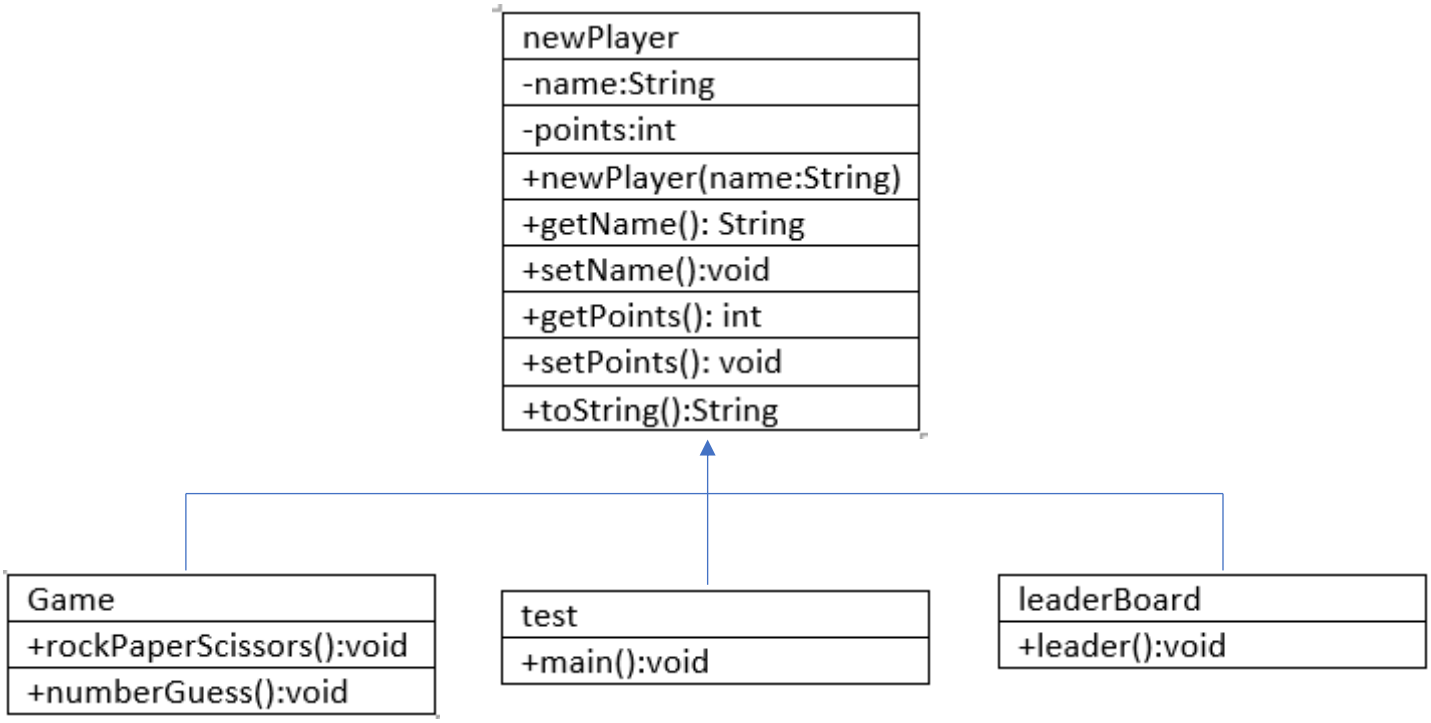
My project has four classes and one text file:

1. Game.java-holds the code for two games rock paper scissors and guess a number.
2. leaderBoard.java- holds the code to display the text file of the leader board.
3. NewPlayer.java- creates a new instance of a player object.
4. Test.java – holds the main method to run the code
5. Leaderboard.txt is the text file.



All initial instances of a player starting are created as a newPlayer object through the constructors of the class. The constructor is called when a class object is created. The constructors name matches the class name and can have an argument or no argument. This creates a name and a points association with that player until they exit the games. The getters and setters in my code are positioned in the newPlayer class to allow access to the private class variable points and name. This use of the private key word encapsulates the variables into a class and thus protects them. They are then only accessible by the getter and setter methods. This allows the end user the benefit of abstraction, they do not need to know how the variable is implemented they can just get the variable and use it. The newPlayer class has a toString method which overrides the built-in java toString method to print the name points and date of each player to the leader board, an override allows you to change the behaviour of a method that is already defined in a parent or superclass.

Each class has its own method or methods which allow code to be reused and its only actioned when the method is called. Some methods take parameters, and some do not. If nothing is returned from a method it is set as a void, otherwise the method must be of the same type as the return type.

The leader board scores are saved to the leader board text file in the test class which houses the main method and is where the other classes are run from. The leader board class then reads this text file and saves the file to a list parameterised by <string> and orders it by points descending it is this list that prints to the screen for the player to see once the exit from the main menu. This utilises

javas input and output streams. I have also used exceptions handling for these to search for the file and if it does not exist it throws an exception.

The game class extends the newPlayer class through inheritance as each game will have points and a player depicted by name in this case. Inheritance allows one class to inherit the attributes and methods of another class. newPlayer is the parent class and game is the child class. It has two methods one for each game. The first is rock paper scissors which uses a string array to hold rock, paper, scissors, this is so I can use the random import to help randomly pick an array number up to and including the length of the array which is how the computer selects its choice. The points are stored using the setPoints setter until the player exits at which time, they are saved to the leader board text file along with the date and the time they played. The games use do while loop to allow the player to have three interactions with each game before it finishes.

For rock paper scissors the points vary depending on win draw or lose (10,5,0) respectively.

For number guess you receive 15 points for a correct guess and 0 points for an incorrect guess. If the number is guessed correctly the target number changes if the guess is incorrect prompts of higher and lower guide you toward the correct target number, you get three tries.

The test class is the main method that runs the program to run continuously while the player wishes to, I have use two nested while loops that runs the menus until a player selects to exit.

If they are in the game menu selecting -1 will bring them to the main menu where a new player can start a game by pressing one followed by enter or else, they can select 2 in the main menu to quit completely and have the leader board show on screen.