Java Games Systems Project Report

The Project

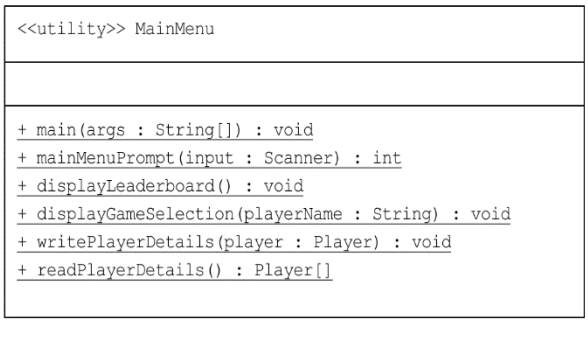
This project is a java program that implements a simple console based game system. Users interact with the system through the console by typing either a character or integer (or in the case of entering a name a string) and pressing enter. On start up the user is presented with the option to create a new player or to exit the system. If the user chooses to create a new player, they are prompted to enter their player name and are asked if they wish to play as a limited player where they are limited to only 15 plays. The user is then provided with a list of playable games. A game can be chosen to play be entering the integer corresponding to the game. Once a game is chosen the user will be prompted for game input and, once input is given, the game outcome will be displayed. The user will continuously be prompted for game input until they decide to quit the game. When a user decides to quit a game, they will be displayed with the game selection menu where they can choose another game to play or to quit to the main menu. When the user finally returns to the main menu, they will again be prompted to either create a new player or to exit the system. When the user finally decides to exit the system, they will be presented with a leader board showing a list of all player names and their total points in descending order of points.

The program makes use of various classes and subclasses. The main access modifier used is public except for the Player and LimitedPlayer instance variables which are protected with private. The private instance variables can be accessed through getter and setter methods. Arrays were used to hold the various game outcomes from which the program will choose, and to hold the past players read in from an external file.

Class Diagrams

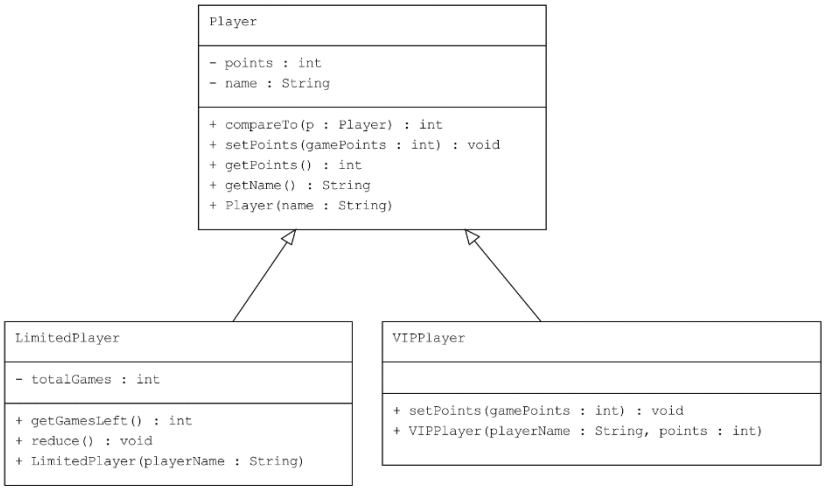
The MainMenu class contains the main method with which the program runs. This contains all the method calls needed to display necessary information, create player instances and to create and run game instances. The mainMenuPrompt method simply prompts the user to either create a new player or exit the system. It reads in and returns an integer. The displayLeaderboard method prints out the past players and their points in descending order of points. It calls readPlayerDetails to create an array of Players. The displayGameSelection simply prints out the selection of games available. The writePlayerDetails method takes in a player as input, opens the 'players.csv' file and appends the given player's name and points. The readPlayerDetails method opens the 'players.csv' file and counts the number of lines. It then uses the count to create a Player array of appropriate size. It then reads each line entry from the file, creates a Player instance with the given details and adds the Player instance to the array. It finally returns the array of Players.

Figure 1. MainMenu Class



The Player class has two private instance variables: points and name. The Player class implements the Comparable interface and overrides the compareTo method to compare Player objects based on their total points. The setPoints method takes in an integer and adds this to the player's total points. If the player's total points are 0 and the game points are negative, nothing happens. The getPoints method is the getter for the Player instance variable points. The getName method is the getter method for the Player instance variable name. The name variable has no setter as it is only set in the constructor.

Figure 2. Player Class and Subclasses

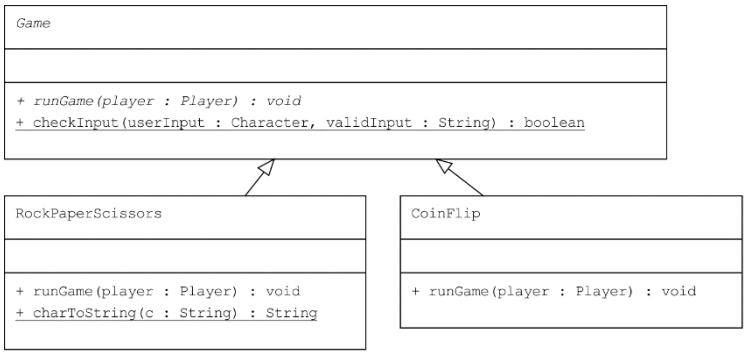


LimitedPlayer is a subclass of player. Its constructor takes in a string for the player name and calls the superclass constructor using this string with "(Ltd.)" appended to it. It has an instance variable called totalGames which is an integer that represents the total amount of plays the player can have. The getGamesLeft method is simply a getter for totalGames. The reduce method simply decrements the LimitedPlayer's totalGames.

VIPPlayer is another subclass of Player. Within the game system if a player earns more than 50 points they become a VIPPlayer. The constructor takes in the player name and the number of points they have earned so far. VIPPlayer overrides the setPoints method and call the super method but with double the game points.

The Game class is an abstract class. It contains the abstract method runGame which will be unique to each game. It also contains the method checkInput which takes in a character and a regex string and checks the character against the string.

Figure 3. Game Class and Subclasses



RockPaperScissors is a subclass of Game. The runGame method implements a simple version of the Rock Paper Scissors game. The charToString method takes in either "R", "P", or "S" as a string and outputs either "Rock", "Paper", or "Scissors" accordingly. Players either win 5 points or lose 5 points.

CoinFlip is another subclass of Game. The runGame implements a simple coin flip game in which the player guesses whether the outcome will be heads or tails. Again, players can either win 5 points or lose 5 points.