CS 111 Introduction to Data Structures
Project 1.1 – List Implementations
Maximum Possible Points: 30

Objectives:

- To gain experience with ArrayLists.
- To gain experience with LinkedLists.
- To gain experience with DoublyLinkedLists.
- To gain experience with Test Driven Development.

Overview:

Lists in Java are versatile data structures for storing collections of Objects. For the first part of this project, you will build upon your experiences from lab and lecture by implementing an ArrayList, a LinkedList with a tail, and a doubly linked list. JUnit tests are provided so that you can perform Test Driven Development, i.e., you know the input and expected output of each method before writing the method. You may reuse lab code in this project, but it will need to be altered considerably to pass all of the tests.

Specifications:

Your Java program must implement the following data structures:

- **ArrayList** – a generic list collection that is backed by an Object array that stored in contiguous memory locations. In addition to an Object array, the list should have a constant DEFAULT_CAPACITY of 10 elements and integer fields for its length (the number of items currently stored in it) and capacity (the max number of elements it can hold). The underlying array must be able to expand when new items are added to it that would exceed the maximum capacity. When this occurs, the array should grow by the default size, e.g., an ArrayList with a default size of 10 should expand to have a capacity of 20 when the 11th element is added to it. The Iterator for this class must implement the methods *hasNext()* and *next().*
- **LinkedList** – a generic list collection that stores Node Objects linked by reference. A Node Object should have 2 fields, one for the generic type of element stored, and another that is a reference link to another node. The list should consist of a head Node, which points to the first element in the list, and a tail Node, which points to the last element of the list. The Iterator for this class must implement the methods *hasNext()* and *next().*
- **DoublyLinkedList** – a LinkedList that consists of DoublyLinkedNode Objects. In addition to a generic value, a DoublyLinkedNode Object has two reference links, one that points to the Object before it in a list and one that points to the object that follows it. The ListIterator for this class must implement the methods *hasNext(), hasPrevious(), next(),* and *previous()*.

Each List must implement the following methods:

- `public int size()`
- `public boolean isEmpty()`

- `public boolean contains(Object o)`
- `public Iterator<E> iterator()`
- `public boolean add(E e)`
- `public boolean remove(Object o)`
- `public boolean addAll(Collection<? extends E> c)`
- `public boolean addAll(int index, Collection<? extends E> c)`
- `public void clear()`
- `public E get(int index)`
- `public E set(int index, E element)`
- `public void add(int index, E element)`
- `public E remove(int index)`
- `public int indexOf(Object o)`
- `public int lastIndexOf(Object o)`
- `public List<E> subList(int fromIndex, int toIndex)`

Instructions:

- This will be an individual programming project. Cheating will not be tolerated, as per the syllabus. All code is subject to automated and manual plagiarism detection.
- Organize your code into files named **ArrayList.java, Node.java, LinkedList.java, DoublyLinkedNode.java,** and **DoublyLinkedList.java**.
- Your project should be organized into the following packages:
  - Test – contains the provided JUnit tests.
  - Lists – contains the data structures you create.
- Use meaningful variable names, helpful comments, and a consistent coding style.
- Each file should have an appropriate class header comment block at the top:
  ```
  /**
  * Description of the class file.
  *
  * @author  YOUR NAME
  * @version 1.0
  * @since   YYYY-MM-DD
  */
  ```
- All methods should have a Javadoc header comment block describing the purpose of the method, input, and output:
  ```
  /**
  * Description of why the method exists and how it
  * accomplishes its goal.
  * @param          Parameter list with descriptions
  * @return         Return value with description
  */
  ```

Deliverables:

You will need to submit the following in eCampus by the due date:

- The files **ArrayList.java, Node.java, LinkedList.java, DoublyLinkedNode.java,** and **DoublyLinkedList.java**,
- A document with screenshots of your program passing the JUnit tests in Eclipse.

Grading:

This project is worth 30 points distributed as follows:

- Coding (25 pts)
    - Successful passing of non-iterator JUnit tests for **LinkedList** (1 pt)
    - Successful inclusion of a tail for **LinkedList** (1 pt)
    - Successful passing of iterator JUnit tests for **LinkedList** (3 pts)
    - Successful passing of all JUnit tests for **ArrayList** (10 pts)
    - Successful passing of non-iterator JUnit tests for **DoublyLinkedList** (4 pts)
    - Successful passing of iterator JUnit tests for **DoublyLinkedList** (6 pts)

- Program Style (5 pts)
    - Meaningful variable names (1 pt)
    - Proper indentation (1 pt)
    - Sufficient comments (3 pts)