

**Assignment 1**  
**Subject: Advanced**  
**Software Engineering**  
**(Software Testing)**  
**By Arghya Bandyopadhyay**  
**Roll no: 20CS4103**

## **Binary Search in Java**

```
import java.util.*;
class BinarySearch {
    // Returns index of key if it is present in arr[l.. r], else return -1

    int binarySearch(int arr[], int l, int r, int key)
    {
        if (r >= l) {
            int middle = l + (r - l) / 2;

            // If the element is present at the middle itself
            if (arr[middle] == key)
                return middle;

            // If element is smaller than middle, then it can only be present in left subarray
            if (arr[middle] > key)
                return binarySearch(arr, l, middle - 1, key);

            // Else the element can only be present in right subarray
            return binarySearch(arr, middle + 1, r, key);
        }

        // We reach here when element is not present in array
        return -1;
    }

    public static void main(String args[])
    {
        BinarySearch ob = new BinarySearch();
        int arr[];
        Scanner sc= new Scanner(System.in);
        System.out.println("Enter the length of the array you want to search in.");
        int length=sc.nextInt();
        arr= new int[length];
        System.out.println("Enter "+length+" elements.");
        for(int i=0;i<length;i++)
        {
            try
            {
                System.out.println("Enter element no. "+(i+1));
                arr[i]=sc.nextInt();
            }
            catch(Exception e)
            {
                System.out.println(e.getMessage());
                i--;
            }
        }
        sc.nextLine();
    }
}
```

```
System.out.println("Enter the no which you want to find in the array input by you :\n Enter Q  
to quit");
```

```
while(true)
{
    String a=sc.nextLine();
    if(a.equalsIgnoreCase("Q"))
    {
        System.exit(0);
    }
    else
    {
        try
        {
            int result = ob.binarySearch(arr, 0, length - 1, Integer.parseInt(a));
            if (result == -1)
                System.out.println("Element not present");
            else
                System.out.println("Element found at index " + result);

        }

        catch(Exception e)
        {
            System.out.println(e.getMessage());
        }
    }
}
}
```

## Boundary Value Check Analaysis

1 variables so total cases=  $4n+1=4*1+1=5$

let sorted array used in above program=[11,25,31,49,52,66,74,89,93,100]

Range:11-100

Min value:11

Min+ value:12

Max value:100

Max- value:93

Nominal value:52

Test cases:

Test No.	Input	Outpur
1	11	Element found at index 0
2	12	Element not present
3	100	Element found at index 9
4	93	Element found at index 8
5	74	Element found at index 6

## Equivalence partitioning

Pre-conditions satisfied, key element in array

Pre-conditions satisfied. Key element not in array

Pre-conditions unsatisfied, key element in array

Pre-conditions unsatisfied. Key element not in array

Input array has a single value

### Binary Search – Test cases

Input array(T)	Key(key)	Output(Found, L)
19	19	true,0
19	0	false,??
19,20,23,25	19	true,0
19,16,28,30,31,51,65	65	true,6
19,20,23,25,35,38,52	20	true,2
19,20,23,25,35,38,52	23	true,2
22,38,41,53,62	53	true,3
21,23,29,33,38	25	false,??

Check Input partitions:

- Do the inputs satisfy the pre condition?
- Is the key in the array?  
Leads to at least 2x2 equivalence classes

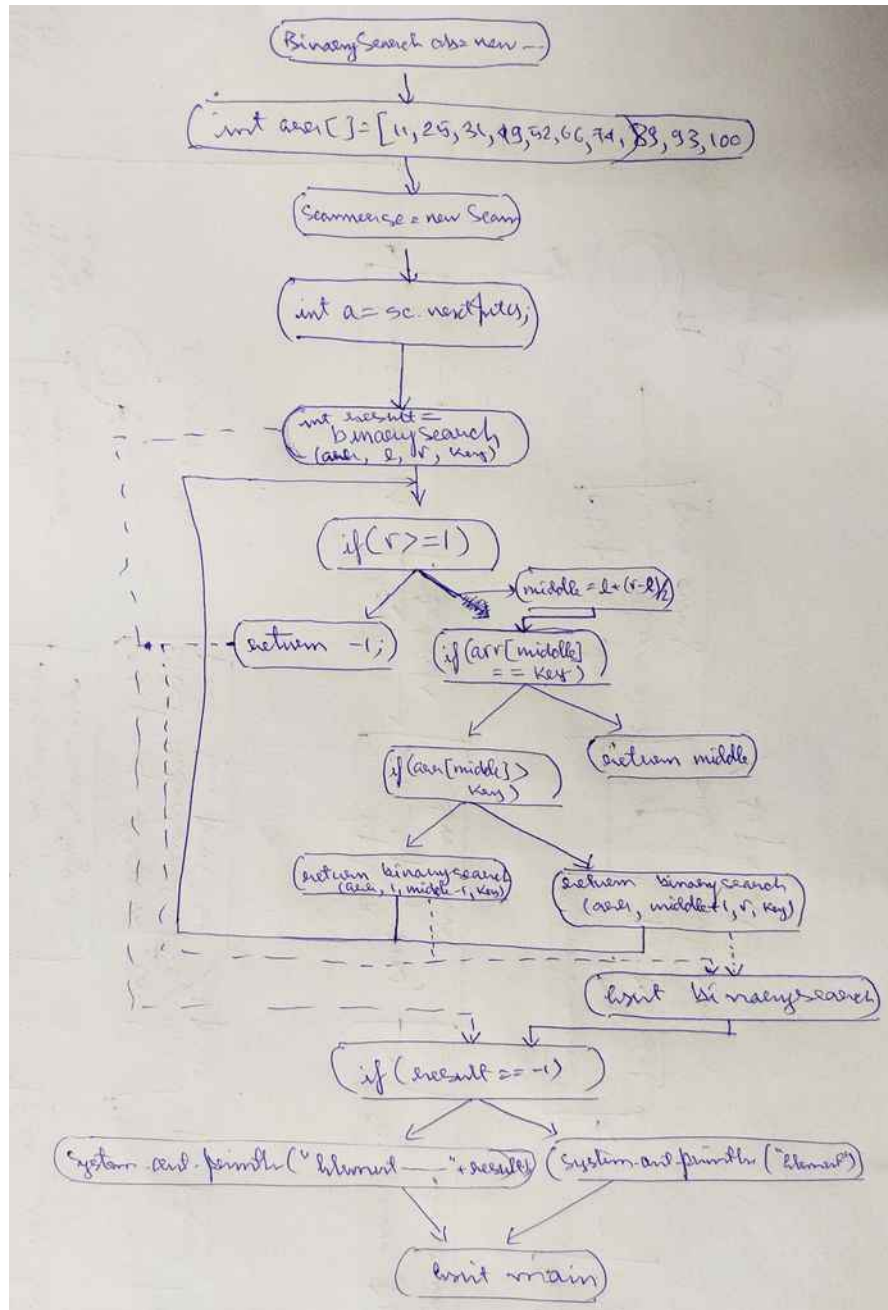
Check boundary conditions

- Is the array of length 1?
- Is the key at the start or end of the array>leads to further subdivisions  
(not all combinations make sense)

### Equivalence Partitioning: Test Data

Test cases	Input	Output
Array length 0	key =27, element={}	false
Array not sorted	key =27, element={43,31,27,28}	exception
Array size 1, key in array	key =27, element={27}	true
Array size 1, key not in array	key =0, element={27}	false
Array size > 1, key is first	key =27, element={27,28,32,33}	true
Array size > 1, key is last element	key =33, element={27,28,32,33}	true
Array size > 1, key is in middle	key =20, element={27,28,32,33}	true
Array size > 1. key not in array	key =50, element={27,28,32,33}	false

## Control Flow Graph:



## Cyclomatic Complexity

No of edges(E)=20

No of nodes(N)=18

$V(G)=E-N+2$

$V(G)=20-18+2=4$