Janaye Jackson

Professor Lehr

CIS-17C

06/08/2024

**Project Write-up**

## **Introduction**

***What are you coding and why did you choose this Game?***

I am coding the card game BlackJack. I chose this game because I know the rules very well and it is among one of the first card games I ever learned to play. I knew implementing the game with these concepts would be simple enough to accomplish.

***How long did you spend, how many lines, classes, etc.....***

I spent a total of 30 hours on this project. There are 5 classes. The first four contain structures and methods related to trees, nodes, graphs, and the cards used in the game. The last one contains the main code and functions. There are about 1355 lines of code. 203 out of those 1355 lines include comments and/or white space.

***Where on github is it located?***

This is located in the project folder in my github repository. It is labeled project 2.

## **Approach to Development**

***Concepts***

For this program I came in with the concept of making it as close to a real game of Blackjack as possible. It consists of a lot of methods, each method contributing to the overall game play method. The majority of the code is in the gameplay method. This method wholes all scenarios in which players lose or win the game. It also includes betting so players can “win” or “lose” money.

***Version Control***

There are 7 total versions

V1 sets up the deck

V2 Implements a basic game of Blackjack

V3 adds on to the game of Blackjack to make it more like Blackjack

V4 Implements ace special cases

V5 Implements player and dealer statistics

V6 adds betting and implement special betting cases

V7 adds splitting the deck

V8 Implements recursion and recursion sorts

V9 Implements trees

V10 Implements hashing functions

## **Game Rules**

1. Your goal is to get as close to 21 without going over.
2. Each round you will have the chance to continue playing or quit
3. When playing you have the option to hit if your first two cards do not result in a Blackjack or bust
4. You can choose not to hit or hit until you bust or are satisfied with your card total
5. If you get 21 exactly that is considered BlackJack and an automatic win
6. If you go over 21 that is considered a bust and an automatic loss
7. If you are under 21 then you play against dealer if dealer has 21 or cards higher than you, you lose, if your cards are higher then dealers you win.
8. If your first two cards are the same face, you may split and play two hands

***Betting***

1. You have the option to play with bets, if you chose to play with bets the game is over once you have no more money or you choose to quit
2. There is a minimum bet of $2
3. If you get Blackjack you receive 1.5 times your bet back
4. If you beat the dealer you receive your 1 times your bet back
5. If you lose you lose your bet
6. If your first 2 cards equal 9, 10, or 11 you have the option to
7. double down, this doubles your bet and you receive only one more card to beat the dealer
8. If your first 2 cards share the same face, you can only split if you have enough to double your bet, same goes with doubling down

## **Sample Input/Output**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

## **Checkoff Sheet**

## **Documentation of Code**

Things to Include in the Documentation - Enough of each of the following to show expertise

**Flowchart – main method**

**A diagram of a computer program

Description automatically generated**

**Pseudo-Code**

Start

Initialize deck of cards

Shuffle deck

Deal initial cards to players and dealer

Player's turn

Check if player has blackjack

If black jack

Adjust money

end round

Check if player bust

If bust

Adjust money

end round

Player decides to hit or stand

If player hits:

Deal additional card to player

Check if player has blackjack

If blackjack

End round

Adjust money

Check if player busts (hand value > 21)

If player busts:

End round

Adjust money

If player stands:

Player's turn ends

Next player's turn

Dealer's turn:

Dealer reveals hidden card

Check for black jack

If black jack

Dealer won

Adjust money

End round

Dealer hits until hand value >= 18

Check if dealer busts (hand value > 21)

If dealer busts:

Players wins

Adjust bet

If dealer does not bust:

Players looses

Adjust bet

End of round

Ask if players want to play again

If yes, go back to Start

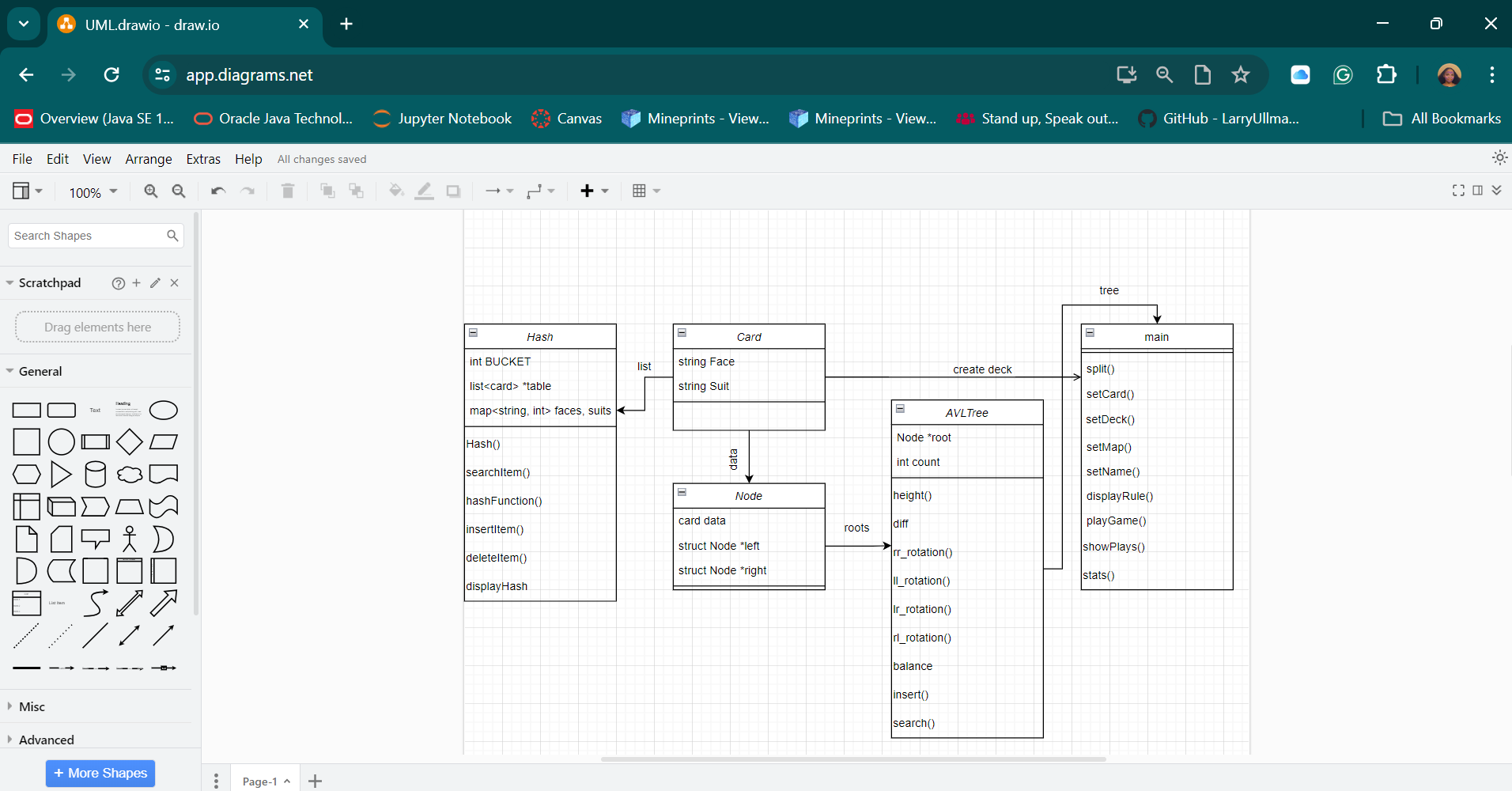
If no, End

Show stats

Show money

Show dealt cards

**UML Class Diagram of each class developed**

****

**Project2 Checkoff Sheet**

**Recursion**

A recursive method is used starting at line 923 of the main file. It is used to handle the dealer and it is dealing with its own cards. It deals cards until the total value of the cards dealt is less than 3. Else it calls the function again.

**Recursive Sort**

I have a recursive merge sort method starting at line 936 of the main file used to sort the player and dealer cards before they are displayed at the end of the game.

**Trees**

I use a tree to hold all of the cards a user has been given and then they can search the tree to see if a specific card has been played.

**Hash**

I use a hash table to store all player and dealer cards. That table is then used to count how many times each card has been played. Cards are hashed based on their face value and their suit value.

**Checkoff Sheet Contents**

1.Container classes (Where in code did you put each of these Concepts and how were they used?

1.Sequences (At least 1)

~~1.~~[~~list~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/List.html)

**2.** [**slist**](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/Slist.html) **– lines 81, 539, 562, 563, 564, and 565**

Line 81 is used to hold the total card values of each play.

Line 539 is used to temporarily hold a randomized list of 13 cards of one suit in order to add suit names to card structure.

Line 562, 563, 564, 565 hold a list of 13 cards of each suit.

~~3.~~[~~bit\_vector~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/bit_vector.html)

2.Associative Containers (At least 2)

**1.**[**set**](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/set.html) **– lines 541 and 621**

Line 541 is iterated through to add a randomized list of cards to a list of cards

Line 621 holds an unordered set of randomized face values

**2.**[**map**](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/Map.html) **– lines 82 and 600**

1. Line 82 uses an already made map
2. Line 600 maps faces to their values

~~3.~~[~~hash~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/hash.html)

3.Container adaptors (At least 2)

**1.**[**stack**](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/stack.html) **– lines 84 and 85**

1. Both line 84 and 85 are used to store the cards played in the game one for the dealer and one for the player

**2.**[**queue**](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/queue.html) **– lines 52, 86, and 561**

1. Lines 52 and 86 are used to hold a shuffled deck of cards
2. Line 561 is used to add cards randomly from the 4 lists of cards with different suits

~~3.~~[~~priority\_queue~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/priority_queue.html)

2.Iterators

1.Concepts (Describe the iterators utilized for each Container)

~~1.~~[~~Trivial Iterator~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/trivial.html)

~~2.~~[~~Input Iterator~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/InputIterator.html)

3**.**[**Output Iterator**](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/OutputIterator.html) **–**

Used to ouput items from set into a list

Used to output items in a list to console

4.[**Forward Iterator**](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/ForwardIterator.html) **–**

used to traverse all items in set to add to list

Used to traverse items in a list to count number of black jacks

~~5.~~[~~Bidirectional Iterator~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/BidirectionalIterator.html)

~~6.~~[~~Random Access Iterator~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/RandomAccessIterator.html)

3.Algorithms (Choose at least 1 from each category)

1.Non-mutating algorithms

~~1.~~[~~for\_each~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/for_each.html)

~~2.~~[~~find~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/find.html)

~~3.~~[~~count~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/count.html)

**4.**[**equal**](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/equal.html) **- lines 142, 143, 147, 152, 249, 250, 254**

All of these are used to compare the face values between cards to determine if player or dealer starts with two aces which is a special case or two of the same cards which is another special case.

~~5.~~[~~search~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/search.html)

2.Mutating algorithms

~~1.~~[~~copy~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/copy.html)

~~2.Swap~~

~~3.Transform~~

~~4.Replace~~

**5.**[**fill**](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/fill.html) **– lines 129, 134, 180, 206, 236, 241, 265, 292, 330, 351, 369, 396, 427, 445, 473, 500, 551, 578, 582, 586, 590, 652, 655, 658, 661, 664, 667, 670, 673, 676, 679, 682, 685, 688, 729, 759, 793, 801, and 809**

Used to fill list with cards, stacks with cards dealt to players, and queue meant to represent deck of cards. Most used algorithm.

**6.Remove – lines 130, 135, 237, 242, 331, 352, 428, 579, 583, 587, 591, 730, 760, 794, 802, 810, 850, and 859**

Mainly used to remove cards added to deck queue and then remove cards from the deck queue

~~7.Random\_Shuffle~~

3.Organization

**1.Sort – line 704**

Used to sort player scores to make counting all blackjack wins take less time

~~2.Binary search~~

~~3.~~[~~merge~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/merge.html)

~~4.~~[~~inplace\_merge~~](http://209.129.8.7/~MarkLehrSyllabi/sgi-stl-docs/docs/inplace_merge.html)

~~5.Minimum and maximum~~