Arrays

The variables that we have used till now are capable of storing only one value at a time. Consider a situation when we want to store and display the age of 100 employees. For this we have to do the following:

1. Declare 100 different variables to store the age of employees.
2. Assign a value to each variable.
3. Display the value of each variable.

Although we can perform our task by the above three steps but just imagine how difficult it would be to handle so many variables in the program and the program would become very lengthy. The concept of arrays is useful in these types of situations where we can group similar type of data items.

An array is a collection of similar type of data items and each data item is called an element of the array. The data type of the elements may be any valid data type like char, int or float etc. The elements of array share the same variable name but each element has a different **index number known as subscript**. For the above problem we can take an array variable age[100] of type int. The size of this array variable is 100 so it is capable of storing 100 integer values. The individual elements of this array are

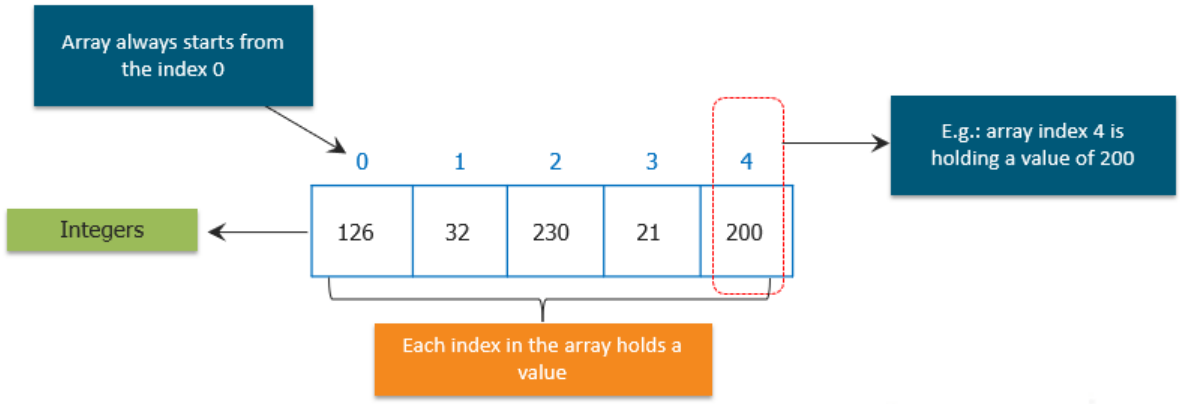
age[0], age[1], age[2], age[3], age[4], age[98], age[99]

In Java the subscripts start from zero, so age[0] is the first element, age[1] is the second element of array and so on.

Arrays can be single dimensional or multidimensional. The number of subscripts determines the dimension of array. A one-dimensional array has one subscript, two dimensional array has two subscripts and son on.

What are Java Arrays?

Arrays in Java are homogeneous data structures implemented in Java as **objects**. Arrays store one or more values of a specific data type and provide indexed access to store the same. A specific element in an array is accessed by its index. Arrays offer a convenient means of grouping related information.

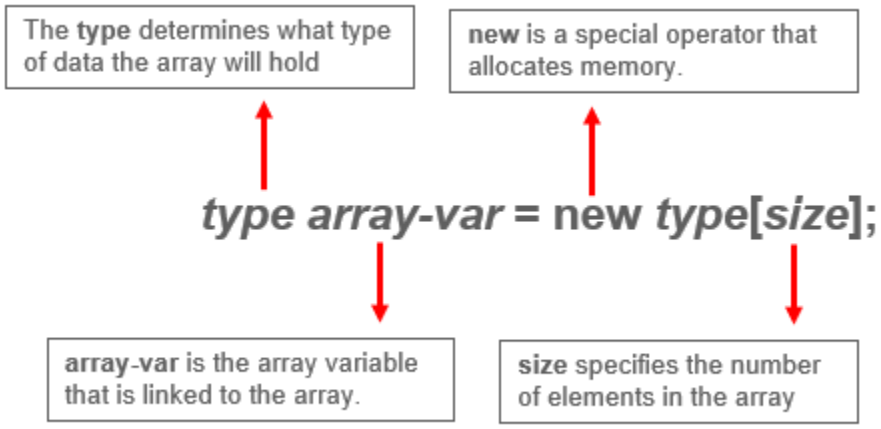


One Dimensional Array:

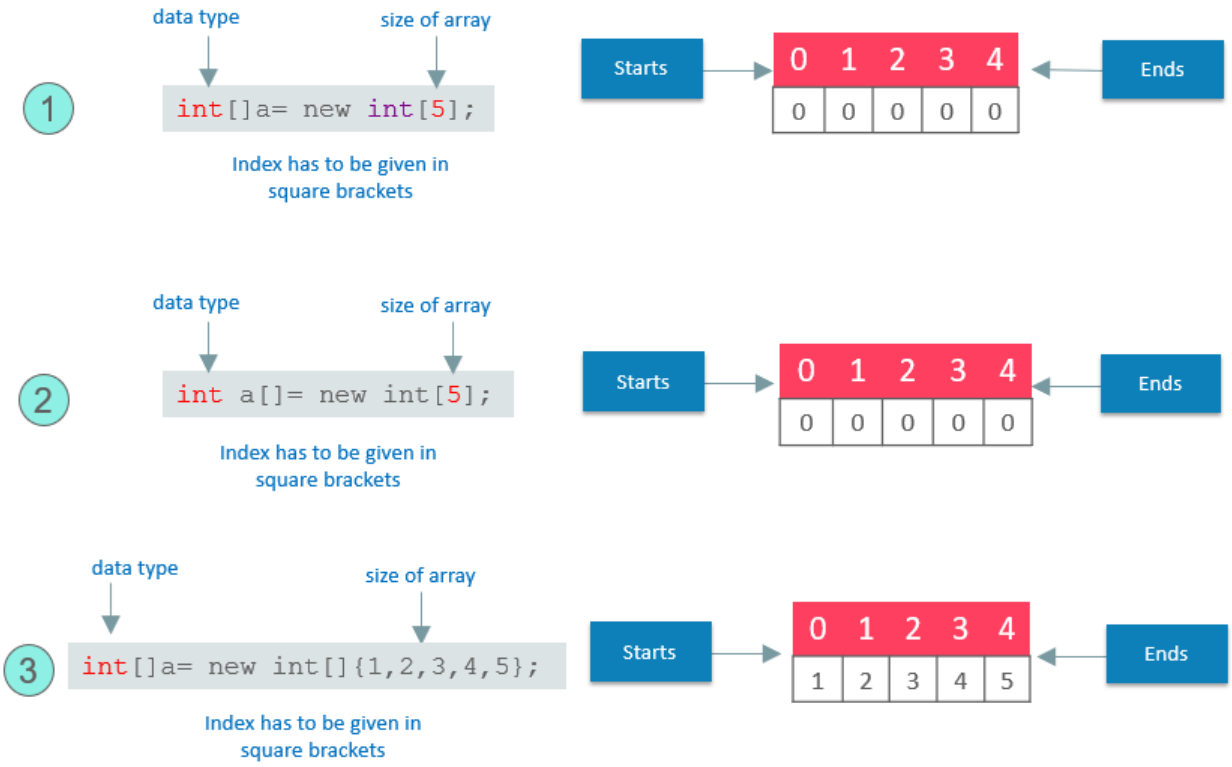
Declaration : In three way we can declare one dimensional array
int[] a;
int []a;
int a[];
Creation : Once we declare array, we can create array (memory allocation)
a= new int[5];
Initialization: using index value we can initialize individual array element
a[0] =10; a[1] = 20; and so on

Declaration and creation of array can be done in one line also
int[] a = new int[5];
Declaration, creation and initialization can be done in single line also (Arrays can be initialized when they are declared. The array will automatically be created large enough to hold the number of elements you specify in the array initializer. There is **no** need to use **new**.)
int[] a = {10,20,30,40,50};

General Form of Java Array Initialization



Example:-



Accessing I-D Array Elements

The elements o fan array can be accessed by specifying the array name followed by subscript or index in brackets. In Java, the array subscripts start from 0. Hence if there is an array of size 5 then the valid index will be from 0 to 4. The last valid index is one less than the size of the array. This last valid subscript is sometimes know as the upper bound of the array and 0 is known as the lower bound of the array.

Let us take an array

int[] arr= new arr[5]; /*Size of array arr is 5, can hold five integer elements*/

The elements of this array are

arr[0], arr[1], arr[2], arr[3], arr[4]

Here 0 is the lower bound and 4 is the upper bound of the array. Index can be any expression that yields an integer value. It can be any integer constant, integer -variable, integer expression or return value(int) from a method call.

For example, if i and j are integer variables then these are some valid indexed array elements

arr[3]

arr[i]

arr[i+j]

arr[2*j]
arr[i++]

Processing 1-D Arrays

For processing arrays we generally use a for loop and the loop variable is used at the place of index The initial value of loop variable is taken 0 since array index start from zero. The loop variable is increased by 1 each time so that we can access and process the next element in the array. The total number of passes in the loop will be equal to the number of elements in the array and in each pass we will process one element.

Suppose arr[10] is an array of int type

(i) Reading values in arr[10]

```
 Scanner s = new Scanner(System.in);
for( i = 0; i < arr.length; i++)
      arr[i] = s.nextInt();
```

(ii) Displaying values of arr[10]

```
           for(int i = 0; i < arr.length; i++)
           System.out.printf(%d ", arr[i]);
```

(iii) Adding all the elements of arr[10]

```
    sum = 0;
     for(int i = 0; i < arr.length; i++)
         sum+=arr[i];
```

- In Java all arrays are dynamically allocated.
- Since arrays are objects in Java, we can find their length using member **length**.
- We can display the length of an array using
  System.out.println(a.length);
- To know the name of array object's corresponding class

  System.out.println(a.getClass().getName());

Program to input values into an array and display them

```
import java.util.*;
 class  ArrayReadAndDisplay
 {
  public static void main (String[] args )
 {
   Scanner s = new Scanner(System.in);
    int[] arr = new int[5];
     int j,i;
     for(i=0;i<arr.length;i++)
         {
          System.out.printf ("Enter the value for arr[ %d] :\n" ,i);
           arr[i] = s.nextInt();
         }
 System.out.println("The array elements are :");
   for(i=0;i<arr.length;i++)
       {
         System.out. printf("%d\t",arr[i]);
} } }
```

Program to add the elements of an array

```
import java.util.*;
 class  ArrayElementSum
 {
  public static void main (String[] args )
 {
   Scanner s = new Scanner(System.in);
  int[]  arr = new int[10];
  int ,i,sum=0;
  for(i=0;i<10;i++) {
         System.out.printf ("Enter the value for arr[ %d] :\n" ,i);
          arr[i] = s.nextInt();
         sum+=arr[i];
   }
System.out.printf ( ("Sum = %d\n", sum) ;
}
}
```

Program to count the even and odd numbers in a array

```
import java.util.*;
 class  ArrayEvenAndOdd
 {
  public static void main (String[] args )
 {
   Scanner s = new Scanner(System.in);
  System.out.println("Enter the size of the array :");
  int size = s.nextInt();
  int[]  arr = new int [size];
 int  I, even=0, odd=0;
 for(i=0;i<arr.length;i++)
   {
    System.out.printf ("Enter the value for arr [%d] :",i);
    arr[i] = s.nextInt() ;
    if(arr[i]%2==0)
       even++;.
    else
      odd++;
   }
System.out.printf ("Even numbers %d, Odd numbers =%d\n", even, odd );
}
}
```

PRESTIGE POINT INDORE

Program to find the maximum and minimum number in an array

```
import java.util.*;
 class  ArrayMinAndMax
  {
   public static void main (String[] args )
  {
    Scanner s = new Scanner(System.in);
    int i, j , arr[ ] = {2 , 5, 4, 1, 8, 9, 11 , 6, 3 , 7} ;
   int min, max;
   min=max=arr[0] ;
   for(i=l;i<arr.length;i++)
     {
      if(arr[i]<min)
        min=arr[i] ;
      if(arr[i]>max)
      max=arr[i] ;
       }
      System.out.printf("Minimum  =  %d, Maximum = %d", min,max);
  }
}
```

Program to reverse the elements of an array

```
import java.util.*;
 class  ArrayReverse
  {
   public static void main (String[] args )
  {
      Scanner s = new Scanner(System.in);
      int i,j,temp,arr[ ] = {1,2,3,4,5,6,7,8,9,10};
     for(i=0,j=arr.length-1;i<j;i++,j- -)
        { temp=arr[i];
          arr [i] =arr [j] ;
            arr[j]=temp;
        }
   System.out.println ("After reversing the array is:" );
      for(i=0;i<arr.length;i++)
            System.out.print(arr[i] + "  ") ;
}
}
```

PRESTIGE POINT INDORE

Program to convert a decimal number to binary number

```
import java.util.*;
 class  DecimalToBinary
 {
  public static void main (String[] args )
 {
     Scanner s = new Scanner(System.in);
    int num, rem, arr [ ] = new int[15], i, j ;
   System.out.println ("Enter a decimal number :");
   num =  S.nextInt();
    i=0;
    while(num>0)
     {
      arr [i) =num%2; ,
      num/=2;
       i++;
    }
   System.out.print ("Binary number is : ") ;
      for(j=i-1;j>=0;j- -)                    /*        print the array backwards          */
        System.out.printf("%d",arr[j));
}
}
```

Program to search for an item in the array

```
import java.util.*;
 class  Search
 {
  public static void main (String[] args )
 {
     Scanner s = new Scanner(System.in);
    int i,int[] ={23,12,56,98,76,14,65,11,19,45};
    int item;
   System.out.println ("Enter the i tern to be searched ") ;
   item = s.nextInt();
  for(i=0;i<arr.length;i++)
      { if(item==arr[i))
         { System.out.printf ("%d found at position %d\n", item, i+1);
           break;
         }
      }
 if (i==arr.length)
      System.out.printf("Item %d not found in array\n" , item) ;
} }
```

PRESTIGE POINT INDORE

Two Dimensional Array

Declaration :

```
data_type array_name[][];
data_type[][]  array_name;    //recommended
data_type    [][]array_name;
data_type[]  []array_name;
data_type[]  array_name[];
data_type    [] array_name[];
```

Creation :

```
array_name = new data_type[rowsize][columnsize];
```

Initialization :

```
array_name [rowno][columnno]= value;  // value must be corresponds to  data type of array
```

```
int[][] a;
a= new int[3][4];
```

This creates a 2D array of int that has 12 elements arranged in 3 rows and 4 columns. Although I haven't mentioned it, there are initializers for 2D arrays. For example, this statement creates the 4-by-3 array that is shown in the picture below:

```
int[][]  a  =  { {  1,  0, 12, -1 },
                 {  7, -3,  2,  5 },
                 { -5, -2,  2, -9 }
               };
```

An array initializer for a 2D array contains the rows of A, separated by commas and enclosed between braces. Each row, in turn, is a list of values separated by commas and enclosed between braces. There are also 2D array literals with a similar syntax that can be used anywhere, not just in declarations. For example,

```
a  =  new int[][] {  {  1,  0, 12, -1 },
                     {  7, -3,  2,  5 },
                     { -5, -2,  2, -9 }
                   };
```

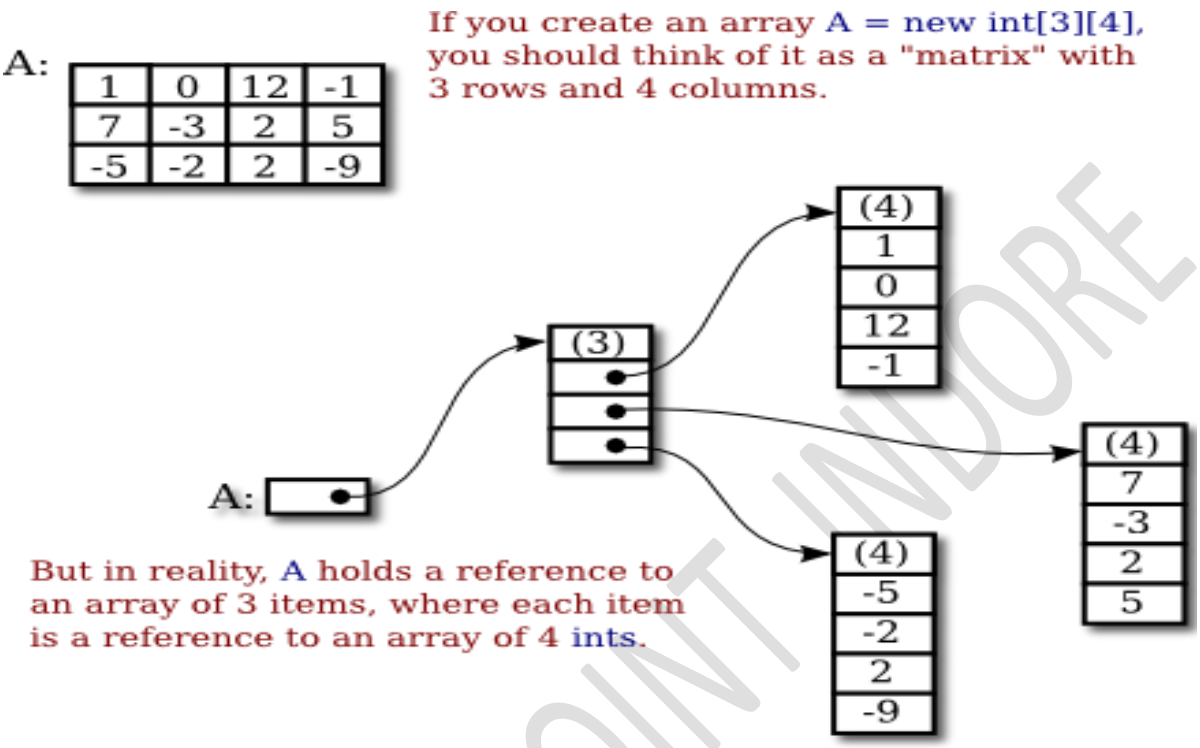
The Truth of 2 D Array: (2 D array is known as 'array of arrays')

Java does not actually have two-dimensional arrays(in matrix style). In a true 2D array, all the elements of the array occupy a continuous block of memory, but that's not true in Java.

The syntax for array types is a clue: For any type BaseType, we should be able to form the type BaseType[], meaning "array of BaseType." If we use int[] as the base type, the type that we get is "int[][] meaning "array of int[]" or "array of array of int."

PRESTIGE POINT INDORE

help to explain this. Consider the 4-by-3 array A defined above.



thinking of a 2D array, **a**, as an array of arrays, we see that **a.length** makes sense and is equal to the number of rows of **a**. If **a** has the usual shape for a 2D array, then the number of columns in **a** would be the same as the number of elements in the first row, that is, **a** [0].length. But there is no rule that says that all of the rows of **a** must have the same length (although an array created with new BaseType[rows][columns] will always have that form). Each row in a 2D array is a separate one-dimensional array, and each of those arrays can have a different length. In fact, it's even possible for a row to be null. For example, the statement

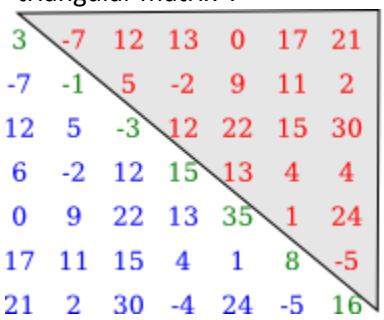
```
int[][] a = new int[3][];
```

with no number in the second set of brackets, creates an array of 3 elements where all the elements are null. There are places for three rows, but no actual rows have been created. You can then create the rows a[0], a[1], and a[2] individually.

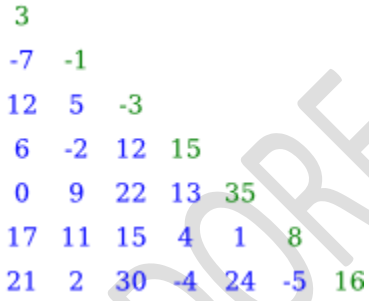
```
a[0] = new int[2];          // this concept is also called jagged array in java
a[1] = new int[3];
a[2] = new int[1];
```

PRESTIGE POINT INDORE

As an example, consider a symmetric matrix. A symmetric matrix, M, is a two-dimensional array in which the number of rows is equal to the number of columns and satisfying M[i][j] equals M[j][i] for all i and j. Because of this equality, we only really need to store M[i][j] for i >= j. We can store the data in a "triangular matrix":



In a symmetric matrix, the elements above the diagonal (shown in red) duplicate elements below the diagonal (blue). So a symmetric matrix can be stored as a "triangular matrix" with rows of different lengths.



It's easy enough to make a triangular array, if we create each row separately. To create a 7-by-7 triangular array of double, we can use the code segment

```
double[][] matrix = new double[7][]; // rows have not yet been created!
for (int i = 0; i < 7; i++) {
   matrix[i] = new double[i+1];  // Create row i with i + 1 elements.
}
```

Jagged Array in Java

Jagged array is array of arrays such that member arrays can be of different sizes, i.e., we can create a 2-D arrays but with variable number of columns in each row. These type of arrays are also known as Jagged arrays.

Following are Java programs to demonstrate the above concept.

```
// Program to demonstrate 2-D jagged array in Java
class JaggedArray
{
   public static void main(String[] args)
   {
      int arr[][] = new int[2][];              // Declaring 2-D array with 2 rows
                                               // Making the above array Jagged
      arr[0] = new int[3];                    // First row has 3 columns
      arr[1] = new int[2];                    // Second row has 2 columns
       int count = 0;                         // Initializing array
     for (int i=0; i<arr.length; i++)
        for(int j=0; j<arr[i].length; j++)
           arr[i][j] = count++;
     System.out.println("Contents of 2D Jagged Array");
     for (int i=0; i<arr.length; i++)
     {
        for (int j=0; j<arr[i].length; j++)
           System.out.print(arr[i][j] + " ");
        System.out.println();
     }     } }
```

```
/* Java Program Example - Two Dimensional Array */

import java.util.Scanner;

public class TwoDimensionalArray
{
  public static void main(String args[])
  {
     int row, col, i, j;
     Scanner scan = new Scanner(System.in);
     System.out.print("Enter Number of Row for Array (max 10) : ");
     row = scan.nextInt();
     System.out.print("Enter Number of Column for Array (max 10) : ");
     col = scan.nextInt();
    int arr[][] = new int[row][col];
     System.out.print("Enter " +(row*col)+ " Array Elements : ");
     for(i=0; i<row; i++)
     {
        for(j=0; j<col; j++)
        {
           arr[i][j] = scan.nextInt();
        }
     }

     System.out.print("The Array is :\n");
     for(i=0; i<row; i++)
     {
        for(j=0; j<col; j++)
        {
           System.out.print(arr[i][j]+ "  ");
        }
        System.out.println();
     }
  }
}
```

Program for addition of two matrices

```
import java.util.Scanner;

public class AdditionOfTwoMatrices
{
  public static void main(String args[])
  {
```

PRESTIGE POINT INDORE

```
      int row, col, i, j;
      Scanner scan = new Scanner(System.in);
     System.out.print("Enter Number of Row for matrix 1 and matrix 2  : ");
      row = scan.nextInt();
      System.out.print("Enter Number of Column for matrix 1 and matrix 2   : ");
      col = scan.nextInt();
     int mat1[][] = new int[row][col];
      int mat2[][] = new int[row][col];
     int mat3[][] = new int[row][col];
     System.out.print("Enter " +(row*col)+ " Elements for first matrix  : ");
      for(i=0; i<row; i++)
      {
         for(j=0; j<col; j++)
         {
            mat1[i][j] = scan.nextInt();
         }
      }

   System.out.print("Enter " +(row*col)+ " Elements for second matrix  : ");
      for(i=0; i<row; i++)
      {
         for(j=0; j<col; j++)
         {
            mat2[i][j] = scan.nextInt();
         }
      }
//Addition
 for(i=0; i<row; i++)
      {
         for(j=0; j<col; j++)
         {
           mat3[i][j] = mat1[i][j]+mat2[i][j];
         }
    }
System.out.printf ("The resultant matrix mat3 is : \n");
 for(i=0;i<row;i++)
   {
     for(j=0;j<col;j++)
        System.out.printf ("%5d" ,mat3 [i] [j]);
       System.out.printf("\n");
   }
}
}
```

PRESTIGE POINT INDORE

write a program to multiply two matrices.

Multiplication of matrices requires that the number of columns in first matrix should be equal to the number of rows in second matrix. Each row of first matrix is multiplied with the column of second matrix then added to get the element of resultant matrix. If we multiply two matrices of order mxn and nxp then the multiplied matrix will be of order mxp.

```
import java.util.Scanner;
class MatrixMultiplication
{
  public static void main(String args[])
  {
    int m, n, p, q, sum = 0, c, d, k;

    Scanner in = new Scanner(System.in);
    System.out.println("Enter the number of rows and columns of first matrix");
    m = in.nextInt();
    n = in.nextInt();

    int first[][] = new int[m][n];

    System.out.println("Enter elements of first matrix");

    for (c = 0; c < m; c++)
      for (d = 0; d < n; d++)
        first[c][d] = in.nextInt();

    System.out.println("Enter the number of rows and columns of second matrix");
    p = in.nextInt();
    q = in.nextInt();

    if (n != p)
      System.out.println("The matrices can't be multiplied with each other.");
    else
    {
      int second[][] = new int[p][q];
      int multiply[][] = new int[m][q];

      System.out.println("Enter elements of second matrix");

      for (c = 0; c < p; c++)
        for (d = 0; d < q; d++)
          second[c][d] = in.nextInt();

      for (c = 0; c < m; c++)
```

PRESTIGE POINT INDORE

```
      {
        for (d = 0; d < q; d++)
        {
          for (k = 0; k < p; k++)
          {
            sum = sum + first[c][k]*second[k][d];
          }

          multiply[c][d] = sum;
          sum = 0;
        }
      }

      System.out.println("Product of the matrices:");

      for (c = 0; c < m; c++)
      {
        for (d = 0; d < q; d++)
          System.out.print(multiply[c][d]+"\t");

        System.out.print("\n");
      }
    }
  }
}
```

Write program to finds out the transpose of a matrix.

Transpose matrix is defined as the matrix that is obtained by interchanging the rows and columns of a matrix. If a matrix is of mxn order then its transpose matrix will be of order nxm.

```
import java.util.Scanner;
public class MatrixTranspose
{
   public static void main(String args[])
   {
      int i, j;
      System.out.println("Enter total rows and columns: ");
      Scanner s = new Scanner(System.in);
      int row = s.nextInt();
      int column = s.nextInt();
      int array[][] = new int[row][column];
      System.out.println("Enter matrix:");
      for(i = 0; i < row; i++)
      {
```

PRESTIGE POINT INDORE

```
            for(j = 0; j < column; j++)
            {
          array[i][j] = s.nextInt();
          System.out.print(" ");
        }
          }
          System.out.println("The above matrix before Transpose is ");
          for(i = 0; i < row; i++)
          {
            for(j = 0; j < column; j++)
        {
          System.out.print(array[i][j]+" ");
        }
        System.out.println(" ");
      }
          System.out.println("The above matrix after Transpose is ");
          for(i = 0; i < column; i++)
          {
            for(j = 0; j < row; j++)
        {
           System.out.print(array[j][i]+" ");
        }
        System.out.println(" ");
      }
    }
}
```

Program to search an element through binary search

```
import java.util.Scanner;
public class BinarySearch
{
   public static void main(String args[])
   {
    Scanner s = new Scanner(system.in);
     int arr[]=new int[10];
     int low,up,mid,i,item;
     System.out.printf("Enter elements of the array(in sorted order)”);
       for(i=0;i<arr.length;i++)
          arr[i]= s.nextInt();
    System.out.printf ("Enter the item to be searched ") ;
      item=nextInt() ;
     low=0; up=arr.length-1;
    while(low<=up && item!=arr[mid)
```


PRESTIGE POINT INDORE

```
{
   mid=(low+up)/2;
    if(item>arr[mid) )
           low=mid+l;      /*Search in right portion */
    if (itern <arr [mid) )
         up=mid-l;         /*Search in left portion */
   if(item==arr[mid)
     System.out.printf ("%d found at position %d\n", item, mid+l);
   if(low>up)
      System.out.printf ("%d not found in array\n", item);

}
}
}
```

Write a program to sort the elements of a I-D array, in ascending order through selection sort.
Sorting is a procedure in which the given elements are ·arranged in ascending or descending order.
For example if the elements of an array are
5, 11, 15, 8, 7, 54, 63, 44
After sorting these elements in ascending order the elements would be :
5, 7, 8, 11, 15,44, 54, 63

Let us take a list of elements in unsorted order and sort them by a example. Elements of the array are
40 20 50 60 30 10

Pass 1:

40 20 50 60 30 10	arr[0] > arr[1] , Exchange
20 40 50 60 30 10	arr[0] < arr[2]
20 40 50 60 30 10	arr[0] < arr[3]
20 40 50 60 30 10	arr[0] < arr[4]
20 40 50 60 30 10	arr[0] > arr[5] , Exchange
10 40 50 60 30 20	

Pass 2:

10 40 50 60 30 20	arr[1] < arr[2] , Exchange
10 40 50 60 30 20	arr[1] < arr[3]
10 40 50 60 30 20	arr[1] > arr[4] , Exchange
10 30 50 60 40 20	arr[1] > arr[5], Exchange
10 20 50 60 40 30	

Pass 3:

10 20 50 60 40 30	arr[2] < arr[3]
10 20 50 60 40 30	arr[2] > arr[4], Exchange
10 20 40 60 50 30	arr[2] > arr[5], Exchange
10 20 30 60 50 40	

PRESTIGE POINT INDORE

Pass 4:
10 20 30 60 50 40 arr[3] > arr[4], Exchange
10 20 30 50 60 40 arr[3] > arr[5], Exchange
10 20 30 40 60 50

Pass 5:
10 20 . 30 40 60 50 arr[4] > arr[5], Exchange
10 20 30 40. 50 60

Sorted array is: 10 20 30 40 50 60

Program of sorting using selection sort

```
import java.util.Scanner;
public class SelectionSort
{
   public static void main(String args[])
   {
    Scanner s = new Scanner(System.in);
    int arr[] = new int[10] ;
     inti,j, temp;
     System.out.printf ("Enter elements of the array : \n") ;
      for(i=0;i<arr.length;i++)
           arr[i] = s.nextInt();
      for(i=0;i<arr.length-1;i++)
              for(j=i+1;j< arr.length;j++)
                {
                 if(arr[i]>arr[j])
                     { temp=arr[i] ;
                       arr [i] =arr [j ] ;
                       arr[j]=temp;
                     }
              }
   System.out. printf ("Sorted array is :");
          for(i=0;i<arr.length;i++)
            System.out.printf("%3d ",arr[i]);
}
}
```

Write a program to sort the elements of a I-D array, in ascending order through bubble sort

If N elements are given, then the procedure for sorting them through bubble sort is as The elements of array are arr[0],arr[I] ,…….,arr[N-1]
Pass 1:

PRESTIGE POINT INDORE

Compare 0^{th} and 1^{st} element, If $0^{th} > 1^{st}$ then exchange them
Compare 1^{st} and 2^{nd} element, If $1^{st} > 2^{nd}$ then exchange them,
Compare 2^{nd} and 3^{rd} element, If $2^{nd} > 3^{rd}$ then exchange them,
……
Compare $N\text{-}2^{th}$ with $N\text{-}1^{th}$, If $N\text{-}2^{th} > N\text{-}1^{nt}$ then exchange them

Pass 2:

Compare 0^{th} and 1^{st} element, If $0^{th} > 1^{st}$ then exchange them
Compare 1^{st} and 2^{nd} element, If $1^{st} > 2^{nd}$ then exchange them,
Compare 2^{nd} and 3^{rd} element, If $2^{nd} > 3^{rd}$ then exchange them,
……
Compare $N\text{-}3^{th}$ with $N\text{-}2^{th}$, If $N\text{-}3^{th} > N\text{-}2^{nt}$ then exchange them

……
……

Pass N-1:
Compare 0^{th} and 1^{st} element, If $0^{th} > 1^{st}$ then exchange them

Let us take a list of elements in unsorted order and sort them by applying bubble sort. .
Elements of the array are

40	20	50	60	30	10	

Pass 1:

40	**20**	50	60	30	10	arr[0]>arr[1], exchange
20	**40**	**50**	60	30	10	arr[1]<arr[2]
20	40	**50**	**60**	30	10	arr[2]<arr[3]
20	40	50	**60**	**30**	10	arr[3]<arr[4] , exchange
20	40	50	30	**60**	**10**	arr[4]<arr[5] , exchange
20	40	50	30	10	60	

Pass 2:

20	**40**	50	30	10	60	arr[0]<arr[1]
20	**40**	**50**	30	10	60	arr[1]<arr[2]
20	40	**50**	**30**	10	60	arr[2]>arr[3] , exchange
20	40	30	**50**	**10**	60	arr[3]>arr[4] , exchange
20	40	30	10	50	60	

PRESTIGE POINT INDORE

Pass 3:

20	40	30	10	50	60	arr[0]<arr[1]
20	40	30	10	50	60	arr[1]>arr[2] , exchange
20	30	40	10	50	60	arr[2]>arr[3] , exchange
20	30	10	40	50	60	

Pass 4:

20	30	10	40	50	60	arr[0]<arr[1]
20	30	10	40	50	60	arr[1]>arr[2] , exchange
20	10	30	40	50	60	

Pass 5:

20	10	30	40	50	60	arr[0]>arr[1] , exchange
10	20	30	40	50	60	

Program of sorting using bubble sort

```
import java.util.Scanner;
public class BubbleSort
{
   public static void main(String args[])
   {
    Scanner s = new Scanner(System.in);
    int arr[] = new int[10];
     int i,j,temp;
    System.out.printf ("Enter' elements of the array : \n") ;
     for(i=0;i<arr.length;i++)
         arr[i]=s.nextInt();
     for(i=0;i<arr.length;i++)
      {
        for(j=0;j< arr.length-1-i;j++)
          {
          if(arr[j]>arr[j+l])
             {
              temp=arr[j] ;
              arr[j]=arr[j+1];
              arr[j+1]=temp;
             }
          }
        }
           System.out.printf ("Sorted array is : \n") ;
```

PRESTIGE POINT INDORE

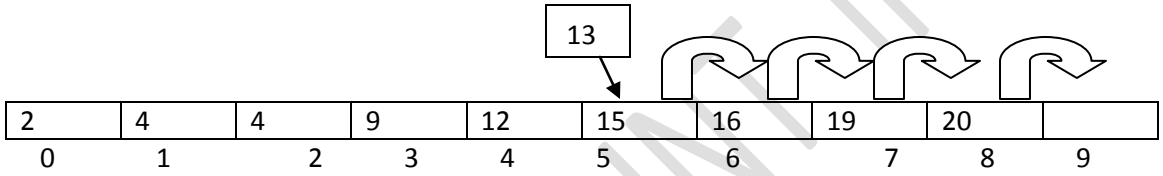
```
        for(i=0;i<arr.length;i++)
          System.out.printf("%3d ",arr[i]);
    }
}
```

Write a program to insert an element in a sorted l-D array at proper place, so that the array remains sorted after insertion also.

Suppose we have an array of size 10 and there are nine elements in it which are in ascending order, one rightmost space is empty for the new element to be inserted.

2	4	4	9	12	15	16	19	20	
0	1	2	3	4	5	6	7	8	9

To insert an item, we'll compare it with the elements of array from the right side and keep on shifting them to the right. As soon as we get an element less than the item we'll stop this process and insert the item. Suppose we have to insert the number 13 in the array



After insertion the array becomes

2	4	4	9	12	13	15	16	19	20
0	1	2	3	4	5	6	7	8	9

```
import java.util.Scanner;
public class InsertAElementInSortedArray
{
   public static void main(String args[])
   {
    Scanner s = new Scanner(System.in);
    int arr [] = new arr[10] ;
    int i, item;
    System.out.println ("Enter elements of the array (in sorted order)");
     for (i=0; i<arr.length-1; i++)          / *rightmost space in array should be empty */
       arr[i] = s.nextInt();
    System.out.println("Enter the item to be inserted ");
       item = s.nextInt() ;
    for(i=arr.length-2; item<arr[i]&&i>=0; i- -)
           arr[i+1]=arr[i];                  /*Shift elements to the right*/
         arr[i+1]=item;                      /*Insert item at the proper place*/
```


PRESTIGE POINT INDORE

```
for(i=0;i<arr.length;i++)
    System.out.printf("%3d ",arr[i]);
}
}
```

Write a program for merging two sorted arrays into a third sorted array

If there are two sorted arrays, then process of combining these sorted arrays into another in sorted order is called merging.

Let us take two arrays arr1 and arr2 in sorted order, we'll combine them into a third sorted array arr3.

arr1 - 5 8 9 28 34

arr2 - 4 22 25 30 33 40 42

We'll take one element from each array, compare them and then take the smaller one in third array. This process will continue until the elements of one array are finished. Then take the remaining elements of unfinished array in third array. The whole process for merging is shown below. arr3 is the merged array, i, j, k are variables used for subscripts of arrI, arr2, arr3 respectively.

```
import java.util.Scanner;
public class MergeSort
{
   public static void main(String args[])
   {
    Scanner s = new Scanner(System.in);
    int arr1[] ,arr2[],arr3[];
    int i,j,k;
     System.out.println ("Enter size of the array  arr1”);
      int size1 = s.nextInt();
     System.out.println ("Enter size of the array  arr2”);
      int size2 = s.nextInt();
   arr1 = new int[size1];
   arr2 = new int[size2];
   arr3 = new int[size1+size2];
 System.out. println ("Enter elements of the array arr1 (in sorted order)”);
  for(i=0;i<arr1.length;i++)
     arr1[i] = s.nextInt();
System.out. println ("Enter elements of the array arr2 (in sorted order)”);
  for(i=0;i<arr2.length;i++)
     arr2[i] = s.nextInt();
i=0,j=0,k=0;

while(i<arr1.length)&&(j<arr2.length))
    {
      if(arr1[i]<arr2[j] )
       arr3[k++J=arr1[i++];
```

PRESTIGE POINT INDORE

```
    else
       arr3[k++J=arr2[j++];
 }

while(i<arr1.length)                /* Put remaining elements of arrl into arr3*/
      arr3[k++]=arrl[i++];
 while(j<aar2.length)                /*Put· remaining elements of arr2 into arr3*/
    arr3[k++]=arr2[j++];

System.out.printf ("Merged array arr3 is :");
  for(i=0;i<arr3.length;i++)
        System.out.printf("%3d ",arr3[i]);
 }
}
```

For-each loop in Java

For-each is another array traversing technique like for loop, while loop, do-while loop introduced in Java 1.5 (Java 5).

It starts with the keyword for like a normal for-loop.
Instead of declaring and initializing a loop counter variable, you declare a variable that is the same type as the base type of the array, followed by a colon, which is then followed by the array name.
In the loop body, you can use the loop variable you created rather than using an indexed array element.
It's commonly used to iterate over an array or a Collections class (eg, ArrayList)

Syntax:

```
for (type var : array)
{
   statements using var;
}
```

is equivalent to:

```
for (int i=0; i<arr.length; i++)
{
   type var = arr[i];
   statements using var;
}
```

// Java program to illustrate for-each loop

Here's an example to iterate through elements of an array using standard for loop:

```
class ForLoop {
  public static void main(String[] args) {

    char[] vowels = {'a', 'e', 'i', 'o', 'u'};

    for (int i = 0; i < vowels.length; ++ i) {
      System.out.println(vowels[i]);
    }
  }
}
```

You can perform the same task using for-each loop as follows:

```
class AssignmentOperator {
  public static void main(String[] args) {

    char[] vowels = {'a', 'e', 'i', 'o', 'u'};
    // foreach loop
    for (char item: vowels) {
      System.out.println(item);
    }
  }
}
```

```
class For_Each
{
   public static void main(String[] arg)
   {
      {
         int[] marks = { 125, 132, 95, 116, 110 };
         int highest_marks = maximum(marks);
         System.out.println("The highest score is " + highest_marks);
      }
   }
   public static int maximum(int[] numbers)
   {
      int maxSoFar = numbers[0];
      // for each loop
      for (int num : numbers)
      {
```

PRESTIGE POINT INDORE

```
        if (num > maxSoFar)
        {
           maxSoFar = num;
        }
      }
   return maxSoFar;
   }
}
```

Limitations of for-each loop

1. For-each loops are not appropriate when you want to modify the array:

```
for (int num : marks)
{
   // only changes num, not the array element
   num = num*2;
}
```

2. For-each loops do not keep track of index. So we can not obtain array index using For-Each loop

```
for (int num : numbers)
{
   if (num == target)
   {
      return ???;   // do not know the index of num
   }
}
```

3. For-each only iterates forward over the array in single steps

```
// cannot be converted to a for-each loop
for (int i=numbers.length-1; i>0; i--)
{
    System.out.println(numbers[i]);
}
```

4. For-each cannot process two decision making statements at once

```
// cannot be easily converted to a for-each loop
for (int i=0; i<numbers.length; i++)
{
   if (numbers[i] == arr[i])
   { ...
   }
}
```

PRESTIGE POINT INDORE

Programming Exercise

1. Write a program to accept n numbers and display the sum of the highest and lowest numbers.
2. Write a program to accept n numbers in array and display the addition of all even numbers and multiplication of all odd numbers.
3. Write a program to sort numbers of a one-d array in descending order using
   (i) selection sort (ii) bubble sort (iii) insertion sort
4. Write a program to modify the elements of an array such that the last element becomes the first element of the array and all other elements are shifted to right.
   1 2 3 4 5 6 7 8 9 -> 9 1 2 3 4 5 6 7 8
5. Write a program to find out the determinant of a matrix.
6. Write a program to count the occurrences of a number in a matrix.
7. Write a program to store all the elements of a 2-D array in a 1-D array row-wise.
8. Write a program to find out whether a matrix is symmetric or not. A matrix is symmetric if transpose of the matrix is equal to the matrix.
9. Write a program to check that the elements of an array are distinct.
10. Write a program to check that the elements of a matrix are distinct.
11. Write a program to find out the sum of elements of principal and secondary diagonals of a square matrix.

PRESTIGE POINT INDORE