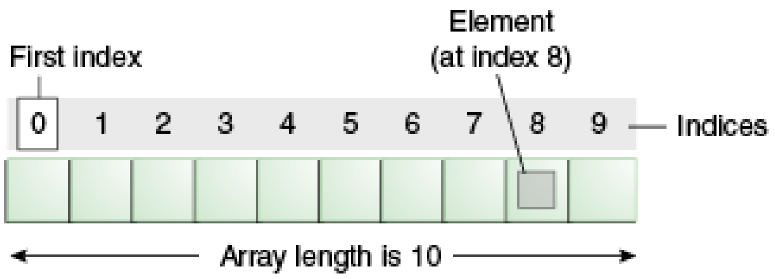
Java Arrays

• Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

• Normally, an array is a collection of similar type of elements which has contiguous memory location.

• **Java array** is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. 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.



Advantages

• **Code Optimization:** It makes the code optimized, we can retrieve or sort the data efficiently.

• **Random access:** We can get any data located at an index position.

Disadvantages

• **Size Limit:** We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.

Types of Array in java

There are two types of array.

- Single Dimensional Array
- Multidimensional Array

Single Dimensional Array in Java

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

```
String[] cars;
```

• Also, we can declare using below types.

```
dataType[] arr; (or)

dataType []arr; (or)

dataType arr[];
```

• We have now declared a variable that holds an array of strings.
• To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

To create an array of integers, you could write:

```
String[] myNum = {1,2,3,4};
```

Access the Elements of an Array

• You access an array element by referring to the index number.
• This statement accesses the value of the first element in cars:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars[0]);
// Outputs Volvo
```

Note: *Array indexes start with 0: [0] is the first element. [1] is the second element, etc.*

Change an Array Element

• To change the value of a specific element, refer to the index number:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
cars[0] = "Opel";
System.out.println(cars[0]);
// Now outputs Opel instead of Volvo
```

Array Length

- To find out how many elements an array has, use the length property:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars.length);
// Outputs 4
```

Loop Through an Array

- You can loop through the array elements with the for loop, and use the length property to specify how many times the loop should run.
- The following example outputs all elements in the **cars** array:

```
   String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
   for (int i = 0; i < cars.length; i++) {
     System.out.println(cars[i]);
   }
```

Loop Through an Array with For-Each

There is also a "**for-each**" loop, which is used exclusively to loop through elements in arrays:

```
for (type variable : arrayname) {
  ...
}
```

The following example outputs all elements in the **cars** array, using a "**for-each**" loop:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
  System.out.println(i);
}
```

- The example above can be read like this: **for each** String element (called **i** - as in **i**ndex) in **cars**, print out the value of **i**.
- If you compare the for loop and **for-each** loop, you will see that the **for-each** method is easier to write, it does not require a counter (using the length property), and it is more readable.

Multidimensional Arrays

- A multidimensional array is an array containing one or more arrays.
- To create a two-dimensional array, add each array within its own set of **curly braces**:

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

- **myNumbers** is now an array with two arrays as its elements.
- To access the elements of the **myNumbers** array, specify two indexes: one for the array, and one for the element inside that array. This example accesses the third element (2) in the second array (1) of myNumbers:

```java
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
int x = myNumbers[1][2];
System.out.println(x); // Outputs 7
```

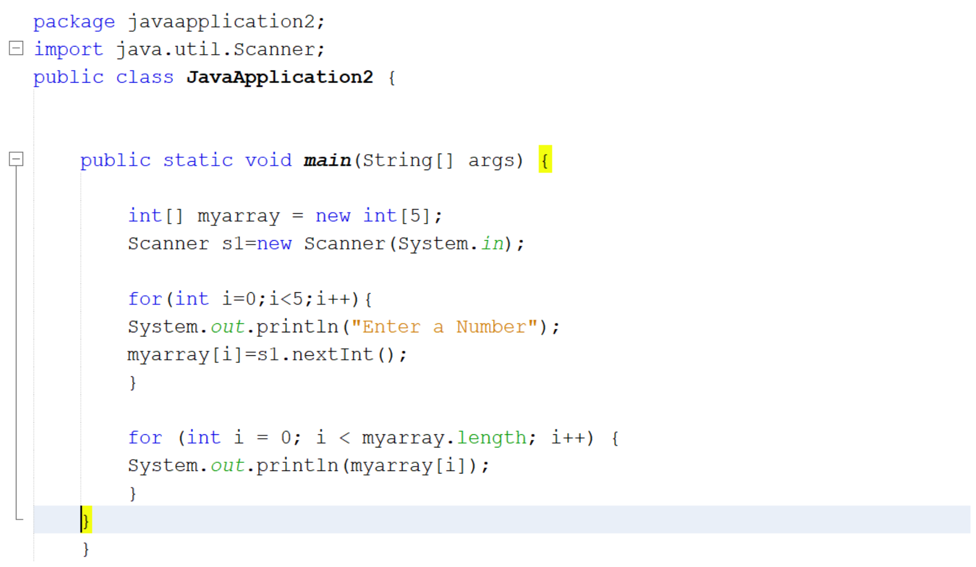
- We can also use a for loop inside another for loop to get the elements of a two-dimensional array (we still have to point to the two indexes):

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

Exercise : Calculate the average value of array elements

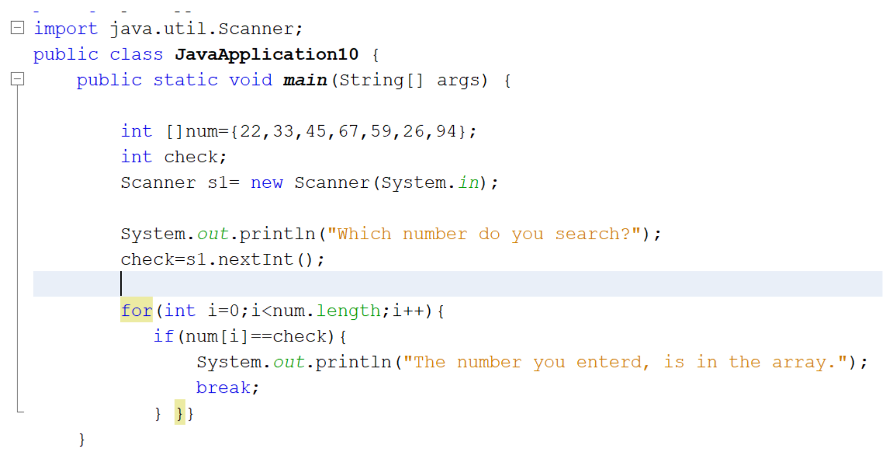
```java
public class Exercise4 {
public static void main(String[] args) {

      int[] numbers = new int[]{20, 30, 25, 35, -16, 60, -100};
      //calculate sum of all array elements
      int sum = 0;
      for(int i=0; i < numbers.length ; i++)
       sum = sum + numbers[i];
      //calculate average value
       double average = sum / numbers.length;
       System.out.println("Average value of the array elements is : " + average);
   }
}
```

Getting user inputs for Array



```
package javaapplication2;
import java.util.Scanner;
public class JavaApplication2 {


    public static void main(String[] args) {

        int[] myarray = new int[5];
        Scanner s1=new Scanner(System.in);

        for(int i=0;i<5;i++){
        System.out.println("Enter a Number");
        myarray[i]=s1.nextInt();
        }

        for (int i = 0; i < myarray.length; i++) {
        System.out.println(myarray[i]);
        }
    }
    }
```

Check the user input value is in a array

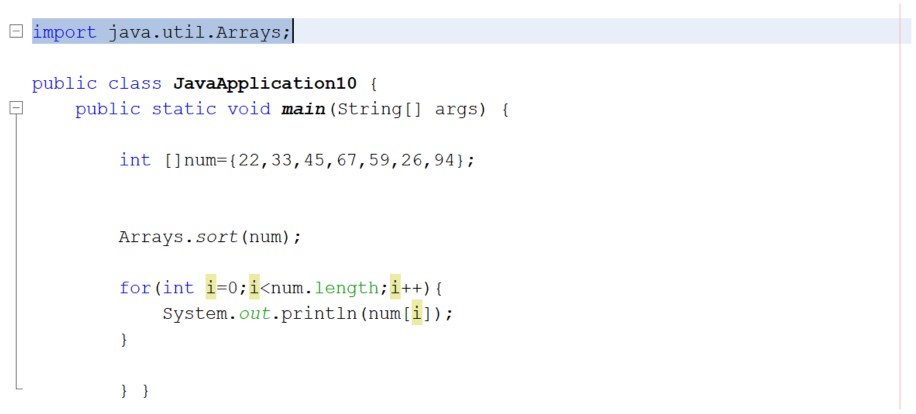


```
import java.util.Scanner;
public class JavaApplication10 {
    public static void main(String[] args) {

        int []num={22,33,45,67,59,26,94};
        int check;
        Scanner s1= new Scanner(System.in);

        System.out.println("Which number do you search?");
        check=s1.nextInt();

        for(int i=0;i<num.length;i++){
           if(num[i]==check){
               System.out.println("The number you enterd, is in the array.");
               break;
           } }}
    }
```

Methods in Java Arrays with examples

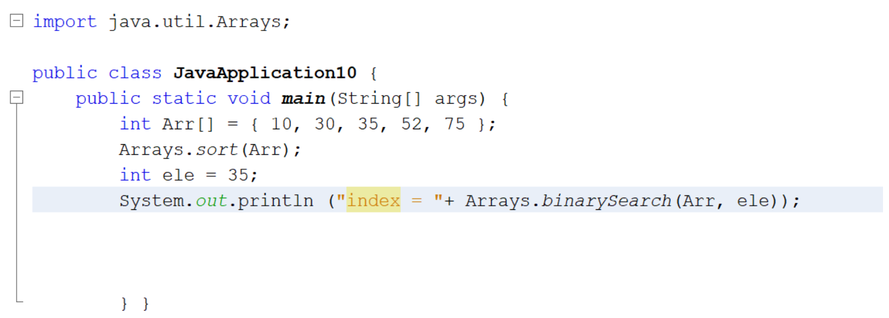
The Arrays class that belongs to the java. The util package belongs to the Java Collection Framework. Array class gives methods that are static so as to create as well as access Java arrays dynamically. Arrays have got only static methods as well as methods of Object class.

1. Sort



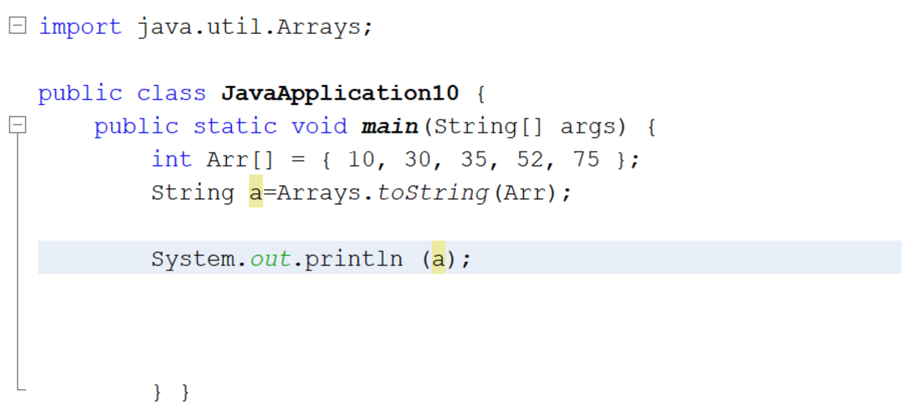
```
import java.util.Arrays;

public class JavaApplication10 {
    public static void main(String[] args) {

        int []num={22,33,45,67,59,26,94};


        Arrays.sort(num);

        for(int i=0;i<num.length;i++){
            System.out.println(num[i]);
        }

        } }
```

2. **binarySearch(itemToSearch) :** This method can search the index number.



```
import java.util.Arrays;

public class JavaApplication10 {
    public static void main(String[] args) {
        int Arr[] = { 10, 30, 35, 52, 75 };
        Arrays.sort(Arr);
        int ele = 35;
        System.out.println ("index = "+ Arrays.binarySearch(Arr, ele));



        } }
```

3. toString – Converting the array into a String.



```
import java.util.Arrays;

public class JavaApplication10 {
    public static void main(String[] args) {
        int Arr[] = { 10, 30, 35, 52, 75 };
        String a=Arrays.toString(Arr);

        System.out.println (a);



        } }
```