ABESIT
COLLEGE OF ENGINEERING

CRC-Training

Core Java – 2D Array

ABESIT
COLLEGE OF ENGINEERING
CRC-Training

Topics

- Introduction
- 2D Array Declaration
- 2D Array Creation
- 2D Array Initialization
- Length
- Looping Through 2D Array Elements
- Passing 2D Arrays to Methods
- Anonymous 2D Arrays

Introduction

- A 2D array is an array of one-dimensional arrays or **array of arrays** i.e. each element of a multi-dimensional array is another array.
- In Java, a two-dimensional array is stored in the form of rows and columns and is represented in the form of a matrix.
- Thus, you can get a total number of elements in a multidimensional array by multiplying row size with column size.

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

2D - Array Declaration

int[][] a,b; // Valid

a and b are 2D array of int type;

2D - Array Creation

❖ **2D - Array Construction**

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

Syntax –

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

                    or

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

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

ABESIT COLLEGE OF ENGINEERING

CRC-Training

2D - Array Creation

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

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

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

2D - Array Creation

int[][] Array = new int[2][5];

```
int[][] Array = new int[2][5];
```

In this, the array has two rows and five columns. The index starts from 0, 0 in the left-upper corner to 1, 4 in the right lower corner.

2D - 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:

The default value is

- 0 for int, short, long, and byte array
- 0.0 for double and float arrays
- null for String array

2D - Array Initialization

❖ **Initializing Array Elements**

b) Initialize array elements using Initialization block

Syntax:

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

Example:

```
int[][] age = { {21, 22},
                {31, 32},
                {41, 42} };
```

2D - 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>,
                                                        <array initialization block>,
                                                        ........ };
```

Example:

```
int[][] arr = new int[][] { {21, 22}, {31, 32}, {41, 42} }; //Ok.
```

Looping Through 2D - Array Elements

There are two ways to loop through the array elements –

- **For-Loop**

```
public class Main
{
    public static void main(String[] args) {
        int arr[][]= { {1, 2, 3},
                       {4, 5, 6},
                       {7, 8, 9} };
      for(int i=0; i<3; i++)
        {
            for(int j=0; j<3; j++)
               System.out.print(" " + arr[i][j]);
           System.out.println();
        }
    }
}
```

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

Looping Through 2D - Array Elements

There are two ways to loop through the array elements –

- **For-Loop with length**

```
public class Main
{
    public static void main(String[] args) {
        int arr[][]= { {1, 2, 3},
                       {4, 5, 6},
                       {7, 8, 9} };
       for(int i=0; i<arr.length; i++)
         {
              for(int j=0; j<arr[i].length; j++)
                 System.out.print(" " + arr[i][j]);
            System.out.println();
         }
    }
}
```

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

Looping Through 2D - Array Elements

There are two ways to loop through the array elements –

- **For each Loop**

```
public class Main
{
    public static void main(String[] args) {
        int arr[][]= { {1, 2, 3},
                       {4, 5, 6},
                       {7, 8, 9} };
       for(int i=0; i<arr.length; i++)
         {
              for(int j : arr[i])
                 System.out.print(" " + j);
            System.out.println();
         }
    }
}
```

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

Passing Arrays to Methods

```
public class Main
{
    public static void main(String[] args) {
        int arr[][]= { {1, 2, 3},
                       {4, 5, 6},
                       {7, 8, 9} };
        System.out.print("Sum = " + sum(arr));
    }
    static int sum(int [][]ar)
    {
        int s = 0;
        for(int i=0; i<ar.length; i++)
         {
             for(int j : ar[i])
                s += j;
         }
         return s;
    }
}
```

```
Sum = 45
```

Anonymous 2D - Arrays

Arrays having no name are called Anonymous arrays in java

Syntax - **new datatype[][] {{1D Array}, {1D Array}, ...}**

```
public class Main
{
    public static void main(String[] args) {
       int result = sum(new int[][]{{1,2,3}, {4,5,6}});
       System.out.print("Sum = " + result);
    }
    static int sum(int [][]ar)
    {
        int s = 0;
        for(int i=0; i<ar.length; i++)
         {
              for(int j : ar[i])
                 s += j;
         }
         return s;
    }
}
```

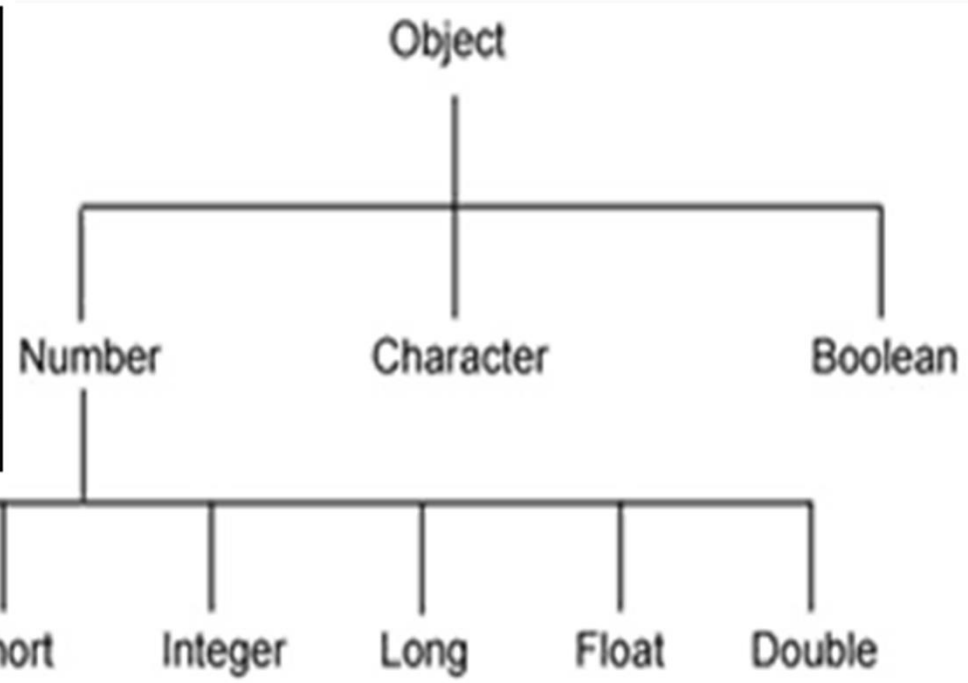
Anonymous 2D Array

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Wrapper Classes

Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
long	Integer
float	Float
double	Double
boolean	Boolean



Wrapper Classes

- Wrapper classes are used to wrap (store) primitive data into an object.
- Hence, primitive types can also be handled just like an object.
- By default, all wrapper classes are final, hence cannot be inherited.
- Also, the instance of all wrapper classes are immutable i.e., the value in the wrapper object cannot be changed.

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Autoboxing and Unboxing

- The Auto & Un boxing first time introduced in **JDK1.5**.

Autoboxing (jdk1.5)

- The java compiler automatically converts Primitive data to Wrapper object is called as autoboxing.

Example:

Before compilation:

```
Integer iRef = 10.
```

After compilation:

```
Integer iRef = new Integer(10);     // Autoboxing
```

ABESIT COLLEGE OF ENGINEERING

CRC-Training

Autoboxing and Unboxing

Autoboxing (jdk1.5)

Example:

```
Integer iRef = 10;          // Autoboxing
        int a = 10;
Integer iRef = Integer.valueOf(a);     // Autoboxing
```

Autoboxing and Unboxing

Unboxing

- The java compiler automatically converts Wrapper object to primitive data is called as unboxing.

Example:

```
int a = 10;
Integer iRef = Integer.valueOf(a);     // Autoboxing

int I = iRef;                          // Unboxing
int I = iRef .intValue();              //unboxing.
```