**Final Report – EECE 350**

Mohamad Abou Salem: mba26

**Description:**

A final report of the project, describing the functionality of the application, brief description of the code and design decisions made, snapshots, the contribution of each member. We will also include a brief PowerPoint presentation about the code and functionalities.

Snapshots are included in the small manual provided.

**A- Functionality of the Application:**

**1- Communication Protocol Design:**

For the client side, scan points to the input stream. A new Socket instance is created containing the server IP and port number as arguments. The nextline() method is used to get the input from the user, while the writeUTF() method is used to write it on the output stream dos. If the entered line is “Exit”, the remote host connection will close and a “Connection closed” message will appear, and the

input stream will also appear and the connection will close. Any errors will be clear with the printStackTrace() method.

For the server side, a server socket ss will be created, and when the connection is established, the client will be accepted, and a message with the client’s information will be displayed (IP + port number). A thread will be created to start the process,

it is a multithread. The multithread class includes the string received that reads the

input stream. A message in the output stream asking to either type LogIn or Exit are to be displayed. If the received input stream is “Exit’, then the connection will be closed. If the received message is “LogIn”, the LogInScreen will be launched.

**2- LogIn, MainMenu and Registration**:

These 3 processes, implemented with a GUI are mainly for creating a new user account before playing the game (Registration), to log in (LogIn) and access the main panel (MainMenu, which contains a convert button to convert from 10 silver coins to 1 gold coin, statistics button, and at first LogIn of each day, 50 gold coins are allocated). The Log in page’s tasks are to verify the fields entered (empty, incorrect…) and updates the values (date, coins) in the database.

If a new user wants to login, a registration screen will pop up to let him enter his account information. The main menu screen displays the statistics and information of the user. **“MyConnection”** connects our application to the database to insert

data input by the user and the local daytime, and retrieves information from the database after a login (firstname lastname, gold, silver).

**3- Driver**

This was implemented to show database in a cleaner way.

**5- Game mode menu, to choose single or multi:**

When launching the application, it contains a random generator of a number with 4 distinct characters, and sets it as the goal number to be guessed. There is a game mode menu which allows us to choose the single player game. It contains the game logic that will be briefly explained later. We can also choose the multi-player

mode.

**6- Multiplayer (logic no connection):**

In this mode, 2 players can compete. The logic of the game was implemented correctly but the implementation of the connection establishment between 2

players didn’t work as smoothly as desirable. The 2 players (we also implemented a Player class) logic was correctly implemented.

**7- JDBC:**

This mode creates a database on the machine of the user.

**8- Character limit (single digit):**

This functionality limits the input to only 1 digit per text field.

**9- Player class:**

This class contains the needed information for the player to compete. Int. TotalScore stores the overall score of the player. It gets his name and updates the total score using setTotalScore.

**B- Brief Description of the Game Logic:**

This part will include a small description of the game logic code implemented:

The game will take as input a string “play” provided by the player in a single game mode, which is set to the playerID, starting with 0 guesses, 0 goldScore and 0 silverScore initially. This will also get the secret number from the random number generator discussed earlier. We will first check if the number provided by the user is negative, greater than 9 or if it is containing 2 similar digits, since we do not want that. Then we did a loop to check the correctness of the guess. At the end the

Int. guesses will be incremented, we will then decrement initGold by 5 to allow for a new guess. If there is a perfect guess, i.e. all numbers are correctly in place, the goldScore will be increment by 50, equating initGold, and initSilver will also be incremented by the silverScore. A pop up message will appear saying it is correct.

Basically, correctIn is when the correct digit is guessed with its correct position, while correctOut is when it is guessed in a different position.

When we have less than 5 coins remaining, we just update the silver and gold scores and exit.

For the multi-player part, we can choose it from the game mode panel, with a pop- up message with instructions of the game, with a deduction of 500 gold coins. If the correct guess was made, the gold coins of the winner will be updated to receive

1500 additional ones, and we use the setTotalScore function of the Player Class to update the players’ scores.

**C- Manual:**

Please refer to the manual provided to see the snapshots of the various functionalities.

**D- Note:**

We did not use the files in Java to store the scores and statistics of the players, however we did a better solution consisting of using databases.