Name: Sarvagnya H. Purohit **Registration Number**: 201070056
Date: 10-10-2021 **Branch:** Computer Engineering
Course: DSA **Course Instructor:** Dr. Mahesh Shirole

Lab Assignment 2

Aim: Extend your long integer array class with following new operations
initArray(): void -- initialize all elements with random value [java.util.Random]
bubbleSort()
selectionSort()
insertionSort()
Write a test application for your class to demonstrate above operations.

Theory: A sorting algorithm is used to rearrange a given array or list of elements according to a comparison operator on the elements.

Sorting generally means to sort or order a particular array or collection of elements into a particular sequence i.e., either in ascending order or in descending order. There are many kinds of Sorting techniques to sort a given array of numbers.

1) Bubble sort:
   Bubble Sort is one of the simplest sorting techniques in Java to sort the array elements. The idea is to traverse from the starting element to the last one by comparing the adjacent elements and swapping them if they are not in the specific order. It is called Bubble sort because, at the end of each iteration, the largest number sits at the bottom of the array just like the heaviest bubble settles down in a vessel. The swapping of elements continues until the array is sorted and no more swapping is required. We can also sort the elements in the descending order in which the smallest element goes at the end of the array in each iteration. This can only happen if we inverse the weight of the element.

2) Selection sort:
   The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.
   1) The subarray which is already sorted.
   2) Remaining subarray which is unsorted.
   In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

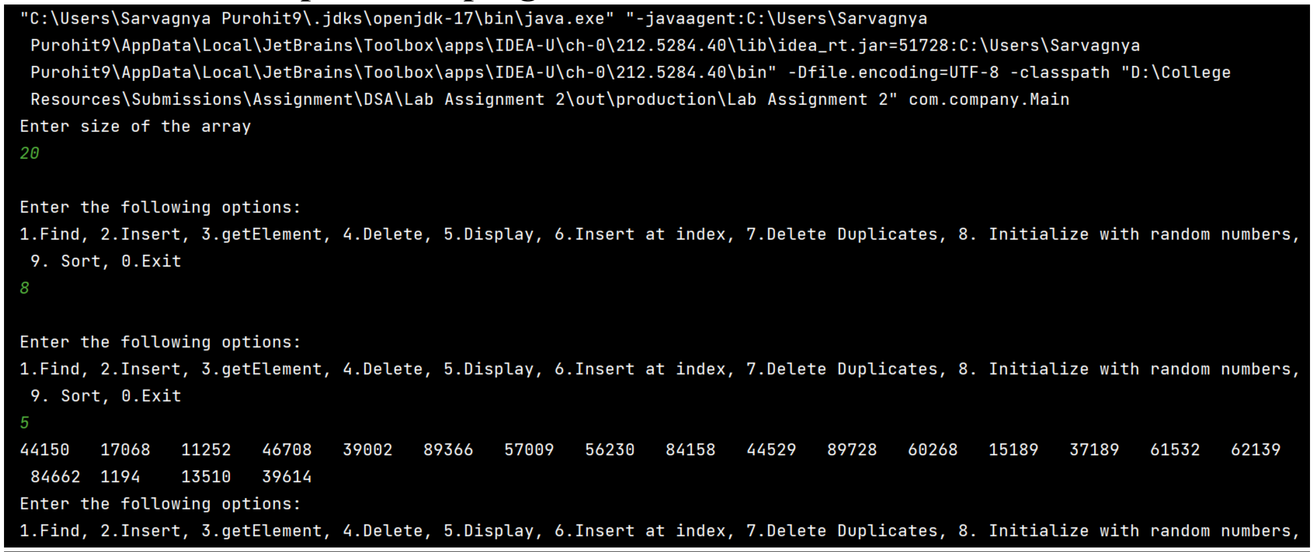
3) Insertion sort:
   The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.
   Algorithm
   To sort an array of size n in ascending order:
   1) Iterate from arr[1] to arr[n] over the array.
   2) Compare the current element (key) to its predecessor.

3) If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

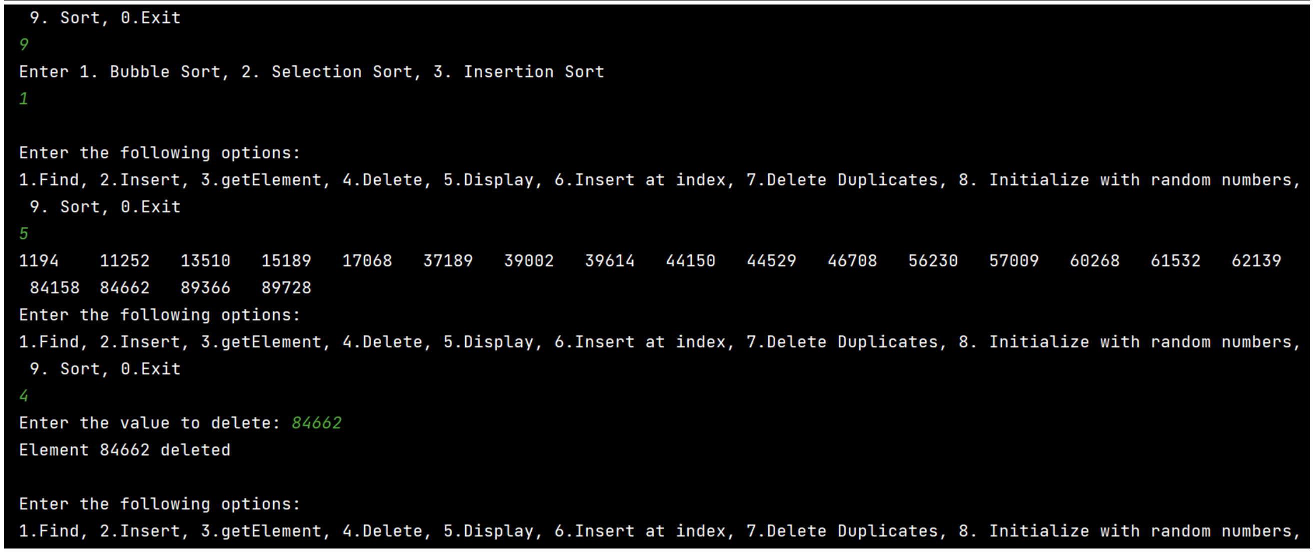
java.util.Random class:
Java Random class is used to generate a stream of pseudorandom numbers.
For using this class to generate random numbers, we have to first create an object of this class and then invoke methods such as nextInt(), nextDouble(), nextLong() etc. on that object.
We can pass arguments to the methods for placing an upper bound on the range of the numbers to be generated. For example, nextLong(1000) will generate numbers(of the type *Long*) in the range 0 to 999 both inclusive.

In this experiment, we will extend our previously created MyLongArray class(inheritance). I.e., our new class MyRandomLongArray will be a child class of the previously created class.

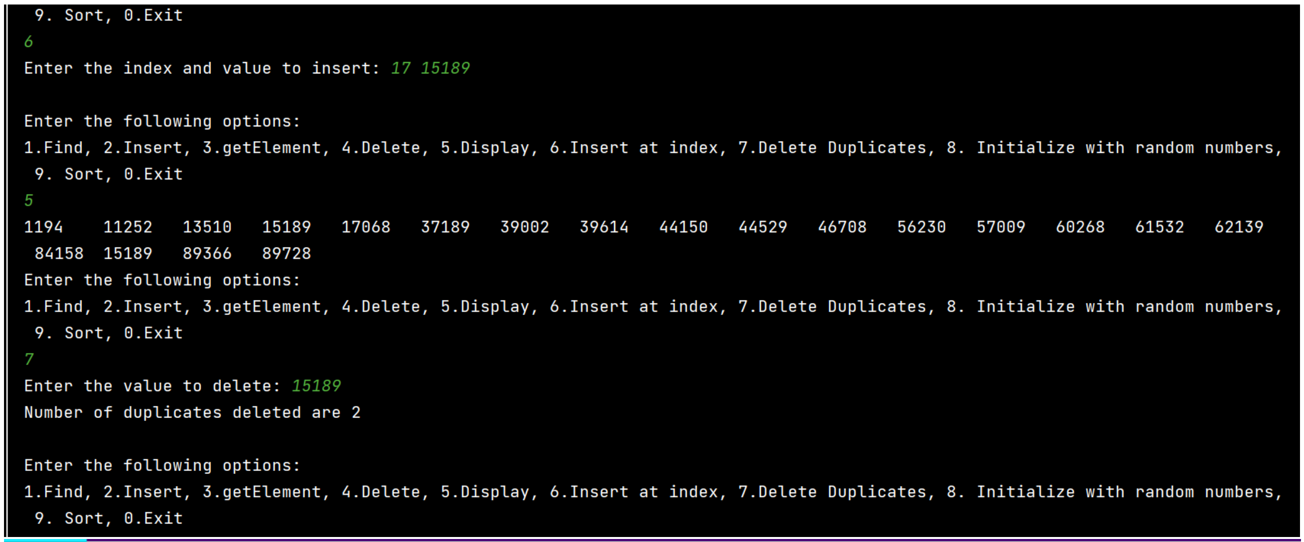
Test Data and Output of the program:



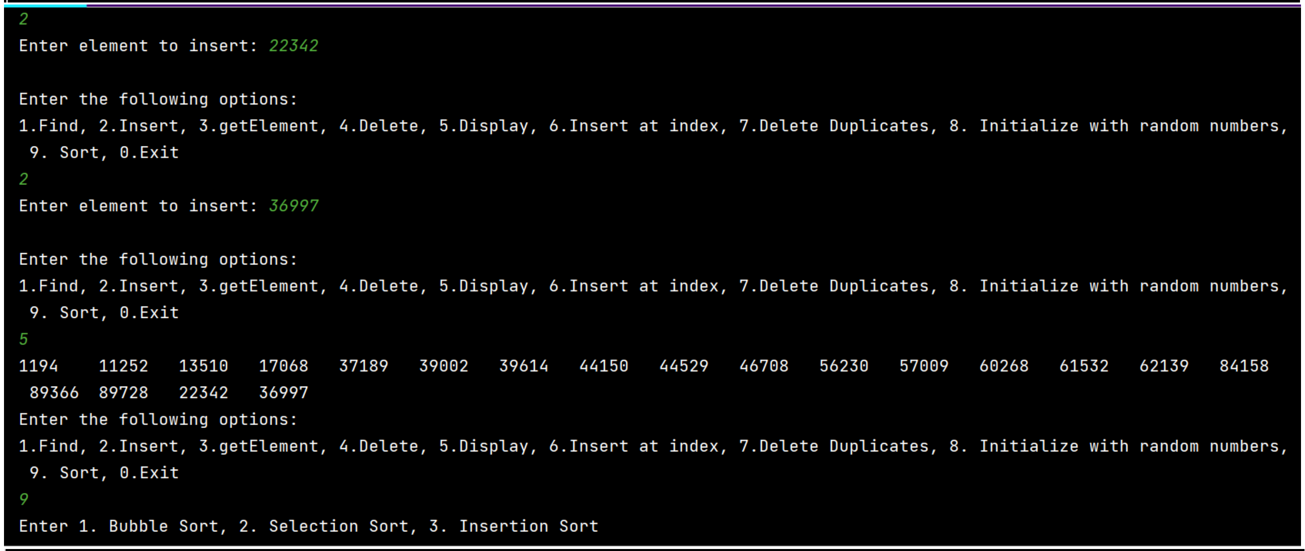
```
"C:\Users\Sarvagnya Purohit9\.jdks\openjdk-17\bin\java.exe" "-javaagent:C:\Users\Sarvagnya
 Purohit9\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch-0\212.5284.40\lib\idea_rt.jar=51728:C:\Users\Sarvagnya
 Purohit9\AppData\Local\JetBrains\Toolbox\apps\IDEA-U\ch-0\212.5284.40\bin" -Dfile.encoding=UTF-8 -classpath "D:\College
 Resources\Submissions\Assignment\DSA\Lab Assignment 2\out\production\Lab Assignment 2" com.company.Main
Enter size of the array
20

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
8

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
5
44150   17068   11252   46708   39002   89366   57009   56230   84158   44529   89728   60268   15189   37189   61532   62139
 84662  1194    13510   39614
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
```



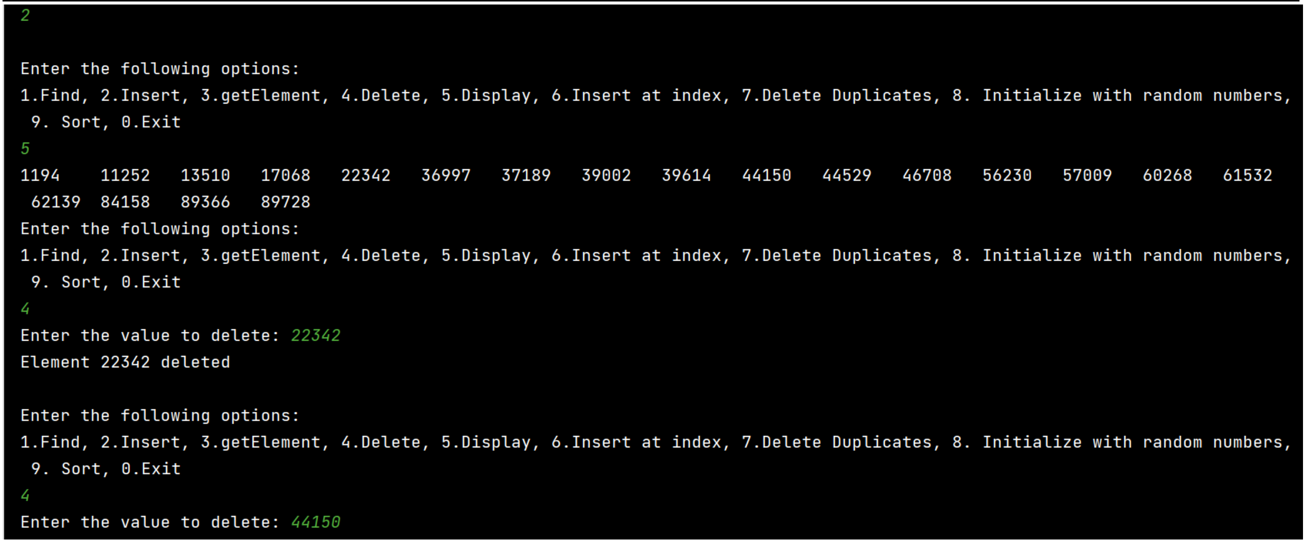
```
 9. Sort, 0.Exit
9
Enter 1. Bubble Sort, 2. Selection Sort, 3. Insertion Sort
1

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
5
1194    11252   13510   15189   17068   37189   39002   39614   44150   44529   46708   56230   57009   60268   61532   62139
 84158  84662   89366   89728
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
4
Enter the value to delete: 84662
Element 84662 deleted

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
```



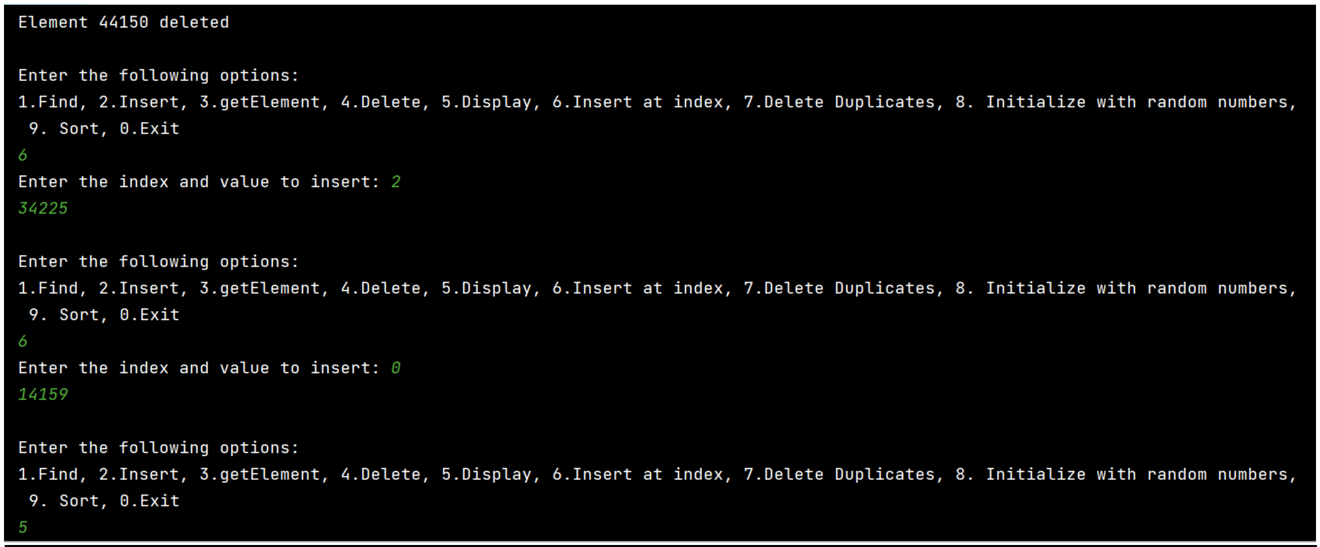
```
 9. Sort, 0.Exit
6
Enter the index and value to insert: 17 15189

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
5
1194    11252   13510   15189   17068   37189   39002   39614   44150   44529   46708   56230   57009   60268   61532   62139
 84158  15189   89366   89728
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
7
Enter the value to delete: 15189
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, 8. Initialize with random numbers,
 9. Sort, 0.Exit
```



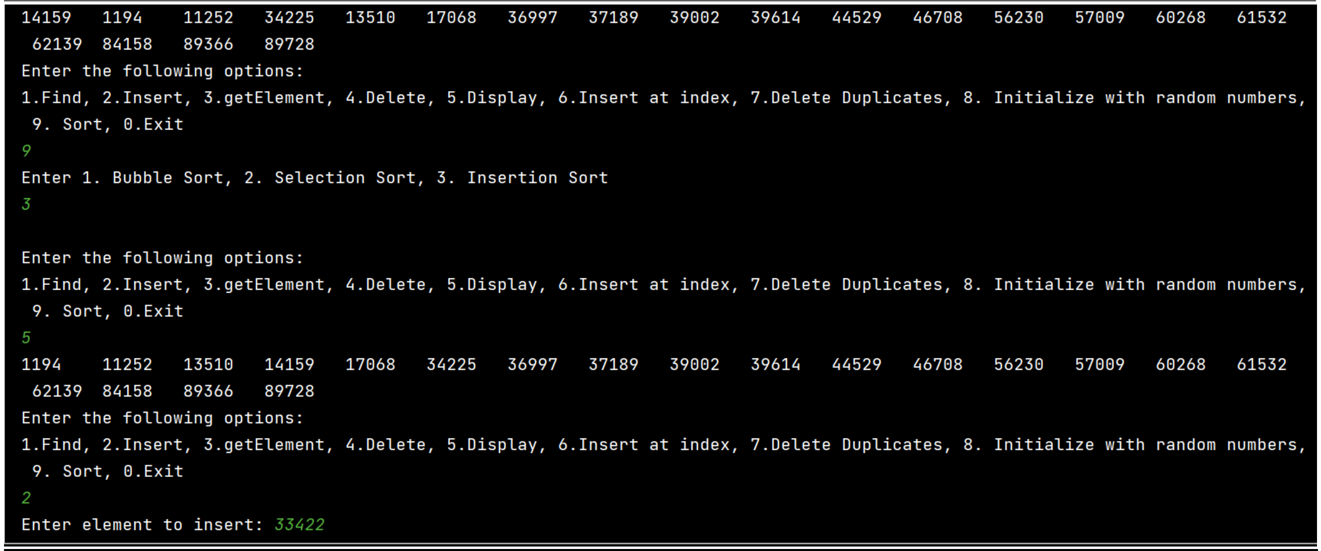
```
2
Enter element to insert: 22342

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
2
Enter element to insert: 36997

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
5
1194    11252   13510   17068   37189   39002   39614   44150   44529   46708   56230   57009   60268   61532   62139   84158
 89366  89728   22342   36997
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
9
Enter 1. Bubble Sort, 2. Selection Sort, 3. Insertion Sort
```



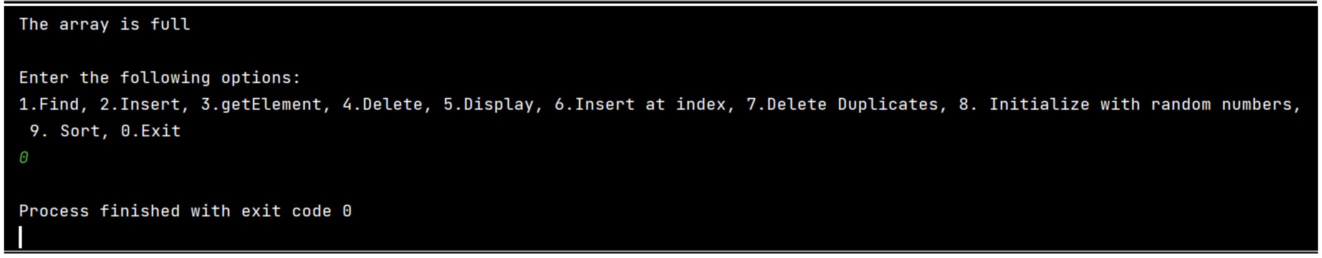
```
2

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
5
1194    11252   13510   17068   22342   36997   37189   39002   39614   44150   44529   46708   56230   57009   60268   61532
 62139  84158   89366   89728
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
4
Enter the value to delete: 22342
Element 22342 deleted

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
4
Enter the value to delete: 44150
```



```
Element 44150 deleted

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
6
Enter the index and value to insert: 2
34225

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
6
Enter the index and value to insert: 0
14159

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
5
```



```
14159   1194    11252   34225   13510   17068   36997   37189   39002   39614   44529   46708   56230   57009   60268   61532
 62139  84158   89366   89728
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
9
Enter 1. Bubble Sort, 2. Selection Sort, 3. Insertion Sort
3

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
5
1194    11252   13510   14159   17068   34225   36997   37189   39002   39614   44529   46708   56230   57009   60268   61532
 62139  84158   89366   89728
Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
2
Enter element to insert: 33422
```



```
The array is full

Enter the following options:
1.Find, 2.Insert, 3.getElement, 4.Delete, 5.Display, 6.Insert at index, 7.Delete Duplicates, 8. Initialize with random numbers,
 9. Sort, 0.Exit
0

Process finished with exit code 0
```

Conclusion:

We learnt about the various algorithms to sort an array: Bubble sort, Selection Sort and Insertion sort. We also learnt to generate arrays of desired length and assign random values to it using Java’s Random class. Here we extended MyLongArray so as to avail all the methods in it to the object of the MyRandomLongArray class. Hence, we can use methods in MyLongArray as well as those in MyRandomLongArray with the help of an object of MyRandomLongArray class. We can apply inheritance in our Java programs using the “*extends*” keyword.