Team #2 - Poker Hands Calculator

Final Report

By: Justin Trussell, Nicholas Medina, Diego Uribe, Mariano Garcia, Brandon Hunt

1. Introduction

Since poker takes a lot of thinking to decide the odds of you winning or losing, this new application will make it easier for you to decide whether you should call, raise, or fold! The motivation behind this project comes from a collective love of playing poker. This will help the user understand the chances of winning according to their in hand cards. Not only will this help the user understand the probability of hands winning, it will also offer as a teaching method to understand what hands are stronger than others.

A bigger motivation for a software like this, is that it will give us some experience in algorithms with probabilities and some more experience in the GUI side of object-oriented programming.

1.1 Project Scope and Objectives

The scope of this project is to develop a web-based application that will operate in harmony with multiple libraries to create our poker software.

Phases:

1. Develop a system regarding the poker cards
    a. Create the poker cards
    b. Draw Cards from the deck according to the number of players
    c. Shuffle the cards
2. Test Product using terminal
    a. Test the drawing of the cards and assigning of cards
3. Develop a GUI (Prototype) *Time Pending*
    a. Display Cards of the player's hand
        i. Potentially allows user input to test hands for other players
    b. Display the cards on the table (Flop, River, Turn)
        i. Currently displaces the entire table instead of iteratively (flop, river, turn)
    c. Display percentage of the hands (Flush, Straight, etc)
        i. Current iteration displays total odds of winning

Scope:

1. Phase One

The scope of the first phase is to create the project itself. Create the cards, shuffle the cards, and deal the cards depending on the number of players.[mention calculating hands] Phase one began February 18th and was finished by March 2nd.

2. Phase Two

The scope of the second phase focuses on testing to see if our product is working properly. We test using printouts to the terminal of the cards assigned to each player (depending on the number of players) and cards that were burned and assigned to the table. [mention calculating poker hands]. Phase two began on February 18th and finished by March 8th.

3. Phase Three

The goal of phase three was to create a prototype (functional or non-functional) of our envisioned product. This Graphical User-Interface was created using Java Swing constants. This prototype is to accept input on how many players to account for and display the proper cards to the player and table. Note that this prototype only shows all five cards already revealed on the table, not incrementing reveal as a standard procedure of 3 to 4 to 5.

4.Phase Four

The goal of the fourth phase is to make the software as an executable that performs all outlined tasks and functions correctly. This would be the final version of the calculator that does not require any other turning to the functionality and runs with the expectation of accurate results.

1.2 Supplementary Requirements

The software’s aim is to be just the odds for the player's hand. There is no intention for other software to be used with ours or for the framework of the code to allow subsystems. The goal is for the software to be run when needed and executed efficiently with clear and defined documentation. The software will execute the given hands and reset after all turns have been completed.

We opted to use an API from University of Alberta, Department of Computing Science Computer Poker Research Group. The source of their code can be found here, at http://spaz.ca/poker/doc/ca/ualberta/cs/poker/package-summary.html. This research group focused on getting all of the combinations of each of the faces in a deck and returned a sort of rank that the specific hand and the cards on the table would be. This rank allowed us to know how good this particular hand would be and we compared it to the other players and because of the API we were able to find out what our chances of winning were. The API allows us to present accurate winning odds that is scalable for a total of 2 to 9 player

2. Customer Requirements

This section contains the architecture view of your use-case model. It includes

- use-case diagram(s),
- actor descriptions and prioritized use-case descriptions.

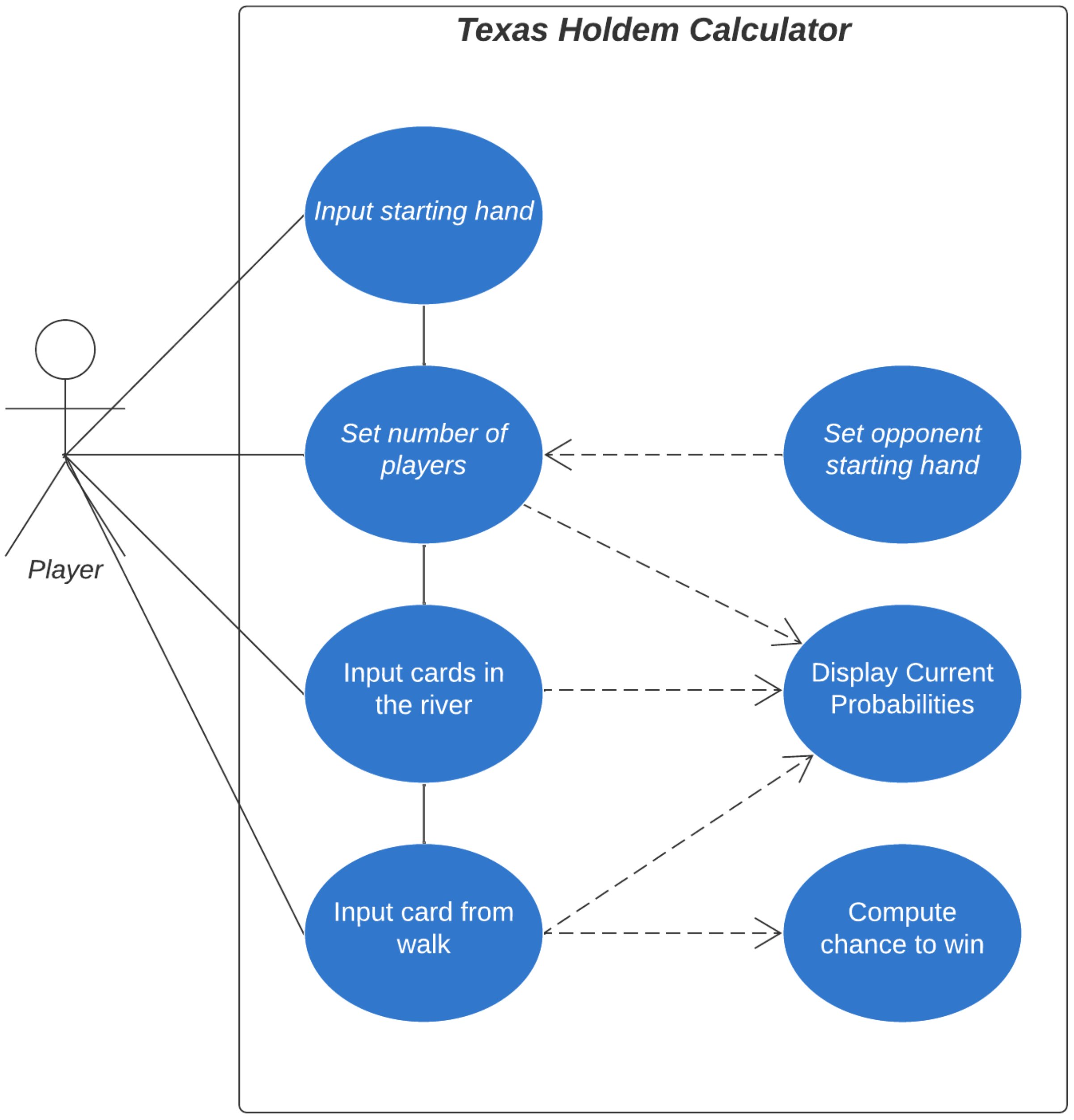


Figure 1: Use Case Diagram

The use case diagram in figure one depicts the player (user) actor interacting with the program. The actor’s choice for interaction is first to set their starting hand by inputting the cards that were dealt to them by the dealer. After inputting their hand the actor then adjusts the amount of players or leaves the value at the program’s default of 2 other players as well as being able to put the other players cards . From here the actor is then shown the current odds of winning and is now prompted to place the cards for the river and shown the new odds for winning. The final interaction takes place after the odds are shown and the actor then inputs the card that is drawn from the walk.

Our use case involves one actor as the player as the rest of the software is run exclusively through the code and without the need of any other actors as administrators. The only parts of the software the actor does not interact with are the calculations of the probabilities and win percentages.

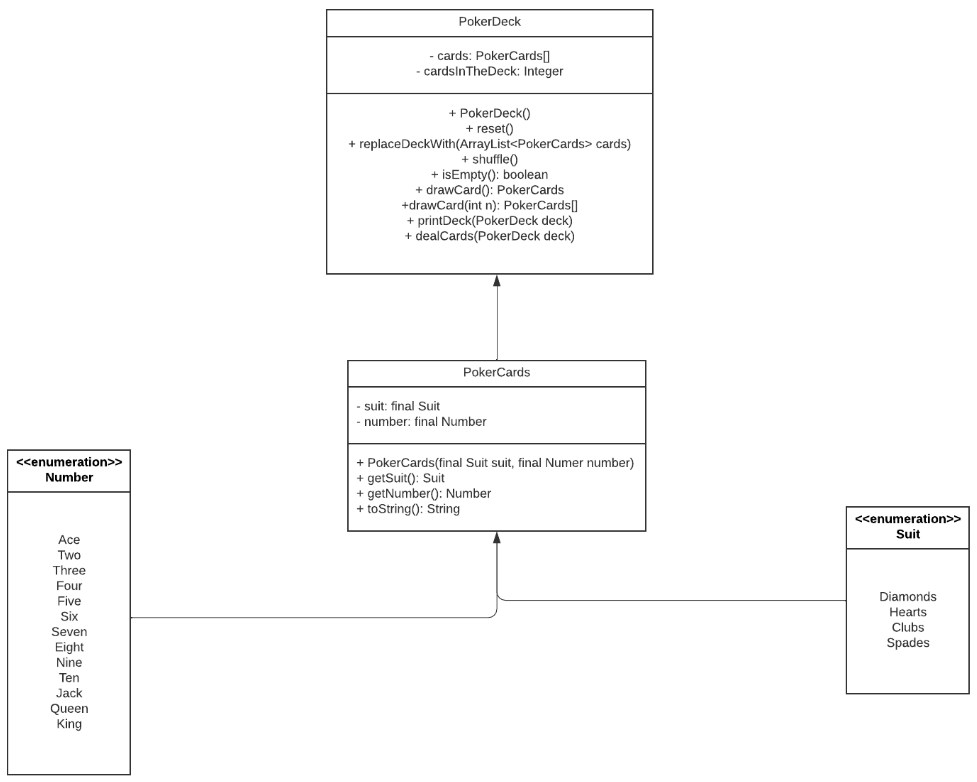
3. Architectural Design

Our softwares architectural design does not involve any subsystems. However, there has been discussion to create a blackjack calculator which would be implemented as a middleware and would inherit from the main poker program.

3.1 Subsystem Architecture

For our software we do not have a subsystem currently. In the future, if we have the time, we would like to implement a black jack calculator and make that a subsystem of our calculator. The subsystem would be a middleware and would directly plug and play with the main code utilizing the deck as well as the deal. The substem would have to employ its own methods of calculating the win and loss odds as it is entirely a different game as well as implement a shoe of decks as that is how blackjack can be played.

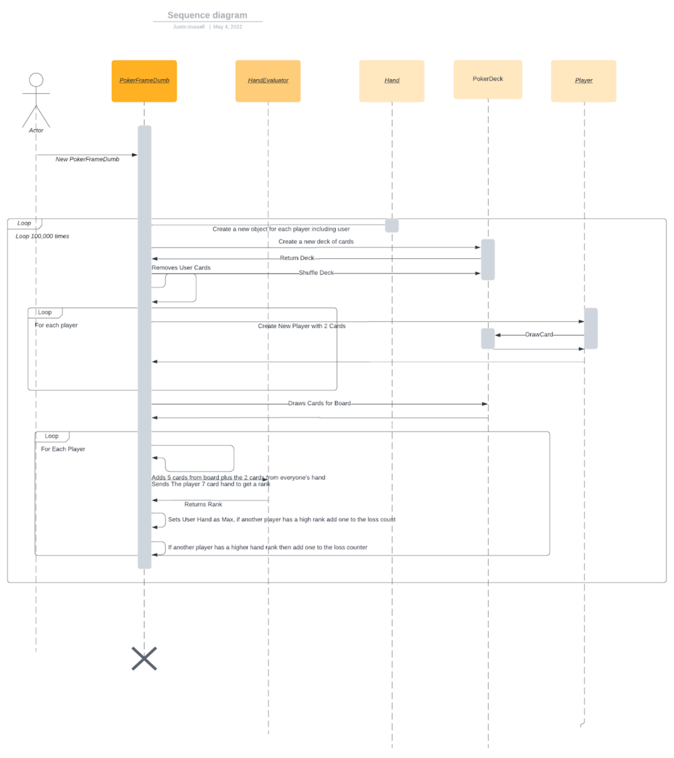
4. Use Case Realization Design



5. Subsystem Design

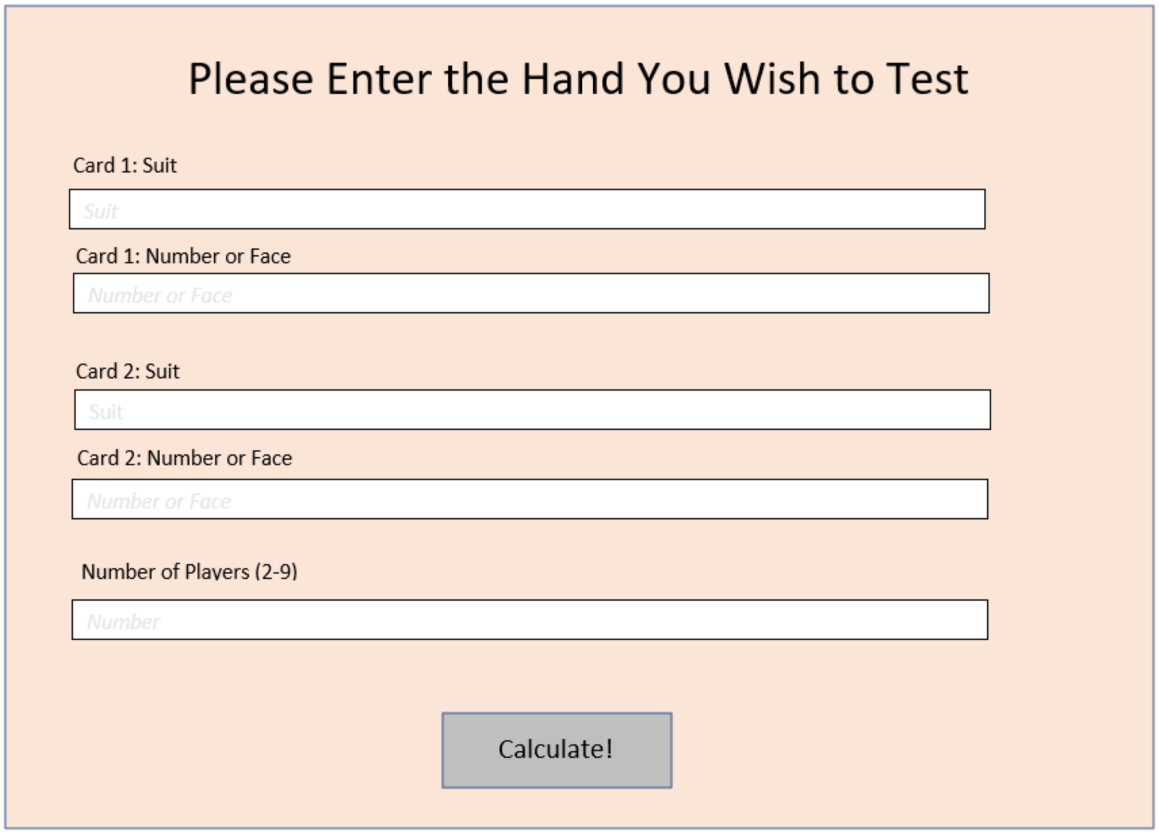
Currently, we are using an API developed to evaluate a player's hand. This API compares the table to the player's hand and decides the player's chance of winning. It does this by running through every single possibility on what cards other players may have, and we opted to do this loop 100000 times so that we will get the most accurate representation of what the chances the user will have of winning with the hand they have been dealt with.

5.1 Hand Evaluation API Sequence Diagram


Sequence diagram
Justin.trussell | May 4, 2022
Actor
PokerFrameDumb
HandEvaluator
Hand
PokerDeck
Player
New PokerFrameDumb
Loop
Loop 100,000 times
Create a new object for each player including user
Create a new deck of cards
Return Deck
Removes User Cards
Shuffle Deck
Loop
For each player
Create New Player with 2 Cards
DrawCard
Draws Cards for Board
Loop
For Each Player
Adds 5 cards from board plus the 2 cards from everyone's hand
Sends The player 7 card hand to get a rank
Returns Rank
Sets User Hand as Max, if another player has a high rank add one to the loss count
If another player has a higher hand rank then add one to the loss counter

6. Human Interfaces

User Input Screen:



Please Enter the Hand You Wish to Test

Card 1: Suit

Suit

Card 1: Number or Face

Number or Face

Card 2: Suit

Suit

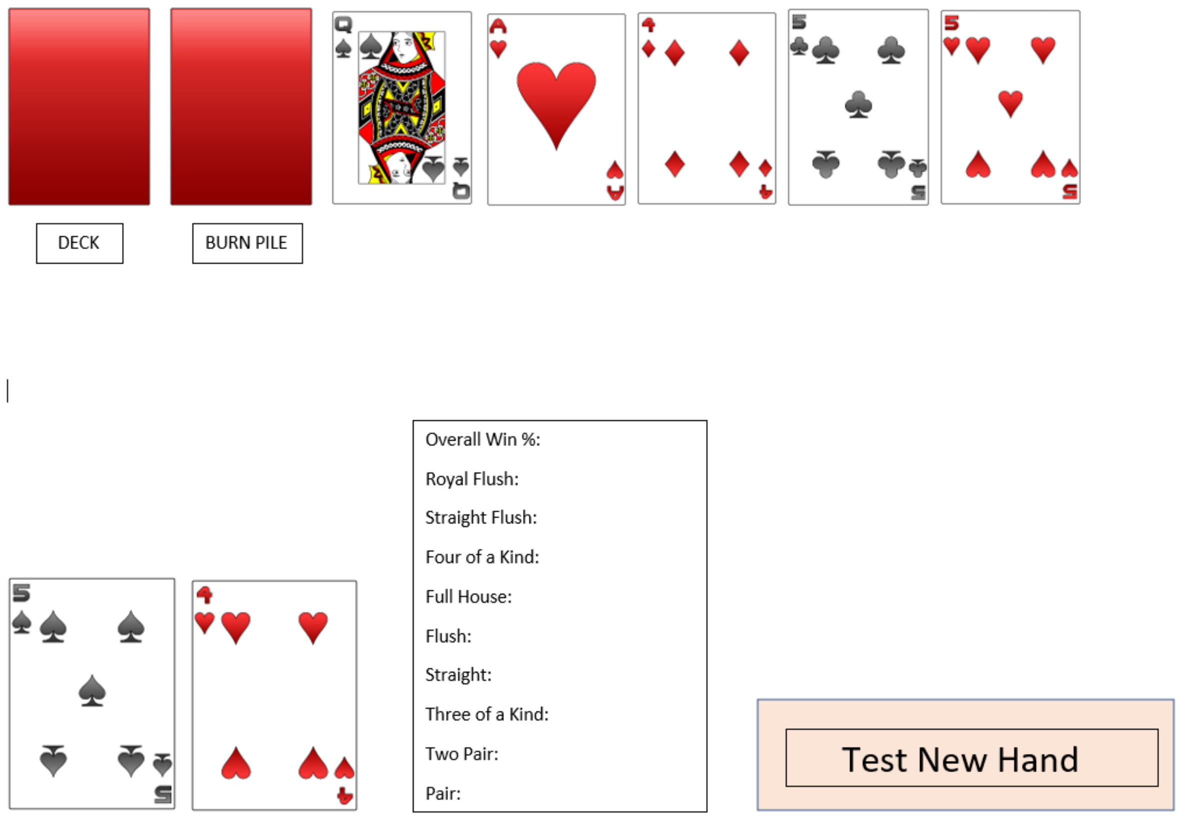
Card 2: Number or Face

Number or Face

Number of Players (2-9)

Number

Final Product:



7. Testing Plan/Results

1) Functionality Testing
    - Cards are created and formatted properly
    - Cards are dealt to the player, opponents, and table properly
    - Test Print out of proper variables
2) Performance Testing
    - The software has proper exception handling, and properly deals with all mentioned hands.
3) User Testing
    - Testing with the GUI was successful, and produced accurate results.

8. Project Status/Summary

8.1 Project Status

Currently we are testing for final bugs. The software works properly and is in a quality check phase.

8.2 Difficulties Encountered

The main difficulty we encountered was an integration problem. We all set out to solve our problem on our own. When we came together to assemble the final product we ran into problems integrating each other's code. This problem took us several days to figure out and to solve this problem we spent several hours altering code in order to integrate our code.

Another difficulty that was encountered was the fact that each of us were friends and we did not communicate efficiently with each other with what needed to be finished and accomplished. There was a lot of rushing to finish and produce what was needed because of the disorganization. Had we been more on top of each other and what was being worked on there would have been a more complete software developed. The lack of communication also led to two different projects being worked on in different languages resulting in a lot of discussion and argument over what the final product would look like.

9. Appendices

Attached in the zipped file is a HTML file for our Java classes index as a Java doc. This is the source code of our project and is to be used in conjunction with the rest of the report.