Unit II
Arrays

Tutor: Pema Galey

Learning Outcomes

In this session, you will learn about:

- Basics of Array
- Types of Array
- Single Dimension array
- Multi-dimension array
- Copy of Arrays

Introduction

- In computer programming, an array is a collection of similar types of data.

- The number of values in the Java array is fixed.

Array Declaration

- In Java, here is how we can declare an array.

```
        dataType[] arrayName;
```

- Example:

```
        double[] data;
```

But, how many elements can array this hold?

```
        // declare an array
        double[] data;
        // allocate memory
        data = new double[10];
```

OR

```
        double[] data = new double[10];
```

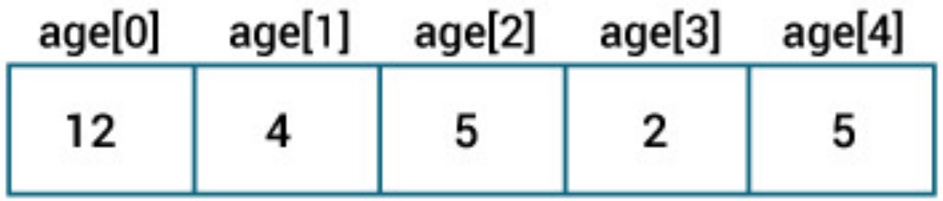
How to Initialize Arrays?

- we can initialize arrays during declaration.
- Example:

```
//declare and initialize and array
// automatically specifies the size
int[] age = {12, 4, 5, 2, 5};
```

- We can also initialize arrays in Java, using the index number.

```
int[] age = new int[5];
// initialize array
age[0] = 12;
age[1] = 4;
age[2] = 5; ..
```



How to Access Elements of an Array?

- We can access the element of an array using the index number.
- Here is the syntax for accessing elements of an array,

```
// access array elements
array[index]
```

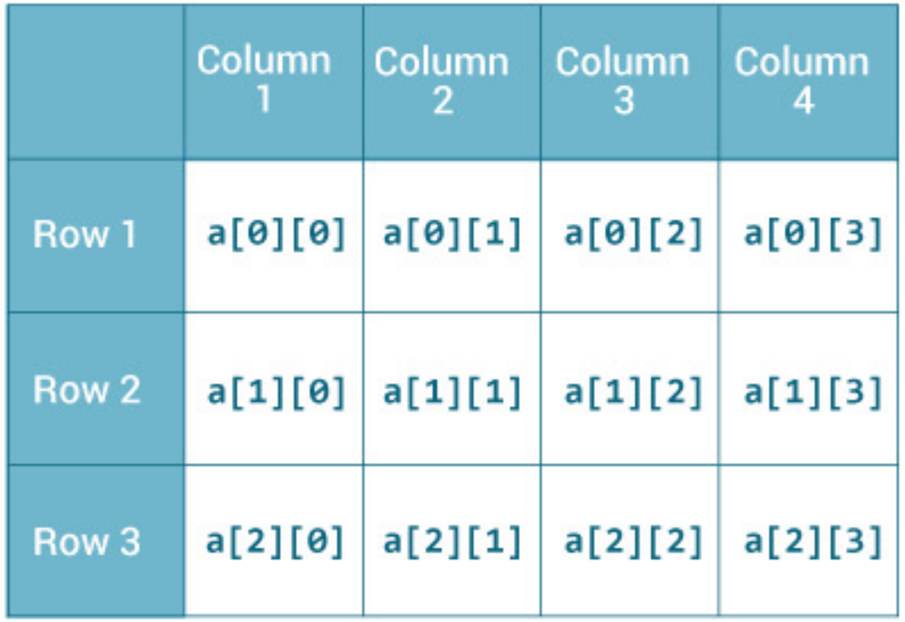
Example

```
class Main {
    public static void main(String[] args) {
        // create an array
        int[] age = {12, 4, 5, 2, 5};
        // access each array elements
        System.out.println("Accessing Elements of Array:");
        System.out.println("First Element: " + age[0]);
        System.out.println("Second Element: " + age[1]);
        System.out.println("Third Element: " + age[2]);
        System.out.println("Fourth Element: " + age[3]);
        System.out.println("Fifth Element: " + age[4]);
    }
}
```

Multidimensional Arrays

- A multidimensional array is an array of arrays. Each element of a multidimensional array is an array itself.
- For example,

```
int[][] a = new int[3][4];
```

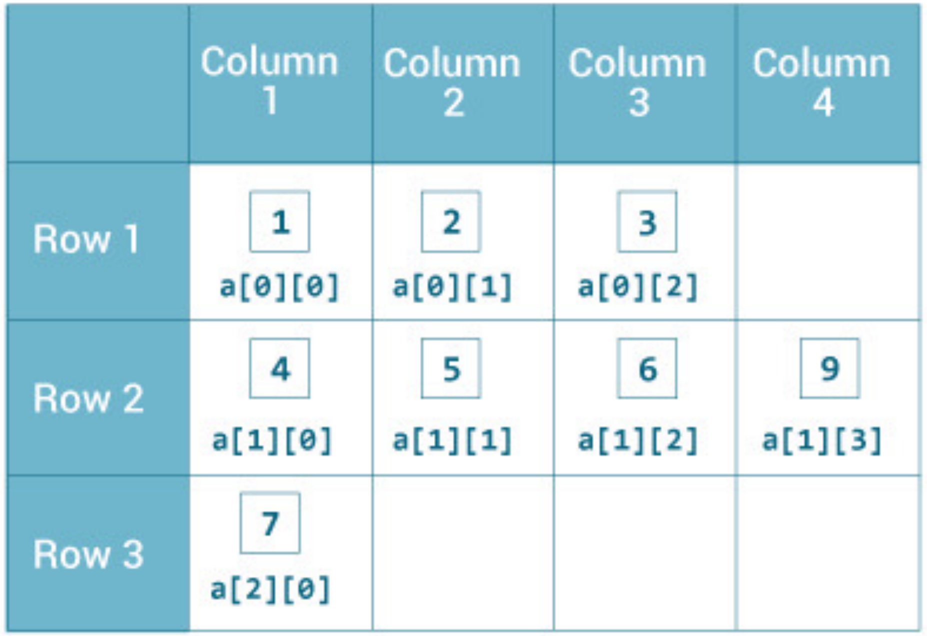


	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

How to initialize a 2d array?

- Here is how we can initialize a 2-dimensional array in Java.

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



	Column 1	Column 2	Column 3	Column 4
Row 1	1 a[0][0]	2 a[0][1]	3 a[0][2]	
Row 2	4 a[1][0]	5 a[1][1]	6 a[1][2]	9 a[1][3]
Row 3	7 a[2][0]			

Example for 2D Array

```
class MultidimensionalArray {
    public static void main(String[] args) {
            // create a 2d array
            int[][] a = {
                  {1, 2, 3},
                  {4, 5, 6, 9},
                  {7},
            };
            // calculate the length of each row
            System.out.println("Length of row 1: " + a[0].length);
            System.out.println("Length of row 2: " + a[1].length);
            System.out.println("Length of row 3: " + a[2].length);
    }
}
```

Example for 2D Array

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

Copy Arrays

- In Java, we can copy one array into another. There are several techniques you can use to copy arrays in Java.
  1. Copying Arrays Using Assignment Operator
  2. Using Looping Construct to Copy Arrays
  3. Copying Arrays Using arraycopy() method
  4. Copying Arrays Using copyOfRange() method
  5. Copying 2d Arrays Using Loop

1. Copying Arrays Using Assignment Operator

- Example:

```
class Main {
  public static void main(String[] args) {
          int [] numbers = {1, 2, 3, 4, 5, 6};
           // copying arrays
          int [] positiveNumbers = numbers;
          for (int n: positiveNumbers) {
                  System.out.print(n + ", ");
          }
   }
}
```

1. Copying Arrays Using Assignment Operator

- However, there is a problem with this technique. If we change elements of one array, corresponding elements of the other arrays also change:

```
class Main {
   public static void main(String[] args) {
            int [] numbers = {1, 2, 3, 4, 5, 6};
            int [] positiveNumbers = numbers;
             // change value of first array
            numbers[0] = -1;
            for (int n: positiveNumbers) {
                        System.out.print(n + ", ");
            }
   }
}
```

Output:

```
-1, 2, 3, 4, 5, 6
```

2. Using Looping Construct to Copy Arrays

```
import java.util.Arrays;
class Main {
    public static void main(String[] args) {
            int [] source = {1, 2, 3, 4, 5, 6};
            int [] destination = new int[6];
            // iterate and copy elements from
            //source to destination
            for (int i = 0; i < source.length; ++i) {
                destination[i] = source[i];
            }
            // converting array to string
            System.out.println(Arrays.toString(destination));
    }
}
```

Output:

```
1, 2, 3, 4, 5, 6
```

3. Copying Arrays Using arraycopy() method

- In Java, the System class contains a method named arraycopy() to copy arrays. This method is a better approach to copy arrays than the above two.

- The arraycopy() method allows you to copy a specified portion of the source array to the destination array.

- For example,

```
arraycopy(Object src, int srcPos,Object dest,
            int destPos, int length)
```

3. Copying Arrays Using arraycopy() method

For example,

```
arraycopy(Object src, int srcPos,Object dest,
             int destPos, int length)
```

Here,

✓src - source array you want to copy

✓srcPos - starting position (index) in the source array

✓dest - destination array where elements will be copied from the source

✓destPos - starting position (index) in the destination array

✓length - number of elements to copy

3. Copying Arrays Using arraycopy() method

```
// To use Arrays.toString() method
import java.util.Arrays;
class Main {
   public static void main(String[] args) {
            int[] n1 = {2, 3, 12, 4, 12, -2};
            int[] n3 = new int[5];
            // Creating n2 array of having length of n1 array
            int[] n2 = new int[n1.length];
            // copying entire n1 array to n2
            System.arraycopy(n1, 0, n2, 0, n1.length);
            System.out.println("n2 = " + Arrays.toString(n2));
            // copying elements from index 2 on n1 array
            // copying element to index 1 of n3 array
            System.arraycopy(n1, 2, n3, 1, 2);
            System.out.println("n3 = " + Arrays.toString(n3));
 } }
```

Output:

```
n2 = [2, 3, 12, 4, 12, -2]
n3 = [0, 12, 4, 0, 0]
```

Copy Arrays

- Explore further on:
  - Copying Arrays Using copyOfRange() method
  - Copying 2d Arrays Using Loop

Thank you!