Cryptogram Project – Status Report

CS207 – Advanced Programming

The cryptogram application is a piece of software implemented in Java that basically consists in solving a cryptogram. The application is a command line interface. When starting the program, the application prompts the user to create a username if the user is new, or to select a username and load it if the user is already registered. The main class that manages the flow of the game is represented by the **Application class** where the game is created and which allows the user to interact with the application by giving different commands.

As any user-friendly application, our program has the “help” functionality implemented. This is achieved using the **HelpMessages** class that allows the user to see the functionality of the application and offers a list with the commands that can be used. In order to consult the help menu, the user has to type in “help”. This facility can be accessed at any time during the game. When the “help” command is accessed from the main menu, the user has the option to load a game, to play a game, to view the scoreboard or to exit. When accessed during a game, the help option allows the player to generate a new cryptogram, to exit the game or to simply resume, continuing.

The cryptogram can be played in two modes, by mapping letters to letters or by mapping letters to numbers, and users get to make this choice at the beginning of each cryptogram. Also, the users get to choose whether they want to play a cryptogram about a historical quote or a cryptogram about a pop culture quote, and they get to choose the length of the cryptogram. These options are managed in the **Cryptogram class** which is in charge of the encryption and decryption of a cryptogram. Also, here happens the generation of the cryptogram game based on the selections of the user. The cryptogram dictionary is generated on the fly and it is randomised, mapping being different for each cryptogram. User can select a letter to map a value to it, and all its occurrences will be completed automatically in the puzzle.

Users can challenge each other by trying to score as well as possible and to enter in the top 10 scores which are shown in the scoreboard. The player has the option to quit the cryptogram at any point. The application keeps track of the state of completion of each cryptogram that each user has played, making it possible for example to stop a cryptogram game half way through, and to resume it later from the point where they have left it.

The scoreboard is formed by taking into consideration different statistics such as: number of cryptograms correctly solved, number of cryptograms played, time spent to solve a cryptogram, number of hints used. The **Player class** is in charge of maintaining and managing the details about the players.

Users have to choose one of the criteria (“by percentage of guess accuracy”, “by number of cryptogram played”, “by number of cryptogram completed”, “by lowest average time”) when they request to view the scoreboard, so that the best 10 results are displayed according to one of these categories. Users are allowed to go to another cryptogram if they do not like the current one. Also, the cryptogram game offers unlimited number of hints that can help the user solve the cryptogram. However, when using a hint, the user gets penalized, such that his/her accuracy will be lowered, and this will affect his/her to be one of the top players that have their score displayed on the scoreboard.

Once created, a player cannot be deleted. However, he/she will eventually get eliminated from the scoreboard, once the game reaches more than 10 users and other players surpass his/her score. For a better user experience, the game displays error messages such as an error message for trying to enter a letter that has already been used. By doing so, the application aims to guide the user towards the right path, and to offer a better level of interactivity. Besides the Java classes, the application makes use of three txt files, one for storing different statistics about the performance of the users, one with information about the games, and another one for storing the different texts for cryptograms.

Another two important classes used are Players class and SavedGames class. **Players class** gets a list with all the players saved in the players.txt file, and every time a user enters the game, it checks in the list whether the name of the user exists in the list or not. If it exists, the player will get loaded and it will have its previous score and games. If not, the player will be added to the list. The **SavedGames** class saves the progress of a game when the user wants to quit an unfinished game, allowing the user to resume at a future time from where he left.

Finally, in order to make the program more stable, the application makes use of the Junit framework. This makes testing easier and helps reduce the debugging time and regression.