Due Date: Friday, October 13th @ 11:59pm

Points: 100

This is an individual assignment.

**Objective: In this lab assignment you will be implementing positional lists and iterators.**

**Create a NetBeans project named Lab107 and ensure it is saved to a location like desktop or your flash drive.**

**In completing this assignment, you must include and use the interfaces and classes for the positional list and iterator ADTs presented in the textbook. You must write the classes for any additional ADTs that you use using the textbook code. You cannot import any ADTs from the Java library.**

**You may use all or part of Iterator Example 1 posted on Blackboard.**

**You do not need to comment any unaltered textbook code or any of the unaltered code from Iterator Example 1.**

**In the project you will do the following (not listed in order):**

1. Complete assignment P-7.60 on page 305 of the textbook that implements the **CardHand** class with **the additional requirements as listed**. When arranging your cards:
   1. you should not worry about the relative values of the suites (i.e. does a Heart come before a Diamond).
   2. Rather you should arrange the suits so that the first suit you are dealt is to the “left” (first) and additional suits are added to the “right” (later) as they are received.
   3. Within a suit you so not need to worry about relative ranks of the cards. Just group all of the cards for a given suit together.
   4. Implement your CardHand class so that adding a new card can be done in constant time, O( 1 ).
   5. **You must use a LinkedPositionalList to represent the card hand.**
   6. **You must use the “four fingers” method suggested in the textbook.**
   7. **You do NOT need to implement the play(s) method described in the text.**
   8. **You DO need to implement the addCard(r,s) [ you can implement an addCard( card ) instead ], iterator() and suitIterator(s) methods described in the text.**
2. Implement a **Card** class that represents a playing card.
   1. A card has a suit ( Club, Diamond, Heart, or Spade )
   2. A card has a value ( 2 through 10, Jack, Queen, King, or Ace) with 2 being the lowest value and Ace being the highest value.
   3. You do not need to support Jokers
3. Implement a **Deck** class that represents a deck of cards. This Deck class must include the following methods:
   1. **Deck( )**: the constructor that creates a deck of 52 cards, one of each possible suit-value combination.
   2. **card( )**: deals a random card from the deck, where: 1) cards are dealt on a non-replacement basis (i.e. once a card is dealt it cannot be dealt again), and 2) on each deal all cards (remaining) in the deck have an equal probability of being dealt.
   3. Selecting the correct data structure should make this easy to implement.
4. Implement a **Game** class that represents a card game. This Game class must include:
   1. An instance variable that represents the number of players in the game (should be passed in as a parameter).
   2. An instance variable that represents the maximum number of cards that can be in each player’s hand (should be passed in as a parameter).
   3. An instance variable for the deck of cards used in the game.
   4. An instance variable that is an array of the player’s card hands.
   5. A method **getCard()** that deals one card to each player in the array of players (item c). When a player gets a new card they should **immediately add** this card to their hand so that they are ordered correctly by suit and value (item 1).
   6. Note that the player must immediately insert each new card into the correct position in their hand. The player cannot wait until they have received all of their cards and then sort the cards.
5. A client that tests your classes by dealing a card game. Your client should:
   1. Create a new instance of a game with four ( 4 ) players, and 13 cards per player (e.g. like a game of Bridge).
   2. Have cards dealt to each player one card at a time and display the hands of each player after each round is dealt. You output should look something like the following (Keep in mind that the order of the suits in each player’s hand is determined by the order the suits are received):

Card 01:

Player 1: 2C

Player 2: KH

Player 3: 8H

Player 4: 2D

Card 02:

Player 1: 2C 7D

Player 2: KH 9S

Player 3: 3H 8H

Player 4: 9D 2D

And so on until

Card 13:

Player 1: 7C QC 10C 9C 5C KC 2C 5D 7D 6D KS AH 4H

Player 2: 3H QH JH KH 4S 9S 7S JS 4D 8D 4D 3D 10C

Player 3: 5H 8H 6H 7H 10H 8C 3C AD KD QS 6S 5S 3S

Player 4: JD QD 2D 9D AS 8S 10S 2S 9H 2H 6C JC AC

The above list of methods represents the methods that must be present. You will most likely need additional methods in each of the classes to complete the assignment.

**Things to turn in:**

* Open a Microsoft Word document named Lab107
* Copy and Paste the source code of each of your classes
* Create a screen capture of your NetBeans IDE that includes the contents of the Output Windows from both the programs and paste it into your Word document below your source code.
* Next, zip the Project folder.
* Finally on blackboard, submit both your Word document and project zipped file as two separate files in one submission.