Lab 06: Repetition

Due date: By the end of your lab session of the week of Monday, September 26th

Goals

- Learn to control the flow of a program using repetition with **do-while / while** loops
- Learn to write decision making programs using **switch** statements

Description

In this lab, you will develop the game "Rock-Paper-Scissors". Your program will allow the user to play this game against the computer. When the game starts, the program prompts the user to provide input for his/her move. This should be either rock, paper or scissors. The computer's choice of the move is done randomly. The program then compares the user's move to the computer's move and decides on the winner. The rules of the game are as follows: the Paper beats the Rock, the Rock beats the Scissors and the Scissors beat the Paper. If the computer's move is the same as the user's move, the result of the game is a tie. The game repeats infinitely and exits if the user chooses to exit the game instead of providing a move.

Hints

- Implement this game in a class called "Game". The file "Game.java" should contain all your code and should reside in a directory called "Lab06" inside your "cs180" directory.
- Use numbers to represent the different moves both in the user choice menu and in the program. For example, to represent the rock move and user choice in the program, you may define in the Game class:

  ```
  private final static int ROCK = 1
  ```

  In order to get the user's input, show a menu that mentions that if the user inputs the number 1, this will be considered a rock.
- The methods shown in the Table 1 should be implemented in the Game class.
- Implement a main method that starts the game by creating an Object of the class Game and calling the method `runGame()` in that Object.
- Use **while** or **do-while** loops to implement the repetition of the game.
- Use **switch** or **if** statements to check the different combinations and find the winner in the method `checkWinner`.
- Use the **Random** class to implement the computer move. If you represent moves as 0, 1 and 2, you will basically need to find random number between 0 and 2.
- Use the Java API to check which methods in the Random class can be helpful.

Method Prototype	Description
`public void runGame()`	Contains the Game driver.
`private int checkWinner(int move1, int move2)`	• Finds the winner in the game based on the input moves. • Returns 0 in case of a tie, 1 if the first move beats the second, and 2 if the second move beats the first one.
`private int simulateComputerMove()`	Simulates the random choice of the computer and returns the chosen move.

Table 1: Object methods defined in the Game class

Demo of a working solution:

```
Welcome
Please enter an option:
1. Rock
2. Paper
3. Scissors
4. Exit
1
You played rock!
The computer played scissors!
You win!
Please enter an option:
1. Rock
2. Paper
3. Scissors
4. Exit
2
You played paper!
The computer played rock!
You win!
Please enter an option:
1. Rock
2. Paper
3. Scissors
4. Exit
3
You played scissors!
The computer played scissors!
Draw!
Please enter an option:
1. Rock
2. Paper
3. Scissors
4. Exit
4
Thanks for playing!
```

Turning in Your Work

You **must** turn in your "lab06" directory before leaving the lab session. Change your current directory to "cs180" and execute the following command:

```
turnin -c cs180=COMMON -p lab06 lab06
```

Rubric

This lab is worth 50 points. The points breakdown is as follows:

- 20 points for a correct `runGame` method
- 10 points for a correct `simulateComputerMove` method
- 10 points for a correct `checkWinner` method
- 10 points for a correct main method