

# Black Jack C++ Project Write-Up

## Engineers:

Adams, Jeffrey - [jladams42@tntech.edu](mailto:jladams42@tntech.edu)

Barrera-Solano, Maribel - [mbarreras42@tntech.edu](mailto:mbarreras42@tntech.edu)

Vasanthan, Vidhula Oviya - [yvasantha42@tntech.edu](mailto:yvasantha42@tntech.edu)

Cecil, Autumn - [abcecil43@tntech.edu](mailto: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 `Game` class based on certain criteria determined from the menu choice. The `Game` class and functions actually initialize and utilize all the other classes and functions which control the storage, data and special class selected for this project.

#### Explanation in the Context of Black Jack:

The `main()` function will manage the flow of the game. It prompts the player to start a new game, load a save file, or allows user to close out of the game by providing an exit option.

---

### 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 ( `Game.cpp` and `Game.h` )

The **Data Class** manages everything related to the game's mechanics, including the player/dealer hands. This class contains methods for 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:

- 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 ( `Deck.cpp` and `Deck.h` )

The **Other Class** is custom-designed to handle all the mechanics of the deck. It reads in a text file that contains an instance of all the possible cards that can be drawn. Once initialized, it assigns these an array that contains each card. This also includes some logic to deal and assign numerical values from the string values read in for each card.

### Key tasks include:

- Reading from a text file for all values available in a deck of cards.
- Dealing a random card based on a random seed when called.
- Assigning numerical values to each card when prompted to be calculated by the game logic.

### Explanation in the Context of Black Jack:

The `Deck` in Black Jack is a normal deck of cards with values ranging from 2 - 11. This class allows the user to select pseudo-random cards from the array like they would in a game of Black Jack.

---

## 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. Game.cpp & Game.h

- Handles the deck, cards, and game logic for playing a round of Black Jack.

### 4. Deck.cpp & Deck.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.pdf

- 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.