**Date:13/09/2023**

**Name: Suvarna**

**Email: Suvarnaballid726@gmal.com**

**Assignment:13**

1) Complete description of Array Class and Array Methods() ?

2)Usage of Array class and package.

**Arrays class in Java**

The **Array class** is contained in **java.util.package.** Java Arrays are created and accessed through the static methods that are provided by this class. The methods of this class can be accessed by the class name. Only static methods are present and the methods of the object class.  
This class contains various methods for manipulating arrays

## Java Arrays Class declaration

Arrays class extends Object class. The general syntax to declare arrays class in java is as follows:

public class Arrays

extends Object

The hierarchy diagram of this class is as below:

java.lang.Object

java.util.Arrays

Before working with arrays, we need to import the statement java.util.Arrays class or entire java.util.\* package in the java program.

Array class in java provides lots of useful utility methods for common array operations such as sorting, searching, comparing arrays, filling array elements, etc.

All these methods provided by the arrays class are static in nature. So, we do not need to create an instance of array class in order to use these utility methods.

We can call these utility methods by using the class name itself.

For example, a sort() method of arrays class can be used for sorting an array in ascending order. Similarly, binarySearch() method is used to search for a key in an array.

Arrays class is available in java.util.Arrays package and it is a member of a java collection framework. This class was introduced in Java 1.2 version.

What is the use of array class in Java

the array class provides static methods to dynamically create and access java arrays. Arrays permits widening conversions to occur during a get or set operation ,but throws an illegal argument Exception if a narrowing conversion would occur.

**Arrays Class Methods in Java**

In addition to methods inherited from Object class, Java Arrays class also provides useful static methods for working with arrays.

Here, we have listed some most commonly important methods with example programs that are as follow

**Method:1 Compare(array1,array2)**

 This method is used to compare two arrays passed as parameters lexicographically. Both arrays array1 and array2 must have the same data types.

The compare() method returns 0 if the first and second arrays are equal and contain the same elements in the same order.

If the first array is lexicographically greater than second array, it returns a value greater than zero.

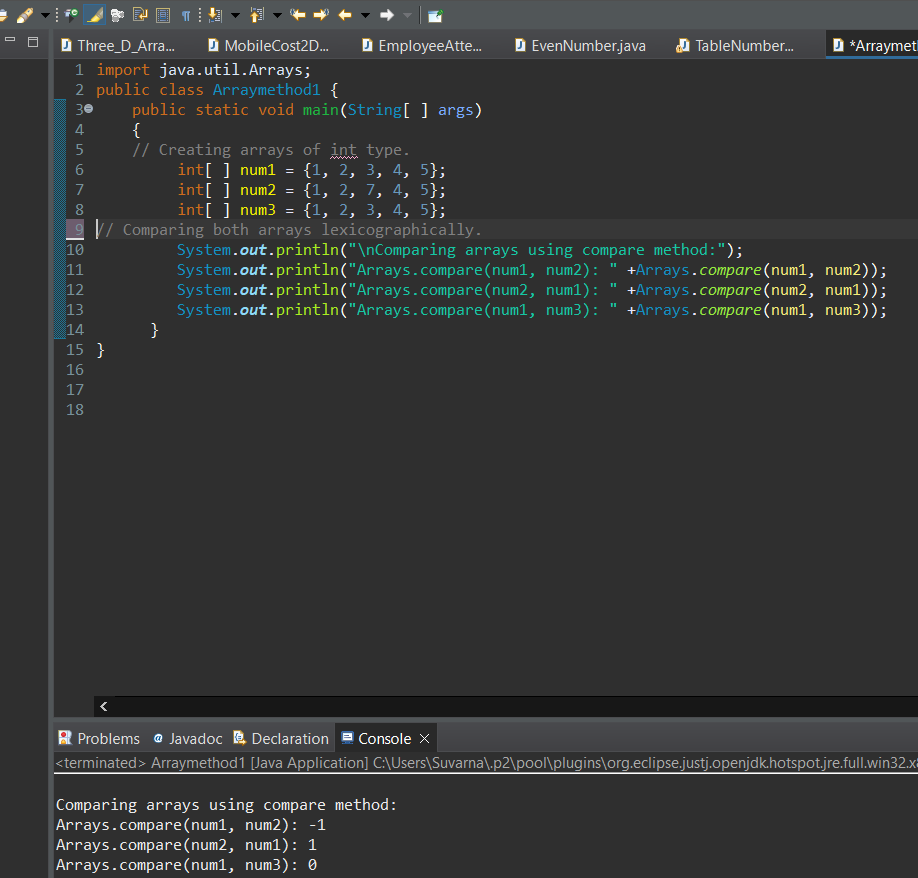
If the first array is lexicographically less than second array, it returns a value less than 0.

The compare() method has the following general form that are as:

* static int compare​(boolean[ ] array1, boolean[ ] array2)
* static int compare​(byte[ ] array1, byte[ ] array2)
* static int compare​(char[ ] array1, char[ ] array2)
* static int compare​(double[ ] array1, double[ ] array2)
* static int compare​(float[ ] array1, float[ ] array2)
* static int compare​(int[ ] array1, int[ ] array2)
* static int compare​(long[ ] array1, long[ ] array2)
* static int compare​(short[ ] array1, short[ ] array2)
* static <T extends Comparable<? super T>> int compare​(T[ ] array1, T[ ] array2)
* static <T> int compare​(T[ ] array1, T[ ] array2, Comparator<? super T> cmp)

The last two forms of compare() methods compare two Object arrays within comparable elements, lexicographically and by using a specified comparator respectively.

**Program code:**



**Method:2 copyOf(arrayOriginal, newLength):** This method was added in arrays class in Java 6 version. It returns an array that is copy of arrayOriginal. newLength represents the length of copy.

If the length of copy is longer than arrayOriginal, then the copy is padded with zeros (for numeric arrays), nulls (for object arrays), or false (for boolean arrays).

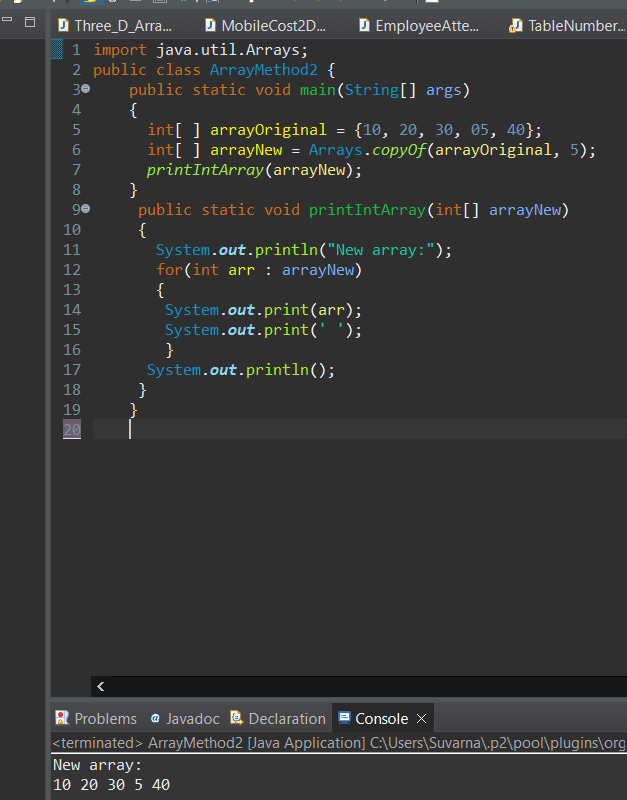
If the length of copy is shorter than arrayOriginal, the copy is truncated. That is, the method does not copy all of the original array’s value.

If newLength is negative, a NegativeArraySizeException is thrown. If arrayOriginal is null, a NullPointerException is thrown. The copyOf() method comes in the following flavour.

* static boolean[ ] copyOf(boolean[ ] arrayOriginal, int newLength)
* static byte[ ] copyOf(byte[ ] arrayOriginal, int newLength)
* static char[ ] copyOf(char[ ] arrayOriginal, int newLength)
* static double[ ] copyOf(double[ ] arrayOriginal, int newLength)
* static float[ ] copyOf(float[ ] arrayOriginal, int newLength)
* static int[ ] copyOf(int[ ] arrayOriginal, int newLength)
* static long[ ] copyOf(long[ ] arrayOriginal, int newLength)
* static short[ ] copyOf(short[ ] arrayOriginal, int newLength)
* static <T> T[ ] copyOf(T[ ] arrayOriginal, int newLength)

Let’s take an example program where we will copy all elements of an array into another array.

**Program code:**



f we write the statement int[ ] arrayNew = Arrays.copyOf(arrayOriginal, 2);, then the output will be 10, 20, 30.

If we write the statement int[ ] arrayNew = Arrays.copyOf(arrayOriginal, 7);, then the output will be 10, 20, 30, 5, 40, 0, 0.

**Method:3 copyOfRange(arrayOriginal, from, to):** This method was also added in arrays class in Java 1.6 version. The copyOfRange() method returns a copy of range (from one index to another) within an array.

If the range is longer than arrayOriginal, the copy is padded with zeros (for numeric arrays), nulls (for object arrays), or false (for boolean arrays).

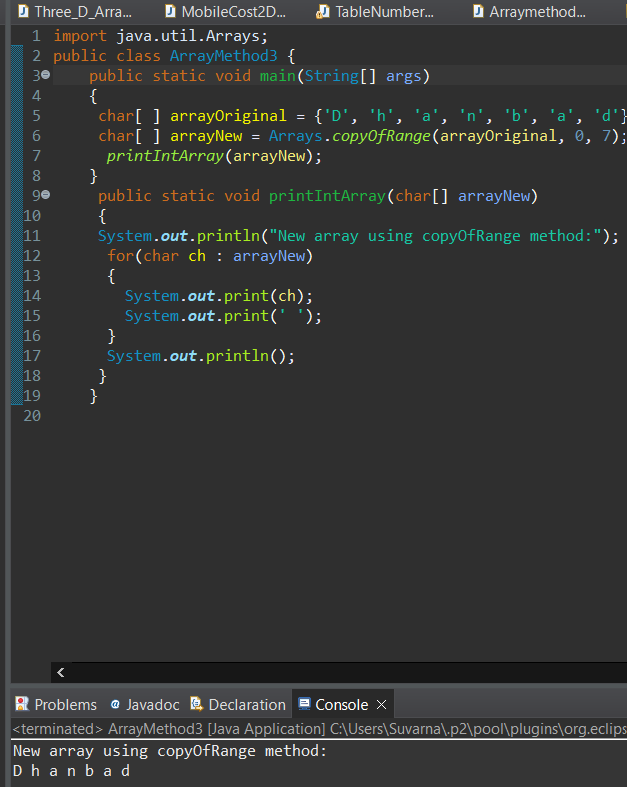
If from is negative or greater than the length of arrayOriginal, an ArrayIndexOutOfBoundsException is thrown. If from is greater than to, an IllegalArgumentException is thrown. If arrayOriginal is null, a NullPointerException is thrown.

The copyOfRange() method comes in the following flavour:

* static boolean[ ] copyOfRange(boolean[ ] arrayOriginal, int start, int end)
* static byte[ ] copyOfRange(byte[ ] arrayOriginal, int from, int to)
* static char[ ] copyOfRange(char[ ] arrayOriginal, int from, int to)
* static double[ ] copyOfRange(double[ ] arrayOriginal, int from, int to)
* static float[ ] copyOfRange(float[ ] arrayOriginal, int from, int to)
* static int[ ] copyOfRange(int[ ] arrayOriginal, int from, int to)
* static long[ ] copyOfRange(long[ ] arrayOriginal, int from, int to)
* static short[ ] copyOfRange(short[ ] arrayOriginal, int from, int to)
* static <T> T[ ] copyOfRange(T[ ] arrayOriginal, int from, int to)

Let’s create a program where we will copy a range of elements of an array into another array.

**Program code:**

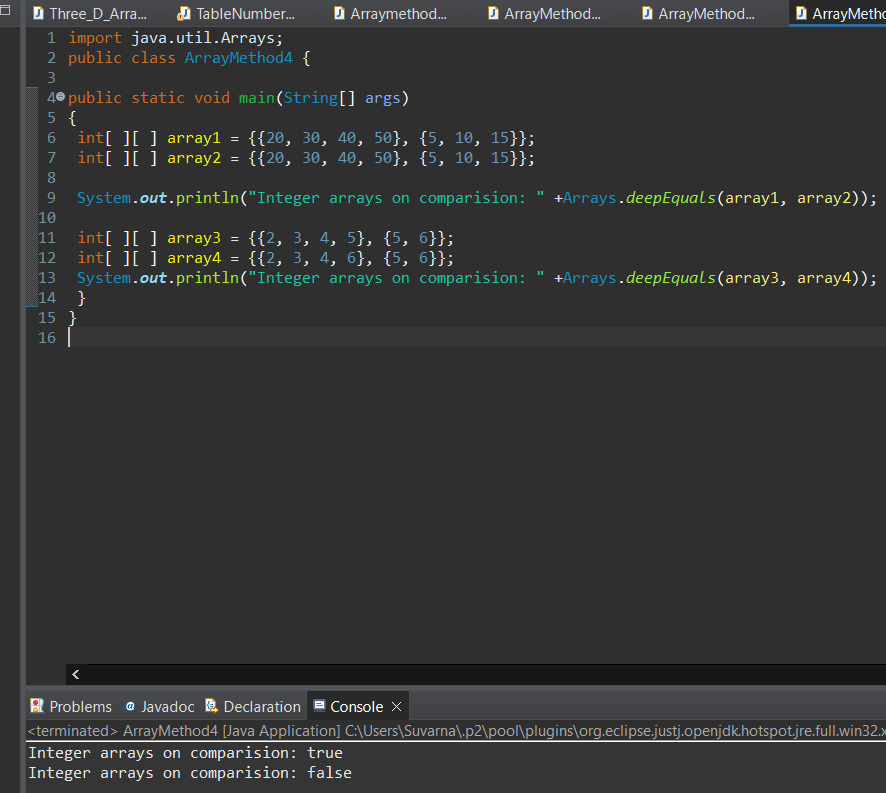


**Method:4 deepEquals():** The deepEquals() method (added in arrays class in Java 5 version) returns true if the two specified arrays are deeply equal to one another. This method works for arrays of 2D or more dimensions. It has only one form:

* static boolean deepEquals(Object[ ] array1, Object[ ] array2)

It returns false if element values of arrays, or any nested arrays, differ. Let’s take an example program based on the deepEquals() method.

**Program code:**

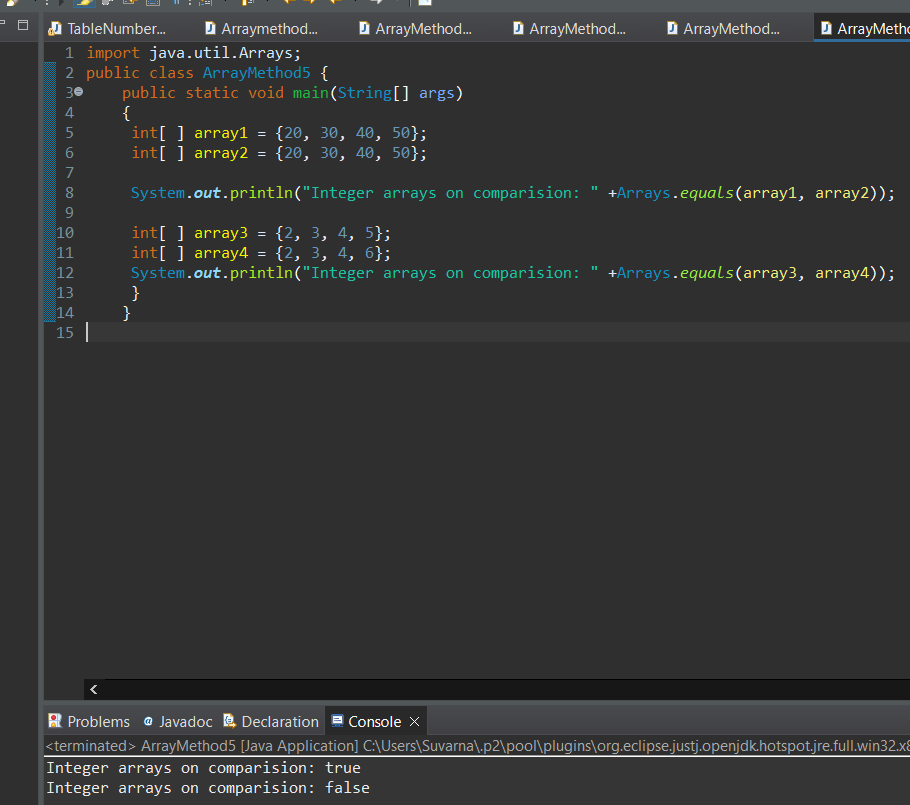


**Method:5 equals(array1, array2):** The equals() method is used to check equality for [one-dimensional array](https://www.scientecheasy.com/2021/08/one-dimensional-array-in-java.html/). It returns true if two arrays are equivalent. The equals( ) method comes in the following flavors:

* static boolean equals(boolean array1[ ], boolean array2[ ])
* static boolean equals(byte array1[ ], byte array2[ ])
* static boolean equals(char array1[ ], char array2[ ])
* static boolean equals(double array1[ ], double array2[ ])
* static boolean equals(float array1[ ], float array2[ ])
* static boolean equals(int array1[ ], int array2[ ])
* static boolean equals(long array1[ ], long array2[ ])
* static boolean equals(short array1[ ], short array2[ ])
* static boolean equals(Object array1[ ], Object array2[ ])

Let’s take an example program based on the equals() method where we will check equality for 1D array.

**Program code:**

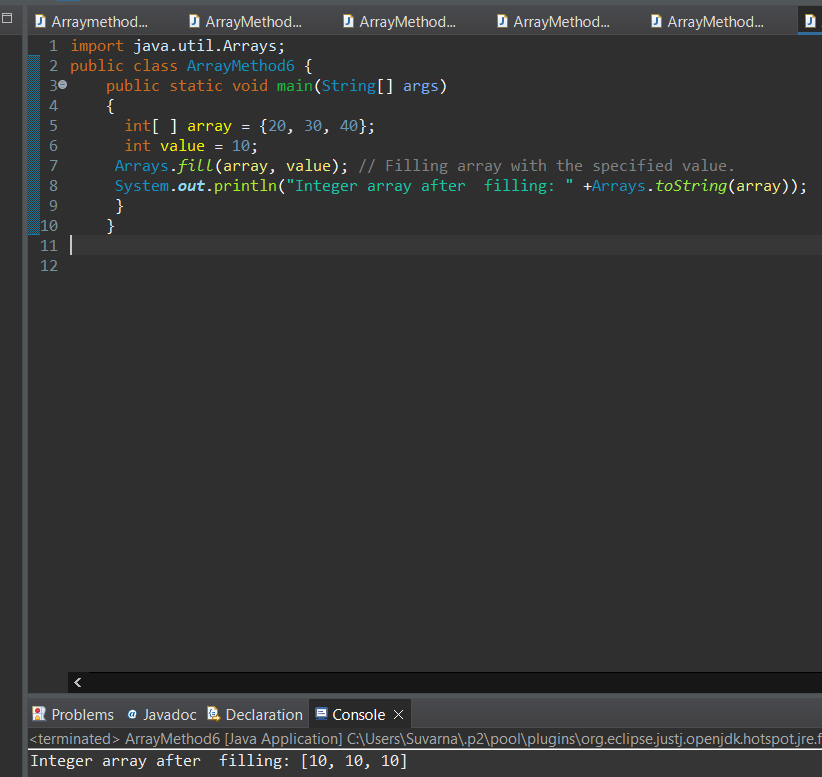


**Method:6 fill(array, value):** The fill() method is used to fill an array with the specified value. It has the following general forms:

* static void fill(boolean array[ ], boolean value)
* static void fill(byte array[ ], byte value)
* static void fill(char array[ ], char value)
* static void fill(double array[ ], double value)
* static void fill(float array[ ], float value)
* static void fill(int array[ ], int value)
* static void fill(long array[ ], long value)
* static void fill(short array[ ], short value)
* static void fill(Object array[ ], Object value)

Let’s create a program to fill an array with a specified value.

**Program code:**

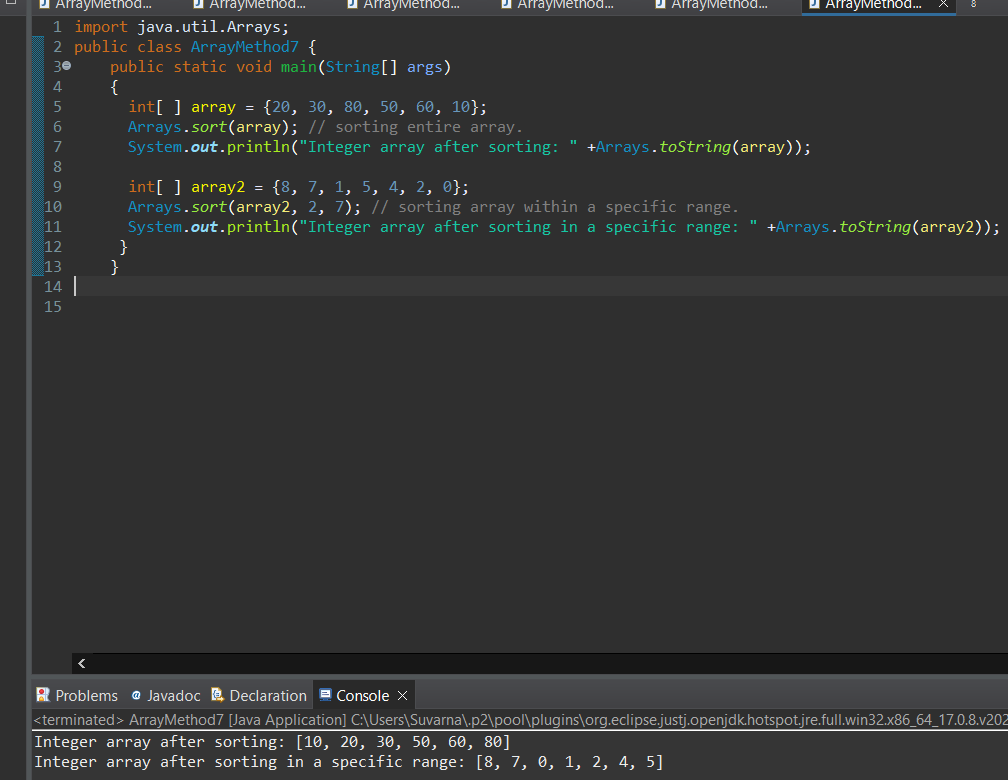


**Method:7 sort(array, from, to):** This version of sort() method is used to sort elements of an array within a specific range in ascending order. The following general forms of this method is available in java:

* static void sort(byte array[ ], int from, int to)
* static void sort(char array[ ], int from, int to)
* static void sort(double array[ ], int from, int to)
* static void sort(float array[ ], int from, int to)
* static void sort(int array[ ], int from, int to)
* static void sort(long array[ ], int from, int to)
* static void sort(short array[ ], int from, int to)
* static void sort(Object array[ ], int from, int to)
* static <T> void sort(T array[ ], int from, int to, Comparator<? super T> c)

Let’s take an example program where we will sort elements of an entire array and within a specific range in ascending order.

**Program code:**



**Method:8 stream(array, from, to):** This version of stream() method is used to convert array into a stream with a specific range. It has the following overloaded forms available in java:

* static DoubleStream stream​(double[ ] array, int fromInclusive, int toExclusive)
* static IntStream stream​(int[ ] array, int fromInclusive, int toExclusive)
* static LongStream stream​(long[ ] array, int fromInclusive, int toExclusive)
* static <T> Stream<T> stream​(T[ ] array, int startInclusive, int endExclusive)

Let’s take an example program based on toString(), and stream() methods.

**Program code:**

