Laboratory work #3 | Arrays

Lab objectives

As a result of this laboratory work you will know how to work with arrays in Java:

- declare and initialize arrays;
- iterate through arrays;
- manipulate with arrays' elements.

Overview

Arrays in programming are one of the fundamental concepts. It is the key feature of many programming languages to create a set of data. There are one and multi-dimensional arrays could be declared in Java and other programming languages. In this task, you will get experience with one and two-dimensional arrays.

Part 1

Write a program, containing three parts:

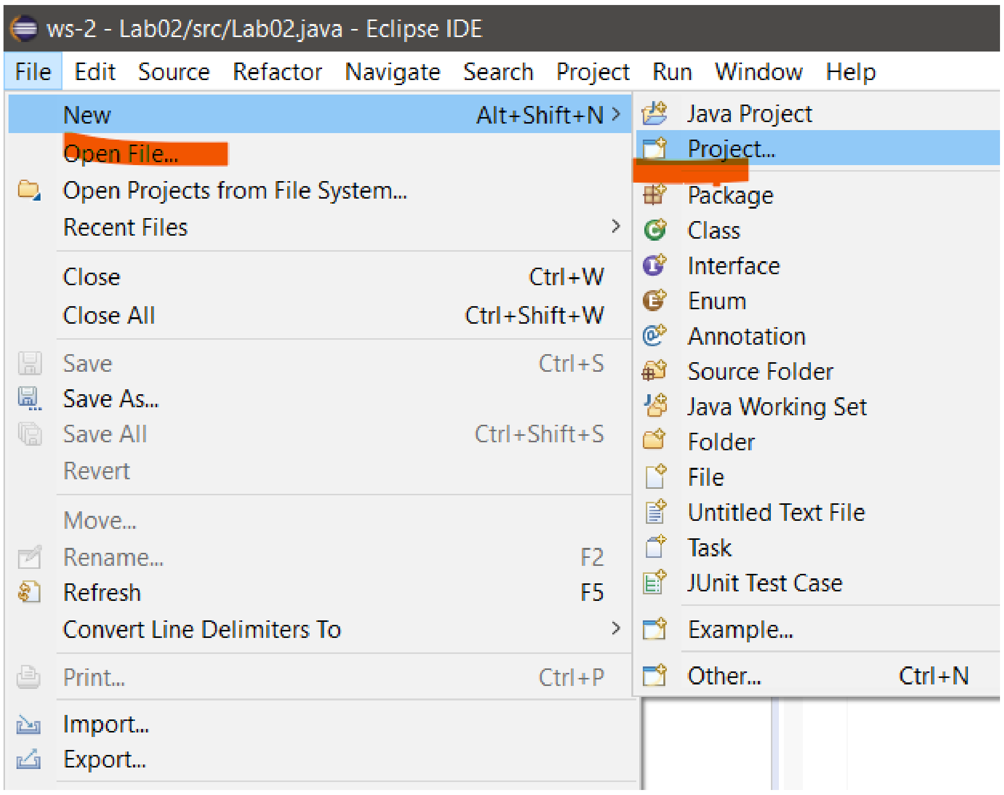
1. Reading of custom number of elements from the standard input into the array of rank 1.
2. Performing given computation with values of array (consider handling border cases)
3. Printing the results of the computation.

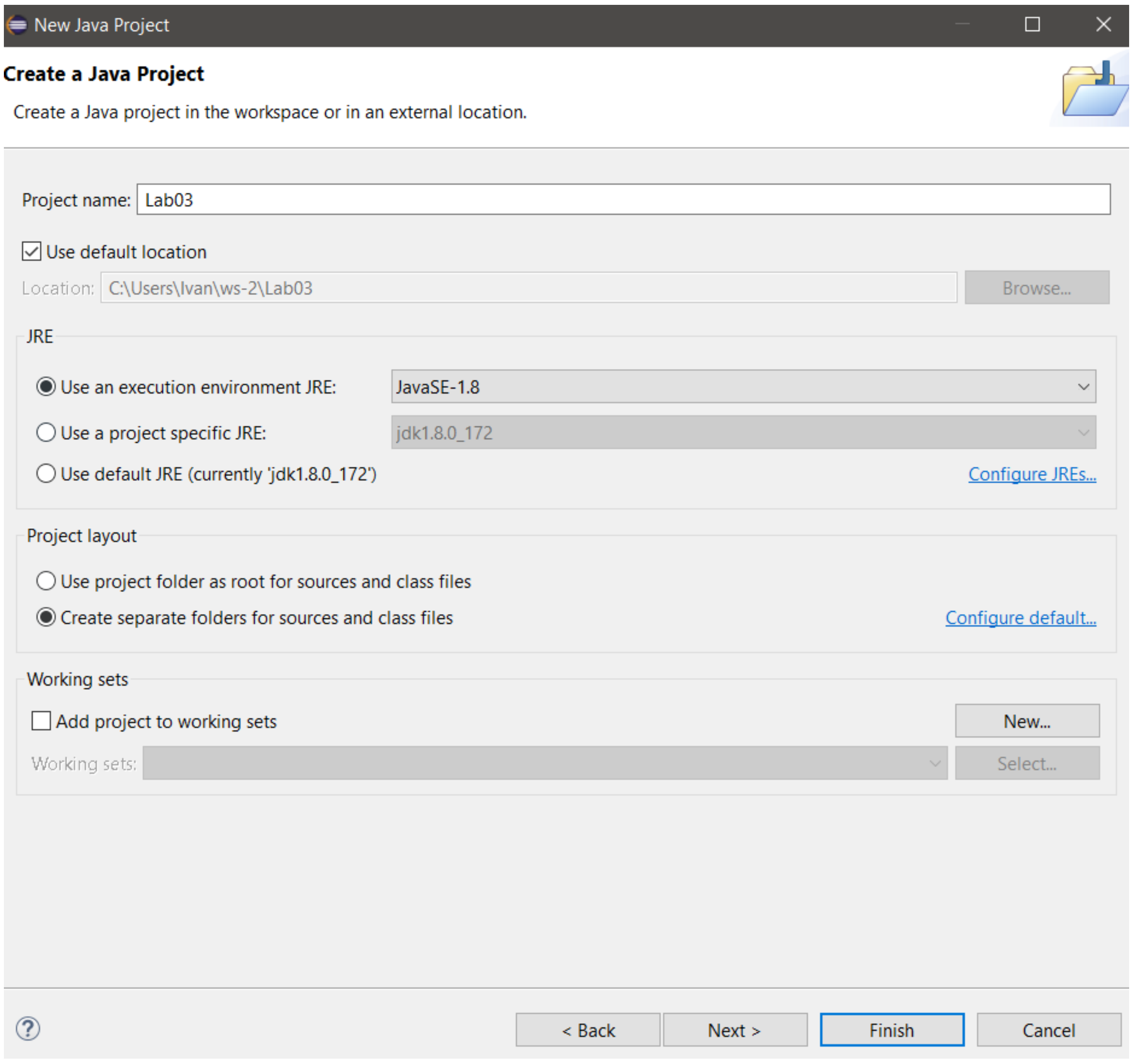
Instructions for the part 1

1. Create a new project as you have done it before.
   1.1. Add a new class, with the method main.
2. In the method main code that does the job to create and fill the array:
   2.1. Prints the message to say to the user input the size of the array, like it shown in example #1.
   2.2. Reads the integer value (size of the array) into variable "size", like it is shown in example #1.
   2.3. Creates variable "array", its type depends on you variant, initialize it with size from the previous step, see an example #2.
   2.4. Write the for-loop to fill the array with random, like it is shown in example #3.
3. Implement the task from your variant.
   3.1. At the end of this document, you can find some examples that can help you. This example shows some basic operations with arrays.

Detailed instructions and example

1. Create a new project as you have done it before.




New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: Lab03

☑ Use default location

Location: C:\Users\Ivan\ws-2\Lab03 Browse...

JRE

● Use an execution environment JRE: JavaSE-1.8

○ Use a project specific JRE: jdk1.8.0_172

○ Use default JRE (currently 'jdk1.8.0_172') Configure JREs...

Project layout

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files Configure default...

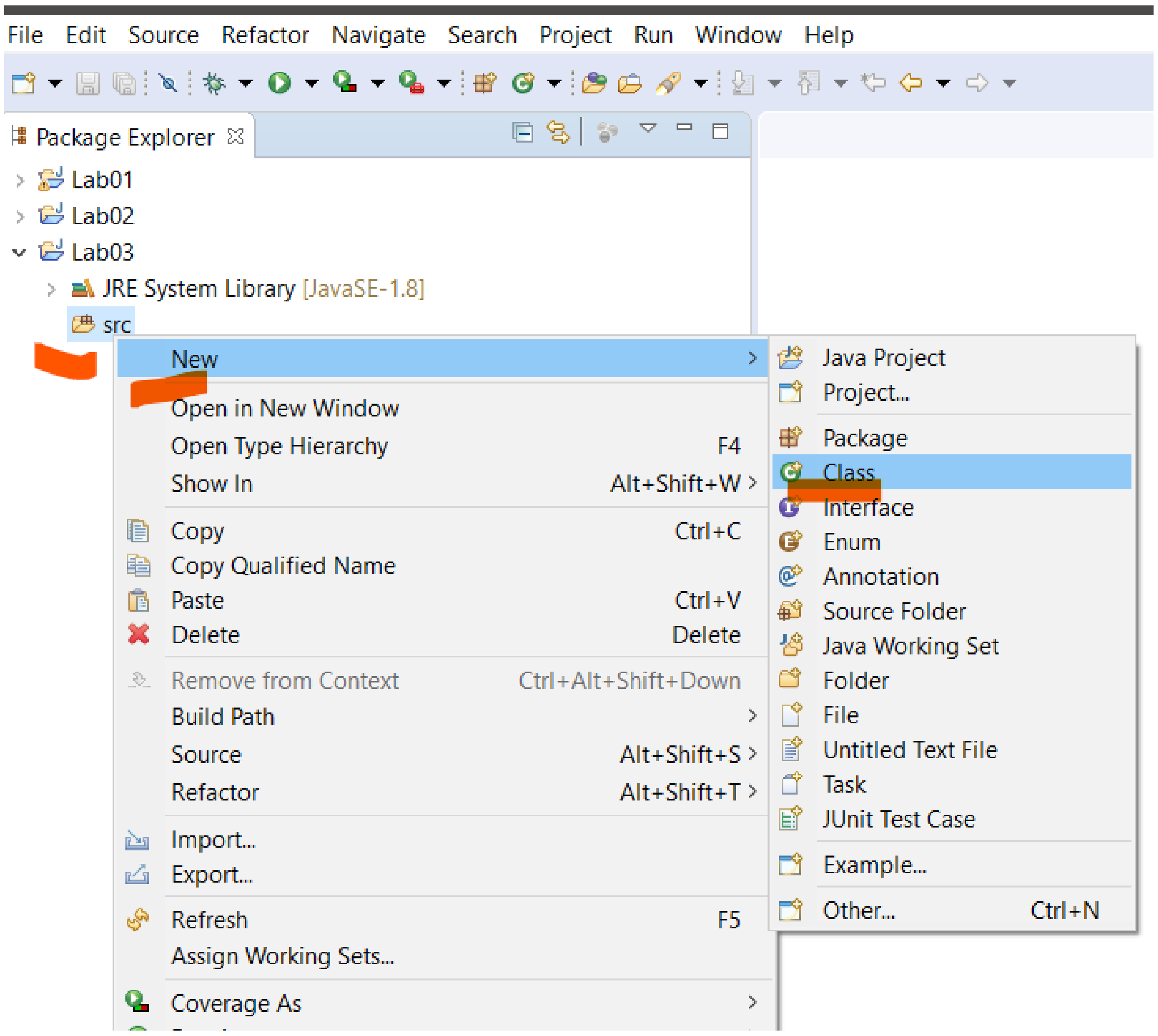
Working sets

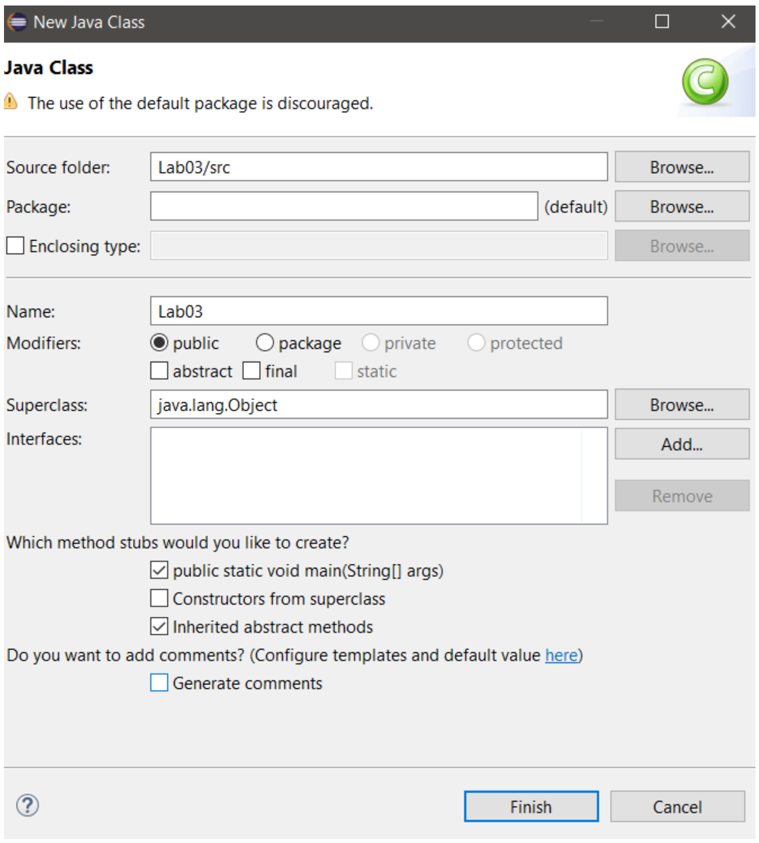
☐ Add project to working sets New...

Working sets: Select...

< Back Next > Finish Cancel

1.1. Add a new class with the method main:




New Java Class

Java Class

The use of the default package is discouraged.

Source folder: Lab03/src

Package: (default)

☐ Enclosing type:

Name: Lab03

Modifiers: ● public ○ package ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☑ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

2. Import the class Scanner to read values from the console (terminal). To do that, write this line your file with code (before the class declaration):

import java.util.Scanner;

2.1. Create the method that will read the size of an array. The best solution to implement it as the separate method, because you will do this work (reading the size) no less than 2 times. In the example implementation section, you can find solution without this method, and it doesn't look nice.

```
public static int readSize(String message) {
      //output the greeting to make clear to the user what's happening
      System.out.print(message);

      Scanner scanner = new Scanner(System.in);
      int size = scanner.nextInt();

      return size;
}
```

2.2. Also, we want to fill this array somehow. We can fill our array by hands (reading the values from the console (terminal)). But in some cases, it could be not easy: what if the size of array, for example, 50? 100? 1000 items? Here are two methods that fill array using the keyboard and the random numbers generator. You can choose the any way to fill an array, or both.

2.2.1.To fill an array "by hands" all what we need is the loop and read values from the console (terminal) in this loop:

```
public static void fillArrayByUser(int[] array) {
      Scanner scanner = new Scanner(System.in);

      for (int i = 0; i < array.length; i++) {
            System.out.print("Enter the " + (i + 1)+" element of "+array.length);
            array[i] = scanner.nextInt();
      }
}
```

2.2.2.To fill an array by the class named Random (that presents the functionality of a random number generator) we should write the second line in our file this code:

import java.util.Random;

2.2.3.After that we can declare the method that implements required functionality.

```
public static void fillArrayByRandom(int[] array) {
      Random random = new Random();
      for (int i = 0; i < array.length; i++) {
            array[i] = random.nextInt(20);
      }
}
```

2.2.4.Now we can create array with a custom (specified by the user) size and fill it by the one of two ways. In this example we will use **fillArrayByRandom** method.

2.2.5.After the filling of an array we want to look at the elements. To do that, we will create a simple method:

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

2.2.6.As the pre-summary we have this code:

```java
import java.util.Random;
import java.util.Scanner;

public class Lab03 {

	public static int readSize(String message) {
		//output the greeting to make clear to the user what's happening
		System.out.print(message);

		Scanner scanner = new Scanner(System.in);
		int size = scanner.nextInt();

		return size;
	}

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

	public static void fillArrayByRandom(int[] array) {
		Random random = new Random();
		for (int i = 0; i < array.length; i++) {
			array[i] = random.nextInt(20);
		}
	}

	public static void fillArrayByUser(int[] array) {
		Scanner scanner = new Scanner(System.in);

		for (int i = 0; i < array.length; i++) {
			System.out.print("Enter the "+(i+1)+" element of "+array.length);
			array[i] = scanner.nextInt();
		}
	}

	public static void main(String[] args) {
		//read the size of the vector (linear array)
		int size = readSize("Count of elements: ");

		int[] array = new int[size];
		fillArrayByRandom(array);

		printArray(array);

		//here we will implement the task.
	}
}
```

2.2.7.Now we can think about the task, and how to implement the algorithm of its solution.

3. In our case we have this task:

For an array of n integer numbers compute:

- Product of positive elements of the array.
- Sum of the negative elements, located before the minimal element.

Reverse the order of the elements.

Here is implementation (on the next page).

```
import java.util.Random;
import java.util.Scanner;

public class Lab03 {

	public static int readSize(String message) {
		//output the greeting to make clear to the user what's happening
		System.out.print(message);

		Scanner scanner = new Scanner(System.in);
		int size = scanner.nextInt();

		return size;
	}

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

	public static void fillArrayByRandom(int[] array) {
		Random random = new Random();
		for (int i = 0; i < array.length; i++) {
			array[i] = random.nextInt(20);
		}
	}

	public static void fillArrayByUser(int[] array) {
		Scanner scanner = new Scanner(System.in);

		for (int i = 0; i < array.length; i++) {
			System.out.print("Enter the " + (i + 1) + " element of " +
array.length);
			array[i] = scanner.nextInt();
		}
	}

	public static void main(String[] args) {
		//read the size of the vector (linear array)
		int size = readSize("Count of elements: ");

		int[] array = new int[size];

		fillArrayByRandom(array);

		System.out.println("Array:");
		printArray(array);
		System.out.println();

		//we need to get the product of elements
		//this variable for save this value.
		//we init it by 1, because we must multiply values and accumulate the
product.
		int product = 1;
		for (int i = 0; i < array.length; i++) {
			if (array[i] > 0) {
				product = product * array[i];
			}
		}

		System.out.println("The product of the positive elements is: " + product);

		//we need to find the index of min value in the array
		//to do that we will save value for the "current minimal value"
		//(current means that for the current i value)
		int min = array[0];
		//and minIndex - because we must find the sum before the min element
		//it is the way to store it's position
		int minIndex = 0;
		for (int i = 0; i < array.length; i++) {
			if (array[i] < min) {
				min = array[i];
				minIndex = i;
			}
		}
		//sum of elements before the min value
		int sum = 0;
		for (int i = 0; i < minIndex; i++) {
			sum += array[i];
		}

		//print the result
		System.out.println("The sum before the minimal elements is " + sum);

		for (int i = 0; i < array.length / 2; i++) {
			int tmp = array[i];
			array[i] = array[array.length - 1 - i];
			array[array.length - 1 - i] = tmp;
		}

		System.out.println("Reversed array:");

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

Variants for the part 1

#	Description
1	For an array of n floating-point numbers compute: • Sum of the elements • Product of elements resided between maximal and minimal elements Sort elements ascending.
2	For an array of n floating point-numbers compute: • Sum of positive elements • Product of elements resided between maximal by modulo and minimal by modulo Sort elements descending.
3	For an array of n integer numbers compute: • Product of elements with even indices • Sum of elements resided between first and last non-zero elements Transform an array so that all elements that are greater than zero would lay in the lowest indices, all elements that are less than zero would lay in the greatest indices.
4	For an array of n floating-point numbers compute: • Sum of elements with odd indices • Sum of elements resided between the first and the last negative elements Transform array so that there would be no elements greater than 1. Other elements should reside in the lowest indices, a corresponding number of unused elements with greatest indices should be filled with zeros.
5	For an array of n integer numbers compute: • Index of the maximum element • Product of elements resided before the first and the second zero elements Transform array so that all elements with odd indices would be in the first half of the array, and all element with even indices would be in the second half of the array.
6	For an array of n floating point numbers compute: • Value of maximal by modulo element • Sum of elements resided between the first and the second positive elements Transform array so that all zero elements would have greatest indices.
7	For an array of n integer numbers compute: • Value of minimal by modulo element • Sum of elements resided after the first zero element Transform array so that all elements with even indices would be in the first half of the array, and all element with odd indices would be in the second half of the array.
8	For an array of n floating point numbers compute: • Count zero elements • Sum of elements resided after minimum element Sort the array by ascending modulo of the elements.
9	For an array of floating-point numbers compute: • Count negative elements • Sum of elements resided after the first minimal by modulo element Replace all negative elements with their values multiplied by itself, then sort all elements ascending.
10	For an array of floating-point numbers compute: • Product of negative elements • Sum of elements resided before maximum element Reverse order of elements.

Part 2

Instructions for the part 2

1. Create a new project as you have done it before.
   1.1. Add a new class, with the method main.
2. In the method main code that does the job to create and fill the array:
   2.1. Prints the message to say to the user input the size of the array, like it shown in example #1.
   2.2. Reads two integer values (count of rows and columns) into variables, like it is shown in example #1.
   2.3. Creates variable “array”, its type depends on you variant, initialize it with size from the previous step, see an example #2.
   2.4. Write the nested for-loops to fill the array with random, like it is shown in example #3.
3. Implement the task from your variant.
   3.1. At the end of this document, you can find some examples that can help you. This example shows some basic operations with arrays.

Variants for the part 2

#	Description
1	For an array of n*m element of integer numbers compute: • Count rows containing no zero elements • The greatest of numbers, presented more than once
2	For an array of n*m element of integer numbers compute: • Count rows containing at least one zero element • Index of row having the longest sequence of identical elements
3	For an array of n*m element of integer numbers compute: • Product of elements for rows having no negative elements • Maximum sum of elements in diagonals (total count of diagonals is 2*(n+(m-1)))
4	For an array of n*m element of integer numbers compute: • Sum of elements for rows having no negative elements • Minimum sum of elements in diagonals (total count of diagonals is 2*(n+(m-1)))
5	For an array of n*n element of integer numbers compute: • Indices of rows having the same elements as column with the same index • Sum of elements for rows having at least one negative elements
6	For an array of n*m element of integer numbers compute: • Remove rows and columns having only zero elements • Find index of the first row having at least one positive element
7	For an array of n*m element of integer numbers compute: • Sort rows ascending by the count of identical elements • Find index of the first column having no negative elements
8	For an array of n*m element of integer numbers compute: • Count rows having at least one zero element • Find number of columns having the longest sequence of identical elements
9	For an array of n*m element of integer numbers compute: • Sum for rows having no negative elements • Minimum sum of elements in diagonals (total count of diagonals is 2*(n+(m-1)))
10	For an array of n*m element of integer numbers compute: • Sort rows descending by the count of identical elements • Count negative elements for rows having at least one zero element

Examples for the part 2

Case	**Example**

Read the integer values to get the size of matrix	Scanner scanner = new Scanner(System.in);
	
	System.out.println("Enter the count of rows: ");
	int rows = scanner.nextInt();
	
	System.out.println("Enter the count of columns: ");
	int cols = scanner.nextInt();

Declare and init the matrix:	int[][] matrix = new int[rows][cols];

Fill the matrix with the Random numbers in the range from 0 to 100.	java.util.Random rand = new java.util.Random();
	for (int i = 0; i < rows; i++) {
	    for (int j = 0; j < cols; j++) {
	        matrix[i][j] = rand.nextDouble();
	    }
	}

Print the matrix:	for (int i = 0; i < rows; i++) {
	    for (int j = 0; j < cols; j++) {
	        System.out.print(matrix[i][j] + " ");
	    }
	    System.out.println();
	}

Example implementation

For matrix N * M (rows * columns) of integers count the negative values per row only if this row contains zero element(s).

```
import java.util.Scanner;
public class HelloWorld {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		System.out.println("Enter the count of columns: ");
		int columns = scanner.nextInt();
		System.out.println("Enter the count of rows: ");
		int rows = scanner.nextInt();
		//declare the matrix
		int[][] matrix = new int[rows][columns];

		//fill the matrix by random numbers
		java.util.Random random = new java.util.Random();
		for (int i = 0; i < rows; i++) {
			for (int j = 0; j < columns; j++)
				//we write -4 to get negative. nextInt generate numbers only from 0
to 10.
				matrix[i][j] = random.nextInt(10) - 4;
		}

		//matrix to save the count of negative elements for each row in matrix
		int[] countByRows = new int[rows];

		//for each row in matrix
		for (int i = 0; i < rows; i++) {
			//we save count in one row (line) of negative elements
			int countInLine = 0;
			//flag to check - are we need to save the count of negative elements
			boolean hasZeroElement = false;

			//for each element in row
			for (int j = 0; j < columns; j++) {
				//if element is less than zero, count it
				if (matrix[i][j] < 0) {
					countInLine++;
				}
				//if we have met the zero, it means
				//that we can save the count of negative elements in row
				if (matrix[i][j] == 0) {
					hasZeroElement = true;
				}
			}
			//save the count of negative elements if we have 0 in row
			if (hasZeroElement) {
				countByRows[i] = countInLine;
			}
		}

		//print the matrix
		for (int i = 0; i < rows; i++) {
			for (int j = 0; j < columns; j++)
				System.out.print(matrix[i][j] + " ");
			System.out.println();
		}
		//print the array with the count of negative elements by rows
		for (int i = 0; i < countByRows.length; i++)
			System.out.print(countByRows[i] + " ");
	}
}
```