JAVA.UTIL.ARRAYS.BINARYSEARCH METHOD

Description

The **java.util.Arrays.binarySearch***T[]a, Tkey, Comparator < ?superT > c* method searches the specified array for the specified object using the binary search algorithm. The array must be sorted into ascending order according to the specified comparator before making this call. If it is not sorted, the results are undefined.

Declaration

Following is the declaration for **java.util.Arrays.binarySearch***super* method

```
public static <T> int binarySearch(T[] a, T key, Comparator<? super T> c)
```

Parameters

- **a** -- This is the array to be searched.
- **key** -- This is the value to be searched for.
- **c** -- This is the comparator by which the array is ordered. A null value indicates that the elements natural ordering should be used.

Return Value

This method returns index of the search key, if it is contained in the array, else it returns $-(insertionpoint - 1)$. The insertion point is the point at which the key would be inserted into the array: the index of the first element greater than the key, or a.length if all elements in the array are less than the specified key.

Exception

- **ClassCastException** -- if the array contains elements that are not mutually comparable using the specified comparator or the search key is not comparable to the elements of the array using this comparator.

Example

The following example shows the usage of java.util.Arrays.binarySearch*super* method.

```
package com.tutorialspoint;

import java.util.Arrays;
import java.util.Comparator;

public class ArrayDemo {

   public static void main(String[] args) {

   // initializing unsorted short array
   Short shortArr[] = new Short[]{5, 2, 15, 52, 10};

   // use comparator as null, sorting as natural ordering
   Comparator<Short> comp = null;

   // sorting array
   Arrays.sort(shortArr, comp);

   // let us print all the elements available in list
   System.out.println("The sorted short array is:");
   for (short number : shortArr) {
```

```
   System.out.println("Number = " + number);
   }

   // entering the value to be searched
   short searchVal = 15;

   int retVal = Arrays.binarySearch(shortArr, searchVal, comp);
   System.out.println("The index of element 15 is : " + retVal);

   }
}
```

Let us compile and run the above program, this will produce the following result:

```
The sorted short array is:
Number = 2
Number = 5
Number = 10
Number = 15
Number = 52
The index of element 15 is : 3
```