Arrays

Arrays: An array represents a group of elements of same data type. Arrays are generally categorized into two types:

- ✓ Single Dimensional arrays (or 1 Dimensional arrays)
- ✓ Multi-Dimensional arrays (or 2 Dimensional arrays, 3 Dimensional arrays, …)

Single Dimensional Arrays: A one dimensional array or single dimensional array represents a row or a column of elements. For example, the marks obtained by a student in 5 different subjects can be represented by a 1D array.

- We can create a 1D array by declaring the array first and then allocate memory for it by using new operator, as:

```
int marks[];              //declare marks array
marks = new int[5];       //allot memory for storing 5 elements
```

These two statements also can be written as:

```
int marks [] = new int [5];
```

Program 1: Write a program to accept elements into an array and display the same.

```
// program to accept elements into an array and display the same.
import java.io.*;
class ArrayDemo1
{       public static void main (String args[]) throws IOException
        {       //Create a BufferedReader class object (br)
                BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
                System.out.println ("How many elements: " );
                int n = Integer.parseInt (br.readLine ());
                //create a 1D array with size n
                int a[] = new int[n];
                System.out.print ("Enter elements into array : ");
                for (int i = 0; i<n;i++)
                        a [i] = Integer.parseInt ( br.readLine ());
                System.out.print ("The entered elements in the array are: ");
                for (int i =0; i < n; i++)
                        System.out.print (a[i] + "\t");
        }
}
```

Multi-Dimensional Arrays (2D, 3D … arrays): Multi dimensional arrays represent 2D, 3D …

arrays. A two dimensional array is a combination of two or more (1D) one dimensional arrays. A three dimensional array is a combination of two or more (2D) two dimensional arrays.

· **Two Dimensional Arrays (2d array):** A two dimensional array represents several rows and columns of data. To represent a two dimensional array, we should use two pairs of square braces [ ] [ ] after the array name. For example, the marks obtained by a group of students in five different subjects can be represented by a 2D array.

We can declare a two dimensional array and directly store elements at the time of its declaration, as:

int marks[] [] = {{50, 60, 55, 67, 70},{62, 65, 70, 70, 81}, {72, 66, 77, 80, 69} };

We can create a two dimensional array by declaring the array first and then we can allot memory for it by using new operator as

```
int marks[][] // declare mark array
marks =new int[3][5] // allot memory for storing five element
This two statement can be written as:
                        int marks[][]=new int[3][5];
```

Program 2: Write a program to take a 2D array and display its elements in the form of a matrix.

```
//Displaying a 2D array as a matrix
class Matrix
{       public static void main(String args[])
        {       //take a 2D array
                int x[ ][ ] = {{1, 2, 3}, {4, 5, 6} };
                // display the array elements
                for (int i = 0 ; i < 2 ; i++)
                {       System.out.println ();
                        for (int j = 0 ; j < 3 ; j++)
                                System.out.print(x[i][j] + "\t");
                }
        }
}
```

Three Dimensional arrays (3D arrays): We can consider a three dimensional array as a combination of several two dimensional arrays. To represent a three

dimensional array, we should use three pairs of square braces [ ] [ ] after the array name.

o We can declare a three dimensional array and directly store elements at the time of its declaration, as:

```
int arr[ ] [ ] [ ] = {{{50, 51, 52},{60, 61, 62}}, {{70, 71, 72}, {80, 81, 82}}};
```

We can create a three dimensional array by declaring the array first and then we can allot memory for it by using new operator as:

```
int arr[ ] [ ] = new int[2][2][3]; //allot memory for storing 15 elements.
```