LAB # 5

ARRAYS IN JAVA

OBJECTIVE

To Study Java One Dimentional and Two Dimentional Arrays.

THEORY

Arrays

An array is a group of like-typed variables that are referred to by a common name. Arrays of any type can be created and may have one or more dimensions. A specific element in an array is accessed by its index. Arrays offer a convenient means of grouping related information.

One-Dimensional Arrays

A one-dimensional array is, essentially, a list of like-typed variables. To create an array, you first must create an array variable of the desired type. The general form of a one dimensional array declaration is

```
int month_days[];
```

Although this declaration establishes the fact that **month_days** is an array variable,
no array actually exists. In fact, the value of **month_days** is set to **null**, which represents an array with no value. To link **month_days** with an actual, physical array of integers, you must allocate one using **new** and assign it to **month_days**. **new** is a special operator that allocates memory.
The general form of **new** as it applies to one-dimensional arrays appears as follows:

```
array-var = new type[size];
```

It is possible to combine the declaration of the array variable with the allocation of
the array itself, as shown here:

```
int month_days[] = new int[12];
```

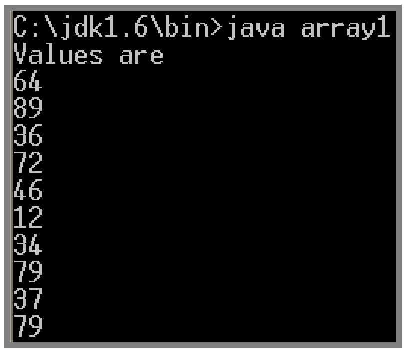
Prorgam # 1

**This program fill an array of 10 elements by randomly generated Integers,
range (1-100).**

```
import java.lang.Math;
public class array1
{
   public static void main(String args[])
   {
    int a[] = new int [10];
    for (int i=0;i<10;i++)
      a[i]=(int)(Math.random()*100);

    System.out.println("Values are");
    for (int i=0;i<10;i++)
       System.out.println(a[i]);
  }
}
```

Output:

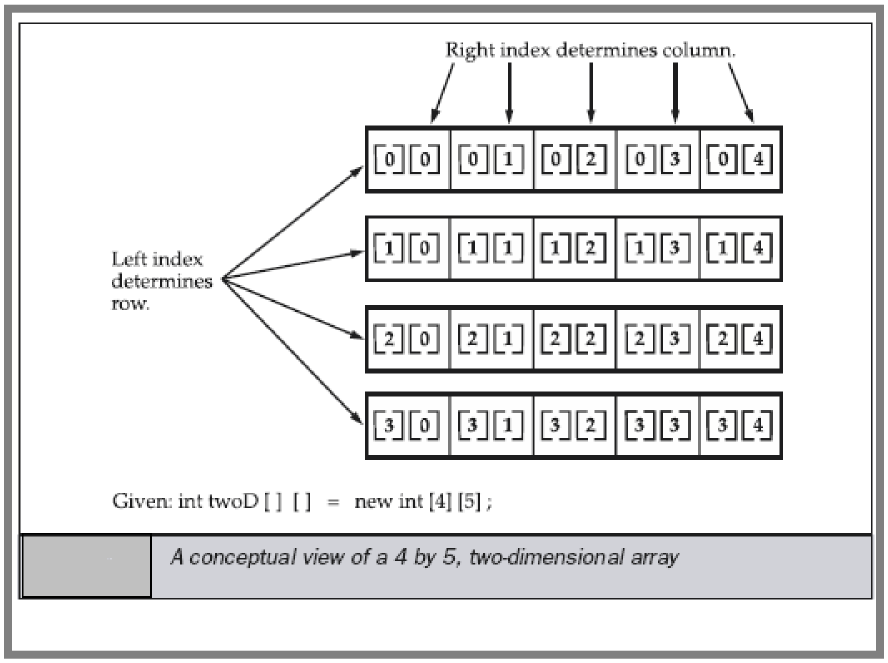


```
C:\jdk1.6\bin>java array1
Values are
64
89
36
72
46
12
34
79
37
79
```

Multidimensional Arrays

In Java, *multidimensional arrays* are actually arrays of arrays. These, as you might expect, look and act like regular multidimensional arrays.
For example, the following declares a two-dimensional array variable called **twoD**.

```
int twoD[][] = new int[4][5];
```

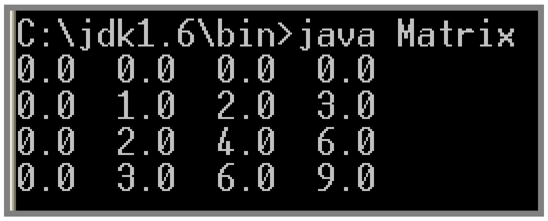


A conceptual view of a 4 by 5, two-dimensional array

It is possible to initialize multidimensional arrays. To do so, simply enclose each dimension's initializer within its own set of curly braces. The following program creates a matrix where each element contains the product of the row and column indexes. Also notice that you can use expressions as well as literal values inside of array initializers.

```
// Initialize a two-dimensional array.
class Matrix {
public static void main(String args[]) {
double m[][] = {
{ 0*0, 1*0, 2*0, 3*0 },
{ 0*1, 1*1, 2*1, 3*1 },
{ 0*2, 1*2, 2*2, 3*2 },
{ 0*3, 1*3, 2*3, 3*3 }
};
int i, j;
for(i=0; i<4; i++) {
for(j=0; j<4; j++)
System.out.print(m[i][j] + "  ");
System.out.println();
}
}
}
```

Output:



```
C:\jdk1.6\bin>java Matrix
0.0  0.0  0.0  0.0
0.0  1.0  2.0  3.0
0.0  2.0  4.0  6.0
0.0  3.0  6.0  9.0
```

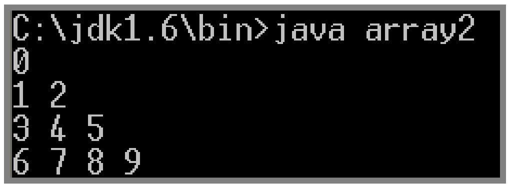
When you allocate memory for a multidimensional array, you need only specify the memory for the first (leftmost) dimension. You can allocate the remaining dimensions separately.

when you allocate dimensions manually, you do not need to allocate the same number of elements for each dimension. As stated earlier, since multidimensional arrays are actually arrays of arrays, the length of each array is under your control. For example, the following program creates a two dimensional array in which the sizes of the second dimension are unequal.

Prorgam # 2
This program manually allocate differing size second dimensions.

```
public class array2 {
public static void main(String args[]) {
int twoD[][] = new int[4][];
twoD[0] = new int[1];
twoD[1] = new int[2];
twoD[2] = new int[3];
twoD[3] = new int[4];
int i, j, k = 0;
for(i=0; i<4; i++)
for(j=0; j<i+1; j++) {
twoD[i][j] = k;
k++;
}
for(i=0; i<4; i++) {
for(j=0; j<i+1; j++)
System.out.print(twoD[i][j] + " ");
System.out.println();
}
}
}
```

Output:



```
C:\jdk1.6\bin>java array2
0
1 2
3 4 5
6 7 8 9
```

LAB TASK

1. Write a program that reads (fictitious) student test scores in the range 0 through 100 and print the following statistics to two decimal places:

   The average (mean) score.
   The student with the highest score.
   The student with the lowest score.
   The number of students whose score equal or exceed the average.

   For each student:

   The difference between the average score and the student's score (this can be either positive or negative).
   The grade letter where
   A is a score of 90 or greater.
   B is a score of 80 through 89.99.
   C is a score of 70 through 79.99
   D is a score of 60 through 69.99
   E is a score of less than 60.

2. Write a program that will allow two users to play **Tic-Tac-Toe** game partially. The program should ask for moves alternately from player X and player O. The program should display the game position as follows:

```
1    2    3
4    5    6
7    8    9
```

   The players enter their moves by entering the position number they wish to mark (*Assume that the player always enters a valid position*). The program then displays the changed board. A sample board configurations are:

```
x    o    3          x    2    3          x    o    3
4    5    6          4    5    6          4    x    6
7    8    9          7    8    9          7    8    9
```