Due Date: Friday, Sep. 14th @ 11:59pm

Points: 100

This is an individual assignment.

Restrictions:

Unless directed otherwise, you cannot use any Java class libraries in this assignment. In particular:

- **You cannot use the ArrayList class nor can you use any methods from the Arrays class.**
- **You cannot use the SinglyLinkedList class defined by Java nor can you use any other collection class from the Java library.**

If needed:

- **You may create and use one or more instances of an array and access the length instance variable for each array.**
- **You may duplicate the Singly Linked List class from the textbook (code fragments 3.14 & 3.15).**
- **You may use the Java Random class.**

In this assignment you will be creating a generic version of ***Bag*** which would be a collection of items that the client would define. The ***Bag*** interface will be a ***Generic interface*** with a ***Type Parameter***. Just like in Lab 102, ***Bag*** will not order the items in any particular order, nor does it prevent any duplicates.

TASK 1 (very similar to Lab102 but generic):

Develop a Generic interface named ***Bag*** that can store a certain number of items (type will be specified by the client). Provide the following methods in the interface:

- A method that returns the current count of items in the bag
- A method that checks if the bag is empty
- A method that adds an item to the bag
- A method that removes a random item from the bag as long as the bag is not empty. This method would return the removed item from the bag.

- A method that removes a specific item from bag. This method will take as parameter the item to be removed, find the first occurrence of the item. Finally, the method returns true if the removal was successful, false otherwise.
- A method that removes all the items from the bag.
- A method that returns the count of occurrences of a specific item in the bag.
- A method that checks if an item exists in the bag.
- A toString method that returns a String representation of the contents of the bag.
- An equals method that returns true if the contents of the two bags are equal, false otherwise.

TASK 2 (very similar to Lab102 but generic):

Design a Generic class called ***`ArrayBag`*** that implements the Generic ***`Bag`*** Interface created earlier. This class will include a Type Parameter as part of the class declaration. A client class using ***`ArrayBag`*** will specify the actual type.

- Declare an instance variable ***`list`*** – an array of ***Generic*** type: This structure will hold the items in the bag.
- Declare another instance variable ***`count`***: This will provide the count of items currently stored in the bag. This count will increment as a new item is added to the bag and decrement as an item is removed from the bag.
- Provide a default constructor that will initialize the instance variable bag to a new array of length 50.
- Provide an overloaded constructor that allows the client to specify the initial capacity of the bag.
- Implement the methods defined in the interface.
- The method that adds an item to the bag should check if the bag is full. When the bag is full it should automatically double the capacity of the bag and add the item.
- The method that removes a specific item which is passed as a parameter should use the object's ***`equals( )`*** method to compare the contents of object in the bag with the contents of the parameter. If there is an object in the bag with the same contents then, it removes that item from the bag and returns true, and returns false if there is no item with the same contents.
- The methods that remove items (randomly or specified item) should automatically shift the remaining items left to fill in the hole left by the removed item.

- Implement the following additional method
    - A method that returns an item at a specific index position in the bag.

TASK 3:

Design a Generic class called ***LinkedBag*** that implements the Generic ***Bag*** Interface created earlier. This class will include a Type Parameter as part of class declaration. A client class using ***`LinkedBag`*** will specify the actual type.

- Declare an instance variable ***`list`*** – a Singly Linked List of ***Generic*** type: This structure will hold the items in the bag.
- Declare another instance variable ***`count`***: This will provide the count of items currently stored in the bag. This count will increment as a new item is added to the bag and decrement as an item is removed from the bag.
- Provide a default constructor that will initialize the instance variable bag with an empty Singly Linked list.
- Implement the methods received from the interface.
- The method that removes a specific item which is passed as a parameter should use the ***`equals( )`*** method to compare the contents of object in the bag with the contents of the parameter.
    - If there is an object in the bag with the same contents then, it removes that item from the bag and returns true,
    - If there is no item with the same contents it returns false.
- Implement the following additional method
    - A method that returns an item at a specific index position in the bag.

TASK 4:

Create a user-defined class called ***`Player`***. Each Player object will have the following attributes (instance variables): ***`name, position played`***, and ***`jersey number`***. Use appropriate data types to define these instance variables and use recommended naming conventions for the variable names. Provide a constructor, implement the accessor and mutator methods for each of the instance variables, and include the toString( ) and equals( ) methods.

TASK 5:

Create a client class named ***Client*** with the ***main( )*** method. Inside the main method do the following:

- Create an object of ***ArrayBag*** called **footballTeam** to store all players' information of NDSU's Men's football team using the overload constructor to make the initial length of the list array equal to 2.

1. Run a ***for loop*** to prompt the user for each Player's information, create the Player object and finally add the player to the team. Enter information for at least 6 players.
2. Display the contents of the Bag.
3. Remove a random player from the team.
4. Display the contents of the Bag.
5. Add a new Player with some made up information.
6. Display the contents of the Bag.
7. Remove the Player that you just added earlier with made up information from the team using appropriate method.
8. Display the contents of the Bag.
9. To demonstrate that your generic class can support objects of different types:
    a. Create an instance of a Bag called **courses** to store the course ids of the courses that you are taking this semester (CSci 161, …..) as Strings.
    b. Populate the Bag with each of your courses.
    c. Display the contents of the Bag.
    d. Remove a random course id from the Bag.
    e. Display the contents of the Bag.

- Create an instance of ***LinkedBag*** called **basketballTeam** to store all the players's information of NDSU's Women's basketball team.
- Repeat steps 1 through 8 above for the **basketballTeam** that uses **LinkedBag**.

Comment your ***Bag*** interface, ***ArrayBag***, ***LinkedBag, Player*** and ***Client*** classes with Java Doc commenting style. Include inline comments as appropriate

Things to turn in:

- Open a Microsoft Word document named Lab103
- Copy and Paste the source code of the ***Bag*** Interface
    - For all copies, make sure to use *Ctrl + A* to select all the source code of the program and *Ctrl + C* to copy).
- Copy and Paste the source code of the ***ArrayBag, LinkedBag, and Player*** classes.
- Copy and Paste the source code of the ***Client*** class
- Copy and paste the output of the client program
- Copy and paste the contents of the Output Window 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.