CS231

Project 1

Yazan Bawaqna

Report: Blackjack Game

Abstract:

This project endeavors to develop a comprehensive simulation of the classic card game, Blackjack, using fundamental data structures, including arrays and ArrayLists. Blackjack, also known as 21, is a renowned card game with origins in casinos and living rooms alike. Our primary objective was to create a text-based rendition of the game, allowing for extensive simulations and the calculation of win percentages. This report presents an overview of our project, with a particular focus on the implementation of arrays and ArrayLists within the context of this digital card game.

Within computer science, an array serves as a fundamental data structure, storing a fixed-size sequence of elements of the same data type. On the other hand, ArrayList represents a dynamic array-like data structure in Java, allowing us to dynamically store and manipulate collections of objects of diverse types. In our implementation, arrays were employed selectively, primarily for managing card suits, while ArrayLists played a pivotal role in dynamically storing and manipulating card objects within the deck and hands.

Results

The main function (Blackjack.java) :

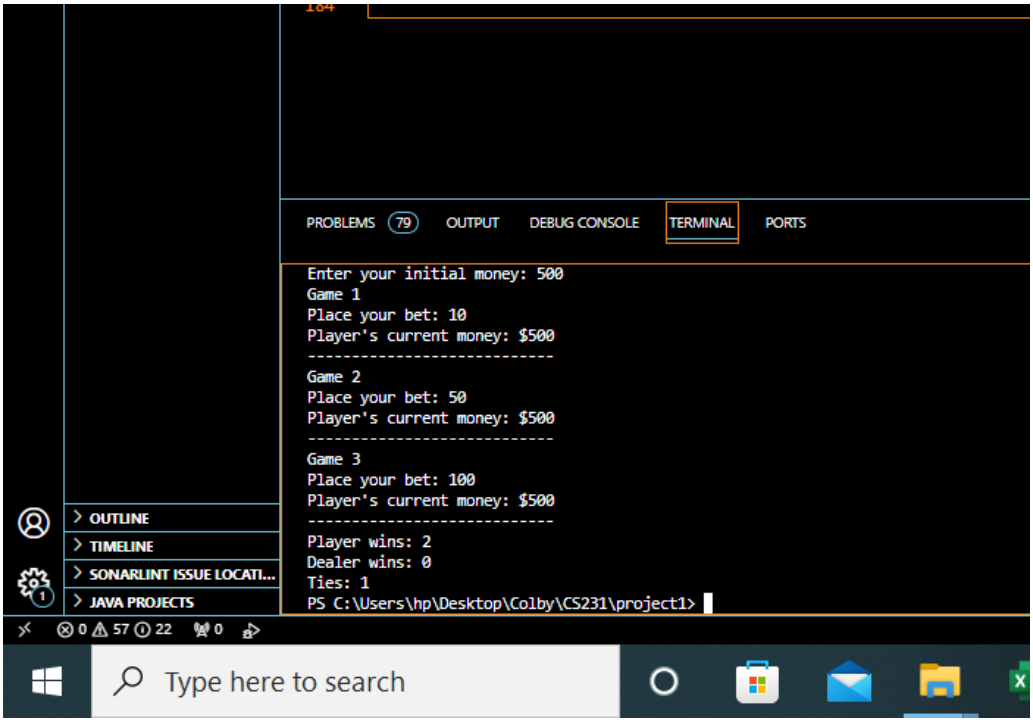

Image-1: A game played where the user initial money was 500, and he betted 1,50, 100 respectively for each round

The main function in the blackjack class asks the user to insert his total money, and then his bets for each round before playing the game and printing the results.

Another way I used to display the results is to print it into a text file from the command prompt by typing “java Blackjack > games1.txt”

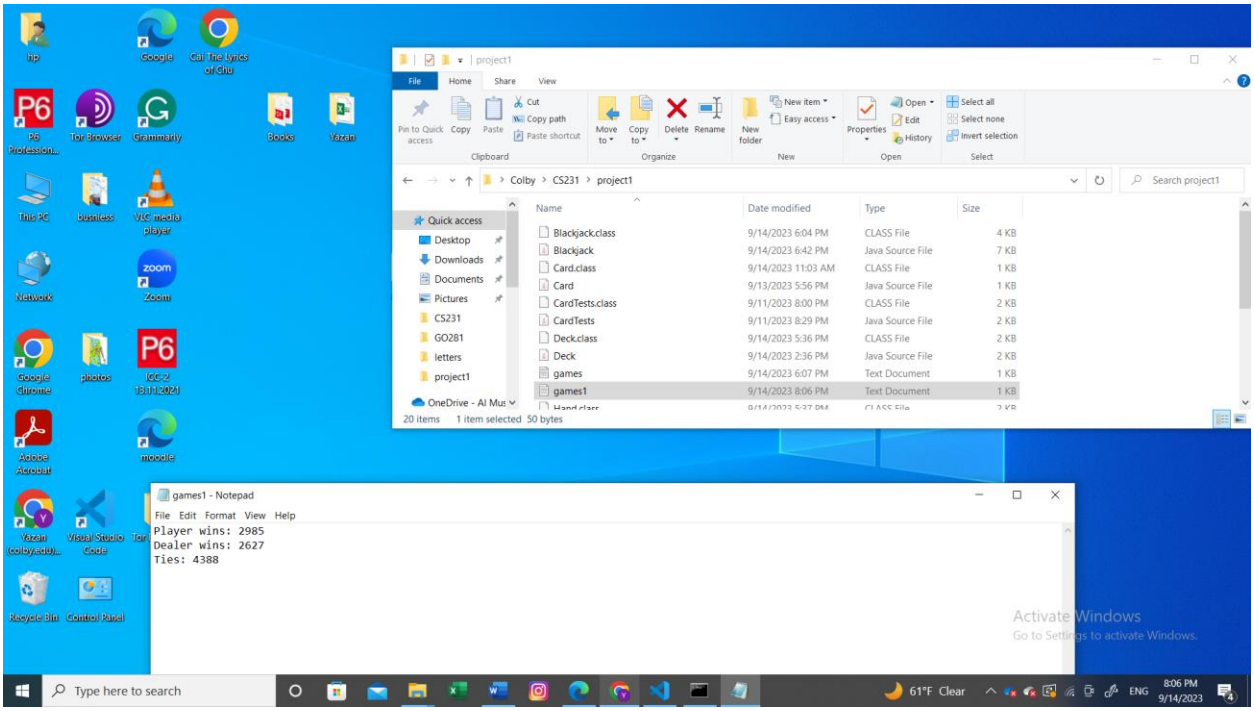

Image-2: The games.txt file got created alongside the results inside. The results are the outputs from the Blackjack main function.

The main function (Simulation.Java)

The simulation class runs a certain number of games, depending on the number of games specified, and returns back the results alongside the percentages for each result.

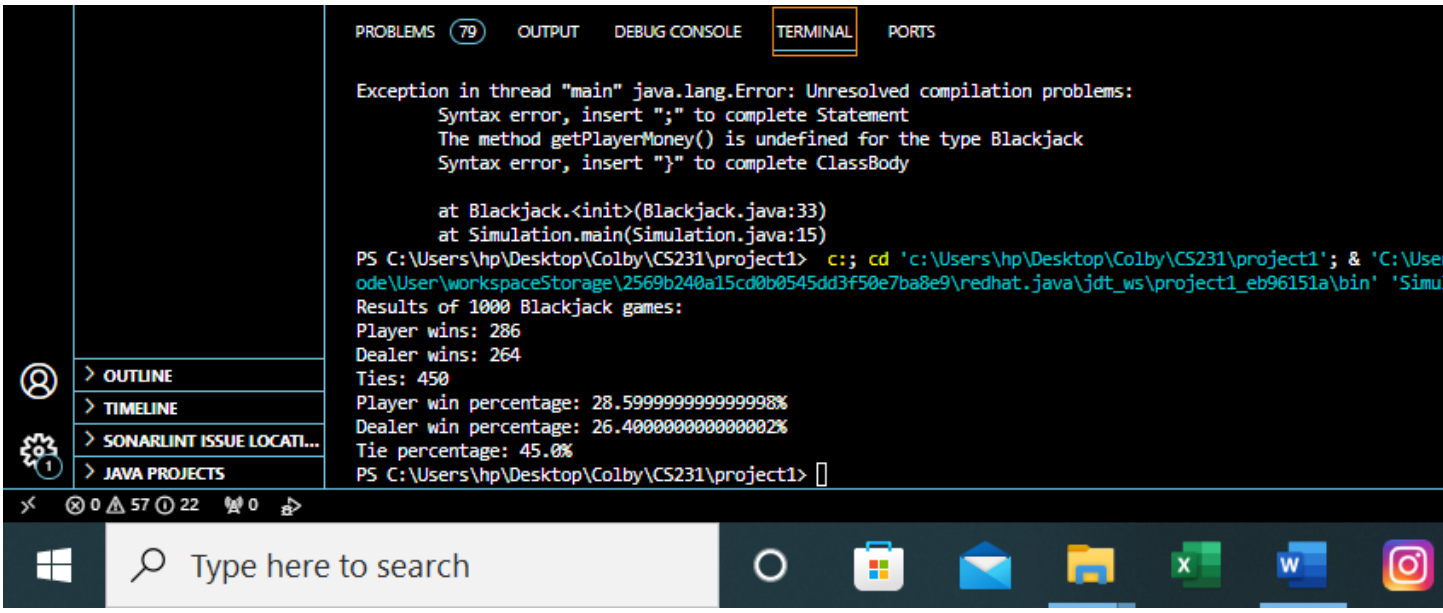

Image-3: The results of one simulation while the number of games being equal to a 1000 games.

Extension

As a significant extension to our Blackjack simulation project, we incorporated a betting mechanism, which introduces an additional layer of strategy and excitement to the game. Players are now able to place bets before each round, and their winnings or losses are directly tied to the outcome of the game. This betting extension adds an element of risk management and decision-making, as players must carefully consider their wagers in relation to their current funds.

We created methods for placing bets and handling payouts. The 'game' method, which orchestrates each round, was extended to include betting logic, where players specify their bets before the dealing phase. We also ensured that players cannot bet more than their available funds. Additionally, we incorporated user prompts to input bet amounts and see updated balances after each round.

```
Card.java | CardTests.java 9 | Hand.java | Simulation.java 7 | Deck.class | Deck.j | Blackjack | Shuffle.java 2
Blackjack.java > Blackjack > game(boolean)
11   public class Blackjack {
89       public int game(boolean verbose) {
123                 System.out.println("Player's hand: " + playerHand);
124                 System.out.println("Dealer's hand: " + dealerHand);
125                 System.out.println(x:"Push (Tie)!");
126             }
127             return 0; // Push (Tie)
128         } else {
129             // This should not happen in a properly implemented game
130             return 0; // Default to tie in case of an error
131         }
132     }
133     public void placeBet(int bet) {
134         if (bet > 0 && bet <= playerMoney) {
135             playerBet = bet;
136         } else {
137             System.out.println(x:"Invalid bet. Please place a valid bet.");
138         }
139     }
140     public int getPlayerMoney() {
141         return playerMoney;
142     }
143
        Run | Debug
144     public static void main(String[] args) {
145
146         Scanner scanner = new Scanner(System.in);  //The player inputs his current total money
147         System.out.print(s:"Enter your initial money: ");
148         int initialMoney = scanner.nextInt();
149         //creating a bBlackjack game object
150         Blackjack blackjack = new Blackjack(reshuffleCutoff:30, initialMoney);
151
152         int playerWins = 0;
153         int dealerWins = 0;
154         int ties = 0;
155         int numGames = 3; // Number of games to play
156
157
158
```

Image-4: part of the extended code, where we added a new method to record and check the validity of the players bets

Acknowledgements

- Kalyan, helped me do the builder method
- Dinesh Varyani (Youtube channel), helped me do and understand the shuttlecutoff function
- ChatGPT, helped me spot my mistakes in the Blackjack main function and correct some part of my Hand.java code
- TAP academy (Youtube), helped me do nested loops in java
- Magzhan Tessiov, Fixed my Reset() method and Verbose() function minor mistakes