Rock Paper Scissors Design Document

Inputs

- User choice of tool
- User choice to set strength of tools
    - User input strengths for tools

Outputs

- Log of what tool the computer chose
- Who won each game played (Computer or User)
- Total scoreboard of overall match (Human Wins: // Computer Wins: // Ties:)

Classes Needed:

Tool

```
Enum FightResult {WIN, DRAW, LOSE};
Int strength;
Char type;
Tool()
Tool(int strength)
setStrength(int strength)
fight(Tool) = 0
```

Rock : Public Tool

```
Rock()  /* set type to 'r' and strength to 1 */
Rock(int strength)
FightResult fight(Tool) {
    Int tempStrength;
    If tool is scissors
        tempStrength = strength * 2
    If tool is paper
        tempStrength = strength / 2

    If tempStrength is greater than other tool's strength
        return WIN
    Else If tempStrength is less than than other tool's strength
        return LOSE
    Else //if equal
        return DRAW
}
```

Paper : Public Tool

```
    Paper()  /* set type to ‘p’ and strength to 1 */
    Paper(int strength)
    FightResult fight(Tool) {
        Int tempStrength;
        If tool is rock
            tempStrength = strength * 2
        If tool is scissors
            tempStrength = strength / 2

        If tempStrength is greater than other tool’s strength
            return WIN
        Else If tempStrength is less than than other tool’s strength
            return LOSE
        Else //if strengths are equal
            return DRAW
    }
```

Scissors : Public Tool

```
    Scissors()  /* set type to ‘s’ and strength to 1 */
    Scissors(int strength)
    FightResult fight(Tool) {
        Int tempStrength;
        If tool is paper
            tempStrength = strength * 2
        If tool is rock
            tempStrength = strength / 2

        If tempStrength is greater than other tool’s strength
            return WIN
        Else If tempStrength is less than than other tool’s strength
            return LOSE
        Else //if equal
            return DRAW
    }
```

RPSGame

```
Tool *humanTool
Tool *computerTool
Int human_wins
Int computer_wins
Int ties

Void play() {
    FightResult results = humanTool fights computerTool
    If results is WIN
        Add 1 to human_wins
    If results is LOSE
        Add 1 to computer_wins
    If results is DRAW
        Add 1 to ties
    Print score
}

Void setComputerTool() {
    Choose random number between 1 and 3
    Num = 1
        computerTool  = Rock
    Num = 2
        computerTool = Paper
    Num = 3
        computerTool = Scissors
}

Void setHumanTool(Tool userToolChoice) {
    If user chose rock
        userTool = rock
    If user chose scissors
        userTool = scissors
    If user chose paper
        userTool = paper
}

Void printScore() {
    Print human_wins, computer_wins, ties to console
}
```

Rock, Paper, Scissors Test Plan and Results

User Input (Human Tool) **• Different Strengths?** **• Chose Tool**	**Computer Tool and Strength**	**Expected Output**	**Actual Output**
Basic Tests with Default Strengths for Rock Paper and Scissors			
Default Strengths R, 1	R, 1	Draw	Draw
	P, 1	Computer Wins	Computer Wins
	S, 1	Human Wins	Human Wins
Default Strengths P, 1	R, 1	Human Wins	Human Wins
	P, 1	Draw	Draw
	S, 1	Computer Wins	Computer Wins
Default Strengths S, 1	R, 1	Computer Wins	Computer Wins
	P, 1	Human Wins	Human Wins
	S, 1	Draw	Draw
Test Plan for setting user input strength for Tools			

User Set Strengths:
R = 10
P = 5
S = 2
Player Chooses:
R
	R	Draw	Draw

	P	Draw	Draw
	S	Human Wins	Human Wins

User Set Strengths:
R = 2
P = 2
S = 2
Player Chooses:
R
	R	Draw	Draw

	P	Computer Wins	Computer Wins
	S	Human Wins	Human Wins

User Set Strengths:
R = 1
P = 1
S = 20
Player Chooses:
S
	R	Human Wins	Human Wins

	P	Human Wins	Human Wins
	S	Draw	Draw
User Set Strengths:	R	Computer Wins	Computer Wins

R = 5
P = 2
S = 2
Player Chooses:
P
	P	Draw	Draw

	S	Computer Wins	Computer Wins

Primary Function Tests

Inputs	**Driver Function(s)**	**Expected Output**	**Actual Output**

computerTool = 2
(randomly generated between 1-3)
pStr = 1
	RPSGame.setComputerTool()	Computer = Paper*	Computer = Paper*
			(strength = 1)

userToolChoice = ‘r’
rStr = 1
	RPSGame.setHumandTool()	Human = Rock*	Human = Rock*
			(strength = 1)

computerTool = 1
(randomly generated between 1-3)
pStr = 5
	RPSGame.setComputerTool()	Computer = Rock*	Computer = Rock*
			(strength = 5)

userToolChoice = ‘s’
rStr = 10
	RPSGame.setHumandTool()	Human = Scissors*	Human = Rock*
			(strength = 10)

Computer = Paper*
(strength = 1)
Human = Rock*
(strength = 1)
	RPSGame.play()	computerWins = 1	computerWins = 1

Computer = Paper*
Human = Scrissors*
	Scissors.fight()	WIN	WIN

Computer = Rock*
Human = Rock*
	Rock.fight()	DRAW	DRAW

Computer = Scissors*
Human = Paper*
	Paper.fight()	LOSE	LOSE

Computer = Paper*
(strength = 10)
Human = Scissors*
(strength = 5)
	Scissors.fight()	DRAW	DRAW

Reflections

The program design and pseudocode that was written out for the Rock, Paper, Scissors program was found to be very similar to the actual code implemented for this project. Since each tool class (Rock, Paper, and Scissors) was derived from the base tool class, writing up the code for each of these derived classes was fairly straightforward since the classes for these only required slight modifications from one another. There were a few things that were handled later on when writing up the actual code such as determining what to return for the fight function for each tool and how setting the strengths for each tool would be implemented.

<u>What to return in the fight function:</u>
When writing up the program design, one thing that needed some group collaboration was figuring out was what to return for the fight function in each of the tool classes. Our group had a discussion on what would make most sense and ultimately we decided to go with returning a fight result enum:

```
Enum FightResult {WIN, DRAW, LOSE}
```

We initially called the enum “Fight” but after some thought we changed it to a more descriptive name, “FightResult,” to make it more clear since we already had a function called Fight.

<u>Manually setting tool strengths:</u>
There were a few different thoughts on how to execute the tool strengths for this piece of the program. One thought was to set the specific tool’s strength after selecting the tool for the human, but this plan didn’t work so well for the computer object since the user doesn’t know what the tool will be due to the computer tool random selection nature. The end result for getting the manual strength setting to work was to have the user to input a strength for each object (Rock, Paper, Scissors) before selecting their tool and have the strengths automatically set when each Tool was created for the human and the computer player.

From the RPSgame class:

```
void RPSGame::setComputerTool(int rStr, int pStr, int sStr);
void RPSGame::setPlayerTool(std::string toolType, int rStr, int pStr, int sStr);
```

The strengths for each tool type are parameters for each of the set tool functions. Depending on the tool selected, the appropriate strength will be assigned.

<u>Random vs. AI:</u>
There was some initial discussion on making the computer object an AI object, but in the end the computer object ended up being simulated by creating a random int between 1-3. If the random int is 1 then the computer tool will be a rock, if the random int is 2 then the computer tool will be paper, and if the random int is 3 the computer tool will be scissors.

Final Thoughts:
In the end, the Rock, Paper, Scissors game program came together quite nicely despite the few adjustments that had to be made. Comparing our expected output in our test plan to the actual output everything was as expected for all of the different combinations for user input and for testing the different driver functions.