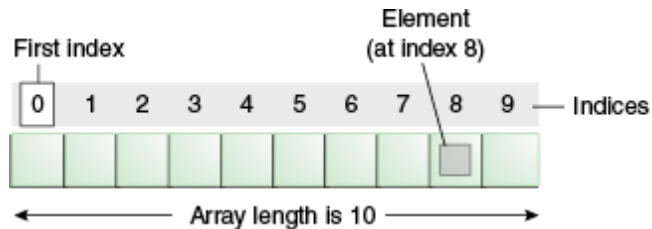


1.6 ARRAYS

- Array is a collection of elements of similar data type stored in contiguous memory location.
- The array size is fixed i.e we can't increase/decrease its size at runtime.
- It is index based and the first element is stored at 0th index.



Advantages of Array

- **Code Optimization:** Multiple values can be stored under common name. Data retrieval or sorting is an easy process.
- **Random access:** Data at any location can be retrieved randomly using the index.

Disadvantages of Array

- **Inefficient memory usage:** Array is static. It is not resizable at runtime based on number of user's input.
- To overcome this limitation, Java introduce collection concept.

Types of Array

There are two types of array.

- One Dimensional Array
- Multidimensional Array

(I) One Dimensional Array

a) Declaring Array Variables

Syntax:

dataType[] arrayName; //preferred way

Or

dataType arrayName [];

Here datatype can be a primitive data type like: int, char, Double, byte etc. arrayName is an identifier.

Example: **int[] a;**

b) Instantiation of an Array

- Array can be created using the *new* keyword.
- To allocate memory for array elements we must mention the array size.

- The size of an array must be specified by an *int* value and not *long* or *short*.
- The default initial value of elements of an array is 0 for numeric types and *false* for boolean.

Syntax:

arrayName=new datatype[size];

Or

dataType[] arrayName=new dataType[size];

Example:

int[] a=new int[5]; //defining an integer array for 5 elements Alternatively, we can create and initialize array using following syntax.

Syntax:

dataType[] arrayName=new dataType[] {list of values separated by comma};

Or

dataType[] arrayName={ list of values separated by comma};

Example:

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

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

The built-in *length* property is used to determine length of the array i.e. number of elements present in an array.

c) Accessing array elements

- The array elements can be accessed by using indices.
- The index starts from 0 and ends at (array size-1).
- Each element in an array can be accessed using *for* loop.

Example:

public class Main

{

 public static void main(String args[])

 {

 int a[]=new int[]{10,20,30,40}; //declaration and initialization//printing array

for(int i=0;i<a.length;i++) //length is the property of array

 System.out.println(a[i]);

 }

```
}
```

Sample Output:

10

20

30

40

The for-each loop:

➤ The for-each loop is used to traverse the complete array sequentially without using an index variable.

➤ It's commonly used to iterate over an array or a Collections class.

(eg, Array- List).

Syntax:

for(type var:arrayName)

```
{
```

Statements using var;

```
}
```

Example:

```
public class Main
```

```
{
```

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

```
    {
```

```
        int a[]=new int[]{10,20,30,40};//declaration and initialization
```

```
        int sum=0;
```

```
        for(int i:a)           // calculate sum of array elements sum+=i;
```

```
            System.out.println("Sum:"+sum);
```

```
    }
```

```
}
```

Sample Output:

Sum:100

(II) Multidimensional Arrays

➤ Multidimensional arrays are arrays of arrays with each element of the array holding the

reference of other array.

➤ These are also known as Jagged Arrays.

Syntax:

dataType[][] arrayName=new datatype[rowsize][columnnsize];

dataType[][][] arrayName=new datatype[][][];

Example:

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

	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Example:

Program to access 2D array elements

```
public class TwoDimEx
{
    public static void main(String args[])
    {
        // declaring and initializing 2D array
        int arr[][] = { {1,1,12},{2,16,1},{12,42,2} };
        // printing 2D array
        for (int i=0; i< arr.length; i++)
        {
            for (int j=0; j < arr[i].length ; j++)
            System.out.print(arr[i][j] + "\t ");
            System.out.println();
        }
    }
}
```

Sample Output:

1	1	12
2	16	1
12	42	2

Jagged Array

Jagged array is an array of arrays with different row size i.e. with different dimensions.

Example:

```
public class Main
{
    public static void main(String[] args)
    {
        int[][] a = {
                        {11, 3, 43},
                        {3, 5, 8, 1},
                        {9},
                    };

        System.out.println("Length of row 1: " + a[0].length);      System.out.println("Length of row 2:
        " + a[1].length);      System.out.println("Length of row 3: " + a[2].length);
    }
}
```

Sample Output:

Length of row 1: 3

Length of row 2: 4

Length of row 3: 1

Passing an array to a method

An array can be passed as parameter to method.

Example: Program to find minimum element in an array

```
public class Main
{
    static void min(int a[])
    {
        int min=a[0];
        for(int i=1;i<a.length;i++)
            if(min>a[i])
                min=a[i];
        System.out.println("Minimum:"+min);
    }
    public static void main(String args[])
    {
        int a[]={12,13,14,5};
        min(a);//passing array to method
    }
}
```

Sample Output:

Minimum:5

Returning an array from a method

A method may also return an array.

Example:Program to sort array elements in ascending order.

```
public class Main
{
    static int[] sortArray(int a[])
    {
        int tmp;
        for(int i=0;i<a.length-1;i++)
        {
            //code for sorting
            for(int j=i+1;j<=a.length-1;j++)
            {
```

```

        if(a[i]>a[j])
        {
            tmp=a[i];
            a[i]=a[j];
            a[j]=tmp;
        }
    }
    return(a);           // returning array
}

public static void main(String args[])
{
    int a[]={33,43,24,5};
    a=sortArray(a);      //passing array to method
    for(int i=0;i<a.length;i++)
        System.out.println(a[i]);
}
}

```

Sample Output:

```

5
24
33
43

```

ArrayIndexOutOfBoundsException Exception

- ArrayIndexOutOfBoundsException gets raised when an array has been accessed with an illegal index.
- The index is either negative or greater than or equal to size of array.

Example:

```

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

```

```

        int[] a={1,2,3};
        System.out.println(a[3]);
    }
}

```

Run Time Error: Exception in thread “main” java.lang.

Array Index Out Of Bounds Exception: 3

ARRAYS CLASS

➤ The Arrays class contains various static methods for sorting and searching arrays, comparing arrays, and filling array elements.

➤ It is available in java.util.Arrays. Some of the commonly used methods in

Example:

```
import java.util.Arrays; public class
```

```
Main
```

```
{
```

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

```
{
```

```
    int a[] = {43, 64, 11, 28, 32, 91, 17, 4, 2};
```

```
    System.out.println(Arrays.toString(a));
```

```
    Arrays.sort(a);
```

```
    System.out.println(“Sorted Array “+Arrays.toString(a));
```

```
    int index = Arrays.binarySearch(a,32);
```

```
    System.out.println(“Position of 32 in sorted arrays is => “ + index);
```

```
    int[] copy = Arrays.copyOf(a, a.length); //To copy the whole array
```

```
    System.out.println(“Copied array => “ + Arrays.toString(copy));
```

```
}}
```

Sample Output:

[43, 64, 11, 28, 32, 91, 17, 4, 2]

Sorted Array [2, 4, 11, 17, 28, 32, 43, 64, 91]

Position of 32 in sorted arrays is => 5

Copied array => [2, 4, 11, 17, 28, 32, 43, 64, 91]