Damon Leaf
Carmen Yon
CSCI 4448

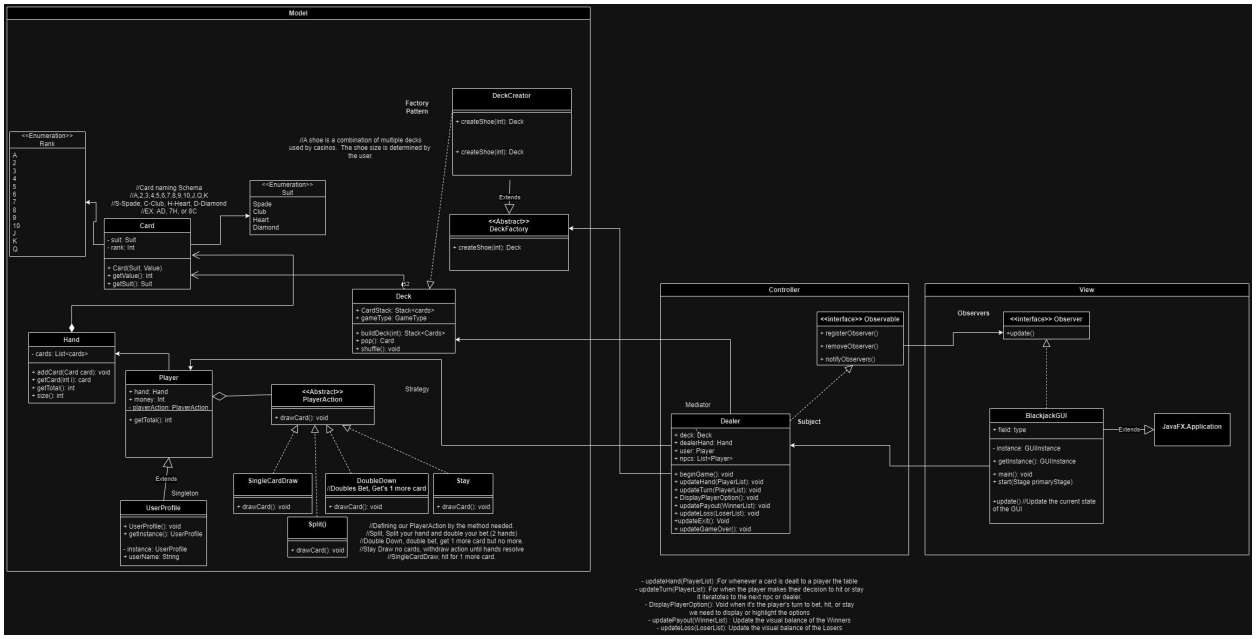
Final Project Report
Carmen & Damon's Blackjack Arena!

Final State of System Statement

The final state of our blackjack simulation is composed of HIT & STAY Blackjack with a variable number of NPCs, NPC difficulty, and number of decks in play. Additionally, the user is able to free bet within their minBet and maxWallet range. While the core functionality promised in the original writeup remains, a few features were not implemented. Features that were not included in our project 5 and 6 writeups were a double down and split system, lucky bonus bets, and save states. The reason these features were not implemented was because of time constraints, developer illness, and the JavaFX learning curve being steeper than expected.

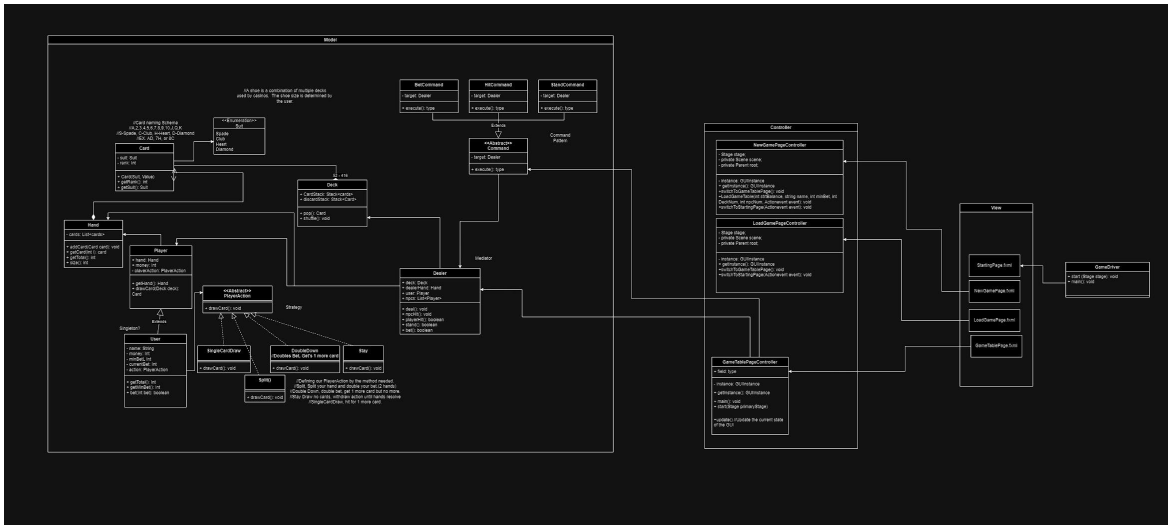
Project 5 UML Diagram:

Share Link: [Link]



Project 6 UML Diagram:

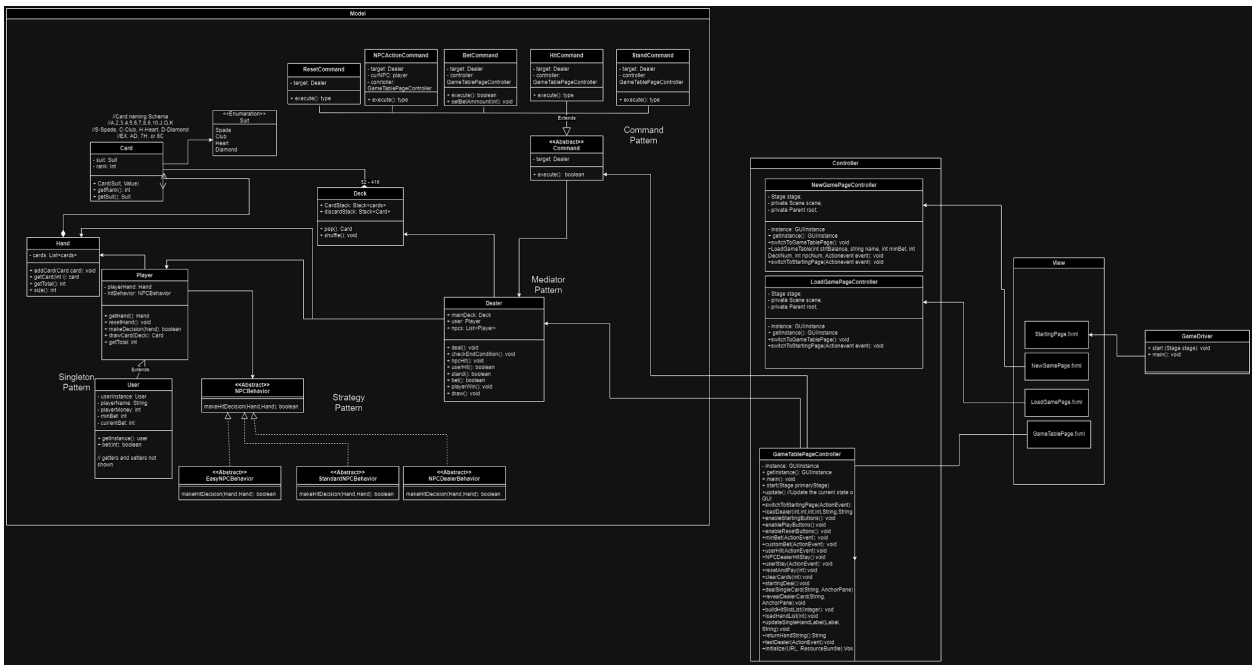
Share Link: [Link]



Project 7 UML Diagram:

Share Link: [Link]

Patterns: Strategy, Command, Singleton, Mediator, and Model View Controller



Key Changes:

The Factory Pattern initially intended to construct our decks was removed because deck objects were not complex or varied enough to justify a factory pattern. Additional controllers for different views were added. The Dealer class was moved into the model section, because it does not act as a controller. We additionally implemented more concrete command classes, as we needed more commands for different functions of the program. We also removed any saving or loading functionality from the User class. User is now implemented as a singleton that extends Player.

Third-Party code vs. Original code Statement

JavaFX: Link

JavaFX was used for our GUI and IO event handling.

Refactoring Guru: Link

The command pattern and singleton implementation was based off of the pseudocode provided on the refactoring guru command pattern page and the singleton page.

Bro-Code JavaFX Course: Link

This course was used for learning the basics of JavaFX: Actions on ButtonPress/user input, Animations of elements, and communication between different controllers/views.

Issues and successes in the design process

Writing requirements and defining use cases went very smoothly for us. However, finding patterns that fit our end goal was more difficult than expected. There were several patterns in our initial plan for the simulator that were either used in a different area than initially planned, or were completely scrapped in favor of a more applicable pattern. We found that attempting to directly implement our original UML diagram was not a winning strategy. We instead began implementing the core functionality to meet our use cases and requirements, causing us to refine our design during development. We also found that pair programming was especially beneficial during this project. Merging after working individually for a while often required refactors to meet each other's expectations and requirements. Pair programming allowed us to take the time to integrate our work and create a final working product.