**Arrays:**

An array is used to store a collection of data of same type, an array is a fixed in size stores the data in sequential manner.

**Assignment:**

iArr = new int[10];

**Declaration:**

int iArr[];

int[] iArr;

**Declaration&Assignment:**

int[] iArr = new int[5];

iArr[0]=10;

iArr[1]=20;

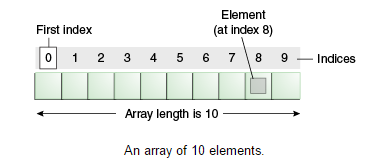
iArr[2]=30;

iArr[3]=40;

iArr[4]=50;

**Alternative way to create array:**

int iArr[] = {10,20,30,40,50,60,70,80,90,100};



Declaring arrays of other types:

byte[] anArrayOfBytes;

short[] anArrayOfShorts;

long[] anArrayOfLongs;

float[] anArrayOfFloats;

double[] anArrayOfDoubles;

boolean[] anArrayOfBooleans;

char[] anArrayOfChars;

String[] anArrayOfStrings;

The array elements are accessed through the **index**. Array indices are 0-based; that is, they start from 0 to **arrayRefVar.length-1**.

**Accessing array elements:**

When processing array elements, we often use either for loop or foreach loop because all of the elements in an array are of the same type and the size of the array is known.

**public** **class** Array

{

**public** **static** **void** main(String[] args)

{

**int**[] iArr = {10,20,30,40,50,60,70,80,90,100};

**for**(**int** i=0;i<iArr.length;i++)

{

System.*out*.println(iArr[i]);

}

**for** (**int** j : iArr)

{

System.*out*.println(j);

}

}

}

**Passing array to method & returning array from a method:**

**public** **class** Array

{

**public** **static** **void** main(String[] args)

{

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

*printArray*(iArr); //pass by reference

*modifyArray*(iArr);

*printArray*(iArr); //10,20,30,40,120

**int**[] rArray = *reverseArray*(iArr);

*printArray*(rArray); //

}

**private** **static** **int**[] reverseArray(**int**[] array)

{

**int**[] revArr = **new** **int**[array.length];

**for**(**int** i=array.length-1,j=0; i>=0 ;i--,j++)

{

revArr[j]=array[i];

}

**return** revArr;

}

**private** **static** **void** modifyArray(**int**[] array)

{

array[4]=120;

}

**private** **static** **void** printArray(**int**[] array) //int iArr[] also valid

{

**for** (**int** j : array)

{

System.*out*.println(j);

}

}

}

**Note:**

int iArr[] = new int[5]; what if I print array? Ans: it contains the default values i.e 0,0,0,0,0

**Two Dimensional Arrays**

 int[][] A = new int[3][4]; // 3 rows , 4 columns

int[][] wrong = new int[][]; // not OK, you must specify 1st dimension

int[][] right = new int[2][]; // OK

To process a two-dimensional array, we use **nested for loops**.

**Code: Java**

**int[][] a2 = new int[10][5];**

**// print array in rectangular form**

**for (int i=0; i<a2.length; i++) {**

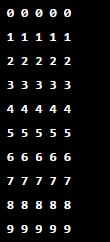
**for (int j=0; j<a2[i].length; j++) {**

**System.out.print(" " + a2[i][j]);**

**}**

**System.out.println("");**

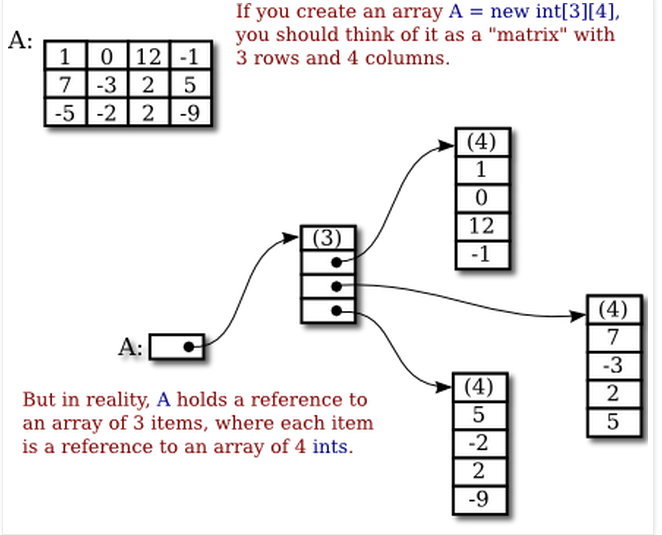
**}**

****

The Java programming language does not really support multi-dimensional arrays. It does, however, supports an array of arrays. In Java, a two-dimensional array 'x' is an array of one-dimensional array. For instance :-

  int[][] x = new int[3][5];

The expression x[i] is used to select the one-dimensional array; the expression x[i][j] is ued to select the element from that array. The first element of this array will  be indexed with the "0" value and the last integer will be referenced by "length-1" indexed value.

****