ECAP615

Programming in Java

Harjinder Kaur

Assistant Professor

Learning Outcomes

After this lecture, you will be able to

- learn the basic concept of Arrays
- understand the different types of Arrays
- implementation difference in one dimensional and multi-dimensional Array

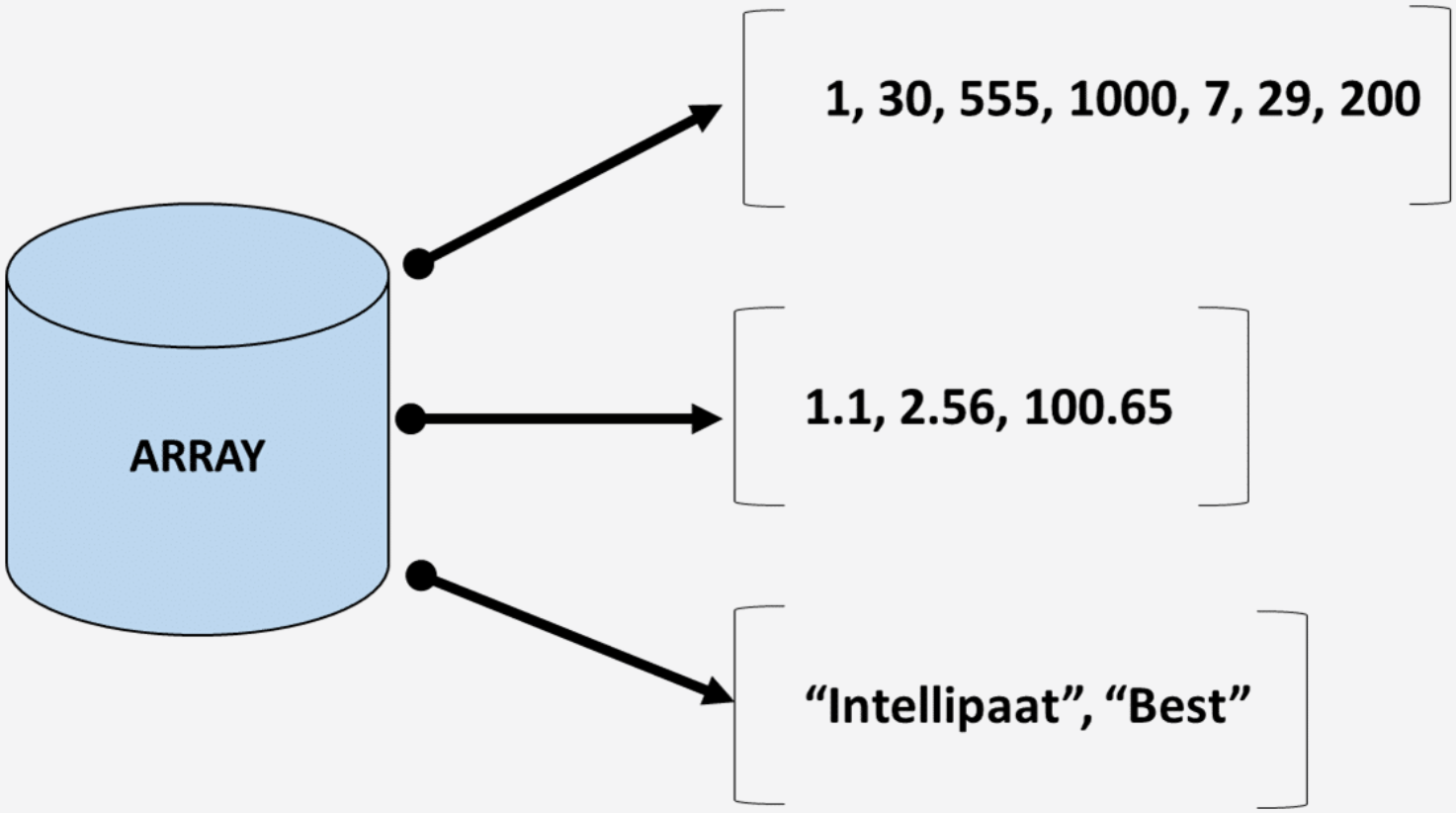
Arrays

- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
- It is an object which contains elements of a similar data type.
- The elements of an array are stored in a contiguous memory location.

Arrays

- We can store only a fixed set of elements in a Java array.
- Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.

Arrays



Arrays

- In Java all arrays are dynamically allocated.

Arrays

- In Java all arrays are dynamically allocated.
- Since arrays are objects in Java, we can find their length using the object property *length*.

Arrays

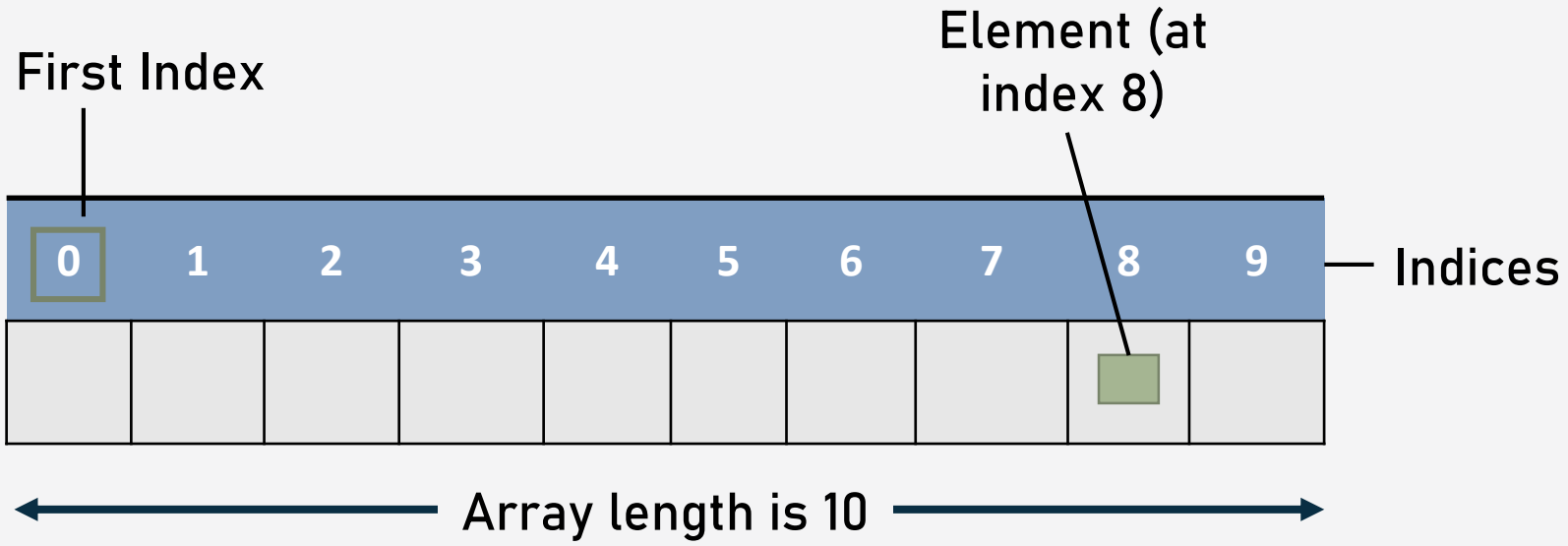
- To declare an array, define the variable type with square brackets:

```
type var-name[];

OR

type[] var-name;
```

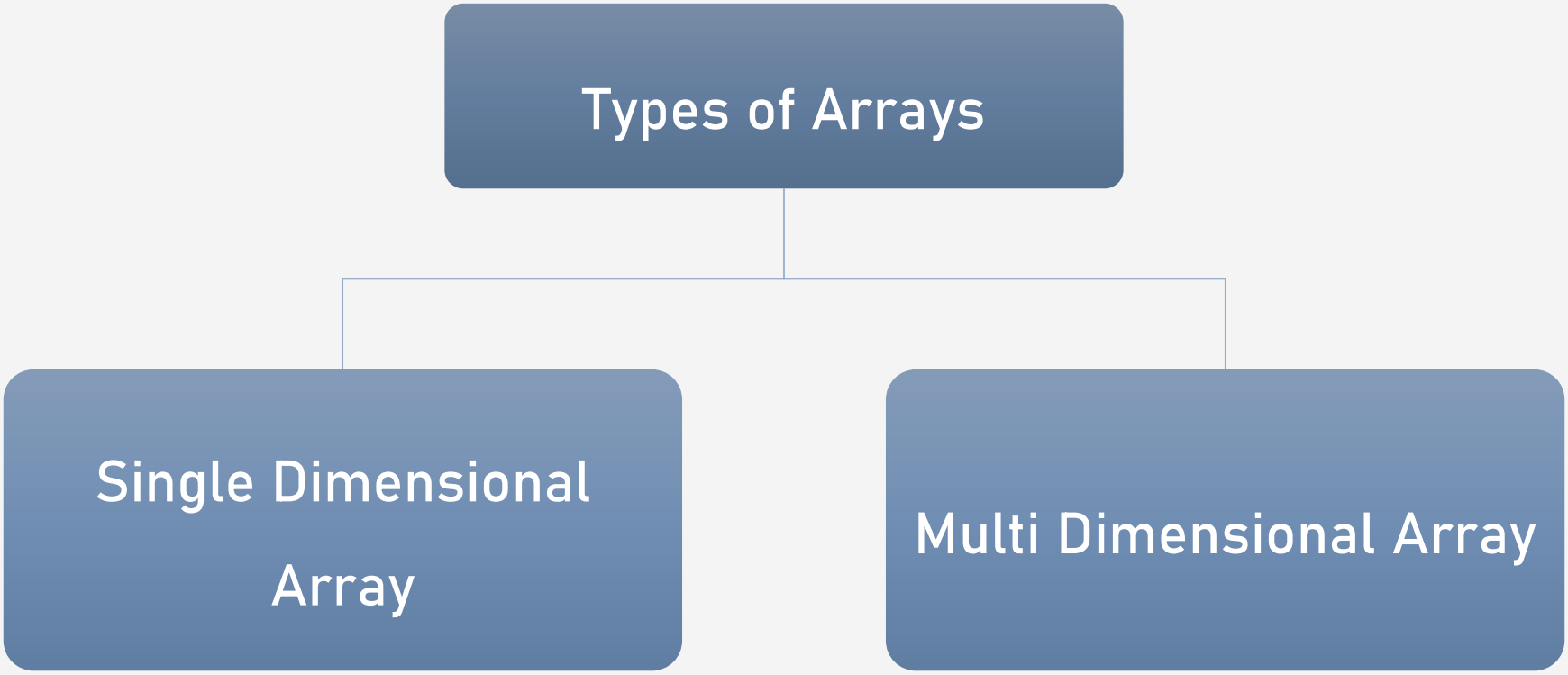
Arrays

- The size of an array must be specified by an int or short value and not long.
- The direct superclass of an array type is Object.
- Every array type implements the interfaces Cloneable and java.io.Serializable.
- Java array can be also be used as a static field, a local variable or a method parameter.

Representation of Array Elements

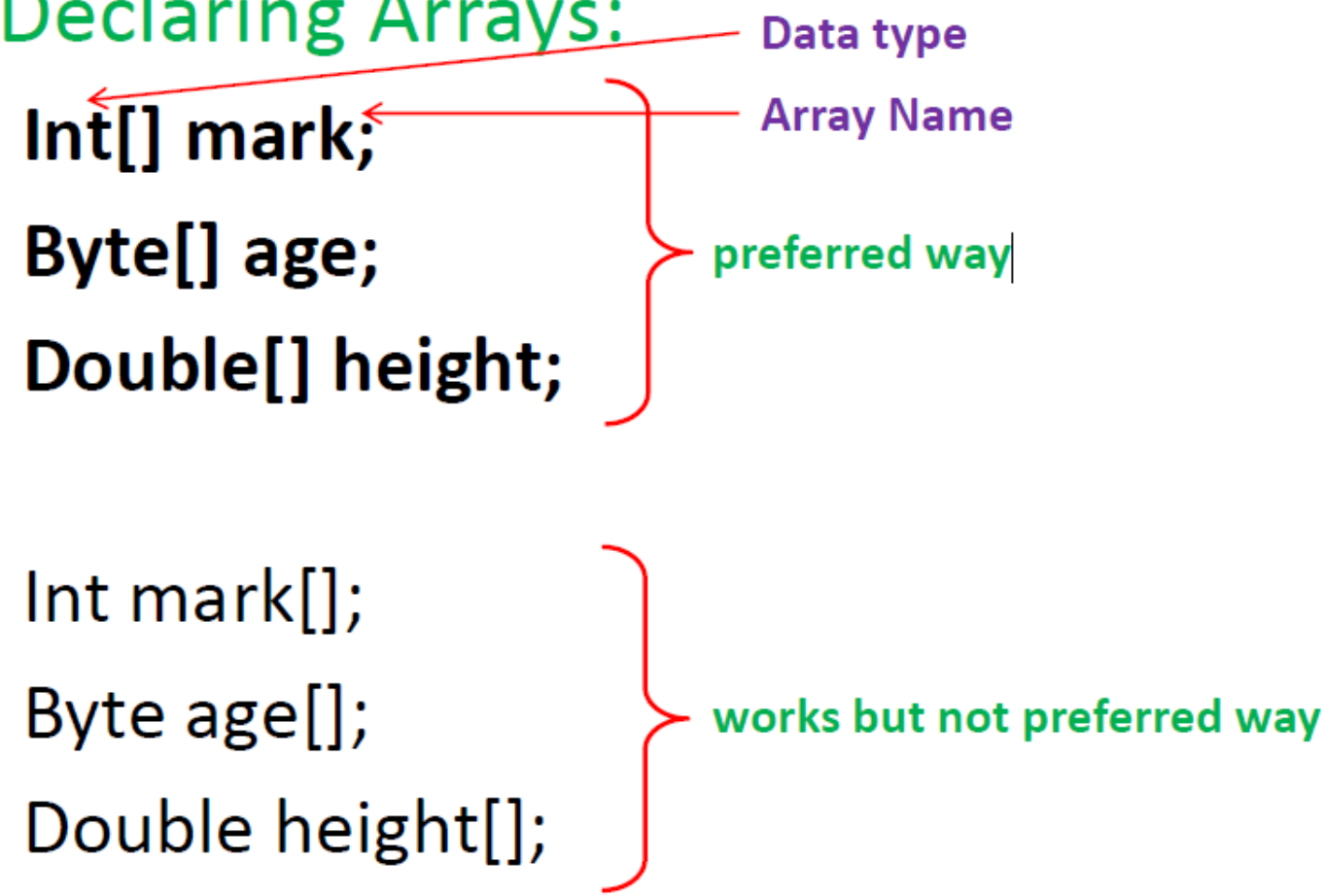


Types of Array



Declaration

- Declaring Arrays:



Single Dimensional Array Declaration

Syntax:

type var-name[];

OR

type[] var-name;

Example:

int intArray[];

or int[] intArray;

Instantiating an Array

- When an array is declared, only a reference of array is created.
- To actually create or give memory to array, you create an array like this:
- The general form of new as it applies to one-dimensional arrays appears as follows:

var-name = new type [size];

Example

- int intArray[]; //declaring array
- intArray = new int[20]; // allocating memory to array

OR

- int[] intArray = new int[20]; // combining both statements in one

Array Literal

- In a situation, where the size of the array and variables of array are already known, array literals can be used.

Example:

```
int[] intArray = new int[]{ 1,2,3,4,5,6,7,8,9,10 };  //
Declaring array literal
```

Accessing Array Elements using for Loop

- Each element in the array is accessed via its index.
- The index begins with 0 and ends at (total array size)-1.
- All the elements of array can be accessed using Java for Loop.

Accessing Array Elements using for Loop

Example: // accessing the elements of the specified array

```
for (int i = 0; i < arr.length; i++)

  System.out.println("Element at index " + i + " : "+ arr[i]);
```

Example

```
class Testarray{

public static void main(String args[])

{

int a[]=new int[5];//declaration and ins
tantiation

a[0]=10;//initialization

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

 //traversing array

 for(int i=0;i<a.length;i++)//length is t
 he property of array

 System.out.println(a[i]);

 }}
```

For-each Loop for Array

- We can also print the Java array using for-each loop.
- The Java for-each loop prints the array elements one by one.
- It holds an array element in a variable, then executes the body of the loop.

For-each Loop for Array

Syntax:

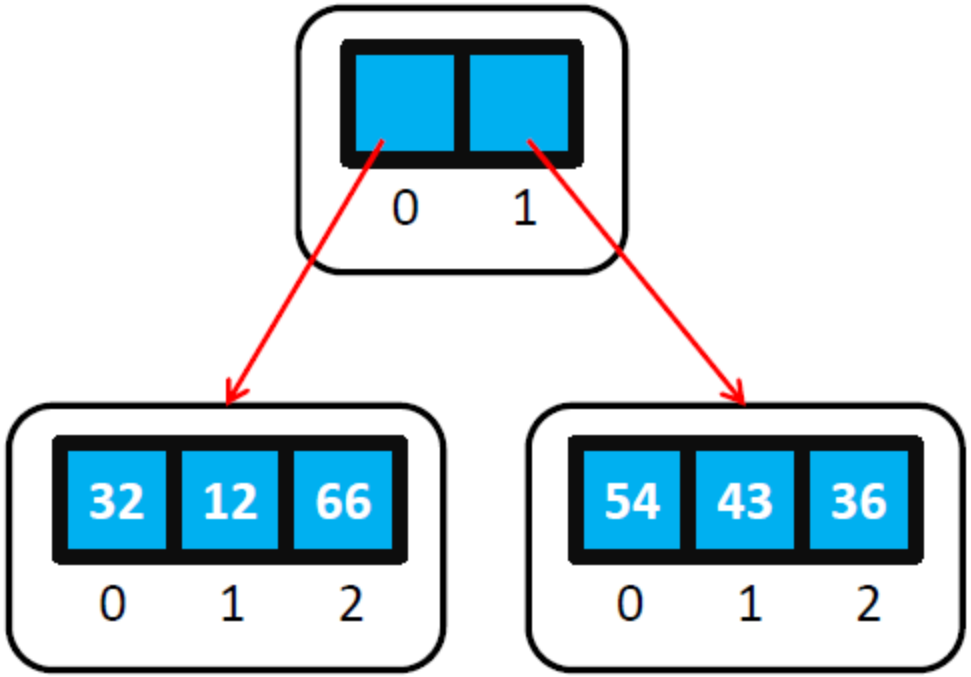
```
for(data_type variable:array)
{
//body of the loop
}
```

Example

```
class Testarray1{

public static void main(String args[]){

int arr[]={33,3,4,5};

//printing array using for-each loop

for(int i:arr)

System.out.println(i);

}}
```

Two Dimensional Array

- Int[ ][ ] har=new int[2][3];
- har[0][0]=32;
- har[0][1]=12;
- har[0][2]=66;
- har[1][0]=54;
- har[1][1]=43;
- har[1][2]=36;



Multidimensional Array

- Multidimensional Arrays can be defined in simple words as array of arrays. Data in multidimensional arrays are stored in tabular form (in row major order).

Syntax:

data_type[1st dimension][2nd dimension][]..[Nth dimension] array_name = new data_type[size1][size2]….[sizeN];

Example

```
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]); // Printing of array elements.
            }
        }    } }
```

Size of multidimensional arrays

- The total number of elements that can be stored in a multidimensional array can be calculated by multiplying the size of all the dimensions.

Example:

The array int[ ][ ] x = new int[10][20] can store a total of (10*20) = 200 elements.

String Array

- A String Array is an Array of a fixed number of String values.
- A string is an immutable object, which means the value of the string can not be changed.
- The String Array works similarly to other data types of Array.

Features of String Array

- It is an object of the Array.
- It can be declared by the two methods:
  - by specifying the size
  - without specifying the size.
- It can be initialized either at the time of declaration or by populating the values after the declaration.

Features of String Array

- The elements can be added to a String Array after declaring it.
- The String Array can be iterated using the for loop.
- The searching and sorting operation can be performed on the String Array.

Declaration of String Array

- String[ ] stringArray1.

  Declaration of the String Array without specifyingthe size

- String[ ] stringArray2=new String[2]

  Declaration by specifying the size

Initialization of String Array

1. String[] strAr1=new String[] {"Ani", "Sam", "Joe"

}; //inline initialization

Initialization of String Array

1. String[] strAr1=new String[] {"Ani", "Sam", "Joe"

}; //inline initialization

2. String[] strAr2 = {"Ani", "Sam", " Joe"};

Initialization of String Array

3. String[] strAr3= new String[3];

```
strAr3[0]= "Ani";  //Initialization after declaration
with specific size

strAr3[1]= "Sam";

strAr3[2]= "Joe";
```

Example

```
String[] strAr = {"Ani", "Sam", "Joe"};

for (int i=0; i<StrAr.length; i++)

{

System.out.println(strAr[i]);

}
```


That’s all for now…