Arrays is a class in the `java.utils` package that contains various methods for manipulating arrays (such as sorting and searching). This class also contains a static factory that allows arrays to be seen as a list. The methods in this class throw a `NullPointerException` if the specified array reference is `null`.

Searching an element in an array

The `Arrays` class provides the `binarySearch()` method to search for a particular element in an array. There are a lot of overloaded `binarySearch()` methods to handle all the primitive types. Some of the important points to note about the `binarySearch()` method are:

1. The array that is passed to the method should be sorted. If the array is not sorted, then the result is undefined.
2. This method returns the index where the element is present in the array. If the element is not present in the array, then the index of the first element greater than the key is returned.
3. If the array contains multiple elements with the specified value, there is no guarantee which one will be found.
4. ClassCastException is thrown if the search key is not comparable to the elements of the array.

As the name suggests, the `binarySearch()` method uses the binary search algorithm to search for an element in the array. It is far better than a linear search. The complexity of the linear search algorithm is **O(n)**, whereas the complexity of the binary search algorithm is **O(log n)**.

The below example shows how we can use the `binarySearch()` method to search an element in an integer array.

```
import java.util.Arrays;

public class ArraysDemo {

    public static void main(String args[]) {

        int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

        int index = Arrays.binarySearch(numbers, 4);

        System.out.println("The index of element 4 in the array is " + index);

    }
}
```

Run Save Reset

It is possible that we may not need to search the entire array. In that case, we can provide the start and end index of the elements in the array that needs to be searched.

```
import java.util.Arrays;

public class ArraysDemo {

    public static void main(String args[]) {

        int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

        int index = Arrays.binarySearch(numbers, 5, 9, 4);

        System.out.println("The index of element 4 in the array is " + index);

        index = Arrays.binarySearch(numbers, 1, 5, 4);

        System.out.println("The index of element 4 in the array is " + index);

    }
}
```

Run Save Reset

Searching a custom class object in an array

Let's say we have created an **Employee** class and we have an array of **Employee** objects. We want to check if a particular **Employee** object is present in the array or not.

We will pass the employee array and the object that we need to search to the `binarySearch()` method as shown in the below example.

The below example will not compile because our Employee class does not implement the Comparable interface. For the search to be successful it is necessary that the objects we have stored in the array should be comparable.

```
import java.util.Arrays;

public class ArraysDemo {

    public static void main(String args[]) {
        Employee[] employees = { new Employee(123, "Jay"), new Employee(124, "Roy"), new Employee(125, "N
                new Employee(126, "Tom") };

        int index = Arrays.binarySearch(employees, new Employee(124, "Roy"));

        System.out.println("The index of employee in the array is " + index);

    }

}
```

Run Save Reset

We have two options to fix this issue. Either our class should implement the Comparable interface or we should pass a Comparator implementation while calling the `binarySearch()` method.

In the below example, we are passing the Comparator implementation while calling the `binarySearch()` method.

ArraysDemo.java

Employee.java

```
import java.util.Arrays;

public class ArraysDemo {

    public static void main(String args[]) {
        Employee[] employees = { new Employee(123, "Jay"), new Employee(124,
                new Employee(126, "Tom") };

        int index = Arrays.binarySearch(employees, new Employee(124, "Roy"),

        System.out.println("The index of employee object in the array is " +

    }

}
```

Run Save Reset