Array in Java

An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number1, number2, ..., and number99, you declare one array variable such as numbers and use number[0], number[1], and ..., number[99] to represent individual variables.

Advantages

- **Code Optimization**: It makes the code optimized, we can retrieve or sort the data efficiently.
- **Random access**: We can get any data located at an index position.

Disadvantages

Size Limit: We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.

Types of Array

- Single Dimensional Array
- Multi-Dimensional Array

Single Dimensional Array

An array with one dimension whether in horizontal or vertical is called single dimensional array. We can access each element by it's index which starts from 0. It has single subscript(**[ ]**).

Declaring Single Dimensional Array Variables

To use an array in a program, you must declare a variable to reference the array, and you must specify the type of array the variable can reference.

Syntax

```
dataType[] variable;            // preferred way.
datatype  []variable;
dataType variabler[];                      // works but not preferred way.
```

The following code snippets are examples of this syntax −

```
int[] myList;  // preferred way.
Int []myList;  // works but not preferred
int myList[];  // works but not preferred way.
```

Creating Arrays

You can create an array by using the **new** operator with the following syntax −

Syntax

myList=new myList[size];

The above statement does two things
–

- It creates an array using new Operator
- It assigns the reference of the newly created array to the variable myList.

We can also create an array along with declaration

dataType[] myArray = new myArray[arraySize];

Example

int[] myArray = **new myArray**[**10**];

Array myList is created with size 10 and the value is accessed by it's index which starts from 0 to n-1.

0	1	2	3	4	5	6	7	8	9
11	71	13	20	74	54	89	12	25	36

myArray[0]…………….. ……….. … . ……… ………………….. ………… **myarray[9]**

Processing Arrays

```
class Testarray
{
    public static void main(String args[])
    {
          int  a[]=new  int[5];                //declaration  and
instantiation
           a[0]=10;
           a[1]=20;
           a[2]=70;
           a[3]=40;
           a[4]=50;
```

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

Declaration, Instantiation and Initialization of Java Array

int a[]={33,3,4,5};   **//declaration, instantiation and initialization**

Example

```
public class TestArray {

  public static void main(String[] args)
  {
      int[] myList = {1, 2, 3, 4};

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

        int total = 0;
     for (int i = 0; i < myList.length; i++) {
         total += myList[i];
     }
    System.out.println("Total is " + total);
}
```

Example

```
public class TestArray {

  public static void main(String[] args)
  {
        int[] myList = {1, 2, 3, 4};
```

```
          for (int element: myList)
          {
                System.out.println(element);
           }
      }
}
```

Passing Arrays to Methods

Just as you can pass primitive type values to methods, you can also pass arrays to methods. For example, the following method displays the elements in an **int** array –

```
class Testarray2
{
    static void min(int arr[])
    {
        int min=arr[0];
        for(int i=1;i<arr.length;i++)
         {
            if(min>arr[i])
            {
                min=arr[i];
            }
         }
     System.out.println(min);
    }


  public static void main(String args[])
  {
      int a[]={33,3,4,5};      //declaring and initializing array
        min(a);                   //passing array to method
  }
}
```

Sorting an Array Element

Anonymous Array in Java

Java supports the feature of an anonymous array, so you don't need to declare the array while passing an array to the method.

```
public class TestAnonymousArray
{
      static void printArray(int arr[])
      {
              for(int i=0;i<arr.length;i++)
              {
                  Systemout.println(arr[i]);
              }
       }

      public static void main(String args[])

      {

          printArray(new int[]{10,22,44,66});        //passing anonymous array to method

        }

}
```

Returning an Array from a Method

A method may also return an array. For example, the following method returns an array that is the reversal of another array –

```
class TestReturnArray
{
    static int[] get()
     {
         return new
        int[]{10,30,50,90,60};
    }
    public static void main(String args[])
    {
         int arr[]=get();
        for(int i=0;i<arr.length;i++)
        {
           System.out.println(arr[i]);
         }
}
```

Example

```
public static int[] reverse(int[] list)
{
    int[] result = new int[list.length];

     for (int i = 0, j = result.length - 1; i < list.length; i++, j--)
     {
        result[j] = list[i];
      }
   return result;
}
```

Multi-Dimensional Array

An array with two dimension in both horizontal and vertical is called single dimensional array. We can access each element by it's index which starts from 0. It has double subscript(**[ ][ ]**). data is stored in row and column based index (also known as matrix form).

Syntax:

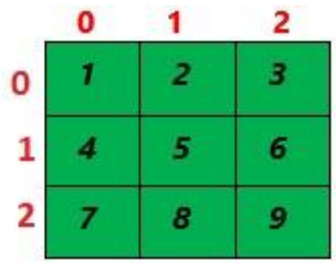
dataType[][] arrayRefVar; (or)
dataType [][]arrayRefVar; (or)
dataType arrayRefVar[][]; (or)
dataType []arrayRefVar[];

Example to instantiate Multidimensional Array in Java

```
int[][] arr=new int[3][3];      //3 row and 3 column
```

Example to initialize Multidimensional Array in Java

```
arr[0][0]=1;  arr[0][1]=2;
arr[0][2]=3;  arr[1][0]=4;
arr[1][1]=5;  arr[1][2]=6;
arr[2][0]=7;  arr[2][1]=8;
arr[2][2]=9;
```



	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

```
class Testarray3
{
   public static void main(String args[])
   {
      int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
      for(int i=0;i<3;i++)
      {
         for(int j=0;j<3;j++)
         {
            System.out.print(arr[i][j]+" ");
          }
         System.out.println();
       }
    }
}
```

ArrayIndexOutOfBoundsException

The Java Virtual Machine (JVM) throws an ArrayIndexOutOfBoundsException if length of the array in negative, equal to the array size or greater than the array size while traversing the array.

```
public class TestArrayException{
    public static void main(String args[]){
        int arr[]={50,60,70,80};
        for(int i=0;i<=arr.length;i++){
            System.out.println(arr[i]);
        }
    }
}
```

Jagged Array, class name of java array, copying a java array, cloning a java array