Assessment: Two Games

Submit Assignment

Due Nov 25, 2019 by 9am **Points** 100 **Submitting** a website url

Overview

This purpose of this assessment is to demonstrate your proficiency in basic Java syntax, console input and output, variables, flow control, expressions, methods, and arrays.

Two Games

Implement two games: **Rock, Paper, Scissors** and **Four in a Row**. Games are a great way to combine the individual skills we've practiced in isolation into a complete product.

> Rock, Paper, Scissors is your priority. Focus on that first.

Rock, Paper, Scissors

Before you begin

It's very important to have a plan before you write *any* code. For some people, flowcharts work best. For others, a list of logical steps that include jumps back to other steps is effective. It's also a good idea to sketch out the UI, get an idea of how one "screen" transitions to another.

Once you have a plan, implement. You'll probably notice that your plan changes. That's perfectly normal. The important thing is to have a plan to get started. Then, adjust as necessary.

Rules

1. Each player chooses Rock, Paper, or Scissors.
2. If both players choose the same thing, the round is a tie.
3. Otherwise:
    a. Paper wraps Rock to win
    b. Scissors cut Paper to win
    c. Rock breaks Scissors to win

Requirements

This program will be a Java Console Application called RockPaperScissors.

- The program first asks the user how many rounds he/she wants to play.
  - Maximum number of rounds = 10, minimum number of rounds = 1. If the user asks for something outside this range, the program prints an error message and quits.
  - If the number of rounds is in range, the program plays that number of rounds. Each round is played according to the requirements below.
- For each round of Rock, Paper, Scissors, the program does the following:
  - The computer asks the user for his/her choice (Rock, Paper, or Scissors).
    - Hint: 1 = Rock, 2 = Paper, 3 = Scissors
  - After the computer asks for the user's input, the computer randomly chooses Rock, Paper, or Scissors and displays the result of the round (tie, user win, or computer win).
    - Hint: use the Random class.
- The program must keep track of how many rounds are ties, user wins, or computer wins.
  - Hint: Create three variables to keep track of these items and update them correctly after each round.
- The program must print out the number of ties, user wins, and computer wins and declare the overall winner based on who won more rounds.
- After all rounds have been played and the winner declared, the program must ask the user if he/she wants to play again.
  - If the user says No, the program prints out a message saying, "Thanks for playing!" and then exits.
  - If the user says Yes, the program starts over, asking the user how many rounds he/she would like to play.

Four in a Row

Known commercially as *Connect Four*.

Two players take turns dropping game pieces into a seven by six vertical game board. Pieces fall to the first empty space in a column. The first player to place four of their pieces in a row horizontally, vertically, or diagonally wins.

Example Game Play and UI

1. An empty board

```
1 2 3 4 5 6 7
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
- - - - - - -
```

2. Player one chooses column 3.

```
1 2 3 4 5 6 7
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ X _ _ _ _
```

3. Player two chooses column 4.

```
1 2 3 4 5 6 7
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ X 0 _ _ _
```

4. Player one chooses column 4.

```
1 2 3 4 5 6 7
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ X _ _ _
_ _ X 0 _ _ _
```

5. Player two chooses column 5.

```
1 2 3 4 5 6 7
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ X _ _ _
_ _ X 0 0 _ _
```

6. Player one chooses column 5.

```
1 2 3 4 5 6 7
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ X X _ _
_ _ X 0 0 _ _
```

7. Player two chooses column 6.

```
1 2 3 4 5 6 7
_ _ _ _ _ _ _
```

```
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ X X _ _
_ _ X 0 0 0 _
```

8. Player one chooses column 5.

```
1 2 3 4 5 6 7
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ X _ _
_ _ _ X X _ _
_ _ X 0 0 0 _
```

9. Player two chooses column 7 and wins with four in a row!.

```
1 2 3 4 5 6 7
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ X _ _
_ _ _ X X _ _
_ _ X 0 0 0 0
```

Requirements

The game is divided into three distinct phases: set up, game play, and summary/play again. The set up phase should ask users for their names. Those names should be used in prompts during the game play and summary phases.

```
Player #1, enter your name: Nemo
Hello, Nemo.

Player #2, enter your name: Dori
Hello, Dori.

(Randomizing...)
It's Dori's turn.

1 2 3 4 5 6 7
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _

Dori, choose a column: 3

1 2 3 4 5 6 7
_ _ _ _ _ _ _
_ _ _ _ _ _ _
```

```
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ X _ _ _ _

Nemo, choose a column: five
That's not a valid column.

Nemo, choose a column:_
```

Game play must track who's turn it is. Since it's a two player game, a boolean is enough. Alternate making moves until someone wins or all moves have been exhausted. (A draw *is* possible.)

A user need only provide one number, the column, to make a move.

Validate moves. The only valid columns are 1 - 7. If a column has six pieces, it is full. It is invalid to place an additional piece.

On win or draw, display the result and ask if the players would like to play again.

```
1 2 3 4 5 6 7
_ _ _ _ _ _ _
_ _ _ _ _ _ _
_ _ _ _ _ X _
_ _ _ 0 X X _
_ _ 0 X 0 X 0
_ 0 X X 0 X 0

Dori Wins!

Play again? [y/n]: _
```

Technical considerations

- Game board should be a two dimensional array.
- Create a method that can be called repeatedly to display the board.
- Invalid input should not crash your program and should not allow the user to proceed.

Stretch Goals

- Add an option for a computer player who plays randomly.
- Add an option for a computer player who plays intelligently.

Basic Programming Concepts 2.4

Criteria	Ratings			Pts
Flowchart The flowchart for Rock, Paper, Scissors reasonably reflects the flow required for the activity.	**15.0 pts** **Meets Expectations**	**8.0 pts** **Needs Improvment**	**0.0 pts** **No Submission**	15.0 pts
Application Name Application is named correctly.	**5.0 pts** **Meets Expectations**	**3.0 pts** **Needs Improvment**	**0.0 pts** **No Submission**	5.0 pts
Variables Apprentice names and uses variables correctly.	**10.0 pts** **Meets Expectations**	**6.0 pts** **Needs Improvment**	**0.0 pts** **No Submission**	10.0 pts
Loops Apprentice uses loops correctly.	**10.0 pts** **Meets Expectations**	**6.0 pts** **Needs Improvment**	**0.0 pts** **No Submission**	10.0 pts
Conditionals Apprentice uses conditionals correctly.	**10.0 pts** **Meets Expectations**	**6.0 pts** **Needs Improvment**	**0.0 pts** **No Submission**	10.0 pts
Methods Apprentice uses main methods correctly.	**10.0 pts** **Meets Expectations**	**6.0 pts** **Needs Improvment**	**0.0 pts** **No Submission**	10.0 pts
Arrays Apprentice uses arrays correctly.	**10.0 pts** **Meets Expectations**	**6.0 pts** **Needs Improvment**	**0.0 pts** **No Submission**	10.0 pts
Operators Apprentice uses arithmetic operators correctly.	**10.0 pts** **Meets Expectations**	**6.0 pts** **Needs Improvment**	**0.0 pts** **No Submission**	10.0 pts

Criteria	Ratings			Pts
Debugging Apprentice can explain the mechanics of debugging.	**10.0 pts** **Meets Expectations**	**6.0 pts** **Needs Improvment**	**0.0 pts** **No Submission**	10.0 pts
Code Style Code is written with appropriate indents, naming conventions, and comments so that other developers can read the code easily	**10.0 pts** **Meets Expectations**	**6.0 pts** **Needs Improvment**	**0.0 pts** **No Submission**	10.0 pts
				Total Points: 100.0