Name: Sarvagnya H. Purohit **Registration Number**: 201070056
Date: 30-09-2021 **Branch:** Computer Engineering
Course: DSA **Course Instructor:** Dr. Mahesh Shirole

Lab Assignment 1

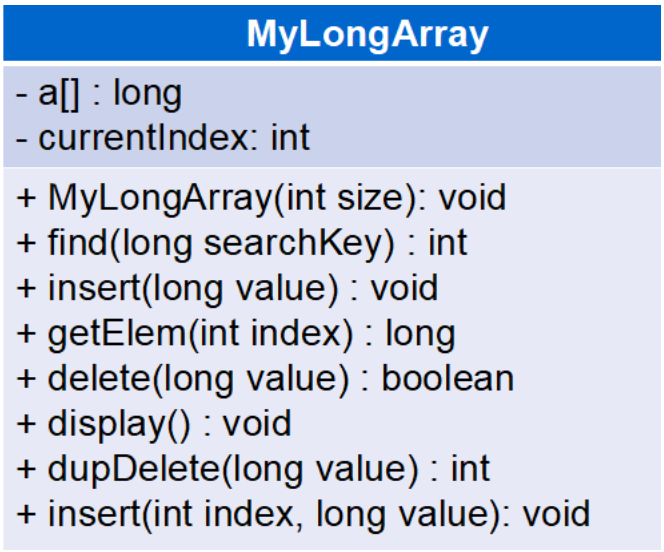
Aim: Design your own long integer array class with the following interface. Write a test application for your class.

Theory: In Java, array is an object which can hold multiple elements of the same data type. The elements of an array are stored in contiguous memory locations, meaning that random access of any element is possible. In general, an array is a data structure where we can store a specific number of elements of the same data type. In Java, all arrays are dynamically allocated (they are stored on the heap). We can store only a predefined number of elements in a Java array. We can perform various operations with the help of user defined methods. In Java, a static array is declared by:

```
data_type[] array_name = new data_type[size];
```

Arrays offer a vast application set in the real world with a variety of applications. In fact, array is the most commonly used data structure.

The UML Class Diagram for the Array class being implemented is:



Data members:

- a[]: long -- this is an array of type long, which holds elements of the array in this class
- currentIndex: int -- this is the index of last element available in the arrayMember

Class Methods:

- MyLongArray(int size): void -- This is a constructor method. This initializes the size of the internal array variable of this class.

- find(long searchKey) : int -- This search operation on array searchKey will be found in the array and it's index will be returned.
- insert(long value): void -- This method allows inserting the data in the internal array at currentIndex location and currentIndex is incremented by one.
- getElement(int index): long -- This method returns the element in the internal array at specified index if index <= currentIndex otherwise -1
- delete(long value): boolean -- This method deletes specific value's first occurrence from the internal array. Elements after that location are shifted one location up. currentIndex counter decremented by one; and true is returned. If the specified value is not available then false is returned.
- display():void -- This method displays all elements of the array.
- dupDelete(long value): int -- This is the same as delete method but removes all entries of the said value and return number of elements are deleted.
- insert(int index, long value): void -- This method allows inserting said value at given index if index <= currentIndex. Remaining elements are shifted to higher indexes and currentIndex is incremented by one. If index > currentIndex then element is inserted atcurrentIndex and currentIndex is incremented by one; also display a warning message.

Test Data and Output of the program:

```
"C:\Users\Sarvagnya Purohit9\.jdks\openjdk-17\bin\java.exe" "-javaagent:C:\Users\Sarvagnya Purohit9\
Enter size of the array
5
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
2
Enter element to insert: 3
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
2
Enter element to insert: 4
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
2
Enter element to insert: 3
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
6
```

```
Enter the index and value to insert: 2 1
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
5
3   4   1   3   Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
1
Find the element to search: 1
Element 1 found at index 2
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
3
Enter the index: 3
Element at index 3 is 3
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
2
Enter element to insert: 5
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
5
3   4   1   3   5   Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
4
Enter the value to delete: 2
Element could not be deleted
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
4
Enter the value to delete: 1
Element 1 deleted
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
5
3   4   3   5   Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
7
Enter the value to delete: 3
Number of duplicates deleted are 2
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
5
4   5   Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 0.Exit
0

Process finished with exit code 0
```

Conclusion: In this assignment, we learnt how to implement a dynamic array in Java by creating a class containing that array and the various operations required to perform on it. We learnt to insert, delete, access elements and also learnt to traverse through all the elements of that array for the search operation. We learnt how to resize an array based on the number of elements present in it at that specific instance. Such arrays have multiple applications. They can be used to solve many problems where we work with a large set of data and are required to perform a bunch of operations on that data.