Arrays in Java

Let's start with few important points.

In Java all arrays are dynamically allocated.

A Java array variable can also be declared like other variables with [ ] after the data type.

The variables in the array are ordered and each have an index beginning from 0.

Java array can be also be used as a static field, a local variable or a method parameter.

The size of an array must be specified by an int value and not long or short.

One-dimensional Array

Declaration

dataType[] arrayRefVar; ***// preferred way.***

or

dataType arrayRefVar[]; ***// works but not preferred way.***

An array declaration has two components: the type and the name.

***type* declares the element type of the array. The element type determines the data type of each element that comprises the array.**

Creating Array

arrayRefVar = new dataType[arraySize];

- **It creates an array using new dataType[arraySize].**
- **It assigns the reference of the newly created array to the variable arrayRefVar.**

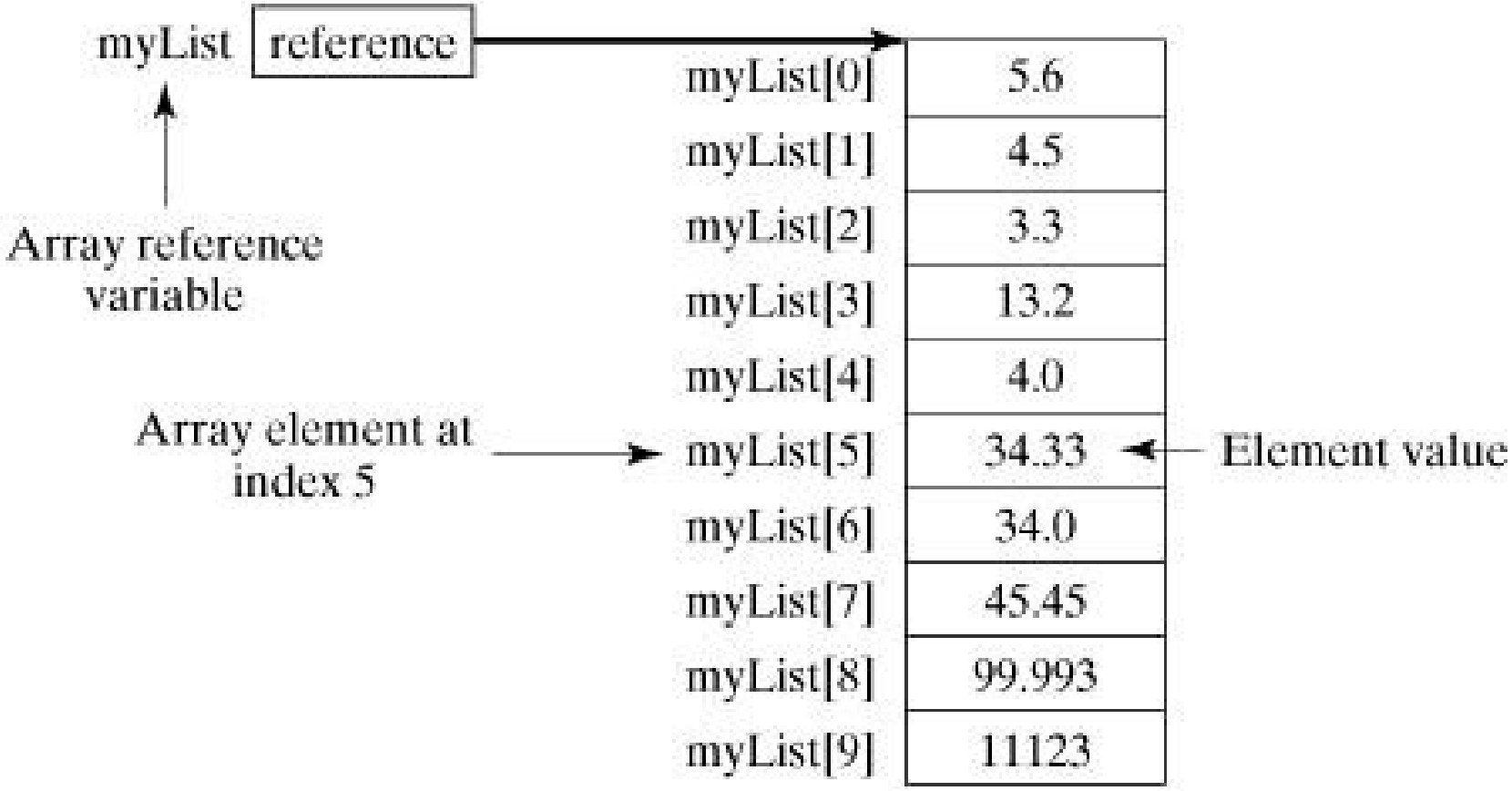
Other Alternative way of creating array

dataType[] arrayRefVar = new dataType[arraySize];
dataType[] arrayRefVar = {value0, value1, ..., valuek};

The array elements are accessed through the index. Array indices are 0-based; that is, they start from 0 to arrayRefVar.length-1.

Example:-

double[] myList = new double[10];



Accessing elements using for loop

```
public class TestArray {

   public static void main(String[] args) {
      double[] myList = {1.9, 2.9, 3.4, 3.5};

      // Print all the array elements
      for (int i = 0; i < myList.length; i++) {
         System.out.println(myList[i] + " ");
      }

      // Summing all elements
      double total = 0;
      for (int i = 0; i < myList.length; i++) {
         total += myList[i];
      }
      System.out.println("Total is " + total);

      // Finding the largest element
      double max = myList[0];
      for (int i = 1; i < myList.length; i++) {
         if (myList[i] > max) max = myList[i];
      }
      System.out.println("Max is " + max);
   }
}
```

Accessing elements using foreach loop

```
public class TestArray {

   public static void main(String[] args) {
      double[] myList = {1.9, 2.9, 3.4, 3.5};

      // Print all the array elements
      for (double element: myList) {
         System.out.println(element);
      }
   }
}
```

Passing Arrays to Methods

```
public static void printArray(int[] array) {
    for (int i = 0; i < array.length; i++) {
        System.out.print(array[i] + " ");
    }
}
```

Invoke the method as

printArray(new int[]{3, 1, 2, 6, 4, 2});

Returning an Array from a Method

```
public static int[] reverse(int[] list) {
   int[] result = new int[list.length];

   for (int i = 0, j = result.length - 1; i < list.length; i++, j--) {
      result[j] = list[i];
   }
   return result;
}
```

Multidimensional Array

dataType[][] arrayRefVar; (or)
dataType [][]arrayRefVar; (or)
dataType arrayRefVar[][]; (or)
dataType []arrayRefVar[];

Example:-

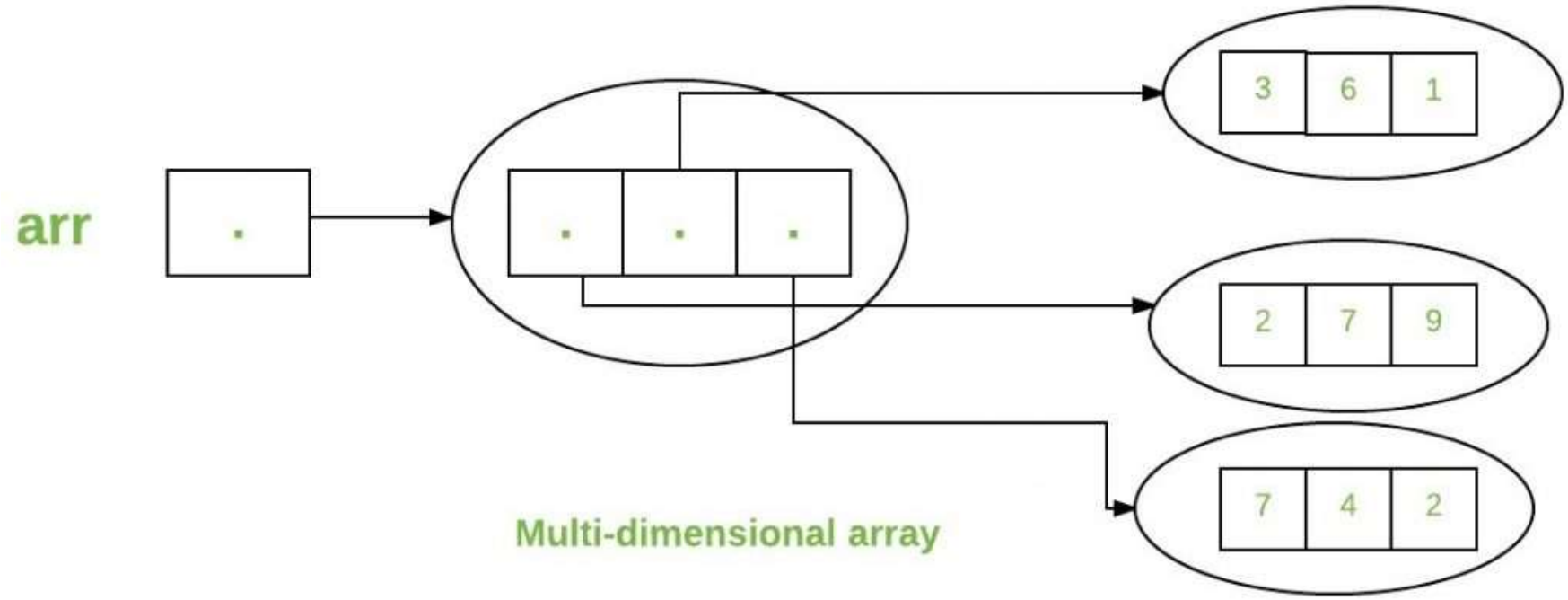
```
int[][] arr=new int[3][3];
//3 row and 3 column
```

Multidimensional arrays are arrays of arrays with each element of the array holding the reference of other array. These are also known as Jagged Arrays.

```
int[][] intArray = new int[10][20];
//a 2D array or matrix

int[][][] intArray = new int[10][20][10];
//a 3D array
```

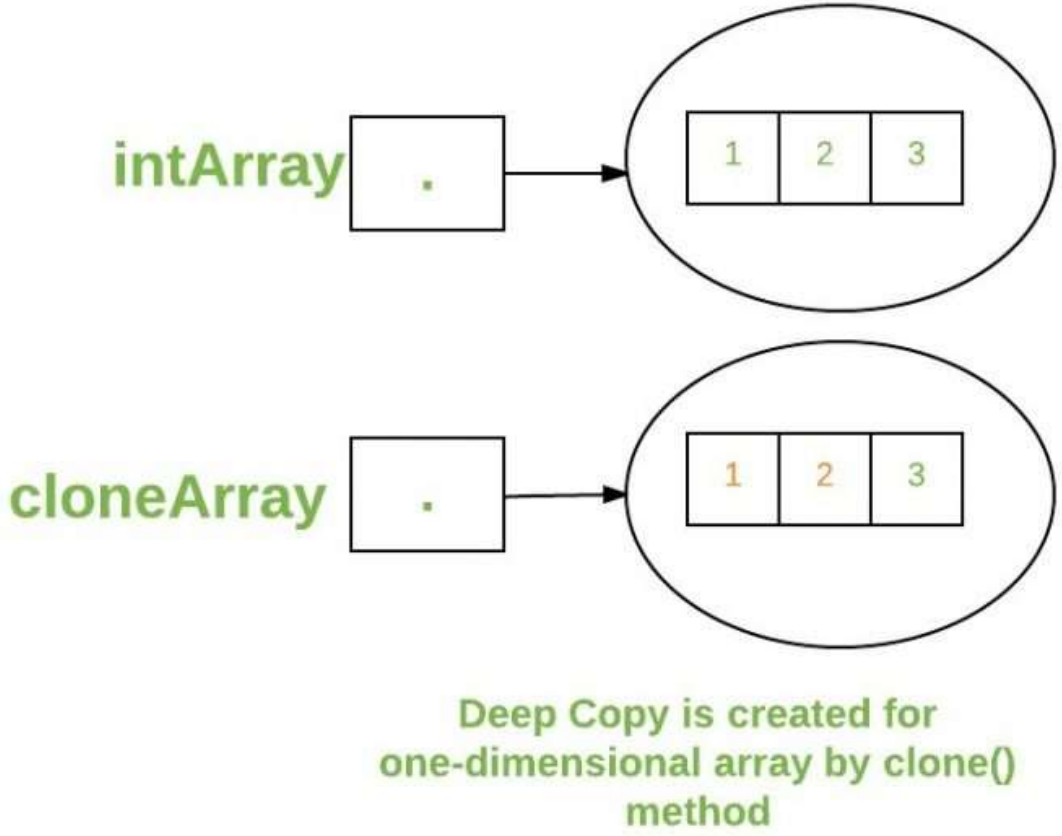
int arr[][] = { {2,7,9},{3,6,1},{7,4,2} };



Cloning of arrays

int intArray[] = {1,2,3};

int cloneArray[] = intArray.clone();



java.util.Arrays

The java.util.Arrays class contains a static factory that allows arrays to be viewed as lists.

- **This class contains various methods for manipulating arrays (such as sorting and searching).**

- **The methods in this class throw a NullPointerException if the specified array reference is null.**

Some of the methods of Arrays class

Sort

```
int numArr[] = { 23,43,26,65,35,16,74,27,98 };
Arrays.sort(numArr);
```

Binary search

static int binarySearch (int[] a, int fromIndex, int toIndex, int key)

a=> array to be searched

fromIndex=> starting index of the range over which the key is to be searched

toIndex=> the index of the last element in the range

key=> key to be searched for

index will be returned as output. If not found then -1

Arrays.binarySearch(numArr,3,7, 35)

Output: 4

Equals

static boolean equals (int [] a, int [] a2)

Returns true if both arrays are equal.

```
int[] array_One = { 1, 3, 5, 7 };
int[] array_Two = { 1, 3, 5, 7 };
Arrays.equals(array_One, array_Two);
```

Output: true

Further methods will be given as reference in the site.

Next Session: Character Class