# Practical-04: Inheritance and Polymorphism

**Due** 8 Sep by 22:00 **Points** 100 **Available** after 15 Aug at 0:00

This assignment will focus on using inheritance and polymorphism to build a simple Rock, Paper, Scissors game.

### Submission

```
https://version-control.adelaide.edu.au/svn/<YOUR-ID>/2019/s2/fcs/week-05/practical-04

save this project as
+ Main.java
+ Player.java
+ Human.java
+ Computer.java
+ Referee.java,
 and any other classes you create in this practical.
```

## The problem

This is a **one** round game. You are asked to implement a simple Rock, Paper, Scissors game (RPS game) in Java.  The game is simple. Two players compete and, independently, choose one of Rock, Paper or Scissors. They then simultaneously declare their choices. The winner of the game is determined by comparing the choices of the players. Rock beats Scissors, Scissors beats Paper, Paper beats Rock.

This programming assignment is a simulation of the RPS game, where we have one human player (you) and one computer player. As we cannot simulate truly simultaneous choice, this program will take your choice and, without giving information on that choice to the computer player, ask the computer player for their choice. The two choices will then be compared to determine who has won.

Your task is to produce a set of classes that will allow a human player, typing instructions from the keyboard, to interact with a computer player. An independent referee class will decide whether the human or computer has lost. When you design this code you must have, at least, a human player class, a computer player class and a referee class. You probably want to use a game controller class of some sort as well but the details are left up to you.

For ease of marking, you should use the filenames Main.java (for your main code), Player.java Human.java, Computer.java and Referee.java to describe the relevant components of your system.

```
Main.java: Main driver of your software. This class should instantiate a referee object;
Referee.java: class that controls the game. In this class you should instantiate the players and control the flow of the RPS game.

Player, Human and Computer: consists of the classes that define the behaviour of each player. Human should require user input, as Computer should draw a random move (Rock, Paper or Scissor).
```

## 1. Taking input

You must configure your code to take input from the keyboard as a string. To do that, you can see **read input from console (http://javadevnotes.com/java-tutorial-read-input-from-console)** .

You will need to specify what you consider to be valid input **by yourself**. Does a user need to type all of Rock, Paper or Scissors? Do you accept scissor, Scissor, Scissors and scissors as all being the same thing? **Ensure that you are detailed in your design as to what you consider to be acceptable.** (e.g.You should spend sometime to design your output so that user know how

to play this game. Please make sure your program will not crash when user input some invalid keyword.)

## Part 0: Design

Create a design for how you wish to solve the problem, based on the stages below.

As part of your design, you should:

- Design your inheritance tree.
- Sketch out how you are going to build and test the code.

You can write your idea in the comments in Main.java

## Stage 1: A human player

Your code must start up, create a human player object and prompt for the human player's move. This answer should be passed to the referee object who will determine if the move is valid and report an error if it is not.

## Stage 2: A human and a computer player

Now, your code must also create a computer player and the computer player is instructed to make a selection.

## Stage 3: Use inheritance to refactor your code

The class declarations of Human and Computer will be very similar, and duplicate information unnecessarily. Human and Computer should extend (inherit from) a class called Player that contains a private member. In this class, it must have one method which should be overridden by Human and Computer classes:

- performMove() // This method will return a string variable to show what this player choose.

You can design the rest of methods by yourself.

Build and test your basic game before continuing.

## Stage 4: Use polymorphism in your Referee class

Both your and the computer's responses are then to be fed to the referee who determines who has won, as well as reporting errors.

A referee is an official who watches a game or match closely to ensure that the rules are adhered to and (in some sports) to arbitrate on matters arising from the play. In this problem, the Referee class is the class that organise the entire game, such as user interface for adding name, determining whether the player is computer or human etc.

Feel free to add to your Referee class user interface such as menu, etc.

| **Basic Marking Scheme** |
| --- |

| Criteria | Ratings | Pts |
| --- | --- | --- |
| Compilation<br>In order to achieve full marks - your code must compile and run; | | 20.0 pts |
| Basic Functionality<br>Your code (1) perform all the functions correctly, (2) use latest concepts learned in class, (3) has a clear, creative and sophisticated way of solving the problem. | | 40.0 pts |
| Functionality Extension<br>Your code (1) perform all the functions correctly, (2) use latest concepts learned in class, (3) has a clear, creative and sophisticated way of solving the problem, and (4) you propose novel ways to solve the problems - or extra functionalities. | | 10.0 pts |
| Code Formatting<br>Your code (1) has the right proportion of comments and place line and block comments correctly, (2) follow correct indentation every new indentation level, (3) has good variable naming, (4) has clear organization between tabs and is easy to read. | | 30.0 pts |
| | | Total points: 100.0 |