Poker

For our next project, we will be implementing a version of the card game Poker. The object of the game is to have the “best” hand for each round. The game starts by dealing 5 cards to each of two players (you can add more players later if you wish!). Each player has one opportunity to replace one or more cards from their hand to try and make a better hand. Then, the player's compare hands and the winner is declared. Here is the list of the possible combinations a player may have in their hand.

The project will be broken up into several parts to keep it from being too overwhelming. Each part will have its own requirements and build upon the work done in prior part(s). Each part will also have its own assignment in Canvas for submission. You should complete the parts IN ORDER. When one part has been submitted, begin work on the next part immediately using your code from the submission as your starting point for the next part.

You have been provided a zip file with some starter code: GameOfPoker2021Starter. This file contains:

1. Complete Card and Deck classes. These should not need to be changed/modified.
2. A partial Hand class. You will be adding a number of additional variables and methods to this class throughout the phases of the project
3. Tester files for the first two phases of the project

Phase 1: Basic Hand methods, first set of Poker Hand Checks

For this phase, we will begin building up the Hand class by adding a new instance variable and several methods to control the Hand. We will also begin to write the methods we will use to check our Cards for various types of Poker hands.

- In addition to the existing instance variables, create an integer array named `totalNumber`. Update the constructor to instantiate this array so that it can hold 15 values. This array will be used to keep track how many of each `Card` type is in the hand.

- Update the `addCard` method as follows
    - When `Cards` are added to the hand, they should be added IN ORDER based on their values from low to high. For example, if the `Cards` 7, 4, 8, Jack and 2 are added to the `Hand`, the `Cards` should end up in the order 2, 4, 7, 8, Jack
    - As each `Card` is added to the `Hand`, update the `totalNumber` array as follows. Look at the value of the `Card`. Increment the count of the array at that index. For example, if the `Card` being added is the Six of Clubs, totalNumber[6] should be incremented. Note that totalNumber[0] and totalNumber[1] will be UNUSED (always = 0).

- Write the `discardCard` method. This method takes an integer input and removes the `Card` at that index from the `Hand`. The method should return the removed `Card`. In addition, the totalNumber array needs to be updated. For example if the Three of Spaces was to be removed, totalNumber[3] would need to be decremented.

- Write the `checkRoyal` method. This method takes no parameters. It returns `true` if the `Hand` contains one Ten, one Jack, one Queen, one King and one Ace. It returns `false` otherwise.

- Write the `checkFlush` method. This method takes no parameters. It returns `true` if all of the `Cards` in the `Hand` are the same `suit`. Returns `false` otherwise.

- Write `CheckStraight` method. This method takes no parameters. It returns `true` if the `Cards` in the `Hand` are in sequential order. For example, 7, 8, 9, 10, Jack. Returns `false` otherwise.

Make sure that the Phase1Tester passes all tests prior to submitting Phase 1. Once this part is complete, proceed immediately to Phase 2.

Phase 2: Complete the Hand Class

In this phase, we will write the methods to check for the remaining types of poker hands and create the method to determine the best possible hand we can make based on the `Cards` in our `Hand`.

- Write the `printHand` method. This method takes no parameters. Prints the `Cards` in the `Hand` in as a numbered list. For example, output might look like this for a typical `Hand`:
    1. Two of Hearts
    2. Four of Spades
    3. Four of Diamonds
    4. Seven of Diamonds
    5. Jack of Spades

- Write the `numberOfAKind` method. This method takes no parameters. Returns an integer that represents the largest number of `Cards` with the same `rank` in the `Hand`. For example, if the `Hand` held: 2, 2, 2, 7, 7, the method would return 3 since there are three 2's. (Hint: use the `totalNumber` array from Phase 1!). Eventually, we can use this method to check for four of a kind, three of a kind, and pairs.

- Write the `checkFullHouse` method. This method takes no parameters. Returns `true` if the `Hand` contains three `Cards` of one `rank` and two `Cards` of another `rank`. Returns `false` otherwise. You might find the totalNumber array helpful here.

- Write the `checkTwoPair` method. This method takes no parameters and returns `true` if the hand contains two `Cards` of one `rank` and two `Cards` of another `rank` with the last `Card` matching nothing. Returns `false` otherwise. Again, you might find the totalNumber array helpful here.

- Write the `checkBestHand` method. This method takes no parameters. Returns an integer that represents the best poker hand that can be made from the `Cards` in the `Hand`. Refer to the table at the top of this document for a list of hands and their relative rankings. A royal flush, the best possible hand, would return a 1. A straight flush would return a 2, etc. In addition, a message should be printed to indicate the type of hand that was found. For example "You have a flush!" or "You have a pair." You will need to make use of the other methods written in Phases 1 and 2 to make this method work.

Make sure that the Phase2Tester passes all tests prior to submitting Phase 2. Once this part is complete, proceed immediately to Phase 3.

Phase 3: The Game Class

The `Game` class is where control of your overall game occurs. This will be your main method.

Step 1: Set up the game

Before the game can start, you will need to set everything up for the players. In this case, you will have two players, so we will need two `Hands`. Here are the things your main method will need to do as part of the setup process:

1. Create a `Deck` of `Cards`
2. Shuffle the Deck and print the message "Shuffling…" so the user knows what is happening.
3. Create two `Hands` (one for each player)
4. Print a message: "Dealing…" and then deal five `Cards` to each player alternating between the players.
5. When the setup is complete, print the message: "Ready! Hit Enter to continue." Use a Scanner and the nextLine() command to cause the program to wait for the user to hit Enter.

Step 2: Playing the hand

Each player will look at the hand they have been dealt (in step 1 above) and decide if they want to replace any of their cards. If so, they will select the cards to remove from their hand and new cards will be dealt to them to bring their `Hand` back up to a total of five `Cards`. After each player has made their choices, the two hands are compared and the winner declared. Here are the basic steps you will need to follow:

1. Print one of the player's `Hands`.
2. Ask the player how many `Cards` they would like to discard and read in their response.
3. Ask the player to enter the number of a `Card` they want to discard (from the numbered list) and read in their response.
4. Remove the selected `Card` from the `Hand` and reprint the remaining `Cards`.
5. Repeat steps 3 and 4 until the desired number of `Cards` have been discarded.
6. Deal new `Cards` to the player to bring their `Hand` total back up to five `Cards`.
7. Repeat steps 1 - 6 above for the second player.
8. Once both players have discarded and drawn new Cards, print both hands. Then, compare the two player's `Hands` to determine the winner. Print a message that indicates what type of hand each player has and then print a message indicating who has the best hand.

Note: This basic game might end in a tie. The tiebreaker is NOT part of this project. You are certainly welcome to add it on your own however!

You must test the game ON YOUR OWN. There is no automated tester. Run your program MULTIPLE times to ensure it works before submitting!

A sample of the output of the program for one round of play is shown below.

```
Shuffling...
Dealing...
--------------------------------
Player 1's Hand
1. Ten of Clubs
2. Jack of Diamonds
3. Jack of Clubs
4. King of Clubs
5. Ace of Clubs
--------------------------------
Player 1 - How many cards do you want to discard? 1
Which card do you want to discard? (1,2,3,4,5)2
1. Ten of Clubs
2. Jack of Clubs
3. King of Clubs
4. Ace of Clubs
--------------------------------
Player 2's Hand
1. Three of Clubs
2. Four of Spades
3. Eight of Diamonds
4. Eight of Hearts
5. Nine of Diamonds
--------------------------------
Player 2 - How many cards do you want to discard? 3
Which card do you want to discard? (1,2,3,4,5)1
1. Four of Spades
2. Eight of Diamonds
3. Eight of Hearts
4. Nine of Diamonds
Which card do you want to discard? (1,2,3,4,5)1
1. Eight of Diamonds
2. Eight of Hearts
3. Nine of Diamonds
Which card do you want to discard? (1,2,3,4,5)3
1. Eight of Diamonds
2. Eight of Hearts
Hit Enter to Continue:
```

```
--------------------------------
Player 1            Player 2
Five of Clubs       Six of Diamonds
Ten of Clubs        Eight of Diamonds
Jack of Clubs       Eight of Hearts
King of Clubs       Jack of Spades
Ace of Clubs        King of Spades
--------------------------------
Player 1's Results
You have a flush!
--------------------------------
Player 2's Results
You have a pair.

And the winner is:
--------------------------------
Player 1 wins!
```