Black Jack C++ Project Write-Up

Engineers:
Adams, Jeffrey - jladams42@tntech.edu
Barrera-Solano, Maribel - mbarreras42@tntech.edu
Vasanthan, Vidhula Oviya - vvasantha42@tntech.edu
Cecil, Autumn - abcecil43@tntech.edu

Project Overview:
This project is a simple Black Jack game designed to help students learn Git while working collaboratively. The game allows players to play against the dealer and bet on their hands, keeping track of winnings and high scores. The program will be structured in layers, each class having a distinct responsibility, promoting modularity and team collaboration.

Sections Breakdown:

1. Driver Class (`Driver.cpp`)

The **Driver Class** will only contain the `int main()` function and be responsible for calling all other functions from various classes to execute the game. The main function initializes objects of the `Storage`, `Deck`, `Hand`, and `HighScores` classes, controls the game loop, and coordinates the interaction between the user and the game logic.

Explanation in the Context of Black Jack:
The `main()` function will manage the flow of the game. It prompts the player to start the game, places bets, initiates card draws, checks win/loss conditions, and keeps track of player money and high scores by calling the appropriate functions in the other classes.

---

2. Storage Class (`Player.cpp` and `Player.h`)

The **Storage Class** holds all player-related information, such as their username, total money available for betting, and their current bet. This class is responsible for updating the player's balance after each game and for saving this data for future sessions if needed.

Key tasks include:

- Storing and updating player’s username.
- Keeping track of the player's current balance.
- Modifying the player's balance after each game (win/loss adjustments).

Explanation in the Context of Black Jack:
In a Black Jack game, players bet on each hand. The `Player` class manages the amount of money the player can bet and tracks it throughout the game. It ensures that the player can only bet up to the amount of money they have left.

---

3. Data Class (`GameData.cpp` and `GameData.h`)

The **Data Class** manages everything related to the game's mechanics, including the deck of cards and player/dealer hands. This class contains methods for shuffling the deck, drawing cards, and calculating the value of both the player's and dealer's hands. It also checks for winning conditions and handles Black Jack-specific rules (e.g., the Ace being worth 1 or 11).

Key tasks include:

- Initializing, shuffling, and managing the deck.
- Drawing cards for the player and dealer.
- Calculating the total hand value and checking for a bust or a win.
- Handling rules specific to Black Jack (e.g., the value of an Ace).

Explanation in the Context of Black Jack:
This class handles the core game logic, like when a player hits or stays. For example, when the player asks for another card, this class will provide the card, update the player's hand, and recalculate the total value.

---

4. Other Class (HighScores.cpp and HighScores.h)

The **Other Class** is custom-designed to track high scores across multiple games. It records the highest winnings by a player during a game session. This class maintains a list of high scores, with functions to add new scores and display the top scores at the end of each game session.

Key tasks include:

- Storing and updating the highest winnings.
- Saving and displaying past players' high scores.
- Comparing current game results to past high scores.

Explanation in the Context of Black Jack:
The `HighScores` class is responsible for tracking the best performances in the game. It will store and display the top winnings from all previous players and update when someone beats the highest score. For example, if a player leaves with $500, the class checks if that is a new high score and updates accordingly.

Files Breakdown:

1. **Driver.cpp**
   - Contains the `int main()` function.
   - Calls the functions from the `Player`, `GameData`, and `HighScores` classes.
2. **Player.cpp & Player.h**
   - Manages player-specific information, such as username and balance.
3. **GameData.cpp & GameData.h**
   - Handles the deck, cards, and game logic for playing a round of Black Jack.
4. **HighScores.cpp & HighScores.h**
   - Tracks and displays the highest winnings of all previous players.
5. **Makefile**
   - Automates the compilation process for easy building and testing.
6. **TEST_CASE.txt**
   - Contains detailed test cases to check all aspects of the game, including edge cases, such as running out of money or getting a Black Jack.