6.178 - Problem Set 3

Blackjack!

You will be implementing various methods to make a working Blackjack game! In particular, you will be expanding `Card.java` and `Hand.java` to implement the interface in `BlackjackValuable.java`, by leveraging the utilities you wrote in PSet 1. Then, you will use the other implementations you wrote in PSet 1 to finish `Hand.java` and `Deck.java`. Once you have a working Card/Deck/Hand system, you will use them to finish the game logic in `BlackjackGame.java`. Finally, you will apply your knowledge of File I/O to complete the statistics logging system in `Logfile.java`.

Do not change method signatures (the names of methods and their parameters), or your code will not pass the tests.

Rules of 6.178 Blackjack

In Blackjack, the player first makes a bet. Then, the dealer deals two cards to the player and then two cards to his or herself (the dealer). The player can see both of his or her cards, but can only see the dealer's second card. The player then chooses to either hit (have another card dealt to them) or stand (take no more cards). This process repeats until the player either busts or stands. Busting is explained later on.

To calculate the value of any given hand, the value is counted by adding up the values of each card in that hand. Each numeric card is worth the value shown on that card. Each face card is worth ten points. An ace can either be worth one or eleven points, whichever benefits the holder of that card. These should be familiar from problem set 1.

If at any point the value of the player's hand exceeds 21, the player busts, and loses immediately. The player also loses his or her bet.

If at any point the value of the player's hand equals 21, the player has hit blackjack, and wins immediately. The play keeps his or her bet and also gains 1.5x that bet as his or her winnings. (Note that if the dealer gets blackjack in the first hand, the game should say so and the player should lose their initial bet immediately.)

In two player Blackjack, the hand ends when the player stands. Once the player stands, the dealer reveals the value of his hidden card. The dealer then plays according to the following strategy:

- If the value of the dealer's hand is less than 17, the dealer hits.
- If the value of the dealer's hand is greater than or equal to 17, the dealer stands.

If the value of the player's hand is greater than the value of the dealer's hand, the player wins, keeps his or her bet and gains an additional amount equal to their bet.

If the value of the player's hand is less than the value of the dealer's hand, the player loses and loses

his or her bet.

If the value of the player's hand is equal to the value of the dealer's hand, the hand is a draw, and the player keeps his or her bet with no additional winnings (the hand does not change the amount of money a player has).

1 Finish Deck.java

This section reviews content you already know in the context of Object Oriented Programming. You should be able to complete this section after Lecture 4.

`Deck.java` contains an unimplemented method, `drawInitialHand`, similar to the one seen in `BlackjackUtils.java` in Problem Set 1.

`drawInitialHand` draws an initial Blackjack hand from the top of the `Deck` and returns that `Hand`. It takes in a single argument `isFirstCardHidden` which is true if the first card of the resulting `Hand` should be hidden in the `toString()` function of that `Hand`.

Read the method's documentation for more information.

2 Implement the BlackjackValuable interface for `Card` and `Hand`

You should be able to complete this section after Lecture 6

In order to play Blackjack, we need to know how much a given `Hand` or Card is worth in Blackjack. We also need to be able to compare two given `Hand`s or `Card`s. So that we can achieve these objectives, we have developed a `BlackjackValuable` interface found in `BlackjackValuable.java`.

HINT: If you're struggling trying to turn a `Rank` into its Blackjack value, you should take a look at Java's `Enum.ordinal()` function. Its Javadoc can be found here: `https://docs.oracle.com/javase/8/docs/api/java/lang/Enum.html#ordinal--`

You should read all applicable method documentation and implement this interface for `Card.java` and `Hand.java`.

3 Finish Hand.java

You should be able to complete this section after Lecture 6, because it depends on the BlackjackValuable interface.

`Hand.java` contains two unimplemented methods, `getHandAfterHit` and `getHandAfterDealerDraw`.

`getHandAfterHit` takes in a `Deck` and returns the `Hand` that would result if a Blackjack player hit from that deck while holding the `Hand` it is called upon.

`getHandAfterDealerDraw` simulates the behavior of a Blackjack dealer drawing from the `Deck` that is passed as an argument while holding that `Hand`. It should return the a `Hand` representing the hand resulting from that drawing.

Read the methods' documentations for more information.

4 Finish BlackjackGame.java

This section contains material from Lecture 5, but you may not be able to complete it until after Lecture 6, because it depends on the BlackjackValuable interface.

`BlackjackGame.java` contains 5 unimplemented methods:

`doesPlayerHaveBlackjack` returns true if the current state of the game dictates that the player has a Blackjack (the value of the player's hand is equal to 21 exactly). Otherwise, it returns false.

`doesDealerHaveBlackjack` returns true if the current state of the game dictates that the dealer has a Blackjack (the value of the dealer's hand is equal to 21 exactly). Otherwise, it returns false.

`isPlayerBusted` returns true if the current state of the game dictates the player is busted (the value of the player's hand is greater than 21). Otherwise, it returns false.

`isDealerBusted` returns true if the current state of the game dictates the dealer is busted (the value of the dealer's hand is greater than 21). Otherwise, it returns false.

`determineWinner` updates the state of the game as necessary to determine a winner at the end of a `Hand`. The method then returns the proper `HandResult` indicating who won and if they won with a Blackjack.

From any state of a Blackjack game, the winner is determined in the following manner:

- If both the dealer and player have Blackjack, or both parties are busted, the hand is a draw.
- If only one party has Blackjack, that party wins.
- If either party is busted, that party loses.
- If there is no winner yet, the dealer draws.
- After the dealer draws, the same reasoning is applied as above to determine the winner.
- If no winner is determined yet, the party with the hand Blackjack value closest to 21 without exceeding 21 wins. If both parties have the same hand Blackjack value, the hand is a draw.

Read the methods' documentation for more information.

5 Implement Logfile.java

You should be able to complete this section after Lecture 7

We would like to keep some statistics on how the player does across all of their game sessions. To do this, we plan to write the results of each hand to a file.

`Logfile.java` represents a file on disk in the current working directory named "logfile.txt" that these results are written to as single textual lines. These lines are seperated from each other in the log file by a newline character.

`Logfile.java` contains 2 unimplemented constructors and 3 unimplemented methods:

`Logfile()` is the actual Logfile constructor. It creates a new logfile that can read and write from "logfile.txt" in the current project directory.

`Logfile(String logfileName)` is a constructor used for testing purposes. It initializes a Logfile object that interacts with a file other than "logfile.txt".

`appendLine()` takes in a line as a String and writes that line to the end of the Logfile while ensuring that all lines are separated by a newline character.

`overwriteWith()` takes in a `String` and overwrites all of the contents of the `Logfile` with that string.

`readLines()` returns all lines currently contained in the `Logfile` as an ordered `List`.

Read the methods' and constructors' documentation for more information.

6 Play Blackjack!

Congratulations! At this point you should have a working Blackjack game!

In order to play the game, run the `Main.java` file in the `io` package.