Array

An array is a collection of similar data types. Array is a container object that hold values of homogenous type. It is also known as static data structure because size of an array must be specified at the time of its declaration.

An array can be either primitive or reference type. It gets memory in heap area. Index of array starts from zero to size-1.

**Array Declaration**

**Syntax :**

datatype[] identifier;

or

datatype identifier[];

Both are valid syntax for array declaration. But the former is more readable.

**Example :**

int[] arr;

char[] arr;

short[] arr;

long[] arr;

int[][] arr; //two dimensional array.

**Initialization of Array**

new operator is used to initialize an array.

**Example :**

int[] arr = new int[10]; //10 is the size of array.

or

int[] arr = {10,20,30,40,50};

**Accessing array element**

As mention ealier array index starts from 0. To access nth element of an array. Syntax

arrayname[n-1];

**Example : To access 4th element of a given array**

int[] arr={10,20,30,40};

System.out.println("Element at 4th place"+arr[3]);

The above code will print the 4th element of array arr on console.

**foreach or enhanced for loop**

J2SE 5 introduces special type of for loop called foreach loop to access elements of array. Using foreach loop you can access complete array sequentially without using index of array. Let us see an exapmle of foreach loop.

class Test

{

public static void main(String[] args)

{

int[] arr={10,20,30,40};

for(int x:arr)

{

System.out.println(x);

} }}

**output**: 10

20

30

40

**Copying Arrays**

The System class has an arraycopy method that you can use to efficiently copy data from one array into another:

public static void arraycopy(Object src, int srcPos,

Object dest, int destPos, int length)

The two Object arguments specify the array to copy from and the array to copy to. The three int arguments specify the starting position in the source array, the starting position in the destination array, and the number of array elements to copy.

The following program, ArrayCopyDemo, declares an array of char elements, spelling the word "decaffeinated." It uses the System.arraycopy method to copy a subsequence of array components into a second array:

class ArrayCopyDemo {

public static void main(String[] args) {

char[] copyFrom = { 'd', 'e', 'c', 'a', 'f', 'f', 'e',

'i', 'n', 'a', 't', 'e', 'd' };

char[] copyTo = new char[7];

System.arraycopy(copyFrom, 2, copyTo, 0, 7);

System.out.println(new String(copyTo));

}}

**Output**

caffein

**Array Manipulations**

Arrays are a powerful and useful concept used in programming. Java SE provides methods to perform some of the most common manipulations related to arrays. For instance, the ArrayCopyDemo example uses the arraycopy method of the System class instead of manually iterating through the elements of the source array and placing each one into the destination array. This is performed behind the scenes, enabling the developer to use just one line of code to call the method.

For your convenience, Java SE provides several methods for performing array manipulations (common tasks, such as copying, sorting and searching arrays) in the java.util.Arrays class. For instance, the previous example can be modified to use the copyOfRange method of the java.util.Arrays class, as you can see in the ArrayCopyOfDemo example. The difference is that using the copyOfRange method does not require you to create the destination array before calling the method, because the destination array is returned by the method:

class ArrayCopyOfDemo {

public static void main(String[] args) {

char[] copyFrom = {'d', 'e', 'c', 'a', 'f', 'f', 'e',

'i', 'n', 'a', 't', 'e', 'd'};

char[] copyTo = java.util.Arrays.copyOfRange(copyFrom, 2, 9);

System.out.println(new String(copyTo));

}}

**Other useful operations provided by methods in the java.util.Arrays class, are:**

1. Searching an array for a specific value to get the index at which it is placed (the binarySearch method).
2. Comparing two arrays to determine if they are equal or not (the equals method).
3. Filling an array to place a specific value at each index (the fill method).
4. Sorting an array into ascending order. This can be done either sequentially, using the sort method, or concurrently, using the parallelSort method introduced in Java SE 8. Parallel sorting of large arrays on multiprocessor systems is faster than sequential array sorting.

**Passing Array to method in java**

We can pass the java array to method so that we can reuse the same logic on any array.

Let's see the simple example to get minimum number of an array using method.

class Testarray2{

static void min(int arr[]){

int min=arr[0];

for(int i=1;i<arr.length;i++)

if(min>arr[i])

min=arr[i];

System.out.println(min);

}

public static void main(String args[]){

int a[]={33,3,4,5};

min(a);//passing array to method

}}

**Output**:3

**Multidimensional array in java**

In such case, data is stored in row and column based index (also known as matrix form).

**Syntax to Declare Multidimensional Array in java**

dataType[][] arrayRefVar; (or)

dataType [][]arrayRefVar; (or)

dataType arrayRefVar[][]; (or)

dataType []arrayRefVar[];

**Example of Multidimensional java array**

Let's see the simple example to declare, instantiate, initialize and print the 2Dimensional array.

class Testarray3{

public static void main(String args[]){

//declaring and initializing 2D array

int arr[][]={{1,2,3},{2,4,5},{4,4,5}};

//printing 2D array

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

System.out.print(arr[i][j]+" ");

} System.out.println();

} }}

**Output**:1 2 3

2 4 5

4 4 5

**What is the class name of java array?**

In java, array is an object. For array object, a proxy class is created whose name can be obtained by getClass().getName() method on the object.

class Testarray4{

public static void main(String args[]){

int arr[]={4,4,5};

Class c=arr.getClass();

String name=c.getName();

System.out.println(name);

}}

**Output**:I