|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_2\_\_\_\_\_ AND QUESTION#\_2\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

Write a program that plays the Rock-Paper-Scissors-Lizard-Spock game. Refer to http://en.wikipedia.org/wiki/Rock-paper-scissors-lizard-Spock for more information. Normally, one player is a human and the other is the computer program. However, in this exercise, the program will generate two players who play against each other. The play continues until either of the computer-generated players wins four consecutive times. In this game, two random integers are generated in the range of [1 to 5], one per player. 1 refers to Rock, 2 refers to Paper, 3 refers to Scissors, 4 refers to Lizard, and 5 refers to Spock. For example, if the computer randomly generates integers 2 and 5 in the first iteration, 2 is for the first player and 5 is for the second player. Based on Rule 8 in the following 10 rules, Paper (2) disproves Spock (5), so Player 1 wins. Repeat it to generate one more pair and determine who wins that iteration. Continue the iterations until one player wins four consecutive times.

**2. Description of the Code:**

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
|  |  |  |  |
|  |  |  |  |

**4. Sample Input and Output:**

**The “happy path” test cases are basically all the conditions listed in the play() method. You could also unit test all the outputs from the convert method, given an input.**

**Other test cases aren’t exactly necessary as there is no human interaction with the program other than starting it up. The computer controls all of the inputs… the random generator uses the already tested Math.random function… and because what I understood from the api and requirements is that most of the game logic is in the play method, AND there is no overloaded constructor to be able to inject mocks, it’s not very possible to unit test most of the program.**

**5. Discussion:**