COMP-1502

ASSIGNMENT #1

Dept. of Mathematics & Computing

Mount Royal University

Introduction

The intention of this assignment is to demonstrate skills in designing and implementing classes using Object-oriented programming paradigm. Additionally, it will give you experience working with testing tools, and creating/testing your own code as well.

Team work

- You MUST work with a partner on this project

Assignment Instructions

1. Use **only** Eclipse IDE (**Only** JDK 8 or 11).
2. The due date for this assignment is posted in D2L.

The Problem

In this assignment you will implement a game called Punto Banco. Your software will also save stats and balances for each player, and allow a user to generate reports based on this data. New players will be given a starting balance at their first game, and returning players will continue with their previous balance.

The program must have the following features:

1. Data Storage:

- Player information is stored in the text file `res/CasinoInfo.txt`
  - Each row of text stores data for one player, formatted as `"name,balance,numberOfWins"`
    - for example: `Khosro,100,21`
- If the text file does not exist at startup, your program must create it.
- Player data must be saved back to the text file when the program ends - that file serves as a sort of database.
- When the program starts, the player data in `res/CasinoInfo.txt` must be loaded into an `arrayList` of Player objects

2. Menus and Navigation

Note: The program should treat all user input as case-insensitive, and all menus should display an error message and re-prompt if an invalid input is entered.

- At startup, the program must present a main menu as follows:

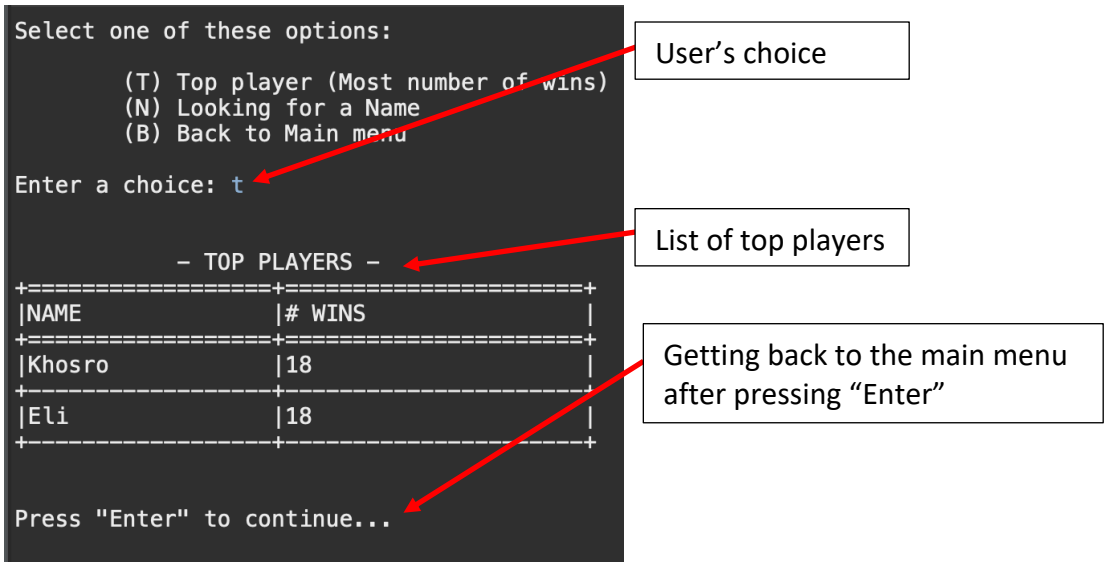
```
Select one of these options:

        (P) Play game
        (S) Search
        (E) Exit

Enter a choice:
```

- **(P) Play Game:** This option lunches the Blackjack game. The rules will be explained later.

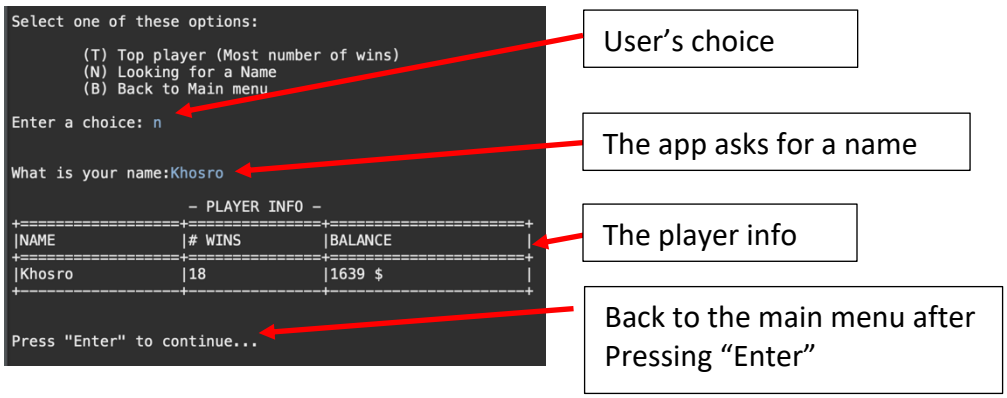
- **(S) Search:** This option shows a sub-menu to the user:

```
Select one of these options:

        (T) Top player (Most number of wins)
        (N) Looking for a Name
        (B) Back to Main menu

Enter a choice: |
```

  - (T) Top player: This option shows the player(s) with most wins and returns to the main menu after the user presses “Enter”:

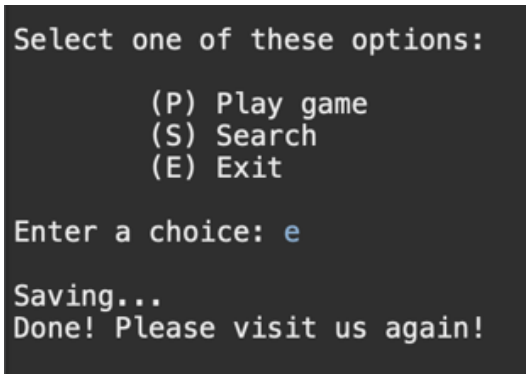


  - **(N) Search for a Name:** This option prompts the user for a name, and displays that player's information if they exist in the database. The user is informed if that player is not in the database. Returns to the main menu after pressing “**Enter**”.



  - **(B) Back to main menu:** This option will take the user to the main menu

- **(E) Exit:** This option in the main menu ends the program. All player data must be saved before the program ends.



BlackJack Game Rules

Blackjack, eequally well known as Twenty-One, is a casino game between a dealer and a player. The rules are simple, the play is thrilling, and there is opportunity for high strategy. In fact, for the expert player who mathematically plays a perfect game and is able to count cards, the odds are sometimes in that player's favor to win.

Object of the Game

Each participant attempts to beat the dealer by getting a count as close to 21 as possible, without going over 21.

Card Values/scoring

It is up to each individual player if an ace is worth 1 or 11. Face cards are 10 and any other card is its pip value (indicated number on the card).

Betting

Before the deal begins, the player places a bet. Minimum is $2 and the player must not be allowed to bet an amount exceeding their available funds.

Shuffling the cards

Before the game starts, the card deck must be shuffled. Additionally, once all the cards in the deck have been dealt, a new card deck should be generated.

The Deal

When the player has placed their bet, the dealer gives one card face up to the player, and then one card face up to themselves. Another round of cards is then dealt face up to the player, but the dealer takes the second card face down. Thus, the player receives two cards face up, and the dealer receives one card face up and one card face down.

Naturals

If the player's first two cards are an ace and a "ten-card" (a picture card or 10), giving a count of 21 in two cards, this is a natural or "blackjack." If any player has a natural and the dealer does not, the dealer immediately pays that player one and a half times the amount of their bet. If the dealer has a natural, they immediately collect the bets of the player if they do not have natural, (but no additional amount). If

the dealer and the player both have naturals, the bet of that player is a stand-off (a tie), and the player takes back his money.

The Play

The player goes first and must decide whether to "stand" (not ask for another card) or "hit" (ask for another card in an attempt to get closer to a count of 21, or even hit 21 exactly). Thus, the player may stand on the two cards originally dealt to them, or they may ask the dealer for additional cards, one at a time, until deciding to stand on the total (if it is 21 or under), or goes "bust" (if it is over 21). In the latter case, the player loses.

The combination of an ace with a card other than a ten-card is known as a "soft hand," because the player can count the ace as a 1 or 11, and either draw cards or not. For example with a "soft 17" (an ace and a 6), the total is 7 or 17. While a count of 17 is a good hand, the player may wish to draw for a higher total. If the draw creates a bust hand by counting the ace as an 11, the player simply counts the ace as a 1 and continues playing by standing or "hitting" (asking the dealer for additional cards, one at a time).

The Dealer's Play

When the dealer has served the player, the dealer's face-down card is turned up. If the total is 17 or more, it must stand. If the total is 16 or under, they must take a card. The dealer must continue to take cards until the total is 17 or more, at which point the dealer must stand. If the dealer has an ace, and counting it as 11 would bring the total to 17 or more (but not over 21), the dealer must count the ace as 11 and stand. The dealer's decisions, then, are automatic on all plays, whereas the player always has the option of taking one or more cards.

The player wins if the value of their hand is higher than the dealer's without going bust. There will be a tie, if both stand on the same number.

Card Deck: A fresh deck of 52 cards (represented as a software object) should be created and shuffled when the program starts. The same deck object should be used (even across games) until it runs out of cards, at which point it should be reset (all 52 cards restored and shuffled).

The Game User Interface

1. Upon launching the game (from the main menu) the user is asked for their name:
   - If the user already exists in the database, a welcome back message is shown as follows:

```
What is your name:khosro

**************************************************************************
***   Welcome back KHOSRO     ---    Your balance is: 1639 $          ***
**************************************************************************
```

   - If the user is new, we create a new account with initial balance of $100:

```
What is your name:John

**************************************************************************
***   Welcome JOHN       ---    Your initial balance is: 100  $  ***
**************************************************************************
```

- If the player has a balance of 0, they cannot play and are returned to the main menu.

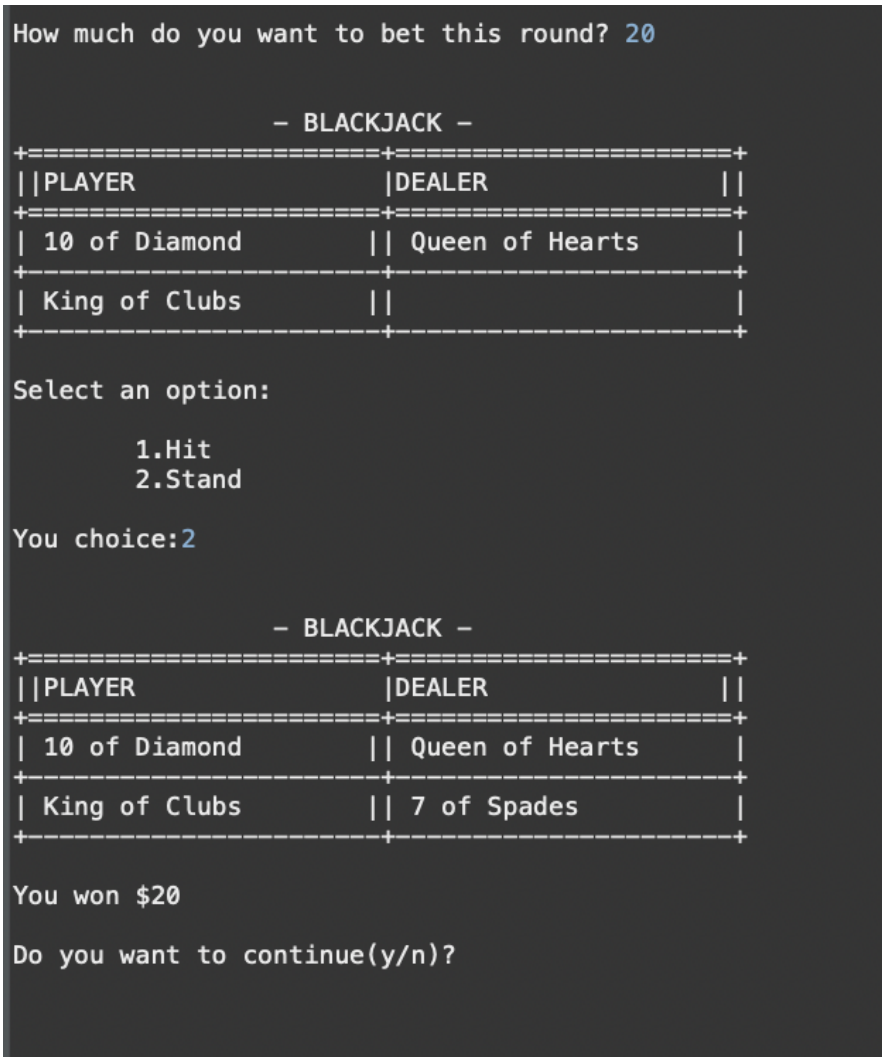
2. The user is then asked about the bet amount (min: $2 max: user's available funds):

```
How much do you want to bet this round?
```

3. The player cannot bet more than their current balance.
4. The game then plays out, the result determined, and the player's balance is adjusted accordingly.
5. The player is asked if they would like to play again, or return to the main menu. If they choose to play again, they return to the betting menu (step 2).

Two sample outputs of the game:

```
Select one of these options:

        (P) Play game
        (S) Search
        (E) Exit

Enter a choice: p

What is your name:Khosro

********************************************************************************
***   Welcome back KHOSRO      ---    Your balance is: 1800 $           ***
********************************************************************************


How much do you want to bet this round? 20


                - BLACKJACK -
+======================+======================+
||PLAYER                |DEALER                ||
+======================+======================+
| 7 of Diamond         || 8 of Hearts          |
+----------------------+----------------------+
| 8 of Spades          ||                      |
+----------------------+----------------------+

Select an option:

        1.Hit
        2.Stand

You choice:1


                - BLACKJACK -
+======================+======================+
||PLAYER                |DEALER                ||
+======================+======================+
| 7 of Diamond         || 8 of Hearts          |
+----------------------+----------------------+
| 8 of Spades          ||                      |
+----------------------+----------------------+
| 7 of Hearts          ||                      |
+----------------------+----------------------+

You lost $20

Do you want to continue(y/n)?
```



Automated Testing

We expect to see reasonable JUnit tests to ensure the behavior of the existing classes (Card, Deck, Player, …) is as expected. Few example, we would expect your tests to:

- Ensure you can create a card and get its suit, rank and correct string representation (especially for Jack, Queen, King, Ace)
- Ensure you can create a hand of cards, add cards to it, and compute its score
- Create a new deck of cards and shuffle it (and get a different arrangement of cards each time)
  - Does the deck behave as expected when the last card is drawn?
- Additionally, any public method you create which does not require user input to function should also have a JUnit test.

Runnable Jar file

- Generate a runnable jar for your project (inside the project folder)

Documentation

- Javadoc files must be generated in the `doc` project folder.

- Reasonable in-line comments (single or multi-line) should be provided

Starter Code

- You have been provided a starter code (skeleton) with the basic project structure and some of the classes' code. Complete the code to meet the requirements above. You can add new classes & code as necessary, but you must not change the basic structure. All classes and methods in the starter code must work as described in the starter code.

Submission Instruction

1. Zip the project folder and submit it into D2L
2. The contribution of each team member must be visible. You can use either GitHub(recommended) or submit a one page task sheet in MSWords.
3. You **MUST** demo your project to your instructor during a demonstration to be scheduled during tutorial/lab time. This demo is mandatory. **No demo = a grade of 0**

Grading

The following items are prerequisite to having your project graded. Failure to satisfy any of these points could result in a **grade of zero**.

- Code must compile and run
- Basic project structure follows starter code
- Clearly marked document in repository outlines each team member's contribution

Projects that satisfy the above criteria will be graded according to the following rubric:

	The Application	**Points**	**Grade**
5.	Runnable JAR file	3	
6.	Generate proper documentation (JavaDoc & Commenting)	7	
7.	All of the inputs are validated	5	
8.	Follow object-oriented structure	10	
9.	Follow naming conventions	5	
11.	Main Menu design	5	
12.	Sub-Menu design	5	
13.	Game runs properly	15	
14.	Top-player search works properly	5	
15.	Search name works properly	5	
16.	The program returns properly from the sub-menu to the main menu	5	
17.	The program saves the record properly into the text file before exiting	10	
18.	Proper use of ArrayLists	10	
19.	Designing proper unit tests	10	
Total		**100**	