Unit 5

Array

Date 11/08
Page

Introduction to Array

↳ An array is a collection of similar type of data items in a single unit.

↳ It is continuous memory locations that stores homogeneous data items.

↳ Each item of an array is called its element. ~~of~~

↳ Every element of an array is identified by index number. Index ranges from 0 to n-1, where n is size of an array

↳ Name of array elements are same as that of array name but index ranges.

Declaration of array in java:

↳ It is the process of creating continuous memory space to store similar types of data items.

↳ Array must be declared before using it in program.

Syntax:

```
data type [ ] arrayname ; // preferred way
OR
data type arrayname [ ] ; // works but not preferred way
```

Example:

```
int [ ] num;
double [ ] marks;
```

Creating array

↳ Array can be created using new keyword for memory allocation.

Syntax:

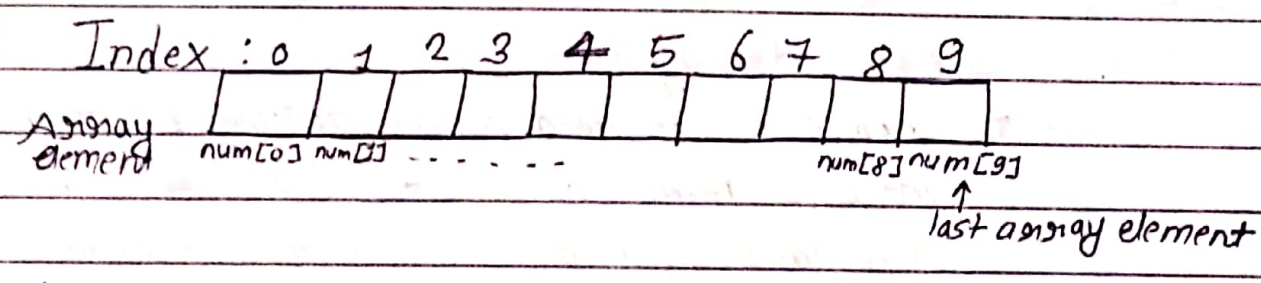
```
array_name = new datatype [size];
```

Example: num = new int [10];

Array declaration and creation can be done in a single line as data_type [] arrayname = new datatype [size];

Example:

```
int [ ] num = new int [10];
```



Array initization

↳ Array elements can be initialized during declarati

Syntax:

```
type [ ] array = new type [ ] { val1, val2 valN };
```

OR,

```
type [ ] array = { val1, val2, ... valN };
```

Example:

```
int [] num = new int [] { 2, 4, 6, 8, 10, 12, 14,
16, 18, 20};
```

Or

```
int [] num = {2, 4, 6, 8, 10, 12, 14, 16, 18, 20};
```

↳ An array can be initialized using index number.

Example:

```
int [] num = new int [5] // array declaration
num [0] = 2;
num [1] = 4;
num [2] = 6;
num [3] = 8;
num [4] = 10;
```

/* Program illustration array initialization using system */

```
class ArrayTest
{ public static void main (String [] args) {
  int [] num = new int [] { 2, 4, 6, 8, 10};
  System.out.println("First array element :"+num
  System.out.println ("Second array element [0]);
                      :"+num[1]);
System.out.println("Third array element :"+num[2]);
System.out.println("Fourth array element :"+num[3]);
System.out.println("fifth array element : "+num[4]);
}
}
```

length property

↳ length property is used to find length of array i.e total number of array elements.

Syntax:

```
array name.length;
```

// Program illustrating length property

```
class ArrayTest
{
  public static void main(String [] args)
  {
    int [] num = new int [5]; //array declaration
    int len;
    len = num.length
    System.out.println("The length of an array is :"+len);
  }
}
```

Accessing array elements (Processing array)

↳ An array elements can be accessed using two ways:

i) Using For() Loop
ii) Using Foreach() Loop

i) Using For() Loop :-

↳ Accessing means reading data from elements of array and display to the output screen to the user.

↳ It uses index number of each array element.

// Program to illustrate accessing array elements:

```
class AccessArray
{
public static void main(String [] args)
{
double [] nums = { 1.5, 2.5, 3.5, 4.5 };
for (int i = 0; i < nums.length; i++)
{
System.out.println("Element :" + nums [i]);
}
}
}
```

/* Program to input age of 10 students and display then using array. */

```
import java.util.*;
class AccessArr
{
public static void main (String [] args)
{
int [] age = new int [10];
```

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter age of student:");
For (int i=0; i<age.length; i++)
{
   age[i] = sc.nextInt();
}
   System.out.println("Age of students are:");
   For (int i=0; i<age.length; i++)
{
   system.out.println(age[i]);
}
}
}
```

/* Java program to input 10 number from user and display sum of them */

```
import java.util.*;
class calculateSum
{
  public static void main (String [] args)
  {
    int total = 0;
    int [] num = new int [10]; // array declaration
    Scanner sc = new Scanner (System.in);
    System.out.println ("Enter numbers:");
    for(int i = 0; i<10; i++)
    {
      num [i] = sc.nextInt();
    }
    for (int i=0; i<10; i++)
    {
      total = total + num[i];
    }
    System.out.println ("Total Sum of number is = "+ total);
  }
}
```

/* Program to input n numbers store in array and display in ascending order. */

```
import java.util.*;
class SortNum
{
  public static void main (String [] args)
```

~~int [] num = new int [100];~~

```
Scanner sc = new Scanner (System.in);
System.out.println("How many numbers :");
    int n = sc.nextInt(); int [] num = new int[n];
System.out.println("Enter numbers:");
  for (int i = 0; i<n; i++)
      num[i] = sc.nextInt();
  int temp;
    for (int i=0; i<n; i++)
     {
      for (int j = i+1; j<n; j++)
    {
       if (num[i] > num[j])
    {
       temp = num[i];
       num[i] = num[j];
       num[j] = temp;
    }
  }
}
System.out.println(" The numbers in ascending
Order : ");
   for (int i=0; i<n; i++)
System.out.println(num[i] + " ");
}
}
```

Processing (accessing) array using foreach loop:

↳ Elements of an array can be accessed using foreach loop.

↳ Foreach loop reads each element of an array and assigns in the counter variable until there is no more elements in array.

↳ Syntax for foreach loop is :-

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

/* Program to input age of 'n' students and display using foreach loop */

```
import java.util.*;
class program
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
System.out.println("How many students:");
int n = sc.nextInt();
int[] age = new int[n];
System.out.println("Age of students ~~are~~:");
for(int i=0; i<n; i++)
age[i] = sc.nextInt();
System.out.println("The Age of students are:");
```

```
for (int i: age)
  {
    System.out.println(" "+i);
  }
}
}
```

Types of Arrays:

There types of array can be declared in Java.

1. One Dimensional Array:

↳ One-dimensional array is strings of data stored in a single line. One dimensional array only contains one continuous row of data. The elements of one-dimensional arrays can be added or printed in a single line using loops.

2. Two Dimensional Array:

↳ Two-dimensional arrays are the more frequently used type of array in Java. They form a matrix of rows and colums and find applications in a lot of fields outside development, such as simulation, robotics, and machine learning.

3. Multi-Dimensional Array:

↳ Arrays can also have more than two dimensions. While arrays with multiple dimensions are not easy to

visualize, their applications are only increasing by the day. They can also hold large amounts of data, which is a useful feature when it comes to data analysis.

- Two Dimensional Array:

↳ An array having two dimensions that stores data in rows and columns i.e in matrix form.

↳ It has two subscripts. Value of first subscript represents no of rows and second subscript represents no of columns.

Declaration of 2-D array:

Syntax:

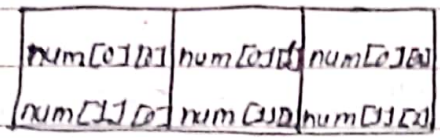
```
type [][] arrayName = new type [row][column];
```

Here;

- ✓ type is valid data type
- ✓ arrayName is name of array i.e valid identifier
- ✓ [row] specifies no of rows
- ✓ [column] specifies no of columns
- ✓ row * column = total no of element in that array
- ✓ index of row ranges from 0 to row-1
- ✓ index of column ranges from 0 to column-1
- ✓ name of array elements are same as that of array name but index of row and column varies.

Example:

```
int [][] num = new int [2] [3];
```



num

2-D array Initialization in java

↳ 2-D array can be initialized during declaration time. Value are initialized row wise.

Syntax:

```
type [] [] arrayName = new type [row] [coloum] {{row0 values}, {row1 values}, ..., {rowN values}};
```

OR

```
type [][] arrayName = {{row0 values}, {row1 values}, ..., {rowN values}};
```

example:

```
int [][] num = new int [][] {{2, 4, 6}, {8, 10, 12}};
int [][] num = {{2, 4, 6}, {8, 10, 12}};
```

```
//Program to demonstrate of 2D array in java.
    public class program
{
  public static void main (String [] args)
{
    int [][] num = new int [2] [3]; //2D array declaration
    num [0] [0] = 2;
    num [0] [1] = 4;
```

```
num[0][2] = 6;
num[1][0] = 8;
num[1][1] = 10;
num[1][2] = 12;
System.out.println("num[0][0]:"+num[0][0]);
System.out.println("num[0][1]:"+num[0][1]);
System.out.println("num[0][2]:"+num[0][2]);
System.out.println("num[1][0]:"+num[1][0]);
System.out.println("num[1][1]:"+num[1][1]);
System.out.println("num[1][2]:"+num[1][2]);
}
}
```

/* Program to input a 2x3 matrix and display its elements */

```
import java.util.*;
class Program
{
public static void main(String []args)
{
Scanner sc = new Scanner(System.in);
int[][] mat = new int[2][3];
System.out.println("Enter element of matrix:");
for(int i=0; i<2; i++)
{
for(int j=0; j<3; j++)
{
mat[i][j] = sc.nextInt();
}
}
```

```
System.out.println("Elements of matrix are :");
for(i=0; i<2; i++)
{
 for(j=0; j<3; j++)
 {
  System.out.println(matnum[i][j]+" ");
 }
 System.out.println("\n");
}
}
```

Arrays class

↳ The Arrays class is built-in class included within java.util package.

This class provides static methods to create and access java arrays dynamically.

Methods of an Arrays class are used to manipulate array in unique ways such as sort search, fill etc.

Some methods of an Arrays class includes:

① Sort() method

↳ Method that sorts an array into ascending numerical order or alphabetical order for strings.

Syntax:

```
Arrays.sort(arr);
```

Here;

Arrays: is built-in class
sort(): is method of Arrays class
arr: argument passed to method

// Program to sort an integer array using Arrays class.

```
import java.util.*;
class Integer
{
  public static void main(String args[])
  {
    int[] arr = {8, 6, 9};
    Arrays.sort(arr);
    for(int i: arr)
      {
        System.out.println(i+" ");
      }
  }
}
```

Methods of Array class:

~~① sort()~~

② fill(): The Arrays.fill() method is used to fill an entire array with a single value.

↳ It can be useful when we want to reset all values in an array and initilize them to a

specific value.

Syntax:

```
Arrays.fill(arr, value);
```

It takes two arguments: name of array and value by which array element are filled.

Examples:

```
import java.util.*;
class program
{
    public static void main(String[] args)
{
    int[] arr = new int[5];
    Arrays.fill(arr, 2);
    for(int i: arr)
    {
        System.out.println(i+" ");
    }
}
}
```

③ toString(): Used to convert an entire array into a string format.

↳ It takes an array as input and returns string representation of the array.

↳ Useful for printing or logging purposes.

Syntax:

```
Arrays.toString(arr);
```

Examples:

```
import java.util.*;
class program
{
    public static void main(String [] args)
{
    int [] arr = {1,2,3};
    System.out.println(Arrays.toString(arr));
}
}
```

O/P

[1, 2, 3]

④ copyOf():

↳ Used to create a new array that is copy of an existing array.

↳ It takes two arguments: original array and length of new array.

↳ Useful when we want to manipulate an array without affecting the original data.

Syntax:

```
Arrays.copyOf(arr, length);
```

Example:

```
int [] original = {1,2,3};
int[] copy = Arrays.copyOf(original,
                           original.length);
```

System.out.println(Arrays.toString(copy));

O/P

[1, 2, 3]

⑤ Arrays.equals():-

↳ equals() method is used to checks if two array are equal, meaning their length, order and element are the same.

↳ It returns logical value either true or false.

Syntax:

Arrays.equals(arr1, arr2);

Example:

```
import java.util.*;
class ArrayTest
{
  public static void main(String [] args)
  {
    int [] arr1 = {1,2,3};
    int [] arr2 = {1,2,3};
  boolean isEqual = Arrays.equals(arr1, arr2);
  System.out.println("isEqual");
  }
}
```

⑥ ~~binary~~ Arrays.binarySearch():-

↳ This method is used to search specific element in a array.

↳ It uses the binary search algorithm which is more efficient than a linear search but required the array to be sorted first.

↳ It takes two arguments : name of array and element to be search.

Syntax :

```
Arrays.binarySearch(arrays, element);
```

Example;

```
import java.util.*;
class ArrayTest
{
public static void main (String [] args)
{
int [] array = {2, 4, 6, 8, 10};
int index = Arrays.binarySearch (arrays, 4);
System.out.println( index);
}
}
```

O/P

1