JAVA PROGRAMMING BASICS

Module 1: Java Overview

Training program

1. Java Fundamentals
2. Start programming with Java, create simple console application
3. Classification of Data Types
4. Primitive types in java
5. Control Flow Statements
6. Arrays

Module contents

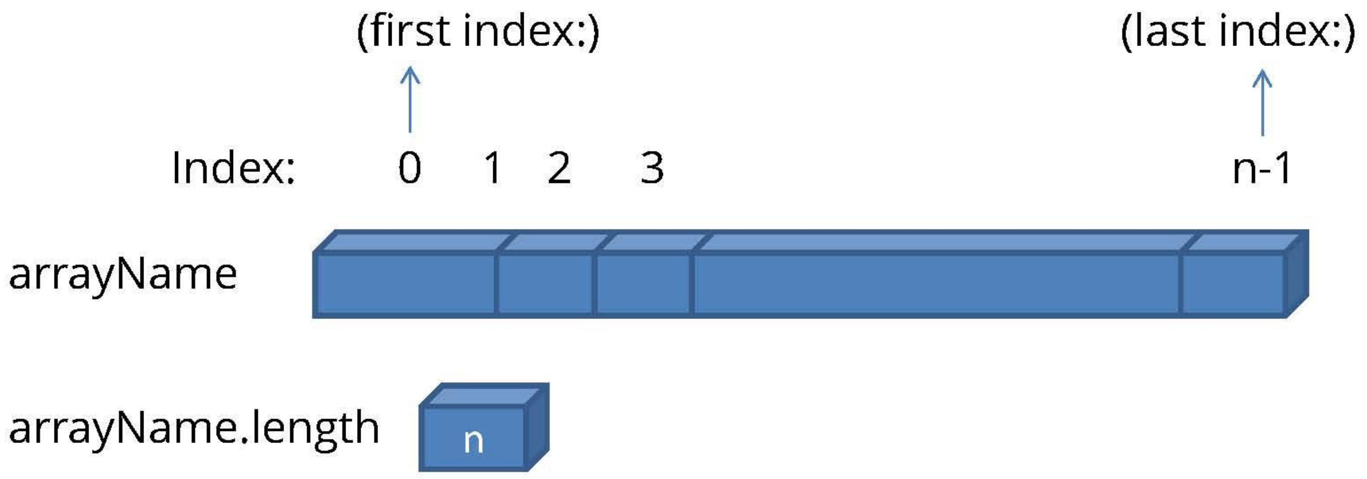
- Arrays
  - One dimensional arrays. Length of array
  - Working with Arrays and Array Elements
  - Multidimensional arrays
  - Resize an array
  - The Arrays Class
  - Array Manipulations: copying, equal check, search element, sort.

Module contents

- Arrays
  - One dimensional arrays. Length of array
  - Working with Arrays and Array Elements.
  - Multi - dimention arrays
  - Resize an array
  - The Arrays Class.
  - Array Manipulations: copying, equal check, search element, sort.

One dimensional arrays 1/2

- An *array* is a container object that holds a fixed number of values of a single type. The length of an array is established when the array is created. After creation, its length is fixed.

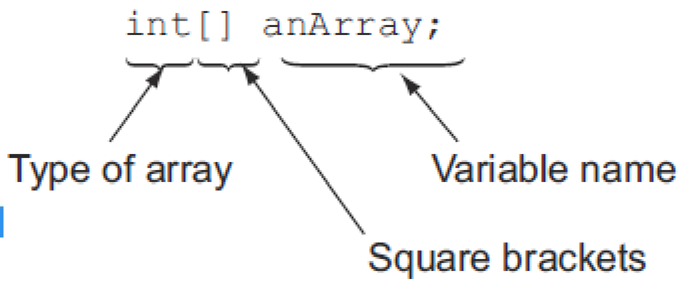


In the Java *arrays* are objects, are dynamically created, and may be assigned to variables of type Object . All methods of class Object may be invoked on an array.

One dimensional arrays 2/2

refers to null

```
1. int[] arr1;  // Declare an int array named marks
2. int arr2[];  // Same as above, (not recommended)
3. arr1 = new int[5];   // Allocate 5 elements
4. // Declare and allocate a 100-element array
5. int[] arr3 = new int[100];
6. // Declare, allocate a 6-element array
7. int[] arr4 = {12, 25, 37, 44, 55, 66};
8. // size of array deduced from the number of items
```



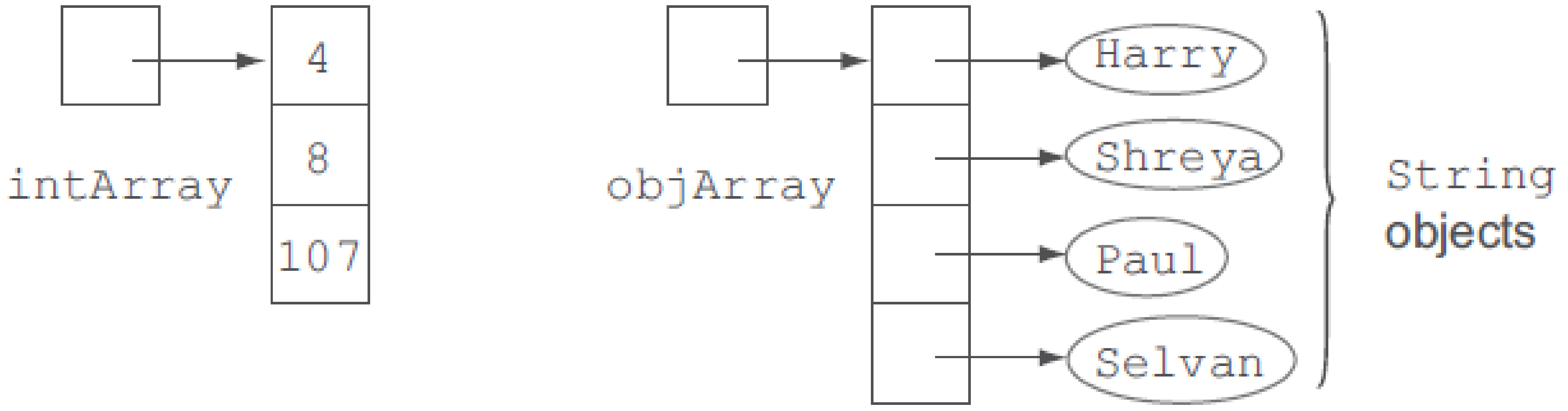
Length of array

In Java, the length of array is kept in an *associated variable* called length and can be retrieved using "*arr*.length"

```java
int[] arr = new int[100];
int num = arr.length;  // num is 100
System.out.println(num);
```

Array of primitive data vs array of objects

```
int[] intArray = {4,8,107};
String[] objArray = {"Harry", "Shreya", "paul", "Salvan"};
```



Array of primitive data

Array of objects

An array of primitives stores a collection of values that constitute the primitive values themselves. (With primitives, there are no objects to reference.)
An array of objects stores a collection of values, which are in fact heap-memory addresses or pointers.

Module contents

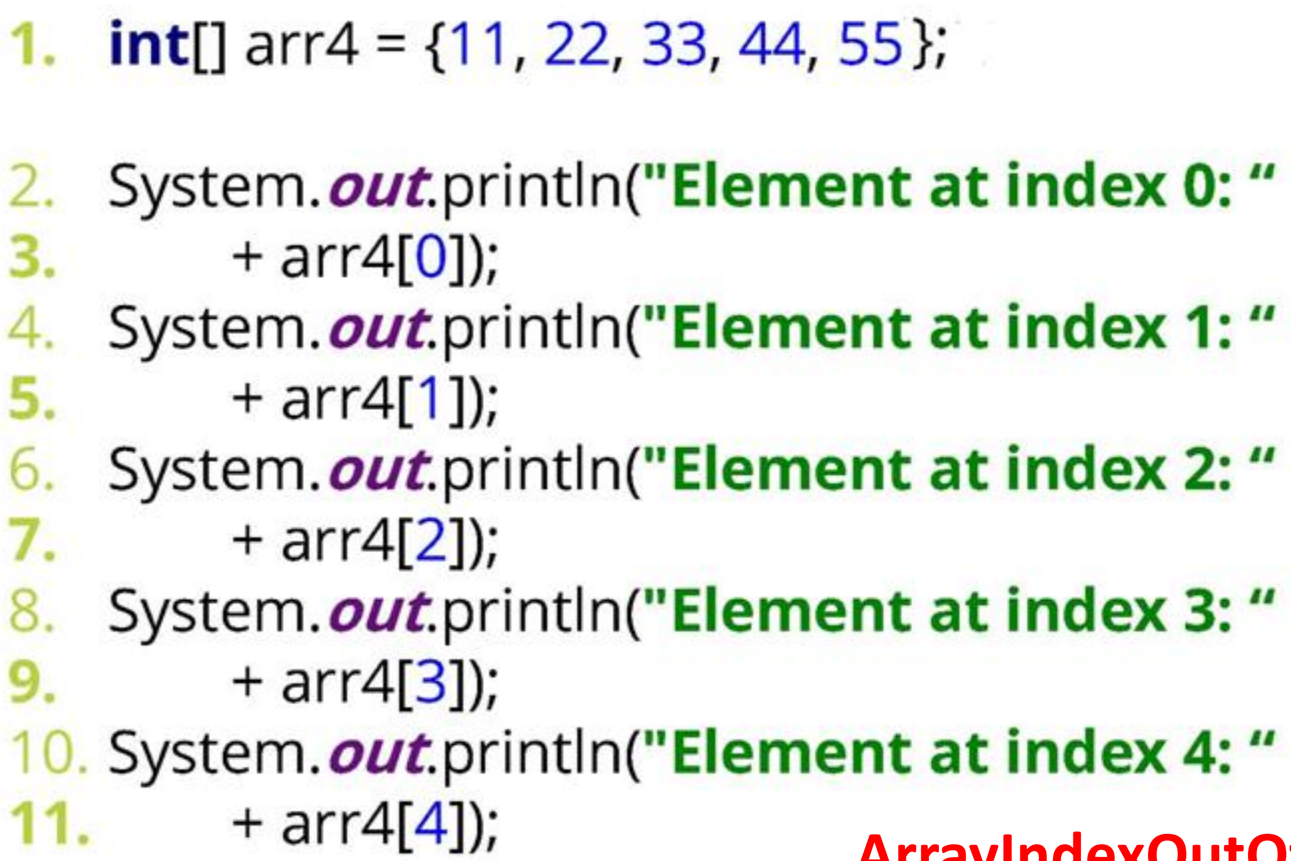
- Arrays
  - One dimensional arrays. Length of array.
  - Working with Arrays and Array Elements.
  - Multi - dimention arrays
  - Resize an array
  - The Arrays Class.
  - Array Manipulations: copying, equal check, search element, sort.

Working with Arrays and Array Elements 1/2

```java
int[] arr = new int[5];
// Assign values to the elements
    arr[0] = 11;
    arr[1] = 22;
    arr[2] = 33;
    arr[3] = 44;
    arr[4] = 55;

    System.out.println(arr[0]);
    System.out.println(arr[4]);
```

Working with Arrays and Array Elements 2/2



```
int[] arr4 = {11, 22, 33, 44, 55};

System.out.println("Element at index 0: “
        + arr4[0]);
System.out.println("Element at index 1: “
        + arr4[1]);
System.out.println("Element at index 2: “
        + arr4[2]);
System.out.println("Element at index 3: “
        + arr4[3]);
System.out.println("Element at index 4: “
        + arr4[4]);
```

ArrayIndexOutOfBoundException

```
System.out.println("Element at index 5: " + arr[5]););
```

Array elements random values generation

```
public static void main(String[] args) {
    int[] rndArr = new int[5];
    for (int i = 0; i < rndArr.length; i++) {
        rndArr[i] = (int) (Math.random() * 10);
        System.out.println(rndArr[i]);
    }
}
```

<u>Output:</u>

```
7
6
4
1
4
```

Foreach loop
(enhanced for loop)

- In Java SE 5 introduced an enhanced for loop for iterating over arrays:

```java
int[] arr1 = { 1, 2, 3, 4, 5, 6 };

for ( int element : arr1 ) {
    System.out.println(element);
}
```

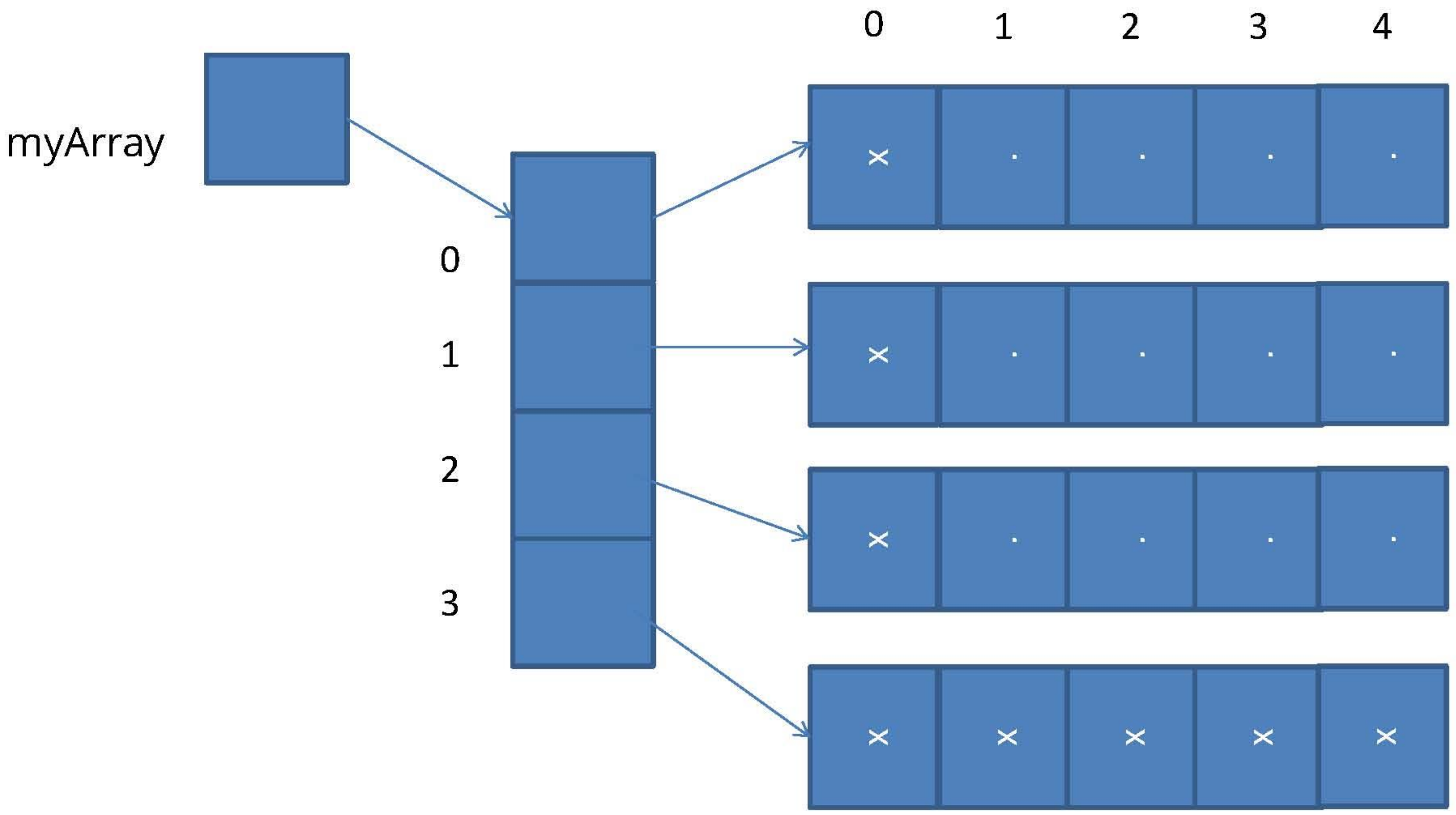
for vs for-each (enhanced for) loop

- The *enhanced for loop* can’t be used to initialize an array and modify its elements.
- The *enhanced for loop* can’t be used to delete the elements of a collection.
- The *enhanced for loop* can’t be used to iterate over multiple collections or arrays in the same loop.

Module contents

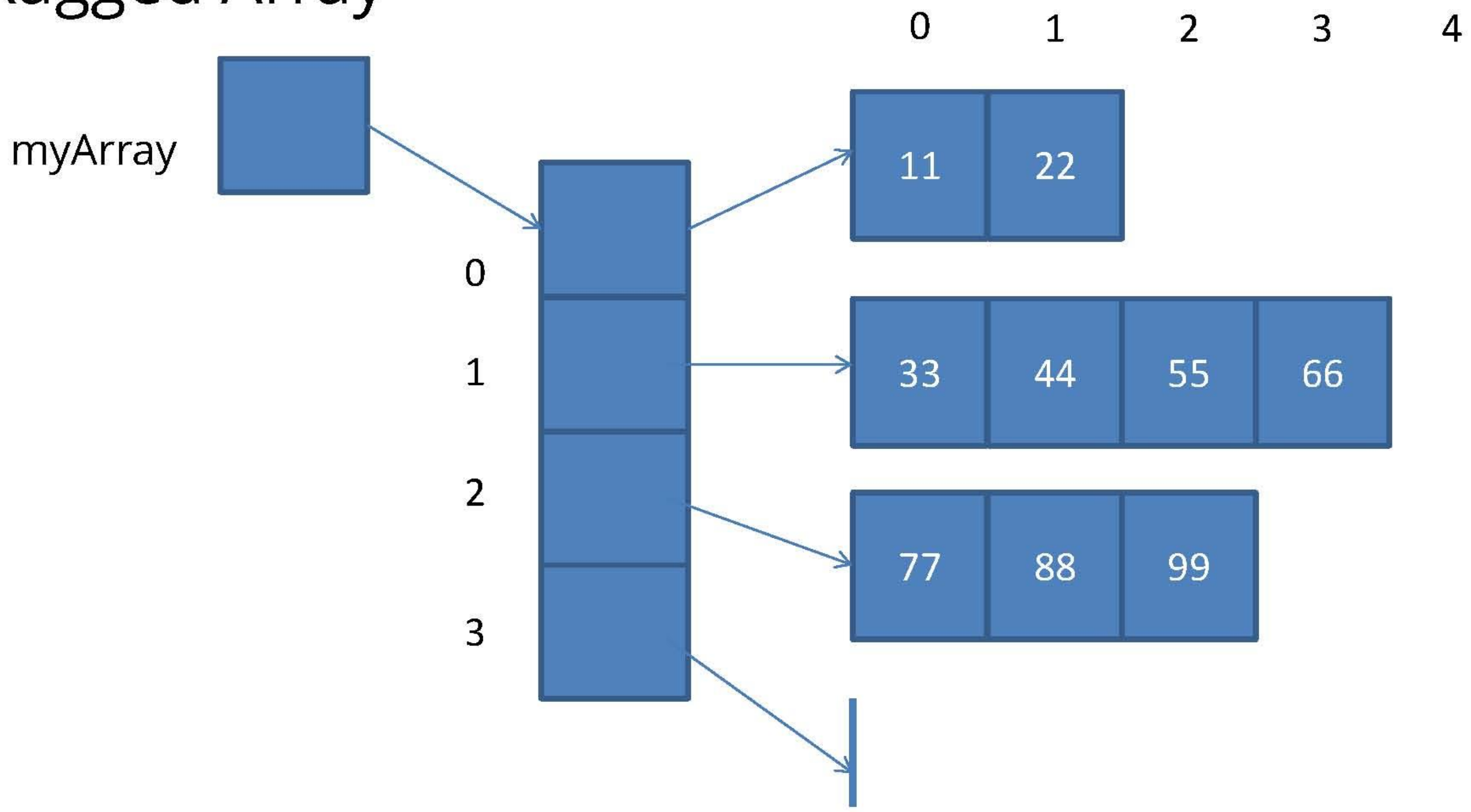
- Arrays
  - One dimensional arrays. Length of array.
  - Working with Arrays and Array Elements.
  - Multi - dimention arrays
  - Resize an array
  - The Arrays Class.
  - Array Manipulations: copying, equal check, search element, sort.

Multi - dimention arrays 1/3



Multi – dimention (Ragged) arrays 2/3

- Ragged Array



Multi - dimention arrays 3/3

```
int[][] marr = {{11,22},{33,44,55,66,},{77,88,99},{}};

marr[1][2] = 123;
marr[2][2] = 555;

System.out.println(marr[0][0]);
System.out.println(marr[1][0]);
System.out.println(marr[0][1]);
```

Module contents

- Arrays
  - One dimensional arrays. Length of array.
  - Working with Arrays and Array Elements.
  - Multi - dimention arrays
  - Resize an array
  - The Arrays Class.
  - Array Manipulations: copying, equal check, search element, sort.

Resize an array

- You cannot resize an array.
- You can use the same reference variable to refer to an entirely new array, such as:

- **int**[] myArray = **new** **int**[6];
- myArray = **new** **int**[10];

Module contents

- Arrays
  - One dimensional arrays. Length of array.
  - Working with Arrays and Array Elements.
  - Multi - dimention arrays
  - Resize an array
  - The Arrays Class.
  - Array Manipulations: copying, equal check, search element, sort.

The Arrays Class 1/2

- Class Arrays providing static methods for common array manipulations:
  - **sort**(array) : Arranges array elements into increasing order.
  - **binarySearch**(array , element) : Determines whether an array contains a specific value and, if so, returns where the value is located.
  - **equal(**array) : Compares arrays.
  - **fill**(array , element) : Places Values into an array.
  - **toString**() : Converts array to String.

The Arrays Class 2/2

- We can copy arrays using **copyof** method of the class Arrays Or using class System's static arraycopy method.
- To use Arrays Class we import it by import java.util.Arrays ;

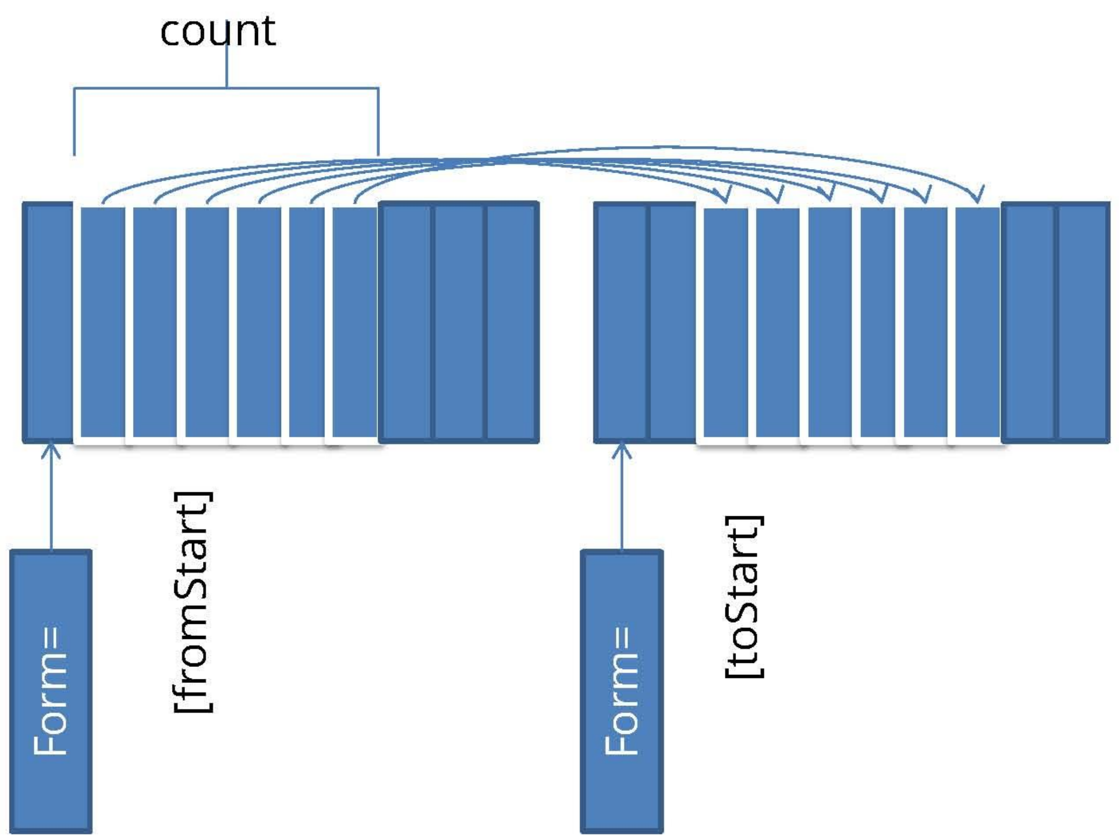
Module contents

- Arrays
  - One dimensional arrays. Length of array.
  - Working with Arrays and Array Elements.
  - Multi - dimention arrays
  - Resize an array
  - The Arrays Class.
  - Array Manipulations: copying, equal check, search element, sort.

Copying Arrays 1/2

```
//The System.arraycopy() method to copy arrays is:
//original array
      int[] arr1 = { 1, 2, 3, 4, 5, 6 };
// new larger array
      int[] arr2 = { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };
// copy all of the myArray array to the hold
// array, starting with the 0th index
System.arraycopy(arr1, 0, arr2, 0, arr1.length);
```

Copying Arrays 2/2



Arrays.equals

```
int[] arr1 = {1,2,3,4,5,6,7,8,9};
int[] arr2 = {1,2,3,4,5,6,7,8,9};
int[] arr3 = {1,2,5,5,5,5,5,8,9};
// compare  references
System.out.println(arr1 == arr2);
// some as above, don't use it
System.out.println(arr1.equals(arr2));
// check equality
System.out.println(Arrays.equals(arr1, arr2));
System.out.println(Arrays.equals(arr1, arr3));
```

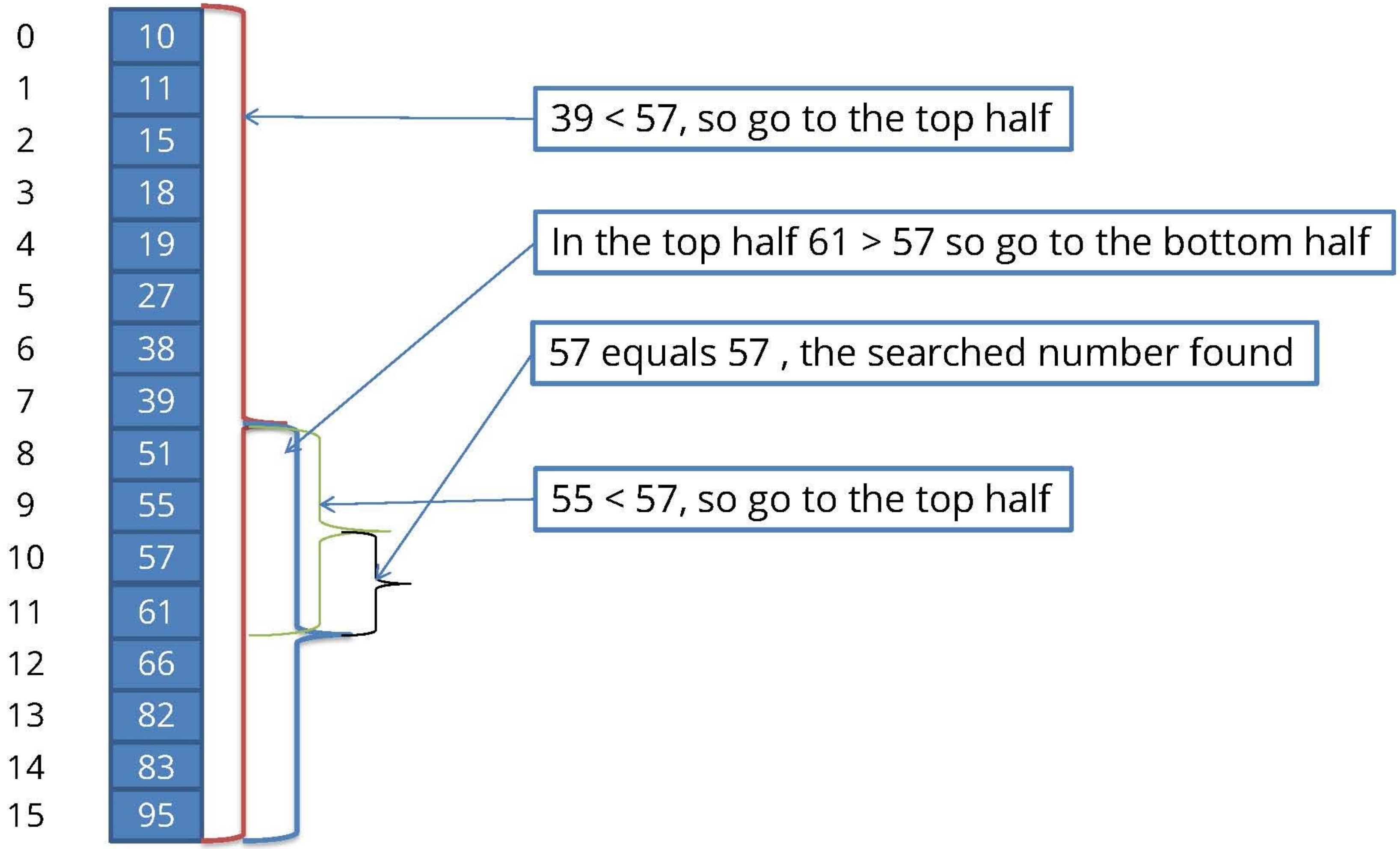
Sort and Search element

```
// initializing unsorted int array
int intArr[] = {55, 57, 61, 66,18, 19, 27, 38,10, 11, 15, 39, 51, 82,
83, 95};

// sorting array
Arrays.sort(intArr);

// let us print all the elements available in list
System.out.println("The sorted int array is:");
System.out.println(Arrays.toString(intArr));
// entering the value to be searched
int searchVal = 57;
int retVal = Arrays.binarySearch(intArr,searchVal);
System.out.println("The index of element 57 is : " + retVal);
```

Binary Search element


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
10
11
15
18
19
27
38
39
51
55
57
61
66
82
83
95
39 < 57, so go to the top half
In the top half 61 > 57 so go to the bottom half
57 equals 57 , the searched number found
55 < 57, so go to the top half