Blackjack Slot Machine

California State University Long Beach

December 8, 2018

Howard Chen

Scott Armia

B. Write a report to describe your algorithm. The report should include:

1. Front page (name of project, school, and group members).

2. Introduction – describe the purpose of the program.

3. Program Analysis and algorithm design.

- Describe any variables involved in the program

- Describe any functions used in the program

- Describe algorithm for main function.

5. Sample run – show “every” possible outcomes of this program.

6. Provide UML diagram that shows relationship between classes you use in the project

NOTE: This program should use at least three classes Card, Player, and Account.

**2. Introduction**

The main purpose of this program is to simulate a text-based version of a Blackjack Slot Machine. The program displays an object-oriented design with and Account class, Black Jack Slot Machine class, Card

**3. Program Analysis and Algorithm Design**

**Important variables**

In the BlackJackSlotMachine we have a few variables used such as the two arrays used to generate the deck a deckIndex which is an integer variable that is a counter for which element in the deck vector is dealt. Then we construct two player objects: dealer and user .The Card class includes a string name called the cardName and the integer variable value which is the value of the card. The Player class has 4 variables. It has two vector variables hand and splitHand which are basically a vector of Card objects of the hand of the plater and their split hand.

The player class also has two booleans: done and doneSplit, which help keep track of when the dealer/player busts while doneSplit keeps track of the splitHand bust since the player doesn't lose untill both hands bust.

The three double variables: pot- the amount of money won for player , totalBet- the amount of money the player bet, and amountWon - overall earnings. The isSplit variable is a boolean that helps us keep track if the player split his hands or not since the player can only split his hand only once and when he has a pair.

In the card class, we have the instance variables: cardname and value of each card object. The card value is of int type and the name is of string type.

The account class has two instance variables : the string accountNumber, and the money in double which keeps track of how much money is left in the account.

**Functions in Classes**

In the account class we have a getter function that just returns the accountNumber the user inputed. There is also another getter to get the private double variable money. We then have two void functions bet and win. The bet function takes in a parameter for the amount of moneyBet by the user. In which it first checks the amount the user requested to bet is less than the amount bet, if it is not enough the program will just bet the remaining amount and then 0 out the money in the account. If the user has enough money then it will just decrement the money instance variable from the account and print out the money bet and the remaining balance. The win function takes in the moneyWon as a parameter and adds the moneyWon to the money instance variable in the account then outputs the amount won with the totalBalance.

In the Card class we only have a few getters and setters. We have a getter that gets the name of the card, a getter that gets the value of the card and a setter that sets the value of the card which will be used later to change the value of the ace card depending on the situation.

Lastly, our player class has quite a bit of functions since it does most of the game’s functions. It has a hit function which takes in a card object as a parameter , it first checks if the card is an Ace or not. If it is an ace the card value is either set to 1 or 11 depending on the users hand. It then pushes the card into the vector hand, lastly checks if the player busted or not and sets the done boolean to true if the player busts. The hitSplit is essentially the same function as hit since it just upadates the splitHand vector and doneSplit instead. Then we have two getters for the done and doneSplit booleans. The removehand function clears the hand vector when the cards are discarded and same idea with the removeSplitHand. We also have getters for the splitTotalValue and getTotalValue which calculates the values of the hand by iterating through the hand vector of cards and added the values of the card objects up. We also have 3 seeHand, seeSplitHand, seeDealer hand functions that display the cards currently in the players hand. The last function called splitAble checks if the user has duplicate cards in his/her hand and returns true. It iterates through the vector hand from 0 to hand.size() -2 since we are only checking for the last card drawn with the rest of the other cards.

**Main Algorithm**

**Generating the Deck of 52 Cards:**

The program first assigns two array variables: one with a string of the names of all the cards called faces[]. The other corresponding array consists with the values of the cards called values []. Then the program uses a nested for loop to construct card objects. The outer loop loops through 13 times since there are 13 different valued cards. The inner loop loops through 4 times since there are 4 of the same value cards (diamond, spades, clubs, hearts). These card objects are assigned to a vector of size 52. Since the cards are in order at the moment we have to shuffle the deck. To shuffle the deck we call srand to create a random number generator based on time rather than using just the pseudorandom number. We used random\_shuffle which is a function in the "algorithm" library.

After that the program asks for user input for account number then constructs the Account object with a starting money balance of $50 defaulted in credit. Then we have a do while loop that asks for the bet amount constructs dealer and user objects. Whenever the player hits, the player hit function is called and the deckIdx is incremented. Then, the results of the move is printed with the seeHand , seeDealerHand() functions and we print the menu with 3 options of Hit, Stand, or Split. Each option is then checked with a while loop for Stand, since whenever the player stands the round will end. In that while loop we also check if the hand i split already or not and then a series of if statements check if the user wants to hit stand or split with the split hands. Whenever the user busts the user is unable to hit again and the program will display " You can't hit anymore" and will continue to loop the menu untill the user chooses to stand. Whenever the user chooses to split the program will call the splitable function first to check if the user has a pair in his hand before spliting the hands. Once the user chooses to stand the program will check for a winner and out the result of the round , then ask the user if he wants to play again or not. The deck is then shuffled and the while loop continues unless the user inputs "No", the total amount betted and amount won will be displayed and the program will exit.