ABESIT
COLLEGE OF ENGINEERING

CRC-Training

Core Java - Array

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Topics

- Introduction
- Array Declaration
- Array Creation
- Array Initialization
- Length
- Looping Through Array Elements
- Passing Arrays to Methods
- Returning Arrays from Methods
- Anonymous Arrays
- Cloning of Arrays

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Introduction

- Arrays in Java are **non-primitive** data types that store elements of a similar data type in the memory.
- Arrays in Java can store both primitive and non-primitive types of data in it.
- **primitive data types** (Integer, Character, Float, etc.)
- and **non-primitive data types**(Object).
- **two types of arrays**
  - **single-dimensional** arrays have only one dimension
  - **multi-dimensional** have 2D, 3D, and nD dimensions.

ABESIT
COLLEGE OF ENGINEERING
CRC-Training

Array Declaration

Syntax to Declare an Array in Java

```
dataType[] arr;
(or)
dataType []arr;
(or)
dataType arr[];
```

Example -

```
int[]  x;

int  []x;

int  x[];
```

ABESIT COLLEGE OF ENGINEERING

CRC-Training

1D - Array Declaration

Tick the correct Array Declaration –

char[ ] ch;	√
float ch[];	√
char []ch;	√
char [10]ch;	×
float ch[10];	×
char[10] ch;	×

At the time of declaration we can't write size of array

1D - Array Declaration

❖ **Array Declaration**

Example:

```
int arr1[], arr2[];      //Both arr1 and arr2 are arrays.

int[] arr1, arr2;    //Both arr1 and arr2 are arrays.

int  []arr1, arr2;     //Only arr1 is an array but arr2 is not.
```

2D- Array Declaration

Syntax to Declare an Array in Java

dataType[][] arr;

(or)

dataType [][]arr;

(or)

dataType arr[][];

Example -

int[][] x;

int [][]x;

int x[][];

ABESIT COLLEGE OF ENGINEERING

CRC-Training

2D - Array Declaration

Tick the correct Array Declaration –

char[ ][] ch; √

float ch[][]; √

char [][]ch; √

char[] []ch; √

float[] ch[]; √

char [] ch[]; √

Last three also valid but generally don't used

Array Declaration

int[] a,b; // Valid

a and b are array of int type;

Array Creation

- **Array Construction**
  - In java, arrays are dynamically created at runtime using **'new'** keyword, hence, arrays are objects.

Syntax –

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

or

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

Note – At the time of array creation we must specify the size.

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Array Creation

- int[] arr = new int[10] // 1D - array of int type
- int[][] arr = new int[10][10] // 2D array of int type

- flaot[] arr = new float[10];
- char[] arr = new char[10];

- long[] arr = new long[10];
- double[] arr =new double[10];

Array Creation

```
int[] arr = new int[5];
```

0	1	2	3	4

Size of array = 5 Lower Bound = 0 Upper Bound = 4

Array Initialization

❖ **Initializing Array Elements**

a) Default Initialization

- When an array is created using new keyword, all its elements are automatically **initialized to their default values.**

Example:

```
int[] arr = new int[3];
```

0	0	0

```
boolean[] arr = new boolean[3];
```

false	False	False

```
float[] arr = new float[3];
```

0.0F	0.0F	0.0F

```
String[] arr = new String[3];
```

Null	Null	null

Array Initialization

❖ **Initializing Array Elements**

b) Initialize array elements using Initialization block

Syntax:

```
<array type>[] <array name> = { <array initialization block> };
```

Example:

```
int[] age = {20, 21, 22, 23};
```

Array Initialization

❖ **Initializing Array Elements**

c) Array Declaration, Construction, and Initialization in a single step

Syntax:

```
<array type>[] <array name> = new <array type>[]{<array initialization block>};
```

Example:

```
int[] arr = new int[]{10, 20, 30};   //Ok.
```

Loops (Iteration Statements)

- **Enhanced for (or for-each) loop**
  - This is first time introduced in **jdk1.5**. **This is the most convenient loop for retrieving elements from array or collection.**
  - This loop cannot be used for general purpose.
  - This loop iterate elements always in forward direction but not in reverse direction.

Syntax:

```
for(<element declaration> : <array or collection name>){
        <body>
}
```

Where type of **<element declaration>** is same as **array type or collection type**.

Looping Through Array Elements

There are two ways to loop through the array elements –

- **For-Loop**

```
int[] arr = {1,2,3,4,5,6};
for(int i=0;i<arr.length;i++)
    System.out.print(arr[i] +" ");
```

- For-each loop

```
int[] arr = {1,2,3,4,5,6};
for(int val:arr)
    System.out.print(val +" ");
```

Loops (Iteration Statements)

- **Enhanced for (or for-each) loop**

Example

int[] arr = {10,20,30};

Basic for loop	Enhanced for loop

for(int i = 0; i < arr.length; i++)
{
SOP(arr[i]);
}
 	
The basic for loop uses array index to read array data.
	for(int e : arr)
	{
	SOP(e);
	}
	
	The Enhanced for loop directly uses element but not index.
	Hence, simple to use.

Loops (Iteration Statements)

❖ **Enhanced for (or for-each) loop**

Note:

- Enhanced for loop is used only with arrays and collections but not for general purpose.
- Enhanced for loop cannot be used to retrieve elements in reverse order.

Passing Arrays to Methods

```
public static void main(String[] args) {
    int[] arr = {1,2,3,4,5};
    System.out.println(sum(arr));
}
static int sum(int[] a){
    int s=0;
    for(int val:a)
        s=s+val;
    return s;
  }
```

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Returning Arrays from Methods

```
public static void main(String[] args) { // Main Method
    int[] arr = {1,2,3,4,5};
    int[] result = sum(arr); // Calling function
    for(int val:result)
        System.out.println(val);
}
static int[] sum(int[] a){   // Sum Method
    for(int i=0;i<a.length;i++)
        a[i]=a[i]+2;
    return a; // Returning array
  }
```

Array Initialization

❖ **Initializing Array Elements**

d) Anonymous array

➢ Neither array name nor array size is specified is called as anonymous array.

Syntax:

```
new <array type>[] {<array initialization block>};
```

Example:

```
new int[] {1, 2, 3, 4};
```

Array Initialization

❖ **Initializing Array Elements**

d) Anonymous array

- **Usage:** Anonymous arrays are most oftenly used in method calling.

Example:

```
add(new int[]{1, 2, 3}, new int[]{4, 5, 6});  //method calling
```

Anonymous Arrays

Arrays having no name are called Anonymous arrays in java

Syntax - **new datatype[] {values separated by comma}**

```
public static void main(String[] args) {
    int result = sum(new int[]{1,2,3,4,5});
    System.out.println(result);
}
static int sum(int[] a){
    int s=0;
    for(int val:a)
    s=s+val;
  return s;
  }
```

Anonymous Arrays

ABESIT COLLEGE OF ENGINEERING

CRC-Training

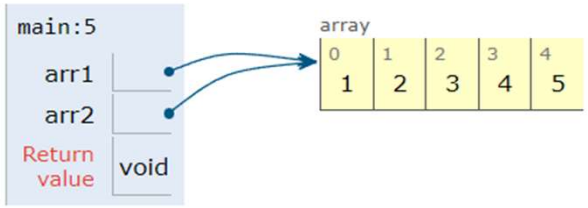
Some Observation

- When we call a method and pass array as a arguments –
  - then copy of array passed or reference passed

Cloning of Arrays

Cloning an array in Java is nothing but creating a new array, where data is copied from the existing one using the clone() property.

```
1  public class YourClassNameHere {
2      public static void main(String[] args) {
3      int[] arr1 = {1,2,3,4,5};
4      int[] arr2 = arr1;
5      }
6  }
```




```
1  public class YourClassNameHere {
2      public static void main(String[] args) {
3      int[] arr1 = {1,2,3,4,5};
4      int[] arr2 = arr1.clone();
5      }
6  }
```