing

1. Loop until rounds = 3
    1. loop for each player
    2. call getWeapon():int 1=player1, 2=player2 3=computer
    3. displayRoundWinner(int,int) :player1 weapon, player2 weapon:, computer weapon and place winner in the winner array

2. displayRoundWinner(int,int,int):int :player1 round totals player2 round totals, computer round totals and returns the Game winner

getWeapon():int
loop: until choice is valid
1. prompt the player for input
2. validate choice is between range [1..4] (1=Rock, 2=Paper, 3=Scissors, 4=Saw)

return weapon.

displayRoundWinner(int,int,array<>)
1. compare player weapon with computer weapon using the rules.
2. Insert players and round totals in the array
3. display round winner name and stats

displayGameWinner(array<>)
1. compare total the number of games the.
2. Insert players and games totals iwon in the array
3. display game winner name and stats

5.5 Show Stats

This module displays the player statistics.

5.5.1 Responsibilities

This module is responsible for:
- Displaying the wins. Losses, and ties of the games
- Displaying the game champion on the screen

5.5.2 Uses/Interactions

This module is called by the gameMenu module.

5.5.3 Constraints

Preconditions
- The player must have selected 3 on the game Menu.
- Must use a 2 dimensional array to keep track of the statistics
- Must display the number of rounds each player won, lost, or had a tie against the computer, each players
- Must display stats on new line
- Stats for each player will have his name and rounds and games stats will be shown on same line
- Must display the overall human winner based on the most won games and lost least games against the computer compared to other human players

5.5.4 Composition

Function	Description
showStats()	Displays the game stats of players on the scree

5.5.5 Resources

None

5.5.6 Processing

ShowStats()

1. Displays the stats and winner name on the same line
2. Displays the overall human winner by iterating through the wins of both players and declaring a game champion.
3. Each player's aggregate round scores will be displayed.
4. The players aggregate game scores will be displayed.
5. The player with the greatest number of wins will be declared the winner. If more than one
6. with the greater number of wins If both players had same number of wins and losses for games, a tie will be announced for the overall winner

5.6 Rules

This module displays the rules that are used in the game.

5.6.1 Responsibilities

This module is responsible for:

- Displaying the rules
  - Rock breaks scissors and Saw therefore rock wins over scissors and saw. Rock loses against paper.
  - Scissors cut paper therefore scissors win over paper. Scissors lose against rock and Saw.
  - Paper covers rock therefore paper wins over rock. Paper loses against scissors and saw
  - Saw cuts through scissors and paper therefore saw wins over scissors and paper. Saw loses against rock.
  - If player and computer make the same selection, there is a tie

5.6.2 Uses/Interactions

This module is called by the gameMenu

5.6.3 Constraints

Preconditions

- The player must have selected 2 on the game Menu.

5.6.4 Composition

Function	Description
showRules	Displays the rules to the screen and how the player can win.

5.6.5 Resources

none

5.6.6 Processing

ShowRules()

- Displays the game rules.

5.7 Exit Game

This module will exit the game

5.7.1 Responsibilities

This module is responsible for:
- Displaying a message “Goodbye!”.
- Ending the game
- Exiting the program

5.7.2 Uses/Interactions

This module is called by the Game Menu when the player selects this option

5.7.3 Constraints

Player must select option 4 of the menu

5.7.4 Composition

Function	Description
exitGame()	Displays the Message “Goodbye!”

5.7.5 Resources

None

5.7.6 Processing

ExitGame()
1. Display Message GoodBye
2. Exits Program

Appendix A: Glossary

AI
Artificial Intelligence

GUI
Graphical user interface (screen)