<MyArrayList> Design

Name: Rahul Jayaraman Period: 3

**Description:**

This class implements a smart array that extends the functionality of a basic array. Basic arrays (defined by int[] myNewArray = new int[6]) can store values; actions that can be performed on them include setting an element to a new value, getting the value of an element at an index, and getting the length. However, this lacks the crucial functionalities of adding or removing an element at a specific location and appending an element to the end. This class solves these problems, all while maintaining the methods present in the basic arrays of Java.

**Services:**

The following services are available:

int size()- This method returns the size of an ArrayList object, or how many elements are stored in that particular data structure. It will use the .length() command that is used on regular arrays in Java, and will run in O(1) time because it just takes the last element’s index and adds 1 to that, thus giving the size (indexing starts at 0 for arrays in Java). It must work for an empty ArrayList too.

boolean add(E obj)- This method appends a value/object to the end of a given instance array. This will run in O(1) time *on average* because once the array is full, it will double the size of the array. Therefore, for future uses of the add() method, it will run in O(1) time as it is a simple usage of the assignment operator.

void add(int index, E obj)- This method will add an object at the desired index and then shift all the objects to its right one index down. This will be accomplished through the use of for loops – use a for loop from the end of the array to add 1 to each index and set it to the previous index’s value; finally, set myNewArray[index] to the object input. Index cannot be negative, nor can it be greater than the size of the array. This runs in O(n) time (for loop).

E get (int index) – This method gets the desired value at a specified index. It is not very hard to implement; in fact, we can merely use the command for a simple array (myNewArray[index]). However, we need to have a “container” (i.e. a variable) that can store the result once it is output. This runs in O(1) time (it is only one calculation). Index is positive and less than the length of the array.

E set (int index, E obj) – This method changes a given index’s value to the value that is input into the argument. Once again, we use the assignment operator (=) to set myNewArray[index] = obj. This will run in O(1) time because it is a simple assignment. Once again, index cannot be negative or greater than the length of the array.

E remove (int index) – This removes the value at a given index and shifts all the values after that back one. This will run in O(n) time due to the use of for loops – this loop will shift the indices back, like the add method’s for loop will shift the indices forward. Index must be positive and less than the length of the array.

Iterator<E> iterator() – Returns null; will be developed in a later lab.

ListIterator<E> listIterator() – Returns null; will be developed in a later lab.

**Internal Data Structures and State**

When the ArrayList constructor is initialized, a new array is formed with the values input. The array is of the type specified. The ArrayList object does not have any state variables; size() is a method that does not take any parameters or state variables. In fact, none of the methods take in parameters that are state variables, precluding the need for one. However, an int variable is maintained to help the size() method run efficiently.