CS101- Algorithms and Programming I

Lab 10

Lab Objectives: `Arrays`

- For all labs in CS 101, your solutions must conform to the CS101 style guidelines (rules!)

1. Create a project, Lab10. You will create an application to play the game Blackjack / Twenty-One. Blackjack uses a deck of cards, and the player and dealer each start with 2 cards. Both try to get as close to 21 without going over.

   Your project should include the following classes and functionality. I recommend that you test each class independently after creating to make sure they are working as expected.

2. **Class: Card**
   - Download this class from Moodle, you should not make any changes to this class.
   - Explore the class to understand the functionality.
   - Each Card has a value from 1 – 13. Special cards have the following values:
     - Ace: 1
     - Jack: 11
     - Queen: 12
     - King:13
   - The following values represent each suit:
     - Hearts♥: 1
     - Diamonds♦: 2
     - Spades ♠ : 3
     - Clubs ♣: 4
2. **Class: Deck**
   - Represents a Deck of 52 cards.
   - **Data Members:**
     - `deck`: *array* of Cards (52)
     - `topCard`: index of the first unplayed Card in the deck.
   - **Methods**:
     - Constructor:
       - Creates the deck and initializes 52 Cards in the deck using the `initialize()` method.
       - Shuffles the deck using the `shuffle()` method.
       - Initializes the `topCard` to zero.
     - `initialize()`: Creates a Card to represent each value (1-13) for each suit (1-4) and stores them in the deck.
     - `shuffle()`: 'unsorts' the deck. You may choose your own algorithm, or you may swap 2 randomly selected cards repeatedly (example: 1000 swaps). Use java.util.Random to create a random number generator. The nextInt(int bound) will generate a random integer value between zero and the bound, not inclusive of the bound.
     - `dealCard()`: returns the top card and updates `topCard`.
     - `toString()`: returns a String which stores the String representation of all Cards in the deck, each on their own line.

3. **Class: Game**
    - Represents a BlackJack (21) card game.
    - **Data Members:**
        - `gameDeck`: Deck of Cards.
        - `playerHand`: array of Cards, max length of 11.
        - `dealerHand`: array of Cards, max length of 11.
        - `playerCards`: current number of Cards in the Player's hand
        - `dealerCards`: current number of Cards in the Dealer's hand
    - **Methods**:
        - Constructor:
            - Initializes the `gameDeck`.
            - Sets the game by initializing the `playerHand` and `dealerHand` to empty arrays, `playerCards` and `dealerCards` to zero.
        - `printHand()`: prints the appropriate hand (player or dealer), should not print null array elements (hand will usually not have all 11 Cards).
        - `determineResult()`**:** determines and displays the result of the game according to the rules of play.
        - `play()`: Implement the play method according to the rules below. You may use helper methods (recommended) to implement some of the functionality. **Note:** the rules below are simplified.

          BlackJack / Twenty-One Rules:

            - The goal of blackjack is to beat the dealer's hand without going over 21.
            - Face cards (Jack, Queen, King) are worth 10. Aces are worth 1.
            - The player and the dealer both are dealt two cards, the dealer's cards are hidden until their turn.
            - The player may 'hit' and ask for another card or 'stay' and end their turn.
            - If the player goes over 21 he/she busts, and the dealer wins regardless of their hand.
            - Once the player's turn has ended (and they have not gone over 21) the dealer will hit until their cards total 17 or higher.
            - Once the dealer stays or busts, the game is over.
            - If the player and dealer have the same total, the dealer wins.
            - See sample run below for the game play.

4. **Class: GameApp**
    - `main` method does the following:
        - creates a BlackJack `Game` and `play()`**.** When the game is over, displays the result.
        - After each `Game` is over prompt if the user wants to play a *new* `Game` or quit, until the user chooses to quit.

Sample Game:

```
Your hand:
Ace of Hearts
King of Diamonds

(1)Hit or (2)Stay: 1
Your hand:
Ace of Hearts
King of Diamonds
Queen of Clubs
BlackJack!!

Dealer hand:
Jack of Diamonds
8 of Hearts
Player:21 Dealer:18
Congrats - You win!
```

```
Play again? (y/n): y
Your hand:
6 of Diamonds
Ace of Diamonds

(1)Hit or (2)Stay: 1
Your hand:
6 of Diamonds
Ace of Diamonds
Jack of Hearts

(1)Hit or (2)Stay: 2
Dealer hand:
Queen of Clubs
Queen of Diamonds
Player:17 Dealer:20
Sorry - you lose

Play again? (y/n): y
Your hand:
Queen of Clubs
2 of Clubs

(1)Hit or (2)Stay: 1
Your hand:
Queen of Clubs
2 of Clubs
8 of Clubs

(1)Hit or (2)Stay: 2
Dealer hand:
10 of Spades
Queen of Hearts
Player:20 Dealer:20
Sorry - you lose

Play again? (y/n): y
Your hand:
King of Diamonds
Jack of Hearts

(1)Hit or (2)Stay: 2
Dealer hand:
Ace of Spades
4 of Diamonds
10 of Spades
Jack of Spades
Player:20 Dealer:25
Congrats - You win!

Play again? (y/n): n
Thanks for playing - Goodbye!
```