Design Document

Instructions

- To run the code, run the main method within the Main file of the src file's com.company
- All tests are within the testGame file of the test directory and can be individually run.

Design Choices

- The user can hit and stand as per BlackJack options. However, they can't split.
- Dealer class inherits the Player class, the former standing in as a computer player and the opponent for the user.
- The user can type in 'Hit' / 'hit' for a new card or 'Stand' / 'stand' to keep their cards as is.
- All card display and user input commands remain within the terminal, so the user can play and only have to interact with the terminal window.
- Since BlackJack just depend on the card rank not its suit, I only printed the ranks.
- The data structures I used are ArrayLists and arrays
    - ArrayLists correspond to the players' **open** cards in the Player class as the players may hit multiple times and and to the **players** in the BlackJack class as I first planned to implement multiple computer Players.
    - Arrays correspond to the int array called **deck** and the String array called **POSSIBLE** in the BlackJack class. A standard deck of cards totals in 52, so each element in **deck** initially has 4 for each card rank (A, 1, 2, ..., J, Q, K). **POSSIBLE** refers to the possible card ranks and allows for easy translation from a random int to a String card rank.

Tooling

- For the language, I chose Java as I have the most experience in it.
- For the libraries, I used the ArrayList and Scanner libraries. The first is because of its ability to dynamically increase its size if needed. The second is to take in the user commands for hit, stand, start a new game, or stop.
- For the test runner, I only ever used JUnit in past projects, so I continued to do so for this challenge.