

# ARRAYS

# LOOPS

# AGENDA

- Arrays
- While
- For
- Break, continue
- Search item in the array
- Sorting
- Practical tasks

# Array

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.

```
String dayWeek[] = new String[7];
```

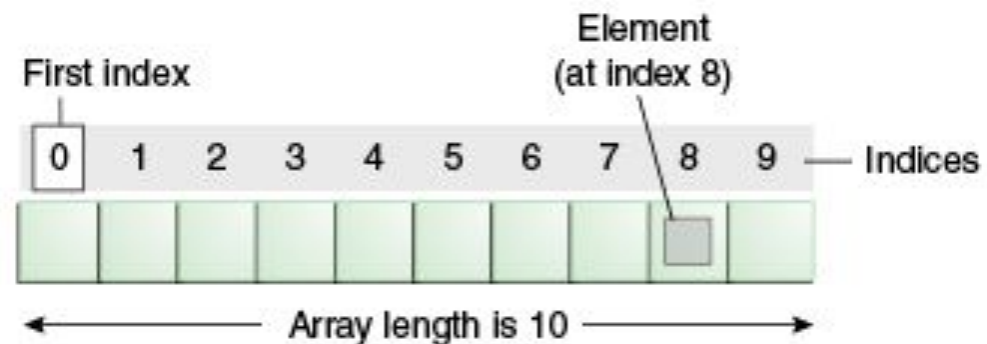
```
String[] month;
```

```
month = new String[12];
```

Each item in an array is called an *element*, and each element is accessed by its numerical *index*. As shown in the preceding illustration, numbering begins with 0.

```
dayWeek[0] = "Monday";
```

```
month[11] = "December";
```



# Array

```
int[] monthDays = {31, 28, 31, 30, 31, 30,  
                  31, 31, 30, 31, 30, 31} ;
```

```
int[] monthDays = new int[12];  
monthDays[0] = 31;  
monthDays[1] = 28;  
// . . .  
monthDays[11] = 31;
```

```
int n = monthDays.length;    // n = 12  
System.out.println(monthDays);    //[I@659e0bfd
```

Need override toString();  
or use Arrays.toString(monthDays);

# Array

```
char twod1[][]= new char[3][4];
```

```
char[][] twod2= new char[3][4];
```

```
double[][] m= { {0, 1, 2, 3},  
                {4, 5, 6, 7},  
                {8, 9, 10, 11},  
                {12, 13, 14, 15} };
```

```
int[][] twoD= new int[4][];
```

```
twoD[0]= new int[5];
```

```
twoD[1]= new int[2];
```

```
twoD[2]= new int[3];
```

```
twoD[3]= new int[7];
```

```
int[][] irregular={{1},{2,3,4},{5},{6,7}}; // Rows not equal
```

```
double m[4][4]; // Error;
```

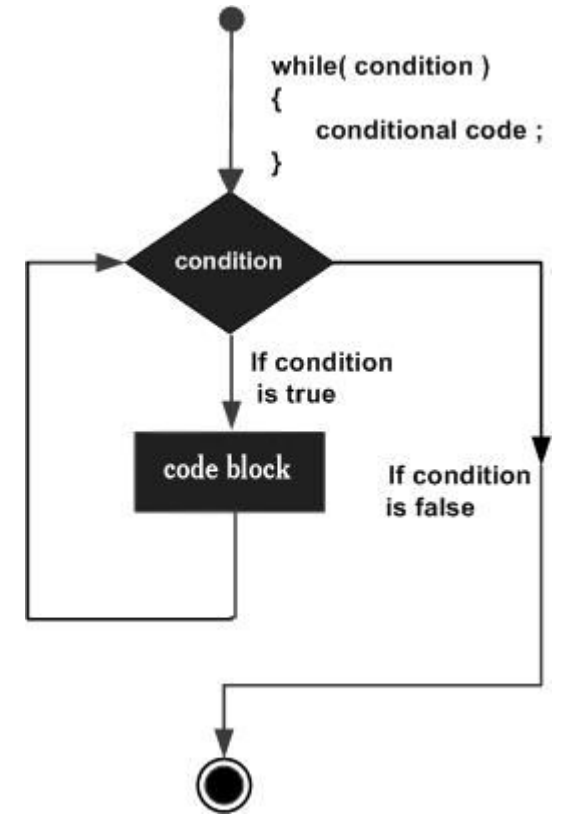
# while

A **while** loop statement repeatedly executes a target statement as long as a given condition is true.

```
while (condition){  
    statements;  
}
```

What will be displayed?

```
int number = 0;  
while (number <= 5) {  
    System.out.println(number);  
    number++;  
}
```



softserve

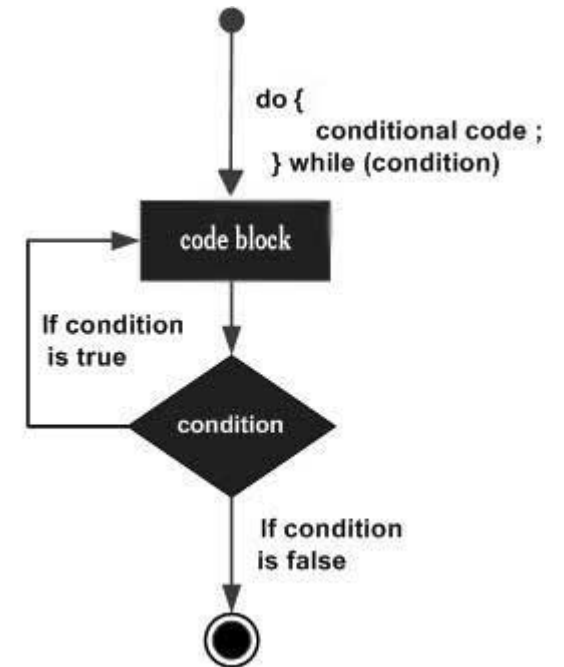
# do while

The difference between **do-while** and **while** is that do-while evaluates its expression at the bottom of the loop instead of the top. Therefore, the statements within the do block are always executed at least once

```
do {  
    statements;  
} while (condition);
```

What will be displayed?

```
int i = 0;  
do {  
    System.out.print(++i);  
} while (i < 5);
```



softserve

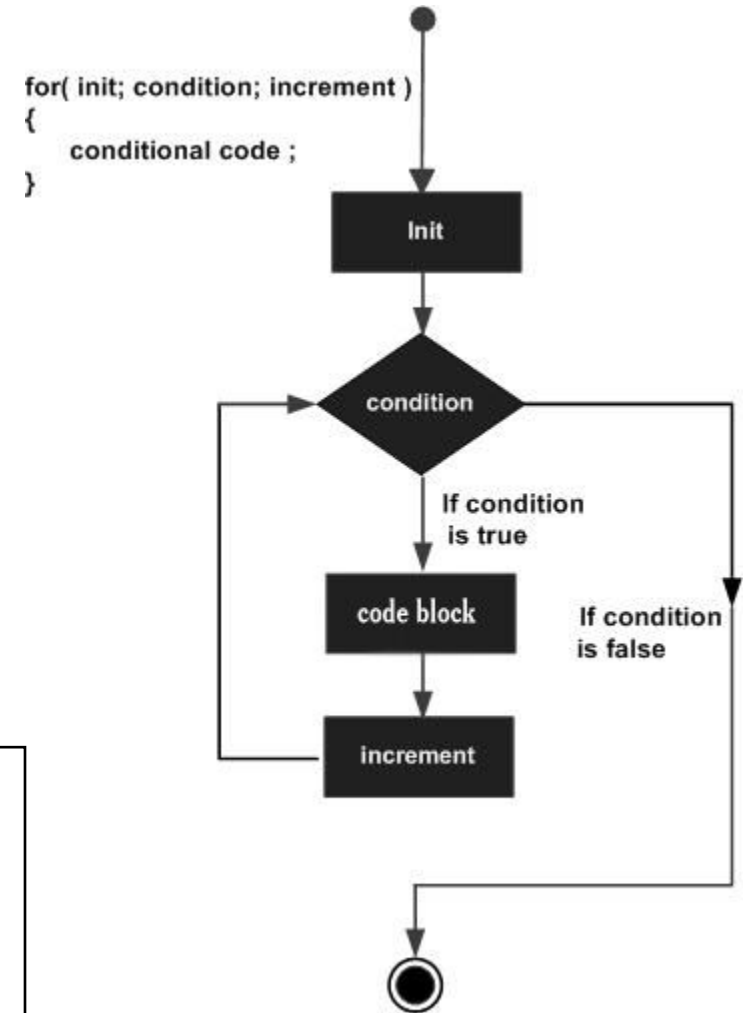
# for

The **for** statement provides a compact way to iterate over a range of values.

```
for (init; condition; increment) {  
    statement(s);  
}
```

What will be displayed?

```
int[] numbers = {5, 6, 8, 3, 5, 7, 9};  
for(int i = 0; i < numbers.length; i++){  
    System.out.println(numbers[i]);  
}
```





# for

Another representation of statement **for**

```
for (type variable : collection) {  
    statement(s);  
}
```

```
int[] workHours = { 8, 6, 8, 7, 7 };  
  
for (int h = 0; h < workHours.length; h++) {  
    System.out.println(workHours[h]);  
}  
  
or  
  
for (int h : workHours) {  
    System.out.println(h);  
}
```

# break

The **break** statement terminates the for, while and do-while loop

What will be displayed?

```
Scanner sc = new Scanner(System.in);
int n = 0;
for (int i = 0; i < 5; i++) {
    System.out.println("Input number");
    n = Integer.parseInt(sc.nextLine());
    if (n < 0){
        break;
    }
}
System.out.println(n);
sc.close();
```

# continue

The **continue** statement skips the current iteration the **for**, **while** and **do-while** loop

What will be displayed?

```
Scanner sc = new Scanner(System.in);
int sum = 0;
int n;
for (int i = 0; i < 5; i++) {
    System.out.println("Input number");
    n = Integer.parseInt(sc.nextLine());
    if (n < 0){
        continue;
    }
    sum += n;
}
System.out.println(sum);
sc.close();
```

# Sum, product, amount

There's an array `int[] arr = {2, -5, 7, -4, 8};`

What will results after running next code?

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

```
int product = 1;
for (int i = 0; i < 5; i++) {
    if (arr[i] > 0) {product = product * arr[i];}}
System.out.println("Product = " + product);
```

```
int amount = 0;
for (int a : arr) {
    if (a > 0 && a <= 7) { amount++; }
}
System.out.println("Amount = " + amount);
```

# Minimum, maximum ...

There's an array `int[] arr = {2, -5, 7, -4, 8};`  
What will be results after running next code?

```
int max = arr[0];
int imax = 0;
int i = 0;
while (i < arr.length) {
    if (arr[i] > max) {
        max = arr[i];
        imax = i;
    }
    i++;
}
System.out.print("Maximum = " + max);
System.out.println(" is in " + (imax + 1) + " place");
```

# Sorting

There's an array `int[] arr = {2, -5, 7, -4, 8};`  
What will be results after running next code?

```
int tmp;
for (int i = 0; i < arr.length - 1; i++) {
    for (int j = i + 1; j < arr.length; j++) {
        if (arr[i] < arr[j]) {
            tmp = arr[i];
            arr[i] = arr[j];
            arr[j] = tmp;
        }
    }
}
for (int i = 0; i < arr.length; i++) {
    System.out.println(arr[i]);
}
```

Sorting elements by descending  
using ***Bubble Sort***

**softserve**

# The **Arrays** class

- The **Arrays** class in **java.util** package is a part of the **Java Collection Framework**.
- This class provides methods *to dynamically create* and *access arrays*.
  - **void sort(originalArray)** sorts the complete array in ascending order.
  - **int binarySearch(elementToBeSearched)** searches for the specified element in the array with the help of Binary Search algorithm.
  - **String toString(originalArray)** returns a String representation of the contents of this Arrays.

# The Arrays class

Example:

```
int[] numbers = { 2, -5, 8, -4, 7 };  
  
Arrays.sort(numbers);  
int index = Arrays.binarySearch(numbers, 7);  
String elements = Arrays.toString(numbers);  
  
System.out.println("Numbers = " + elements);  
System.out.println("Position of element with value 7 is " + index);
```

```
Numbers = [-5, -4, 2, 7, 8]  
Element with value 7 was found in 3-th position
```



# JUnit Framework

- JUnit is a unit testing framework for the Java programming language.
- JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit that originated with SUnit.

## Testing Problems

- Programmers should write tests
- As you probably know programmers always busy, and they have no time to write tests
- They need some tool that can help them
- Main requirements for this tool:
  - A few lines of code then test should run
  - To write test that won't run, then write the code that will make run

# JUnit plugin for Eclipse IDE



Mr. Erich Gamma who is a one of developers of JUnit framework also known as a Eclipse IDE developer

JUnit well integrated into Eclipse IDE

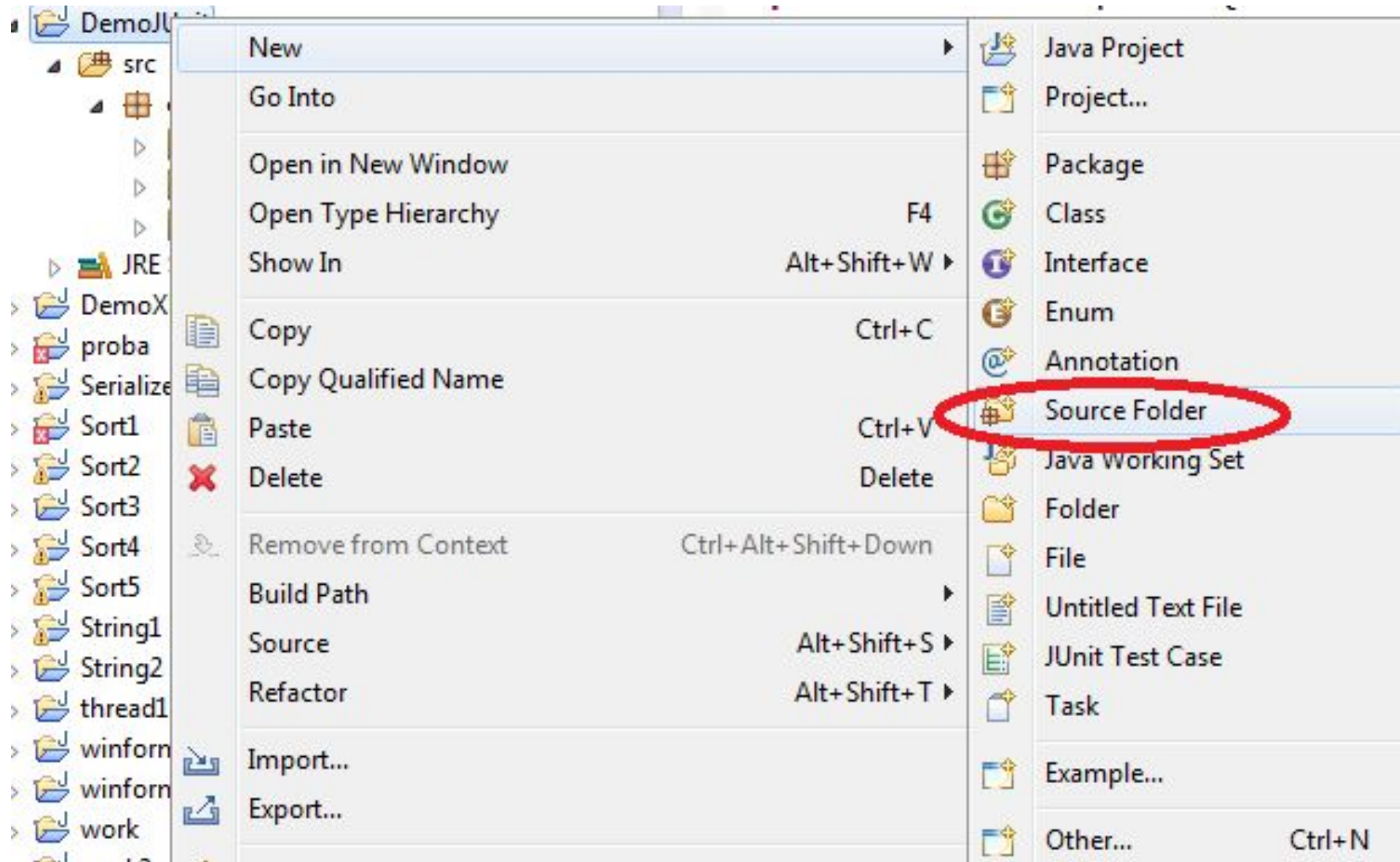


**softserve**

# JUnit assertion methods

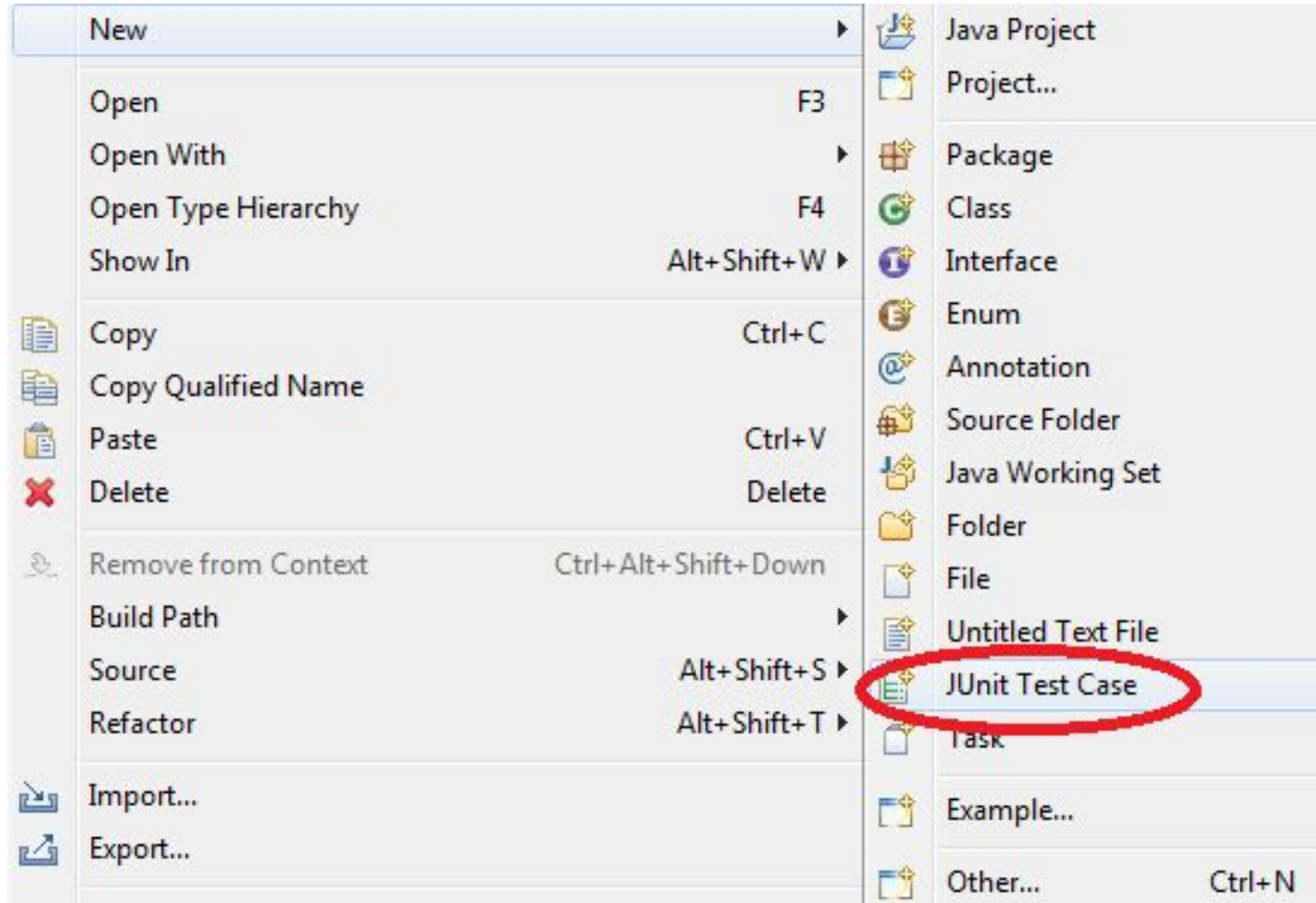
Method	Description
<code>assertEquals()</code>	See if two objects or primitives have the same value
<code>assertNotEquals()</code>	
<code>assertSame()</code>	See if two objects are the same object
<code>assertNotSame()</code>	
<code>assertTrue()</code>	Test a Boolean expression
<code>assertFalse()</code>	
<code>assertNull()</code>	Test for a null object
<code>assertNotNull()</code>	

# JUnit



softserve

# JUnit



# Class Calc

```
public class Calc {  
    public int add(int a, int b) {  
        return a + b;  
    }  
  
    public int div(int a, int b) {  
        return a / b;  
    }  
}
```

# Class CalcTest

```
import static org.junit.Assert.*;
import org.junit.Test;
public class CalcTest {
    Calc calc = new Calc();

    @Test
    public void testAdd() { assertTrue(calc.add(1, 5) == 6);}

    @Test
    public void testDivPositive() {
        int actual = 4;
        int expected = calc.div(9, 2);
        assertEquals(actual, expected);
    }

    @Test(expected = Exception.class)
    public void testDivZero() {
        int actual = calc.div(23, 0);
    }
}
```

# Practical tasks

1. Create an array of ten integers. Display
  - the biggest of these numbers;
  - the sum of positive numbers in the array;
  - the amount of negative numbers in the array.

What values there are more: negative or positive?

2. Create a class *Employee* with fields *name*, *department number*, *salary*. Create five objects of class *Employee*. Display
  - all employees of a certain department (enter department number in the console);
  - arrange workers by the field *salary* in descending order.



# HomeWork (online course)

- UDEMY course "Java Tutorial for Complete Beginners":  
<https://www.udemy.com/java-tutorial/>
- Complete lessons 8, 9, 12, 14 - 16:

▶ 8. While Loops
<a href="#">Learn Java Tutorial for Beginners (Video), Part 4: While Loops</a>
▶ 9. For Loops
<a href="#">Learn Java Tutorial for Beginners (Video), Part 5: For Loops   Cave of Programming</a>
▶ 10. "If"
<a href="#">Learn Java Tutorial for Beginners (Video), Part 6: If   Cave of Programming</a>
▶ 11. Getting User Input
<a href="#">Learn Java Tutorial for Beginners (Video), Part 7: Getting User Input</a>
▶ 12. Do ... While
<a href="#">Learn Java Tutorial for Beginners (Video), Part 8: Do ... While</a>
▶ 13. Switch
<a href="#">Learn Java Tutorial for Beginners (Video), Part 9: Switch</a>
▶ 14. Arrays
<a href="#">Learn Java Tutorial for Beginners (Video), Part 10: Arrays</a>
▶ 15. Arrays of Strings
<a href="#">Learn Java Tutorial for Beginners (Video), Part 11: Arrays of Strings</a>
▶ 16. Multi-Dimensional Arrays
<a href="#">Learn Java Tutorial for Beginners (Video), Part 12: Multi-dimensional Arrays</a>

# Homework

1. Ask user to enter the number of month. Read the value and write the amount of days in this month (create array with amount days of each month).
2. Enter 10 integer numbers. Calculate the sum of first 5 elements if they are positive or product of last 5 element in the other case.
3. Enter 5 integer numbers. Find
  - position of second positive number;
  - minimum and its position in the array.

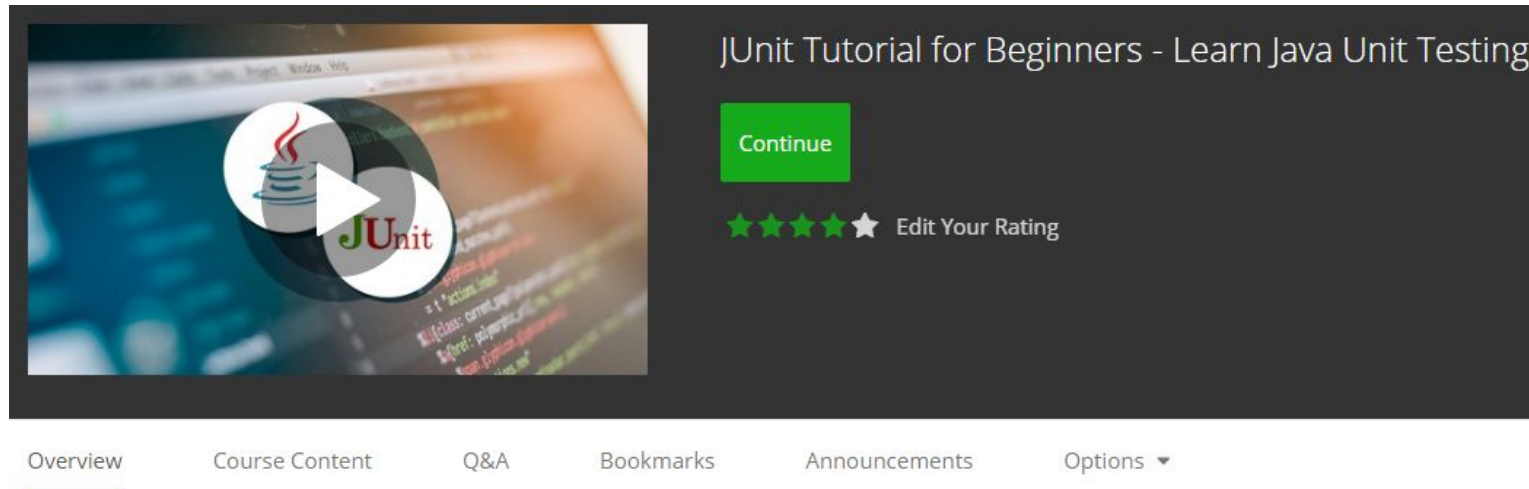
Organize entering integers until the first negative number. Count the product of all entered even numbers.

4. Create class *Car* with fields type, year of production and engine capacity. Create and initialize four instances of class *Car*. Display cars
  - certain model year (enter year in the console);
  - ordered by the field year.
- 5\*. Add Unit Tests to each task, publish code on GitHub

# Unit Testing with JUnit

- Short step-by-step online course:

<https://www.udemy.com/junit-tutorial-for-beginners-with-java-examples/learn/v4/overview>



The screenshot shows a Udemy course interface. On the left is a video player with a play button and a 'JUnit' logo. The course title is 'JUnit Tutorial for Beginners - Learn Java Unit Testing'. Below the title is a green 'Continue' button and a rating of 4.5 stars with the text 'Edit Your Rating'. At the bottom, there is a navigation menu with the following items: Overview (underlined), Course Content, Q&A, Bookmarks, Announcements, and Options with a dropdown arrow.

# THANKS

softserve