Arrays

Sensitivity: Internal & Restricted

Agenda

1 **Arrays**

Sensitivity: Internal & Restricted

Objectives

At the end of this module, you will be able to :

- Work with one dimensional and two dimensional arrays in Java

Arrays

Arrays

- An array is a container object that holds a fixed number of values of a single type
- When an array is created, the length of an array is fixed
- Array elements are automatically initialized with the default value of their type, When an array is created
- Array can be created using the new keyword

Ex:

```
int[] x =  new int[5]; // defining an integer array for 5 blocks
```

Arrays (Contd.).

- Alternatively, we can create and initialize array as below format

```
int[] x = {10, 20, 30};
int[] x = new int[]{10, 20, 30};
```

- Here the length of an array is determined by the number of values provided between {and}
- The built-in length property determines the size of any array

Ex:

```
int[] x = new int[10];
int x_len = x.length;
```

Array - Example

```
public class ArrayDemo {
    public static void main(String[] args) {
        int[] x; // declares an array of integers
        x = new int[5]; // allocates memory for 5integers
        x[0] = 11;
        X[4] = 22;
        System.out.println("Element at index 0: " + x[0]);
        System.out.println("Element at index 1: " + x[1]);
        System.out.println("Element at index 4: " + x[4]);
    }
}
```

Output:

```
Element at index 0: 11
Element at index 1: 0
Element at index 4: 22
```

Array Bounds, Array Resizing

- Array subscripts begin with 0
- Can't access an array element beyond the range
- Can't resize an array. Can use the same reference variable to refer new array

```
int x[] = new int [5];
x= new int [10];
```

Sensitivity: Internal & Restricted

Array copy

- To copy array elements from one array to another array, we can use arraycopy static method from System class
- **Syntax:**

```
public static void arraycopy(Object s,int
sIndex,Object d,int dIndex,int lngth)
```

- **Ex:**

```
int source[] = {1, 2, 3, 4, 5, 6};

int  dest[]  = new int[10];

System.arraycopy(source,0, dest,0,source.length);
```

Array Copy - Example

```
public class ArrayLengthDemo {
    public static void main(String[] args) {
        // creates and initializes an array of integers
        int[] source = {100, 200, 300};
        // creates an integer array with 3 element
        int[]  dest = new int[3];
        // copying an elements from source to dest array
        System.arrayCopy(source, 0, dest, 0, source.length);
        for (int i =0;  i <  dest.length;  i++)
        System.out.println("Element at index " + i + ":  " +
          dest[i]);
    }
}
```

```
Output:
Element at index 0: 100
Element at index 1: 200
Element at index 3: 300
```

Two-Dimensional Arrays

- Two-dimensional arrays are arrays of arrays
- Initializing two-dimensional arrays:

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

The 1st dimension represent rows or number of one dimension, the 2nd dimension represent columns or number of elements in the each one dimensions

- The curly braces { } may also be used to initialize two dimensional arrays
- Ex:

```
int[][] y = { {1,2,3}, {4,5,6}, {7,8,9} };
int[][] y = new int[][] { {1,2,3}, {4,5,6}, {7,8,9} };
```

Two-Dimensional Arrays (Contd.).

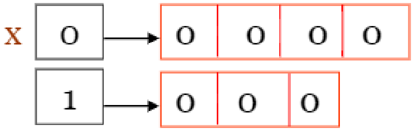
- You can initialize the row dimension without initializing the columns but not vice versa

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

int[][] x = new int[][3];  //error
```

- The length of the columns can vary for each row and initialize number of columns in each row
- Ex1:

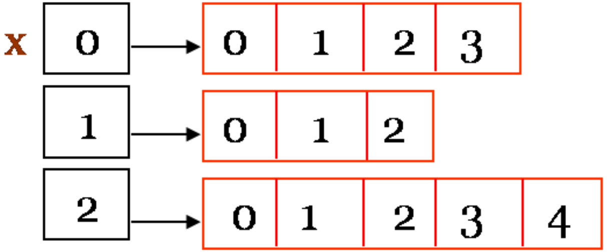
```
int [][]x = new int [2][];
x[0] = new int[5];
x[1] = new int [3];
```



Two-Dimensional Arrays (Contd.).

Ex2:

```
int [][]x = new int [3][];
x[0] = new int[]{0,1,2,3};
x[1] = new int []{0,1,2};
x[2] = new
int[]{0,1,2,3,4};
```



Two-Dimensional Array - Example

/* Program to under stand two-dimensional arrays */

```
class TwoDimDemo {
    public static void main(String[] args) {
     int [][] x = new int[3][];   // initialize number of rows
     x[0] = new int[3];  // define number of columns in each row
       x[1] = new int[2];
       x[2] = new int[5];
       for(int i=0;  i < x.length; i++) {    // print array elements
            for (int j=0;  j < x[i].length;  j++) {
                 x[i][j] = i;
               System.out.print(x[i][j]);
             }
             System.out.println();
        }
    }
}
```

Output:

```
000
11
22222
```

Quiz

Select which of the following are valid array definition

```
1. int[] a;
   a = new int[5];
2. int a[] = new int[5]
3. int a[5] = new int[5];
4. int a[] = {1,2,3};
5. int[] a = new int[]{1,2,3};
6. int[] a = new int[5]{1,2,3,4};
```

Quiz (Contd.).

What will be the result, if we try to compile and execute the following codes

```
class Sample {
    public static void main(String[] args) {
        int[] a = new int[5]{1,2,3};
            for(int i : a)
            System.out.println(i);
    }
}
```

Quiz (Contd.).

What will be the result, if we try to compile and execute the following codes

1.
```
class Sample {
        public static void main(String[] args) {
            while(false)
                System.out.println("while loop");
        }
    }
```

2.
```
class Sample {
        public static void main(String[] args) {
            for( ; ; )
                System.out.println("For loop");
        }
    }
```

Quiz (Contd.).

What will be the result, if we try to compile and execute the following code?

```
class Test {
   public static void main(String [ ] args) {
      int [ ] x=new int[10];
      System.out.println(x[4]);
   }
}
```

Sensitivity: Internal & Restricted

Quiz (Contd.).

What will be the result, if we try to compile and execute the following code?

```
class Test {
   public static void main(String [ ] args) {
      int x[ ][ ]=new int[10] [ ];
      System.out.println(x[4][0]);
   }
}
```

Sensitivity: Internal & Restricted

Assignment

Summary

In this session, you were able to :

- Understand how to work with single and two dimensional arrays in Java

Thank You

Sensitivity: Internal & Restricted