Software Development
Continuous Assessment: Project [20%]
Rock-paper-scissors Game

"**Rock-paper-scissors**" is a hand game usually played by two people, where players simultaneously form one of three shapes with an outstretched hand"[1].

The rules of the games, depicted also in Figure 1, are the following:

- "rock" beats "scissors"
- "scissors" beat "paper"
- "paper" beats "rock"
- if both players form the same shape, then it is a draw

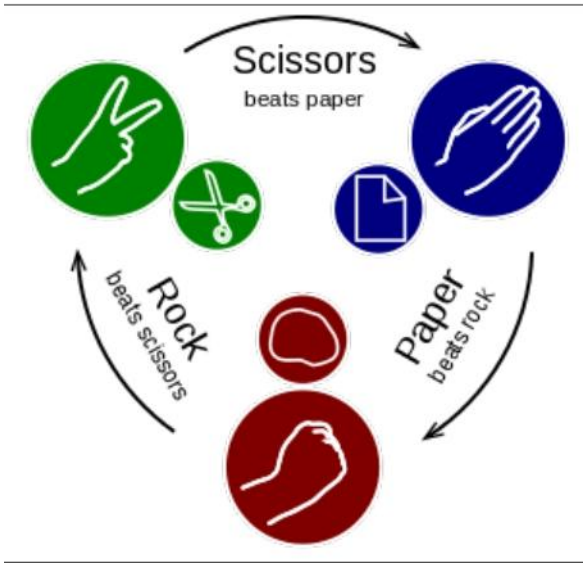

Figure 1: Rock-paper-scissors game rules[1].

[1] http://en.wikipedia.org/wiki/Rock-paper-scissors

Develop the "rock-paper-scissors" game that will allow a user to play repeatedly the game with the computer. The "rock-paper-scissors" game starts by asking the user the numbers of games he/she would like to play.

At the start of a game the user is given 3 lives. In each round of the game the user must choose one of the three shapes, namely "rock", "paper" or "scissors". Similarly, in each round of the game the computer will randomly pick one shape from the list of three shapes stored in an array of shapes. After each round, the game displays the computer's choice. The user loses one life each time he/she loses a round. After each round the game displays the user's number of lives, whether the user or the computer won the round, lost the round or it was a draw. In addition, after each round the game shows the number of rounds won and the number of rounds lost by both the user and the computer. A game finishes when the user lost all the lives. The overall winner of the game is the player who won more rounds. At the end of the game, the game displays who the winner is, and a history of all the shapes chosen by both the user and the computer.

At the end of all games display a history of games played. The history shows for each game the number of rounds won and lost by both the user and the computer.

For example, a possible output of the program could look like:

- After first round

```
User Lives: 3
User chose: rock
Computer chose: scissors
User won!

User rounds won: 1
User rounds lost: 0

Computer rounds won: 0
Computer rounds lost: 1
```

- After second round

```
User Lives: 2
```

```
User chose: rock
Computer chose: paper
Computer won!

User rounds won: 1
User rounds lost: 1

Computer rounds won: 1
Computer rounds lost: 1
```

- After third round

```
User Lives: 2
User chose: paper
Computer chose: paper
It's a draw!

User rounds won: 1
User rounds lost: 1

Computer rounds won: 1
Computer rounds lost: 1
```

- After another round

```
User Lives: 1
User chose: paper
Computer chose: scissors
Computer won!

User rounds won: 1
User rounds lost: 2

Computer rounds won: 2
Computer rounds lost: 1
```

- At the end of a game

```
Winner: computer
Shapes played:
User shapes: rock, rock, paper, paper, ...
Computer shapes: scissors, paper, paper, scissors, ...
```

- At the end of *all* games

```
Game 1
User: rounds won: 2
User: rounds lost: 3

Computer: rounds won: 3
Computer: rounds lost: 2

Game 2
User: rounds won: 6
User: rounds lost: 3

Computer: rounds won: 3
Computer: rounds lost: 6

...
```

Deliverables

The Project materials **must be submitted via moodle**, they cannot be submitted by email. You will submit a .zip file on moodle containing:

- Complete Java source code (.java files) of the "rock-paper-scissors" game
- A report which contains
  1. The application's
     - input
     - process

- output

2. The Class Diagram of the application
3. Screenshots of the application's screens/output
   (a) Once you finished writing the code, compile and run your application.
   (b) Play the game with the computer and take some screenshots while playing the game (for example, the output of your program after the first round of the first game, the output of your program after the third round, the output of your program at the end of a game, the output of your program at the end of all games).
   (c) You can take a screenshot by using the key "Print Screen" of the keyboard. On the keyboard, the key may have one of the following labels *Print Scrn, Prnt Scrn, Prt Scn, Prt Scr, Prt Sc or Pr Sc.*

The .zip file you submit should be named using the following pattern

- YourFirstName_YourLastName_ProgrammeCodeYear_SDEV.zip
- Please check moodle for the **submission deadline**

Guidelines

- Use instantiable classes
- Please include in all the files (i.e. all .java files and the report) your name, your NCI email and your student ID.
- All the .java files should contain comments, both documentation comments and concise comments for the important parts of the programs. Each .java file should contain the following documentation comment:

```
/**
ClassName.java -- short description of the class
@author <your first name and last name>
studentID <your student ID>
@date file creation date
*/
```

Hint

1. The provided flowchart (included in the "Rock_paper_scissors_flowchart.pdf" file – available on moodle) aims to help you visualize the main tasks required

Development Approach

1. Read carefully the project description and identify
   - What is the goal of the project?
   - What does the application do?
2. Design
   - In addition to using the provided flowchart, you could write down the pseudo-code, namely you can write in plain english the main tasks/steps required to implement the game. Note that this step is just to clarify what you have to implement, you do not need to include the pseudo-code in the project report.
   - Identify and write in the report
     - the input
     - the process
     - the output
   - Draw the Class Diagram. The Class Diagram **must be included** in the report.
3. Write code (you should start writing code only after you completed the desing section)
   - Write Code
     - Write code step by step, one piece of functionality at a time. Once you have one functionality done, save it, compile it, and debug it. You should save versions of your program such as *JavaFileName_v1.java*, *JavaFileName_v2.java*, *etc.* to ensure that, if needed, you can retrieve code you have written before. Save all the versions of your application in another folder, for example named *backup*.

- Test the code (i.e. play the game, and provide all the possible inputs) to make sure that the application works as described on pages 1-2 of this document.
  - If needed, debug the code.
    * Change **only one thing** at a time in your program. Do not forget, **continue saving versions** of your application.

Assessment Criteria

The maximum mark you can get for the project is **100**. The project will account for **20%** of the total Continuous Assessment mark.

The criterion used for marking of the project is the following:

- **15** Compiling and Running Code
  - **7.5** All code compiles without errors
  - **7.5** All code runs without runtime errors/exceptions (i.e. the application does not crash while running)
- **20** Documentation/Project report
  - Evidence of identifying the input, processing and output prior writting the source code
  - Class diagram with short explanation
  - Demonstration that you tested your application and is working according to the project requirements (i.e. add several screen shots in the report)
- **20** Code Design and Code Structure
  - Use of classes
  - Suitable method signatures, class naming and variable naming standard conventions
  - Use of comments: both documentation comments and code comments
  - Use code indentation

- **45** Functionality Implemented
  - List of shapes – array construction
  - Random selection of shapes
  - Validation of user input
  - Play the game while the user still has lives
  - Computation to decide who is the winner per round
  - Computation to decide who is the winner per game
  - Display the history of shapes chosen by both players per game
  - Play as many games as the user wants
  - Display statistics for all the games after the user finished playing all the games (i.e numbers of rounds won and lost per game by each player)

References

- [**1**] http://en.wikipedia.org/wiki/Rock-paper-scissors
- [**2**] http://www.nytimes.com/interactive/science/rock-paper-scissors.html