

```
1 import java.util.Arrays;
2
3 public class BinarySearchExample {
4     public static void main(String[] args) {
5         int[] arr = {20, 100, 70, 3, 50, 72, 15};
6         System.out.println("Original array: " + Arrays.toString(arr));
7         mergeSort(arr, 0, arr.length - 1);
8         System.out.println("Sorted array: " + Arrays.toString(arr));
9         int key = 15;
10        long start = System.nanoTime();
11        int elementIndex = binarySearch(arr, key);
12        long end = System.nanoTime();
13        if (elementIndex == -1) {
14            System.out.println("Binary Search -" + " Index ");
15        } else {
16            System.out.println("Binary Search -" + " Index " + elementIndex + " = " +
17arr[elementIndex]);
18        }
19        System.out.println("Time Binary search = " + (end - start) + " nanoseconds");
20        start = System.nanoTime();
21        int linearElementIndex = linearSearch(arr, key);
22        end = System.nanoTime();
23        if (linearElementIndex == -1) {
24            System.out.println("Linear Search -" + " Index");
25        } else {
26            System.out.println("Linear Search -" + " Index " + linearElementIndex + "
27= " + arr[linearElementIndex]);
28        }
29        System.out.println("Time Linear search = " + (end - start) + " nanoseconds");
30    }
31
32    public static void mergeSort(int[] arr, int l, int r) {
33        if (l < r) {
34            int m = (l + r) / 2;
35            mergeSort(arr, l, m);
36            mergeSort(arr, m + 1, r);
37            merge(arr, l, m, r);
38        }
39    }
40
41    public static void merge(int[] arr, int l, int m, int r) {
42        int n1 = m - l + 1;
43        int n2 = r - m;
44
45        int[] L = new int[n1];
46        int[] R = new int[n2];
47
48        for (int i = 0; i < n1; i++) {
49            L[i] = arr[l + i];
50        }
51        for (int j = 0; j < n2; j++) {
52            R[j] = arr[m + 1 + j];
53        }
54
55        int i = 0, j = 0, k = l;
56        while (i < n1 && j < n2) {
57            if (L[i] <= R[j]) {
58                arr[k] = L[i];
59                i++;
60            } else {
61                arr[k] = R[j];
62                j++;
63            }
64            k++;
65        }
66        while (i < n1) {
67            arr[k] = L[i];
68            i++;
69            k++;
70        }
71        while (j < n2) {
72            arr[k] = R[j];
73            j++;
74            k++;
75        }
76    }
77}
```

```
58         } else {
59             arr[k] = R[j];
60             j++;
61         }
62         k++;
63     }
64
65     while (i < n1) {
66         arr[k] = L[i];
67         i++;
68         k++;
69     }
70
71     while (j < n2) {
72         arr[k] = R[j];
73         j++;
74         k++;
75     }
76 }
77
78 public static int binarySearch(int[] arr, int key) {
79     int left = 0;
80     int right = arr.length - 1;
81     while (left <= right) {
82         int mid = (left + right) / 2;
83         if (arr[mid] == key) {
84             return mid;
85         } else if (arr[mid] < key) {
86             left = mid + 1;
87         } else {
88             right = mid - 1;
89         }
90     }
91     return -1;
92 }
93
94 public static int linearSearch(int[] arr, int key) {
95     for (int i = 0; i < arr.length; i++) {
96         if (arr[i] == key) {
97             return i;
98         }
99     }
100     return -1;
101 }
102 }
```