```
Arrays
   It refers to index collection of fixed no of homogenous data elements.
   Single variable holding mulitple values which imporves readability of the
program.


Disadvntages
  1. once we create the size cannot be increased/decreased.
  2. It stores only homogenous data elements.

Array declarations
==================
  Single Dimension Array

Declaration of array
====================
 int[] a;//recomended to use as variable is seperated from type.
 int a[];
 int []a;

int[6] a; // compile time error. we cannot specify the size.


Array Construction
==================
   Every array in java is an object hence we create using new operator.

example
   int[]  a;
   a=new int[5];

      or
   int[] a =new int[5];

For every type corresponding classes are available but these classes are part of
java language
but not applicable at the programmer level.
    int[]    [I
    float[]  [F
    double[] [D

Rule1::
   At the time of Array construction compulsorily we should specify the size.
   example::
         int[] a=new int[5];
         int[] a =new int[];//ce:: array dimension is missing.

Rule2::
  It is legal to have an array with size zero.
     example::
          int[] a =new int[0];
          System.out.println(a.length);

Rule3::
   If we declare an array with negative size it would result in Negative Array
size exception.
       example::
            int[] a=new int[-5]; //NegativeArraySizeException.

Rule4::
   The allowed datatypes to specify the size are byte,short,int,char.
        example::
             int[] a =new int[5];
             byte b=10;
             int[] a =new int[b];//valid

             short s=25;
             int[] a =new int[s];//valid

             char c='A';
             int[] a=new int[c];//valid

           int[] a=new int[10L];//CE
           int[] a=new int[3.5f];//CE
```

```
Rule5:: The maximum allowed array size in java is maximum value of int size.
    int[] a=new int[2147483647]; //but valid:: OutOfMemoryError
     int[] a=new int[2147483648]; //CE

ArrayInitialisation
===================
  Since arrays are treated as objects,internally based on the type of data we
keep inside array
  JVM will keep default values.
     eg::int[] a =new int[5];
         System.out.println(a);//[I@....
         System.out.println(a[0]);//0

     eg2:: int[] a=new int[4];
           a[0]=10; a[1]=20; a[2]=30;
           System.out.println(a[3]); //0
           System.out.println(a[4]); //ArrayIndexOutOfBoundsException.
           System.out.println(a[-4]);//ArrayIndexOutOfBoundsException.

Shortcut of way declartion,construction,initialisation in single line
=====================================================================
  int[]a = {10,20,30,40};
  char[] a= {'a','e','i','o','u'};
  String[] a= {"sachin","ramesh","tendulkar","IND"};

Array Element Assignements
==========================
  case 1:: In case of primitive array as an array element any type is allowed
which can be
           promoted to declared type.

   int[] a=new int[10];
    a[0]=97;
    a[1]='a';
    byte b= 10;
    a[2]=b;
    short s=25;
     a[3]=s;
     a[4]=10L;//CE: possible loss of precession

  case 2:: In case of Object type array as an array elements we can provide
either declared type
           object or its child class objects.

eg#1.
     Object[] obj=new Object[5];
     obj[0] =new Object();//valid
     obj[1] =new Integer(10);//valid
     obj[2] =new String("sachin");//valid

eg#2.
```

```
    Number[] num=new Number[10];
     num[0]=new Integer(10);//valid
     num[1]=new Double(10.0);//valid
     num[2]=new String("sachin");//invalid
```

case3:: In case of interface type array as an array element we can provide its implementation
        class Object.

```
      Runnable[] r=new Runnable[5];
       r[0]= new Thread("sachin");
       r[1]= new String("dhoni");//CE
```

case4:: In case of abstract class type array as an array element we can provide its child class
        object.

Array variable assignment
========================

case1:: Element level type promotion is not applicable
          eg:: char value can be type promoted to int, but char[] can't be type promoted to int[].

```
int[] a= {1,2,3};
char[] c={'a','b','c'};
int[] b = a;
int[] a = c;//invalid
```

case2:: In case of Object type array,its child type array can be assigned.

```
          eg:: String[] names={"sachin","saurav","dhoni"};
               Object[] obj=names;
```

```
eg:: int[]=> int[] (valid)
     char[] => int[] (invalid)
     int  => long (valid)
     int[] => long[] (invalid)
     char  => int(valid)
     char[] = int[] (invalid)
     String => Object (valid)
     String[] => Object[](valid)
```

case3:: Whenever we are assigning one array reference to another array reference,its just
        the reference which are being copied not the array elements.
        While copying the reference only its type would be given importance,not its size.

```
eg:: int[] a= {10,20,30,40};
     int[] b= {100,200};
     a=b;
     b=a;
```

case4:: Whenever we are copying the array,its reference will be copied but we should match it with
        the array dimension and its type,otherwise it would result in compile time error.

```
eg:: int[][] a= {{10},{20},{30}};
     int[] b={100,200,300};
         b= a; //CE: incompatible type
```

Note:: In array assignment,its type and dimension must be matched otherwise it

```
would result in
       compile time error.

       int[] a ={10,20,30};S.o.p(a);//[I@..
       float[] f={10.0f,20.0f}; S.o.p(f); //[F@..
       boolean[] b= {true,false}; S.o.p(b); //[Z@..
       Integer[] i={10,20,30}; S.o.p(i);//[L@...
       Float[] f = {10.0f,20.0f}; S.o.p(i); //[L@...

Tricky Questions
================
  a. int[] a,b;  // a-> 1D, b-> 1D
  b. int a[],b[]; // a-> 1D, b->1D
  c. int a[],b;   // a-> 1D, b-> normal variable
  d. int a[],[]b; //CE
  e. int a,[]b;   //CE
  f. int []a,[]b; //CE
  g. int []a,b;   //a => 1D,b=> 1D.


2-D Array
========
   2D-Array =1D-Array + 1D-Array
               (ref)       (data)

Declaration(All are valid)
    int[][] a ;
    int  a[][];
    int  [][]a;
    int[] []a;
    int[] a[];
    int []a[];

ArrayConstruction
    int[][] a =new int[3][2];
            or
    int[][] a= new int[3][];
      a[0]=new int[5];
      a[1]=new int[3];
      a[2]=new int[1];

ArrayInitalisation
   a[0][0] = 10;
   a[2][3] = 5;

Tricky Question
===============
     int[] a,b; // a-> 1D,b->1D
     int[][] a,b; //a->2D,b->1D
     int[] a[],b; //a->2D,b->1D
     int[] a[],[]b; //CE
     int[] []a,b;  //a->2D,b->1D
Rule:If we want to specify the dimension, we need to specify that before the
first variable,but
     from second variable onwards rule is not applicable.

shortcut of working with 2-D array
   int[][] a= {{10,20},{100,200,300},{1000}};


length vs length()
  length=> It is a property of Array type class.
  length() => It is a method of String class.
```

```
int[] a= {10,20,30};
S.o.p(a);
S.o.p(a.length);//3
S.o.p(a.length());//CE::symbol not found

String[] names={"sachin","saurav","dhoni","yuvi"};
S.o.p(names); //[L@....
S.o.p(names[0]);//sachin
S.o.p(names.length);//4
S.o.p(names[0].length());6

int[] a[] ={{10,20,30},{100,200},{1000},{40,50,60,70}};
S.o.p(a);//[[I@...
S.o.p(a[0]);//[I@...
S.o.p(a.length);//4
S.o.p(a[0].length);3
```

Ananomyous Array
================
    An array without a name is called Ananomyous Array.
    These type of array is created just for instance use.

```
eg:: public class Demo{
      public static void main(String... args){
            add(new int[]{10,20,30,40});
            add(new int[]{10,20});
            add(new int[]{});
      }
      public static void add(int[] a){
            sum+=0;
            for(int i=0;i<=a.length;i++){
                  sum+=a[i];
            }
            System.out.println("The sum is ::"+sum);
      }
     }
```

Usage of ananomyous array internally by jvm
===

```
public class Demo{
      public static void main(String []args){
            System.out.println("Working with array");
      }
}

 javac Demo.java
 java  Demo 10 20  => Demo.main(new String[]{"10","20});
 java  Demo        => Demo.main(new String[]{});
 java  Demo 10     => Demo.main(new String[]{"10"});
 java  Demo sachin tendulkar => Demo.main(new String[]{"sachin","tendulkar"});
```

Usage of Arrays Inbuilt class
=============================
 It is a helper class which is available in java.util package.
 This helper class provides few methods through which the user would get a benfit of working with
 few logics.

eg#1.
  toString() is overriden to print the array elements in one line.

```
int[] arr = { 10, 20, 30, 40, 50 };
for (int data : arr) {
    System.out.println(data);
}
System.out.println(Arrays.toString(arr));//10 20 30 40 50

eg#2.
  copyOf()
     This method is used to copy the elements from one array into another array.
  int[] arr = { 10, 20, 30, 40, 50 };
  int[] duplicateArray = Arrays.copyOf(arr, arr.length);
  System.out.println(Arrays.toString(duplicateArray));

eg#3.
   sort()
      It is used to sort the array elements in ascending order.
int[] arr = { 50, 40, 30, 20, 10 };
System.out.println(Arrays.toString(arr));//50 40 30 20 10
Arrays.sort(arr);
System.out.println(Arrays.toString(arr));//10 20 30 40 50

eg#4.
  binarySearch()
    if key found return the index otherwise
    return -(low+1)

 int[] arr = { 50, 40, 30, 20, 10 };
 Arrays.sort(arr);
 System.out.println(Arrays.binarySearch(arr, 2));//-1
 System.out.println(Arrays.binarySearch(arr,35));//-4
 System.out.println(Arrays.binarySearch(arr,100));//-6

eg::
    int[] arr = { 50, 40, 30, 20, 10 };
    Arrays.sort(arr);
    int key = 45;
    System.out.println(Arrays.binarySearch(arr, 2, 4, key));//-4


eg#5. copyOfRange()
         This method is used to copy the array elements of the specified range.
      int[] arr = { 50, 40, 30, 20, 10 };
      System.out.println(Arrays.toString(arr));//50 40 30 20 10

      int[] duplicateArray = Arrays.copyOfRange(arr, 1, 4);
      System.out.println(Arrays.toString(duplicateArray));//40 30 20

eg#6. equals()
         To compare the contents of data in a particular order.

      int[] arr1 = { 50, 40, 30, 20, 10 };
      Arrays.sort(arr1);
      int[] arr2 = { 10, 20, 30, 40, 50 };
      System.out.println(Arrays.equals(arr1, arr2));
```