ARRAYS CLASS IN JAVA

Arrays Class:

The Arrays class in java.util package is a part of the Java Collection Framework. The **Arrays** class in Java is a utility class that provides various methods for manipulating and working with arrays. This class provides static methods to dynamically create and access Java arrays. It consists of only static methods and the methods of Object class. The methods of this class can be used by the class name itself.

Some Methods of **Arrays** class:

1. equals(): This method compares two arrays to determine if they are equal. The arrays are considered equal if they have the same length and contain the same elements in the same order.

Example:

```
import java.util.Arrays;
import java.util.Scanner;

public class ArraysEqualsMethod {

     public static void main(String[] args) {
          Scanner sc= new Scanner(System.in);
          System.out.println("enter the size of first
array:");
          int arr1[]=new int[sc.nextInt()];
          System.out.println("enter the elements into arr1");
          for(int i=0;i<=arr1.length-1;i++) {
               arr1[i]=sc.nextInt();
          }
          System.out.println("enter the size of second
array:");
          int arr2[]=new int[sc.nextInt()];
          System.out.println("enter the elements into arr2");
```

```
            for(int i=0;i<=arr1.length-1;i++) {
                arr2[i]=sc.nextInt();
            }
            if(Arrays.equals(arr1, arr2)) {
                System.out.println("Arrays are Equal");
            }
            else {
                System.out.println("arrays are not equal");
            }
        }
}
```

Output:

```
enter the size of first array:
2
enter the elements into arr1
1 2
enter the size of second array:
2
enter the elements into arr2
1 2
Arrays are Equal
```

2. sort(): This method sorts the elements of an array in ascending order. It uses a modified version of the quicksort algorithm or the mergesort algorithm, depending on the data type of the array.

Example:

```
import java.util.Arrays;
import java.util.Scanner;

public class ArraysSort {

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        System.out.println("enter the size of an array:");
        int arr[]=new int[sc.nextInt()];
        System.out.println("enter the elements into arr");
        for(int i=0;i<=arr.length-1;i++) {
```

```
            arr[i]=sc.nextInt();
        }
        System.out.println("Array before Sorting");
        for(int i=0;i<=arr.length-1;i++) {
            System.out.print(arr[i]+" ");
        }
        System.out.println();
        Arrays.sort(arr);
        System.out.println("Array after Sorting");
        for(int i=0;i<=arr.length-1;i++) {
            System.out.print(arr[i]+" ");
        }
    }
}
```

Output:

```
enter the size of an array:
5
enter the elements into arr
3 2 45 8 0
Array before Sorting
3 2 45 8 0
Array after Sorting
  0  2 3 8 45
```

3. toString(): It converts an array into a string representation. The elements of the array are concatenated and separated by commas, enclosed in square brackets.

Example:

```
import java.util.Arrays;
import java.util.Scanner;

public class ArraysToString {

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        System.out.println("enter the size of an array:");
        int arr[]=new int[sc.nextInt()];
        System.out.println("enter the elements into arr");
```

```
        for(int i=0;i<=arr.length-1;i++) {
             arr[i]=sc.nextInt();
        }
        String str= Arrays.toString(arr);
        System.out.println("Arrays to string "+ str);

    }
}
```

Output:

```
enter the size of an array:
5
enter the elements into arr
2 3 4 5 6
Arrays to string [2, 3, 4, 5, 6]
```

4. copyOf(): This method creates a new array that is a copy of the original array. You can specify the length of the new array, which can be either shorter or longer than the original.

Example:

```
import java.util.Arrays;
import java.util.Scanner;

public class ArrayCopy {

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        System.out.println("enter the size of an array:");
        int arr[]=new int[sc.nextInt()];
        System.out.println("enter the elements into arr");
        for(int i=0;i<=arr.length-1;i++) {
             arr[i]=sc.nextInt();
        }
        int []copyArray=Arrays.copyOf(arr, arr.length);
        System.out.println("Source array: " +
Arrays.toString(arr));
```

```
        System.out.println("Copied array: " +
Arrays.toString(copyArray));

    }

}
```

Output:

```
12 12 123 12 13
Source array: [12, 12, 123, 12, 13]
Copied array: [12, 12, 123, 12, 13]
```

5. fill(): It assigns a specific value to every element in an array. You can specify the range of indices to be filled.

Example:

```
import java.util.Arrays;
import java.util.Scanner;

public class ArraysFill {

    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        System.out.println("enter the size of an array:");
        String arr[]=new String[sc.nextInt()];
        Arrays.fill(arr, "Orange");

        System.out.println("Array filled with colours " +
Arrays.toString(arr));


    }

}
```

Output:

```
enter the size of an array:
3
Array filled with colours [Orange, Orange, Orange]
```