CSC 210, Spring 2018
Blackjack Program (40 points)

The object of the game Blackjack, or 21, is to get as close to 21 points without getting more than 21. The game is played with a standard deck of 52 cards, of Ace, 2 – 10, J, Q, K in 4 suits each (hearts, spades, clubs, and diamonds). In Blackjack, face cards (J,Q,K) are worth 10 points, aces are worth either 1 or 11 points, and other cards are worth their numeric value.

We will play as follows: the player and dealer are each dealt two cards. The player can see both of his/her cards, while only the dealer's top card is visible. During the player's turn, the player can *hit*, in which case they are dealt the next card. A player *stands* when the player is done with his/her turn. If the player's cards total more than 21 points, the player loses. If the player stands with less than 21 points, then the dealer keeps hitting until reaching more than 17 points. If the dealer goes over 21 (i.e., *busts*), then the player wins. If the player is dealt a blackjack (an ace and a 10-K), the player wins, unless the dealer also has blackjack (in which case the game ends in a draw).

The player bets $2 each time they play. If a player wins with Blackjack, they get $5 (they get back their $2 bet plus $3 more). If a player wins otherwise, the player gets $4. If it is a draw (both the player and dealer have equal hands and have not busted), the player gets back the $2. In all other cases, the player loses $2.

Note that for simplicity, we will not consider certain strategies, including taking insurance (if the dealer has an ace showing), or doubling down (doubling your bet and receiving only one more card).

You may work in teams of up to 3 people on this assignment (see next page)

Assignment:

1. Write a Blackjack program that allows the player to play against a dealer (the computer). Your program must include the following methods (parameters are not shown):

    a. welcome() – a method that displays a welcome message
    b. shuffle() – a method that shuffles a deck of cards
    c. calcHand() – a method that calculates and returns the value of a hand (remember that aces are worth 2 points – this value should be chosen based on which produces the best hand).
    d. promptPlayer() – prompts the player for whether he/she wants to Hit or Stand, and returns this value.
    e. hit() – deals a card to a given hand
    f. compareHands() – determines whether the player or computer wins, or whether it is a tie, and returns the value (such as 'P', 'C', or 'T')
    g. displayCards() – displays the user's or dealer's cards. For the dealer, an 'X' is used for the facedown card.
    h. playAgain() – prompts the player for whether he/she wants to play again, and *returns* true if the user wants to play again, or false otherwise
    i. additional methods are recommended

2. Write a separate unit testing program that tests all methods that return a value, except for those mentioned below. Note that unit tests need to be written to cover all branches and types of possibilities. Methods that prompt the user to enter a value do not need to be unit tested.

3. Your program must be user-friendly and provide appropriate feedback, such as who the winner of each game is, and the amount of money the player has (the player starts with $20). After each game, the player is prompted for whether he/she wants to play again, and another game is played until the player wishes to quit.

4. The deck of cards as well as the player's and dealer's hands must be stored using one of the following:
    - A one-dimensional array for the card values (1-13) and another for the suit ('H', 'C', 'D', 'S')
    - A two-dimensional array, with 1 dimension for the card values (1-13) and another for the suit ('H', 'C', 'D', 'S')

5. Your random number generator should be declared at the top of your class, and not in any method.

6. The top of your program must include the authors as well as a brief description. Each method must include a description explaining the parameters, what the method does, and what value is returned, if any.

Submission instructions: Submit your program (the .java files) through Blackboard.