COP-2250 Java Programming

Programming Assignment 3

FIU Knight Foundation School of Computing & Information Sciences

1 Problem Specification

For this programming assignment, you will implement of a java program for playing the simplified version of Black Jack game with only one player. To be familiar with the original game, you can watch the following video: https://youtu.be/VB-6MvXvsKo

1.1 Introduction to Blackjack

The game uses a deck of 52 cards (see Figure 1). Each card has a value specified in the following way:

- cards with numbers 2, 3, . . . , 10, have the value equal to the number written on them.
- each face card (Jack, Queen, or King of any suit) has the value equal to 10.
- ace cards (A♡, A♢, A♣, and A♠) have the value of 1 or 11; whichever the player prefers.

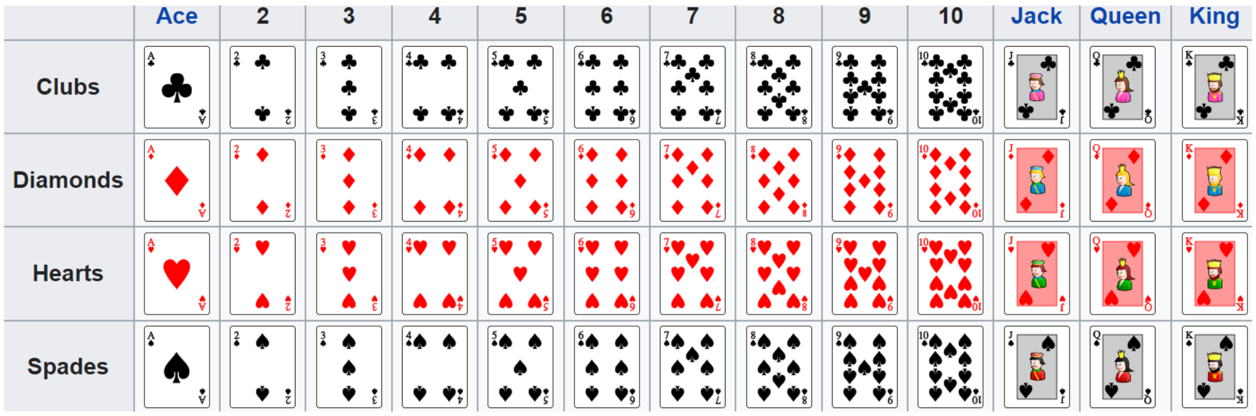

Figure 1: Deck of 52 cards. Each card has one of the four suits ♡, ♢, ♣, and ♠.

For this assignment, we focus on the simplified game with a dealer who deals the cards and a *single* player. The game's result is one of the following three: player wins, dealer wins (or player loses), or the game ends with a tie.

At the beginning, the dealer deals two cards to the player and two cards to the dealer itself. Player sees both of her/his own cards face up, but only sees one of the two dealer's cards (the other one is face down and the player will see its face only at the end of the game).

If sum of the values of cards dealt to the player so far is 21 (e.g. A♣10♢ or A♡K♠), then player wins the game. Otherwise, the game continues as follows:

Step 1: Player Chooses: Hit or Stay

Player chooses one of these two actions: "hit" or "stay". If the player chooses "hit", the dealer deals another card to the player; otherwise, the game continues with the second step. If sum of the values of cards dealt to the player so far is 22 or more, the player is busted and game is lost. If the sum is 21, the player wins, otherwise, the player is given another choice to hit or stay and step 1 is repeated.

Step 2: Dealer's Hand is Exposed

Dealer exposes its cards. If sum of the values of these two cards is 21, dealer is the winner. If sum of the values of these two cards in 17 or more, then there are three possibilities: (1) player wins if sum of the values of cards in player's hand is greater that the dealer's. (2) dealer wins if sum of the values of cards in player's hand is less that the dealer's. and (3) game is a tie if the sums are equal. In the case that sum of the values of dealer's cards is 16 or less, then the dealer must deal another card to itself. At this point, if the sum of the values of dealer's cards is 22 or more, then the player is the winner as the dealer is busted. Otherwise, dealer repeats step 2 with the new card.

2 Program Description

The incomplete program (BlackJack.java) is available on Canvas. In this program, the deck of 52 cards is represented by an array of 52 Strings declared as "deck". Declaration and initialization of the array happens in lines 32 and 33 of the BlackJack.java file: After line

```
32  String[] deck = new String[52];
33  initialize(deck); // initializes and shuffles the deck of cards. Don't delete!
```

33, the deck of cards are shuffled and ready for the game. Every element of the array *deck* contains a string of length two e.g. 2♠, J♣, A♢, etc. The first letter of each string specifies the value of the card and the second letter determines the suit (heart, diamond, spade, or club).

Game Flow

You need to write the rest of given program so that the user can play a complete Blackjack game as its single player while the computer plays the role of dealer in the game. Here is the sequence of events and flow of the game:

Phase 1: Program asks for the player's name. After the name is input via keyboard, the program must store the name in a String variable called "*playerName*" and call the player with the name for the rest of the game. Here is the code:

```
34  Scanner keyboard = new Scanner(System.in);
35  String playerName;
36  System.out.print(”Enter the player’s name: ”);
37  playerName = keyboard.nextLine();
```

Phase 2: Program deals two cards to the player and two cards to itself. These cards which are picked from the first four elements of the array *deck* will be printed on the screen except the first card of the dealer that needs to remain hidden till Phase 4. Also, the program will keep track of the player's score and dealer's score in two *int* variables called *playerScore* and *dealerScore*, both initialized to zero. The score of player is equal to sum of the values of cards dealt to the player and the score of dealer is sum of the values of cards dealt to the dealer. Here is how to implement this step:

```
38  int playerScore = 0;
39  int dealerScore = 0;
40  System.out.println(”Game starts... Dealer is dealing...”);
41  System.out.println(”First card for ” + playerName + ”: ” + deck[0]);//deck[0] for the player
42  playerScore += value(deck[0]);
43  System.out.println(”First card for dealer: Hidden”);//deck[1] for the dealer
44  dealerScore += value(deck[1]);
45  System.out.println(”Second card for ” + playerName + ”: ” + deck[2]);//deck[2] for the player
46  if(deck[2].startsWith(”A”) && playerScore > 10)//Ace’s value is 1 or 11
47      playerScore++;//Ace’s value is 1 here...
48  else
49      playerScore += value(deck[2]);//Ace’s value is 11 here...
50  if(playerScore == 21)//game ends early in favor of the player
51      System.out.println(playerName + ” won!”);
52  System.out.println(”Second card for dealer: ” + deck[3]);//deck[3] for the dealer
53  if(deck[3].startsWith(”A”) && dealerScore > 10)//Ace’s value is 1 or 11
54      dealerScore++;//Ace’s value is 1 here...
55  else
56      dealerScore += value(deck[3]);//Ace’s value is 11 here...
```

As you see in this code, whenever we need to know the value of a card, we use a statement like "*value(deck[i])*" where i is the index of card in the *deck* array; $i = 0, 1, 2, \ldots, 51$. Also, in lines 50 and 51, the program checks to see whether the *playerScore* has reached to the winning value 21. If so, player is the winner and game has ended.

Phase 3: In this step, the program must keep asking the user the following question in an infinite while loop: "*Hit or Stay?*". In the case that user inputs "stay" via keyboard, the program must break from the while loop and moves to Phase 4. However, in the case that user chooses to hit, the program must deal the next card to the player and adds the value of the new card with the variable “playerScore”. If *playerScore* reaches 21, program must end the game after printing *playerName + ” won!”*. If *playerScore* exceeds 21, program must end the game after prining *playerName + ” lost!”*. Otherwise, program remains in the while loop of Phase 3.

Phase 4: In this phase, the program reveals the dealer’s hidden card. Then, while *dealerScore* is less than 17, the program deals a new card to the dealer.

Phase 5: In the fifth and last phase of the program, there are three possibilities:

1. $dealerScore > 21$ or $dealerScore < playerScore$: game is ended in favor of player.
2. $dealerScore = playerScore$: game is ended in a tie.
3. $dealerScore < playerScore$: game is ended in favor of dealer.

3 Submissions

You need to submit a single file named “BlackJack.java”.

4 Grading Criteria

- Code readability: 5%
- Using comments to explain every line of the program: 5%
- Correctness of Java syntax (no compilation error): 5%
- Proper scan of System.in: 5%
- Proper error checking of user input: 5%
- Proper usage and naming of variables: 10%
- Proper traversal of “deck” array for dealing cards: 15%
- Proper usage of conditional and loop structures to implement the program’s logic: 30%
- Correct implementation of the game from start to end: 20%