Array: -

- An Array is an indexed collection of fixed number of homogenous data element.
- Arrays are continuous memory locations having fixed size.
- Where we require storing multiple data elements under single name, there we can use array.
-Arrays are homogenous in nature. It means and integer array can hold only integer values likewise a String array will hold only String values.
- We can create array of byte, short, int, long, double, float, char, String and Object
-The main limitation of array is once we created an array there is no chance of increasing/decreasing size based on our requirement. Hence memory point of view arrays concept is not recommended to use.
-We solve this problem using Collections.

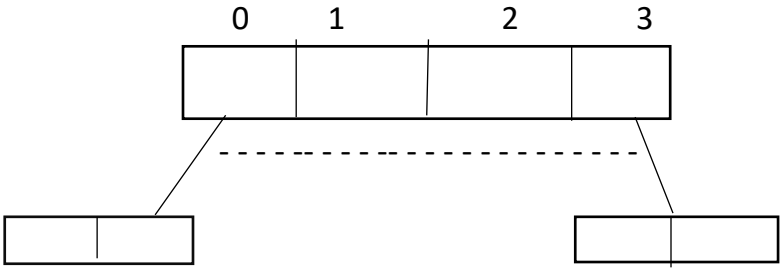
Array Declaration: -

1. One Dimensional Array: -

- One-dimensional arrays can hold values up to one dimension only.
- General form of One-Dimensional array is:
<data-type> <variable-name>[ ];
Example: int a[ ]; //This line only declares a as one-dimensional array, but it is not yet created in ,memory.
- Size is not declared at the time of declaration.
- Valid declarations are:
  - int[] a;
  - int []a;
  - int a[];

2. Two-Dimensional Array: -

- General form of two-dimensional array is:
<data-type> <variable-name>[ ][ ];
Example: int a[ ][ ];
- Valid declarations are:
  - int[][] a;
  - int [][]a;
  - int a[][];
  - int []a[];
  - int[] []a;



- int[] a[];

3. Three-Dimensional array: -

- General form of 3D array is:
<data-type> <var-name>[ ][ ][ ] ;
-Valid declarations are:
  - int[][][] a;
  - int a[][][];
  - int [][][]a;
  - int[][] []a;
  - int[] [][]a;
  - int[] []a[];
  - int[] a[][];
  - int[][] a[];
  - int []a[][];
  - int [][]a[];

Array Creation: -

- In java, array is treated as object. Hence we can create array using 'new' operator.
- At the time of creation of array, we have to specify it's size, else we will get compile time error.

E.g int[] a=new int[4]
int[] a=new int[] //this line will give compile time error

- It is legal to have size '0' in java for array.
E.g: - int[] a=new int[0] //This is legal

- We can specify –ve size of array as well, but it will throw Runtime Exception 'NegativeArraySizeException'.
- To specify array size, allowed data types are: byte, short, int and char. No other data types are allowed. If we use other data types, it will throw compile time error.

E.g. byte b=5;
int[] a= new int[b];

Creation of 2-Dimensional Array: -

- In java, multi-dimensional arrays are not implemented as matrix; rather they are implemented as tree structure.

- We usually call multi-dimensional arrays as 'Array of arrays'
- Main advantage of this approach is memory optimization.

Eg. int[][] a=new int[3]
a[0]=new int[2]
a[1]=new int[2]
a[2]=new int[3]

- While creating a multi-dimensional array we have to specify size to consecutive dimensions.

Ex. int [][]a=new int[4][3] //Allowed
Int [][]=new int[4][] //Allowed
int [][]a=new int[][4] //Not allowed, because 1st dimension doesn't have size.

Adding Elements to Array: -

- To add elements to array we can follow the below given procedure.
  1. Create Array
  2. Initialize size
  3. Add elements to array at separate indexes.

Eg. int a[]=new int[4];
a[0]=11;
a[1]=12
a[2]=13
a[3]=14

- Above method is tedious as it requires more lines of codes to add elements to array.
- A short cut to above method is:

int a[]={11,12,13,14};

- Above line will automatically create an array 'a' with size 4. It will contain 4 elements mentioned in curly braces.
- Same short cut we can use for multi-dimensional arrays as well.

int a[][]={{1,2},{11,12,13},{56}}

- *We can also add elements to an array by using Java loops. Below is the program to insert elements to an array and read inserted elements iteratively*

```
public class ArrayDemo {
  public static void main(String[] args) {
      int[] a = new int[4];
      Scanner sc = new Scanner(System.in);
      for (int i = 0; i < a.length; i++) {
```

```
            System.out.println("Enter a number:>> ");
            a[i] = sc.nextInt();
        }
        for (int i = 0; i < a.length; i++) {
            System.out.println(a[i]);
        }
    }
}
```

length Vs. length(): -

- length is a final variable applicable for array.
- length variable represent size of the array.

```
public class ArrayLength {

    public static void main(String[] args) {

        int[] x = new int[5];
        System.out.println(x.length);

        System.out.println(x.length()); // Compile-Time
Error 'Cannot invoke length() on the array type int[]'
    }
}
```

- length() is a final method applicable for String objects.
- length() returns number of character present in the String.

```
  public class StringLength {

    public static void main(String[] args) {

        String s = "Technology";
        System.out.println(s.length());
    }
}
```

Operators: -

There are different types of Operators in Java.

1. Arithmetic Operator
2. Increment and Decrement Operator.
3. Relational Operator
4. Equality Operator
5. Assignment Operator

Increment and Decrement Operator: -

- We can apply Increment and Decrement Operator only for variables but not for constant value.
- For Final Variable we can't apply increment and decrement operator.
- We can apply increment and decrement operator for every primitive type except boolean.

For Eg: - int x=10;
int y=x++;
System.out.println(y);

Output: - 11

Arithmetic Operator (+,-,*,/,%): -

- Using Arithmetic operator, we perform different operations.
- We perform Addition, Subtraction, Division, Multiplication, Module.
- If we applying any arithmetic operator between two or more than two variables then result type is always.

Max(int, type of 1st Variable, type of 2nd Variable,…,nth Number)

Relational Operator (<,>,>=, =<): -

- We can applied on every primitive type except Boolean.

For eg: -
System.out.Println(10<20);
O/P : - true

System.out.Println(true<false);
//Compile Time error: T he operator < is undefined for the argument type(s) boolean, Boolean
-We cannot apply relational operators for object types.

Equality Operator (== , !=): -

-We can apply on every primitive type including Boolean also.
-We can apply equality operators to object types also.

Difference between == operator and equals() method : -

- In general, we can use == operator for reference comparison (address comparison) and equals() method for contains comparison.

For Eg :- String s=new String("Java");
String s1=new String("Java");
System.out.Println (s==s1); //false
System.out.Println (s.equals(s1)); //true

Assignment Operator: -

- Assignment operators are used to assigning value to a variable. The left side operand of the assignment operator is a variable and right side operand of the assignment operator is a value.
- The value on the right side must be of the same data-type of the variable on the left side otherwise the compiler will raise an error.

For Eg: -
a = a + b can write using Assignment operator as a += b