**Blackjack Card Game Simulation: An Introduction to Java Mini Project**

Allison Juliette Snipes

Department of Computer Science, Johns Hopkins University

EN 605.201.81: Introduction to Programming Using Java

Dr. Sidney Rubey

July 27, 2020

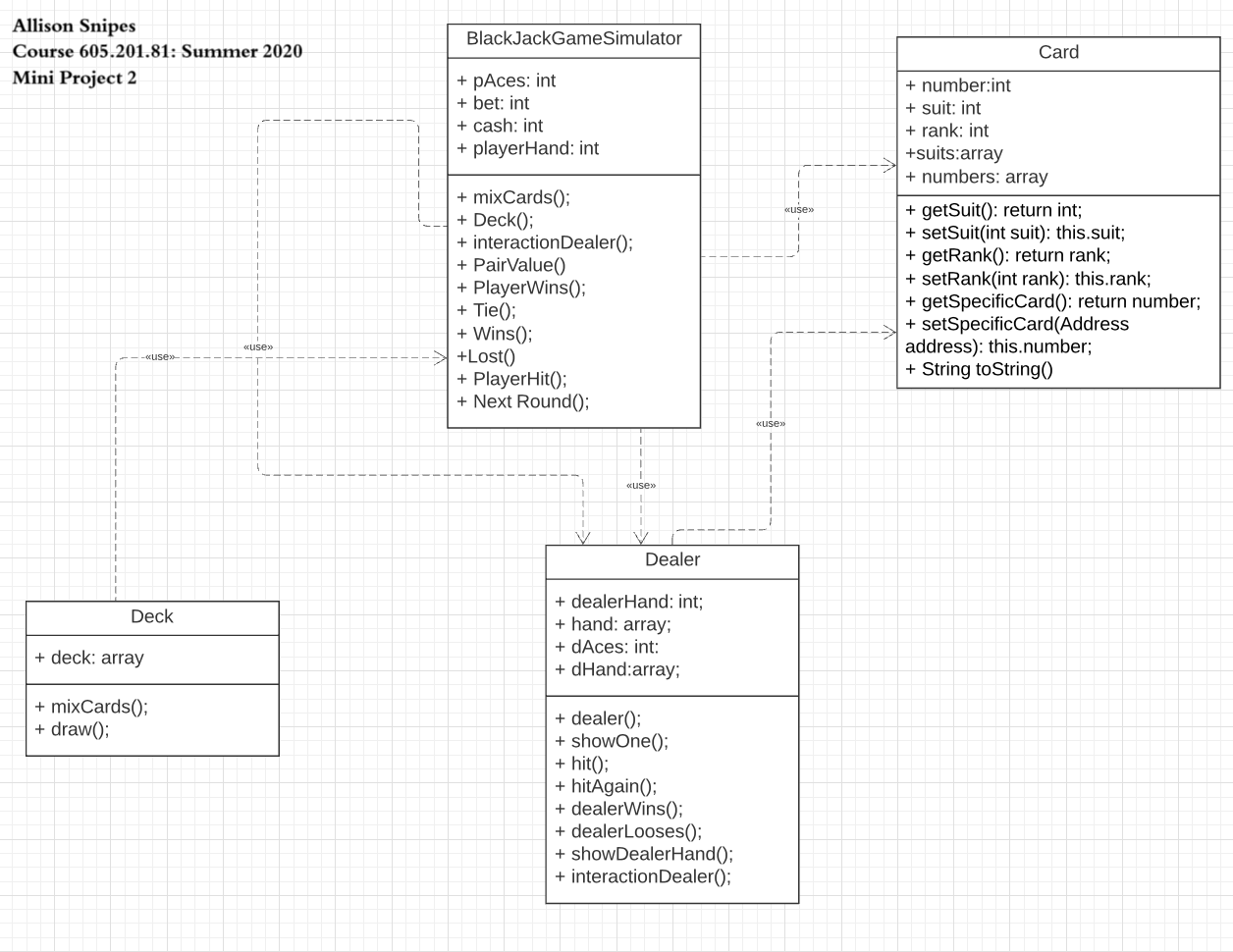
**Introduction**

The purpose of Mini Project Two was to simulate a game of Blackjack between two players: a Dealer, and the application’s user. The cards that the players received are chosen completely at random. The player is able to to gamble any money amount on their hand of cards; so as long as his wager does not exceed the cash reserves they have to play with. The object of the game is for each contender to obtain the sum of twenty-one from their pair of cards, or get as close to it as possible without going over twenty-one. If either the dealer, or player, obtains a pair of cards who’s summation is over twenty-one then they will automatically loose. It is important to note, that either player has the choice to either draw another card, or stay, to help them achieve their goal of reaching twenty-one from their hand of cards.

**Program Design**

The program is organized into four major classes: Black Jack Game Simulator, Card, Dealer, and Deck. The main class is designated as the Black Jack Game Simulator. Within this class all of the program’s control flow, logic, and critical aspects of the three other classes are put together. The Black Jack Game Simulator uses: if statements, for loops, while loops to control the program’s logic flow, scanner utility functions to obtain the user’s input, and simple arrays to store the characteristics of a deck of cards. The main data structures that were used in this program consisted of arrays which populated the pair of cards’ suit, value, and rank per player. Within the Card class the array data structure was utilized to create the rank, value, and suit of each of the fifty-two cards in the deck. Within the Card class getter methods, setter methods, and constructors were utilized to ensure that critical values were stored properly. The Dealer class also relied upon array data structures to build and shuffle the pair of cards in the players’ hand as needed. Lastly, the Deck class held the logic and arrays responsible for being able to shuffle the array and draw cards.

There are a plethora of alternative ways in which this program could have been developed. For instance, the methods for obtaining a hit from the Dealer, staying, and determining who won or lost could have been written as interface classes to incorporate certain efficiencies of Polymorphism. If I would have utilized interface methods, my methods would have been able to be reused throughout my program, and be able to inherit critical aspects of my classes easier. Secondly, I would have been able to pick and choose what characteristics each methods should share, and which ones they should not. Lastly, I believe I should have made two separate classes for both the Dealer and the Player; with the Dealer class inheriting from the Player class or vise-versa. Both classes share important game actions, and tactics, and their differences could be easily coded to be accounted for. I did not utilize those features as it was not something that crossed my mind initially, the idea to do so came whilst I was writing the paper. However, I guarantee in the future I will incorporate these coding mythologies into my projects and homework assignments. The UML diagram of my program is included below:

**Reflection**

I learned how to become more comfortable with creating , manipulating, and working with arrays as a data structures. I hope to find more tutorials and projects online to assist with building my acquired skills. While working on this project, it became more apparent to me that I need to continue to practice my programming logic. This project highlighted the importance of solving my logic issues one by one, instead of all at once. As I was not able to resolve my code misfiring when the user was asked to hit or stay; I believe that I am close to a solution. One thing that I would have done differently was to utilize abstract classes for different users for more than two users.

**Conclusion**

In closing, this was an intense yet rewarding project to work on. There were numerous pieces that one needed to put together logically for the code to work, and basic error handling. I look forward to building my logic skillset to be able to work more quickly and efficiently during these projects.