Programming Assignment #7: Let's Play Blackjack!

Due Date: November 23, 1999

1 The Problem

In this program, you will use *four* classes to write a (very simplified) Blackjack game. Blackjack is a popular card game.

The game is played between two players as follows. Each player takes three cards from the deck. They then add up the points in their respective hands. The point value of a card whose denomination is between two and ten is simply the denomination of the card. For example, the point value of the 'four of clubs' is four points. The point value of a face card (i.e., Jack, Queen or King) is ten points, and the point value of an Ace is eleven points. The player whose point total is closest to 21 *without going over* is the winner. If both players exceed 21 points or if they have the same point value, there is no winner. In addition, if exactly one of the players exceeds 21, the other player is the winner.

2 The Classes

As mentioned above, four classes will be used to write the Blackjack game.

- `class Card`: This class describes the structure of a playing card. *It is already written for you.*
- `class Deck`: Exactly 52 card objects are assembled in an *array* to describe a deck of cards. *This class is written for you.*
- `class Player`: A player can be described by the *hand* he or she is holding. *You will have to write some of the methods in this class yourself.*

- `class BlackJack`: This class is where method `main` is located and is the class that plays a simplified game of Blackjack. *You will write most of this class yourself.*

2.1 `class Card`

We have already seen this class on one of the practice tests.

```
public class Card {

   private int       denom;  //denomination
                     suit;   //suit

   //constructors
                     ...
   //methods
                     ...
}
```

The instance variable `denom` can take on the values from 2 to 14. A Jack has `denom` equal to 11, a Queen has `denom` equal to 12 and a King has `denom` equal to 13. An Ace has `denom` 14. Similarly, the instance variable `suit` can take on values between 0 and 3. Hearts have `suit` equal to 0, diamonds have `suit` equal to 1, clubs have `suit` equal to 2 and spades have `suit` equal to 3.

There is *one* constructor and *five* methods.

- `public Card() {...}` //This constructor creates an initialized Card object.
- `public void displayCard() {...}` //This method displays the calling card in the traditional way, for example, 'ace of hearts', 'two of diamonds', etc.
- `public int getDenom() {...}` //This method returns the value of the calling card.
- `public void setDenom(int v) {...}` //This method sets the value of the calling card to v.

- `public int getSuit() {...}` //This method returns the suit of the calling card.
- `public void setSuit(int st) {...}` //This method sets the suit of the calling card to st.

2.2 `class Deck`

The purpose of this class is to describe the structure of a deck of card objects.

```
public class Deck {

   private Card[]    deckOfCards;  //a deck of cards
   private int       topCardPos;   //position of top of deck

   //constructors
                     ...
   //methods
                     ...
}
```

The instance variable `deckofCards` is a reference to an array of `Card` objects. (When the constructor is called for this class, an array of 52 `Card` objects will actually be created.) The variable `topCardPos` will have a value from 0 to 51 and represents the current 'top of the deck' when a game of Blackjack is in progress. For example, when the game begins, the value of `topCardPos` will be 0, but after a player has picked a card off the top of the deck, the value of `topCardPos` will be changed to 1, indicating that the card in position one is now the new 'top of the deck'.

There is *one* constructor and *three* methods.

- `public Deck() {...}` //This constructor creates a deck of cards (unshu- ed) and sets the position of the top card to 0.
- `public void shuffle(int n) {...}` //This method shu- es the calling deck `n` times and sets the position of the top card to 0.
- `public int getTopCardPos() {...}` //This method returns the position of the top card.

- `public Card getTopCard() {...}` //This method returns the top card on the calling deck and increments `topCardPos`.

2.3 `class Player`

This class describes the structure of a player.

```
public class Player {

   private Card[]      hand;        //a hand of cards
   private int         numOfCards;  //number of cards currently
                                    //in the hand

   //constructors
                       ...
   //methods
                       ...
}
```

The instance variable `hand` is a reference to an array of `Card` objects representing a hand of cards. (When the constructor is called for this class, an array of up to 20 `Card` objects will actually be created. For our Blackjack game, we will only use hands of size three!) The variable `numOfCards` will denote the number of cards the player currently has in his or her hand. For example, when the game begins, the player has *no* cards. However, after he or she has picked a card from the top of the deck, then the player has *one* card, etc.

The class has *one* constructor and *four* methods.

- `public Player() {...}` //This constructor creates a hand of cards and sets the number of cards in the hand to 0.
- `public void takeACard(Deck d) {...}` //This method causes the calling player to take one card from the top of the deck `d` and insert the card at position `numOfCards` in his or her hand. The value of `numOfCards` is then updated. This method should call the method `getTopCard` of the `Deck` class.
- `public int computePoints() {...}` //This method causes the calling player to compute and return the point value of his or her hand.

- `public void showHand() {...}` //This method will cause the calling player to print out his or her hand and the point value of the hand.
- `public void discardHand() {...}` //This method causes the calling player to discard his or her hand.

2.4 `class BlackJack`

This class plays a simplified game of Blackjack.

```
public class BlackJack {

   //methods
                              ...
}
```

In addition to the main method, a static method will be used to decide which of the two players (if anyone) has won the game.

- `public static int whoWins(Player p1, Player p2) {...}` // This method decides whether player 1 or player 2 has won the game. The method returns 1 if player 1 wins and returns 2 if player 2 wins. Otherwise, it returns 3 (in case of a tie or if both players are over 21).
- `public static void main(String[] args) {...}` //The main method plays a game of Blackjack.

The main method should do the following tasks and then terminate. Basically, all the main method does is cause two players two choose three cards from the top of the deck, compares the values of their hands and decides who wins the game.

- Shu- e the deck 100 times before playing. To shu- e the deck, the program uses a *random number generator. The instructor has coded this for you.* All you have to do is call the method `shuffle(int n)` on the deck object with `n = 100`.
- Draw *three* cards from the top of the deck and place them in the hand of player 1.

- Draw *three* cards from the top of the deck and place them in the hand of player 2.
- Print out the hand of player 1 and the point value of the hand.
- Print out the hand of player 2 and the point value of the hand.
- Print out the winner of the game.

3 Program Skeleton

Program skeletons for all the classes described above are available for your use. Dr. Del Greco strongly encourages you to take advantage of these (partially coded) classes. Please see `h:\prof_pub\jdg\c170f99\ProgramSkeletons\Blackjack`. This is a directory that contains the `.java` files for the four classes you will need. *As mentioned before, the* `Card` *class and the* `Deck` *class are written for you.* Do not be concerned if you do not need all the methods in the classes to write your program. Some of them (like method `discardHand` in the `Player` class) can be used if *multiple games are to be played!* Dr. Del Greco will demonstrate such a program in class.

4 What To Turn In

Please submit your `.java` source files in `e:\submit\DelGreco\Comp170\Blackjack(Prog#7)\<yourFolder>`. There will be four files in all: `Card.java`, `Deck.java`, `Player.java`, and `BlackJack.java` (the method `main` will be contained in this file).

5 Late Program Policy

A program will loose 10% of its value each day it is late (excluding weekends). *Starting early on your programs will maximize your chances of earning full credit on your work!*