Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

**Лабораторна робота № 4**

по дисципліні «СТП-1»

|  |  |
| --- | --- |
| Виконав:  студент групи ІТ-51  Давиденко Ігор Володимирович  Дата здачі \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Захищено з балом \_\_\_\_\_\_\_\_\_\_\_ | Перевірено:  ac. Федорчук В. В |

Київ 2017

Завдання

1. assemble your previous labs in one project
2. fix bugs (static methods and other)
3. refactor your code (each sort method in another class, which implement your interface sort and other best practice with OOP)
4. cover all your code by unit test, code coverage should be more then 80%.
5. connect CI for your test (codecov.io and appveyor.com)
6. CI should run your junit test for java 6, java 7, java 8.
7. add widgets for each CI in your readme

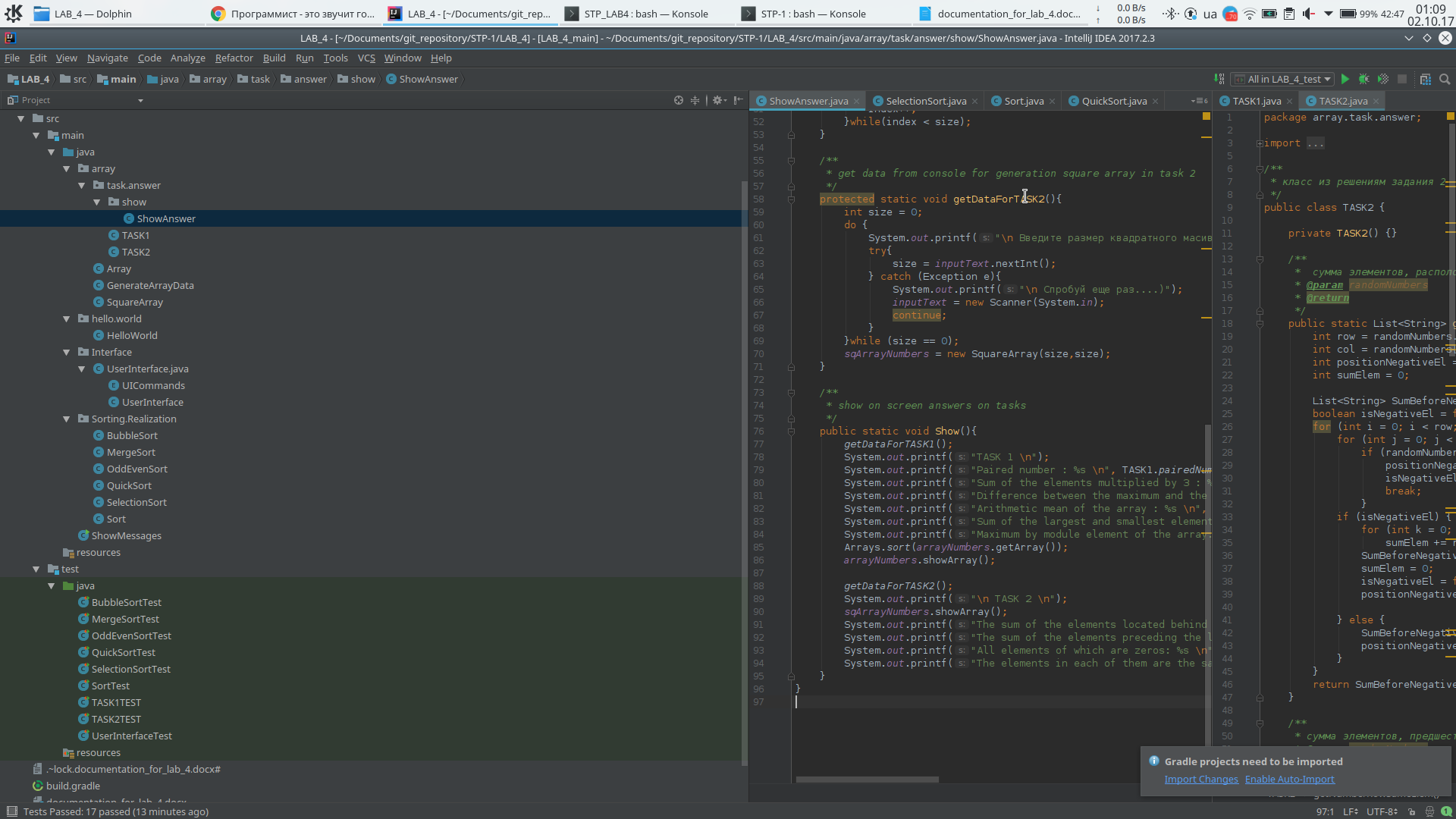
Короткі теоретичні відомості

Unit testing

Модульное тестирование или unit testing - процесс проверки на корректность функционирования отдельных частей исходного кода программы путем запуска тестов в искусственной среде.

Xід роботи

1. Створю проект “LAB\_4”, котрий має наступну структуру:



2. Код програми:

package hello.world;

/\*\*

\* class who show on screen text

\*/

public class HelloWorld {

/\*\*

\* show on screen text hello world

\*/

public static void show (){

System.out.println("Hello World!!!");

}

}

package array.task.answer.show;

import array.Array;

import array.GenerateArrayData;

import array.SquareArray;

import array.task.answer.TASK1;

import array.task.answer.TASK2;

import java.util.Arrays;

import java.util.Scanner;

/\*\*

\* it is class who show task answer

\*/

public class ShowAnswer {

private ShowAnswer (){}

private static Array arrayNumbers = null;

private static SquareArray sqArrayNumbers= null;

private static Scanner inputText = new Scanner(System.in);

/\*\*

\* get data from console for generation array in task 1

\*/

protected static void getDataForTASK1(){

inputText = new Scanner(System.in);

int size = 0;

do {

System.out.printf("\n Введите размер масива: \n");

try{

size = inputText.nextInt();

} catch (Exception e){

System.out.printf("\n Спробуй еще раз....)");

inputText = new Scanner(System.in);

continue;

}

}while (size == 0);

arrayNumbers = new Array(size);

int index = 0;

do {

System.out.printf("\n Введите %s елемент масива: ", index);

try{

arrayNumbers.setElement(index,inputText.nextInt());

} catch (Exception e){

System.out.printf("\n Спробуй еще раз....)");

inputText = new Scanner(System.in);

continue;

}

index++;

}while(index < size);

}

/\*\*

\* get data from console for generation square array in task 2

\*/

protected static void getDataForTASK2(){

int size = 0;

do {

System.out.printf("\n Введите размер квадратного масива: \n");

try{

size = inputText.nextInt();

} catch (Exception e){

System.out.printf("\n Спробуй еще раз....)");

inputText = new Scanner(System.in);

continue;

}

}while (size == 0);

sqArrayNumbers = new SquareArray(size,size);

}

/\*\*

\* show on screen answers on tasks

\*/

public static void Show(){

getDataForTASK1();

System.out.printf("TASK 1 \n");

System.out.printf("Paired number : %s \n", TASK1.pairedNumber(arrayNumbers.getArray()));

System.out.printf("Sum of the elements multiplied by 3 : %s \n", TASK1.SumMultipliedElements(arrayNumbers.getArray()));

System.out.printf("Difference between the maximum and the minimum elements array : %s \n", TASK1.DifferenceMaxMin(arrayNumbers.getArray()));

System.out.printf("Arithmetic mean of the array : %s \n", TASK1.ArithmeticMeanArray(arrayNumbers.getArray()));

System.out.printf("Sum of the largest and smallest elements of the array : %s \n", TASK1.SumLargestAndSmallestElem(arrayNumbers.getArray()));

System.out.printf("Maximum by module element of the array: %s \n", TASK1.maximumModuleElem(arrayNumbers.getArray()));

Arrays.sort(arrayNumbers.getArray());

arrayNumbers.showArray();

getDataForTASK2();

System.out.printf("\n TASK 2 \n");

sqArrayNumbers.showArray();

System.out.printf("The sum of the elements located behind the first negative element in i-th line : %s \n", TASK2.getSumBeforeNegativeEl(sqArrayNumbers.getArray()).toString());

System.out.printf("The sum of the elements preceding the last negative element i-th line : %s \n", TASK2.getSumAfterNegativeEl(sqArrayNumbers.getArray()).toString());

System.out.printf("All elements of which are zeros: %s \n", TASK2.getNumberRowNullElem(sqArrayNumbers.getArray()).toString());

System.out.printf("The elements in each of them are the same: %s \n", TASK2.getNumberRowSameElem(sqArrayNumbers.getArray()).toString());

}

}

package array.task.answer;

import java.util.Arrays;

/\*\*

\* клас из решениям задания 1

\*/

public class TASK1 {

private TASK1(){}

/\*\*

\* метод которий повертает парние числа в масиве

\* @param randomNumbers

\* @return

\*/

public static int pairedNumber(int [] randomNumbers){

if (randomNumbers.length > 0){

int pairedNumbers = 0;

Arrays.sort(randomNumbers);

int arrayLength = randomNumbers.length;

for (int i = 0; i < arrayLength-1; i++)

for(int j = i+1; j < arrayLength; j++ )

if (randomNumbers[i] == randomNumbers[j]){

pairedNumbers++;

break;

}

return pairedNumbers;

}

else

return 0;

}

/\*\*

\* сума елементов умноженых на три

\* @param randomNumbers

\* @return

\*/

public static int SumMultipliedElements(int [] randomNumbers){

if (randomNumbers.length > 0){

int sumMultipliedEl = 0;

int lengthArray = randomNumbers.length;

for (int i = 0; i < lengthArray; i++)

if (randomNumbers[i] % 3 == 0)

sumMultipliedEl += randomNumbers[i];

return sumMultipliedEl;

}

else

return 0;

}

/\*\*

\*

\* разница межу максимальним и минимальним елементом

\* @param randomNumber

\* @return

\*/

public static int DifferenceMaxMin(int[] randomNumber){

if (randomNumber.length > 0){

Arrays.sort(randomNumber);

int maxElemrnt = randomNumber[randomNumber.length-1];

int minElement = randomNumber[0];

return maxElemrnt - minElement;

}

else

return 0;

}

/\*\*

\* середне ариметичне масива

\* @param randomNumbers

\* @return

\*/

public static double ArithmeticMeanArray(int[] randomNumbers){

if (randomNumbers.length > 0){

int sumElements = 0;

int numElements = randomNumbers.length;

for(int i = 0; i < numElements; i++)

sumElements += randomNumbers[i];

return sumElements / numElements;

}

else

return 0;

}

/\*\*

\* сума самых больших и наименьших элементов массива

\* @param randomNumbers

\* @return

\*/

public static int SumLargestAndSmallestElem(int[] randomNumbers){

if (randomNumbers.length > 0){

Arrays.sort(randomNumbers);

int sumMaxElements = 0;

int sumMinElements = 0;

int lengthArray = randomNumbers.length;

int maxElem = randomNumbers[lengthArray-1];

int minElem = randomNumbers[0];

for (int i = 0; i < lengthArray; i++)

if (randomNumbers[i] == minElem)

sumMinElements += randomNumbers[i];

else

break;

for (int j = lengthArray - 1; j > 0; j--)

if(randomNumbers[j] == maxElem)

sumMaxElements += randomNumbers[j];

else

break;

return sumMinElements + sumMaxElements;

}

else

return 0;

}

/\*\*

\* максимальный элемент модуля массива

\* @param randomNumbers

\* @return

\*/

public static int maximumModuleElem(int[] randomNumbers){

if(randomNumbers.length > 0){

int lengthArray = randomNumbers.length;

for (int i = 0; i < lengthArray; i++)

randomNumbers[i] = Math.abs(randomNumbers[i]);

Arrays.sort(randomNumbers);

return randomNumbers[lengthArray - 1];

}

else

return 0;

}

}

package array.task.answer;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* класс из решениям задания 2

\*/

public class TASK2 {

private TASK2() {}

/\*\*

\* сумма элементов, расположенных за первым отрицательным элементом в i-й строке (если все элементы линии неотрицательны, возьмите bi = 100)

\* @param randomNumbers

\* @return

\*/

public static List<String> getSumBeforeNegativeEl(int [][] randomNumbers){

int row = randomNumbers.length;

int col = randomNumbers[0].length;

int positionNegativeEl = 0;

int sumElem = 0;

List<String> SumBeforeNegativeEl = new ArrayList<String>();

boolean isNegativeEl = false;

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++)

if (randomNumbers[i][j] < 0) {

positionNegativeEl = j;

isNegativeEl = true;

break;

}

if (isNegativeEl) {

for (int k = 0; k < positionNegativeEl; k++)

sumElem += randomNumbers[i][k];

SumBeforeNegativeEl.add(sumElem + " ");

sumElem = 0;

isNegativeEl = false;

positionNegativeEl = 0;

} else {

SumBeforeNegativeEl.add(100 + " ");

positionNegativeEl = 0;

}

}

return SumBeforeNegativeEl;

}

/\*\*

\* сумма элементов, предшествующих последнему отрицательному элементу i-й линии (если все элементы линии неотрицательны, то возьмем bi = -l)

\* @param randomNumbers

\* @return

\*/

public static List<String> getSumAfterNegativeEl(int [][] randomNumbers){

int row = randomNumbers.length;

int col = randomNumbers[0].length;

int positionNegativeEl = 0;

int sumElem = 0;

List<String> SumAfterNegativeEl = new ArrayList<String>();

boolean isNegativeEl = false;

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++)

if (randomNumbers[i][j] < 0) {

positionNegativeEl = j;

isNegativeEl = true;

break;

}

if (isNegativeEl) {

for (int k = positionNegativeEl; k < col; k++)

sumElem += randomNumbers[i][k];

SumAfterNegativeEl.add(sumElem + " ");

sumElem = 0;

isNegativeEl = false;

positionNegativeEl = 0;

} else {

SumAfterNegativeEl.add(-1 + " ");

positionNegativeEl = 0;

}

}

return SumAfterNegativeEl;

}

/\*\*

\* все элементы которых являются нулями

\* @param randomNumbers

\* @return

\*/

public static List<String> getNumberRowNullElem(int [][] randomNumbers){

int row = randomNumbers.length;

int col = randomNumbers[0].length;

List<String> zeroRow = new ArrayList<String>();

boolean isNull = false;

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++)

if (randomNumbers[i][j] != 0) {

isNull = false;

break;

} else

isNull = true;

if(isNull){

zeroRow.add(i + "\t");

isNull = false;

}

}

return zeroRow;

}

/\*\*

\* елементи есть одинаковие

\* @param randomNumbers

\* @return

\*/

public static List<String> getNumberRowSameElem(int [][] randomNumbers){

int row = randomNumbers.length;

int col = randomNumbers[0].length;

List<String> sameRow = new ArrayList<String>();

boolean isSame = false;

for (int i = 0; i < row; i++) {

second: for (int j = 0; j < col-1; j++)

for(int k = j+1; k < col; k++)

if (randomNumbers[i][j] != randomNumbers[i][k])

isSame = false;

else{

isSame = true;

break second;

}

if(isSame) {

sameRow.add(i + "\t");

isSame = false;

}

}

return sameRow;

}

}

package array;

import java.util.Arrays;

import java.util.Random;

/\*\*

\* it is class who consist array

\*/

public class Array {

private int arraySize;

private int []arrayElements = null;

private int [] arrayCopy = null;

/\*\*

\*

\* @return size array

\*/

public int getSize() { // get size array

return arraySize;

}

/\*\*

\*

\* @return array

\*/

public int[] getArray() { return arrayElements; }

/\*\*

\* initialize array

\* class constructor

\* @param size array

\*/

public Array(int size){

this.arraySize = size;

arrayElements = new int[size];

generateArray(); // after create array, generate him

}

/\*\*

\* populating an array with random data

\*/

public void generateArray(){ // method whose addition data to our array

int generatedNumber = 0;

arrayElements = GenerateArrayData.generateData(arraySize);

arrayCopy = arrayElements.clone();

}

/\*\*

\* method who return all changes to default values

\*/

public void resetData(){ // repair default data

arrayElements = arrayCopy;

}

/\*\*

\* method who return element array using her index

\* @param position - position element

\* @return retun element using her position

\*/

public int getElement(int position){

return arrayElements[position];

}

/\*\*

\* set data for element in array

\* @param position element

\* @param newData new value array element

\*/

public void setElement(int position, int newData){

arrayElements[position] = newData;

}

/\*\*

\* method who print the value to the console

\*/

public void showArray(){ // method whose showing array data

System.out.printf("Array: %s \n", Arrays.toString(arrayElements));

}

}

package array;

import java.util.Arrays;

import java.util.Random;

/\*\*

\* class who generated data for array

\*/

public class GenerateArrayData {

private GenerateArrayData(){}

private static Random randomNumber = new Random();

/\*\*

\* generate data for array

\* @param sizeArray

\* @return

\*/

public static int[] generateData(int sizeArray)

{

int [] randomNumbers = new int [sizeArray];

// int from = -10;

// int to = 100;

for (int i = 0; i < sizeArray; i++) {

//randomNumbers[i] = from + (int) (Math.random() \* to);

randomNumbers[i] = randomNumber.nextInt();

}

return randomNumbers;

}

/\*\*

\* generate data from square array

\* @param row - rows

\* @param col - column

\* @return

\*/

public static int[][] generateSquareArray(int row, int col){

int from = -10;

int to = 100;

int [][] randomNumbers = new int [row][col];

for (int i = 0; i < row; i++)

for (int j = 0; j < col; j++) {

randomNumbers[i][j] = from + (int) (Math.random() \* to);

//randomNumbers[i][j] = randomNumber.nextInt();

}

return randomNumbers;

}

}

package array;

import java.util.Arrays;

public class SquareArray {

private int arrayNumCol;

private int arrayNumRow;

private int [][] arrayElements = null;

private int [][] arrayCopy = null;

/\*\*

\*

\* @return number colums

\*/

public int getNumberCol() { // get size array

return arrayNumCol;

}

/\*\*

\*

\* @return number rows

\*/

public int getNumberRow() {

return arrayNumRow;

}

/\*\*

\*

\* @return square array

\*/

public int[][] getArray() { return arrayElements; }

/\*\*

\* initialize array

\* class constructor

\* @param number column and rows

\*/

public SquareArray(int row, int col){

this.arrayNumRow = row;

this.arrayNumCol = col;

arrayElements = new int[row][col];

generateArray(); // after create array, generate him

}

/\*\*

\* populating an array with random data

\*/

public void generateArray(){ // method whose addition data to our array

int generatedNumber = 0;

arrayElements = GenerateArrayData.generateSquareArray(this.arrayNumRow, this.arrayNumCol);

arrayCopy = arrayElements.clone();

}

/\*\*

\* method who return all changes to default values

\*/

public void resetData(){ // repair default data

arrayElements = arrayCopy;

}

/\*\*

\* method who return element array using her index

\* @param row,col - position element

\* @return retun element using her position

\*/

public int getElement(int row, int col){

return arrayElements[row][col];

}

/\*\*

\* set data for element in array

\* @param row, col - position element

\* @param newData new value array element

\*/

public void setElement(int row, int col, int newData){

arrayElements[row][col] = newData;

}

/\*\*

\* method who print the value to the console

\*/

public void showArray()

{

int row = this.arrayElements.length;

int col = this.arrayElements[0].length;

for (int i = 0; i < row; i++){

System.out.printf("row "+i+" : ");

for (int j = 0; j < col; j++)

System.out.printf(this.arrayElements[i][j] + " ");

System.out.printf("\n");

}

}

}

import Sorting.Realization.BubbleSort;

import org.junit.Before;

import org.junit.Test;

import java.util.Arrays;

import static org.junit.Assert.assertEquals;

public class BubbleSortTest {

int [] arr = null;

@Before

public void init(){

arr = new int [] {2,1,4,5};

}

@Test

public void SortTest(){

BubbleSort.sort(arr);

assertEquals("[1, 2, 4, 5]", Arrays.toString(arr));

}

}

import Sorting.Realization.BubbleSort;

import Sorting.Realization.MergeSort;

import org.junit.Before;

import org.junit.Test;

import java.util.Arrays;

import static org.junit.Assert.assertEquals;

public class MergeSortTest {

int [] arr = null;

@Before

public void init(){

arr = new int [] {2,1,4,5};

}

@Test

public void SortTest(){

MergeSort.sort(arr);

assertEquals("[1, 2, 4, 5]", Arrays.toString(arr));

}

}

import Sorting.Realization.BubbleSort;

import Sorting.Realization.OddEvenSort;

import org.junit.Before;

import org.junit.Test;

import java.util.Arrays;

import static org.junit.Assert.assertEquals;

public class OddEvenSortTest {

int [] arr = null;

@Before

public void init(){

arr = new int [] {2,1,4,5};

}

@Test

public void SortTest(){

OddEvenSort.sort(arr);

assertEquals("[1, 2, 4, 5]", Arrays.toString(arr));

}

}

import Sorting.Realization.BubbleSort;

import Sorting.Realization.QuickSort;

import org.junit.Before;

import org.junit.Test;

import java.util.Arrays;

import static org.junit.Assert.assertEquals;

public class QuickSortTest {

int [] arr = null;

@Before

public void init(){

arr = new int [] {2,1,4,5};

}

@Test

public void SortTest(){

QuickSort.sort(arr);

assertEquals("[1, 2, 4, 5]", Arrays.toString(arr));

}

}

import Sorting.Realization.BubbleSort;

import Sorting.Realization.SelectionSort;

import org.junit.Before;

import org.junit.Test;

import java.util.Arrays;

import static org.junit.Assert.assertEquals;

public class SelectionSortTest {

int [] arr = null;

@Before

public void init(){

arr = new int [] {2,1,4,5};

}

@Test

public void SortTest(){

SelectionSort.sort(arr);

assertEquals("[1, 2, 4, 5]", Arrays.toString(arr));

}

}

import Sorting.Realization.Sort;

import org.junit.Before;

import org.junit.Test;

import java.util.Arrays;

import static org.junit.Assert.assertEquals;

public class SortTest {

int [] arr = null;

@Before

public void init(){

arr = new int [] {2,1,4,5};

}

@Test

public void SortTest(){

Sort.insertion(arr);

assertEquals("[1, 2, 4, 5]", Arrays.toString(arr));

}

}

import array.task.answer.TASK1;

import array.task.answer.TASK2;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class TASK1TEST {

int Paired = 0;

int sumMultEl = 0;

int DifMinMax = 0;

double ArifMeanAr = 0.0;

int SumMaxMin = 0;

int MaxModEl = 0;

@Before

public void init(){

int [] array = new int [] { 1, 2, 3, 4, 4 };

this.Paired = TASK1.pairedNumber(array);

this.sumMultEl = TASK1.SumMultipliedElements(array);

this.DifMinMax = TASK1.DifferenceMaxMin(array);

this.ArifMeanAr = TASK1.ArithmeticMeanArray(array);

this.SumMaxMin = TASK1.SumLargestAndSmallestElem(array);

this.MaxModEl = TASK1.maximumModuleElem(array);

}

@Test

public void pairedNumbersTest(){

assertEquals(1, this.Paired);

}

@Test

public void SumMultipliedElTEST(){

assertEquals(3, this.sumMultEl);

}

@Test

public void DifferentMaxMinTEST(){

assertEquals(3, this.DifMinMax);

}

@Test

public void ArithmeticMeanArrayTEST(){

assertEquals( 2.0, this.ArifMeanAr, 0.0001);

}

@Test

public void SumLargestAndSmallestElTEST(){

assertEquals(9, this.SumMaxMin);

}

@Test

public void maximumModuleElTEST(){

assertEquals(4, this.MaxModEl);

}

}

import array.task.answer.TASK2;

import org.junit.Before;

import org.junit.Test;

import java.util.List;

import static org.junit.Assert.assertEquals;

public class TASK2TEST {

private int [][] array = new int [][]{ { 1,2 },{ 2, 3 }, {2 , -4 } };

String sumBeforeNeg = new String();

String sumAfterNeg = new String();

String numRowNullEl = new String();

String numRowSameElem = new String();

@Before

public void init(){

sumBeforeNeg = TASK2.getSumBeforeNegativeEl(array).toString();

sumAfterNeg = TASK2.getSumAfterNegativeEl(array).toString();

numRowNullEl = TASK2.getNumberRowNullElem(array).toString();

numRowSameElem = TASK2.getNumberRowSameElem(array).toString();

}

@Test

public void getSumBeforeNegElTest(){

assertEquals("[100 , 100 , 2 ]", sumBeforeNeg);

}

@Test

public void getSumAfterNegElTest(){

assertEquals("[-1 , -1 , -4 ]", sumAfterNeg);

}

@Test

public void NumberRowsWithNullElTest(){

assertEquals("[]", numRowNullEl);

}

@Test

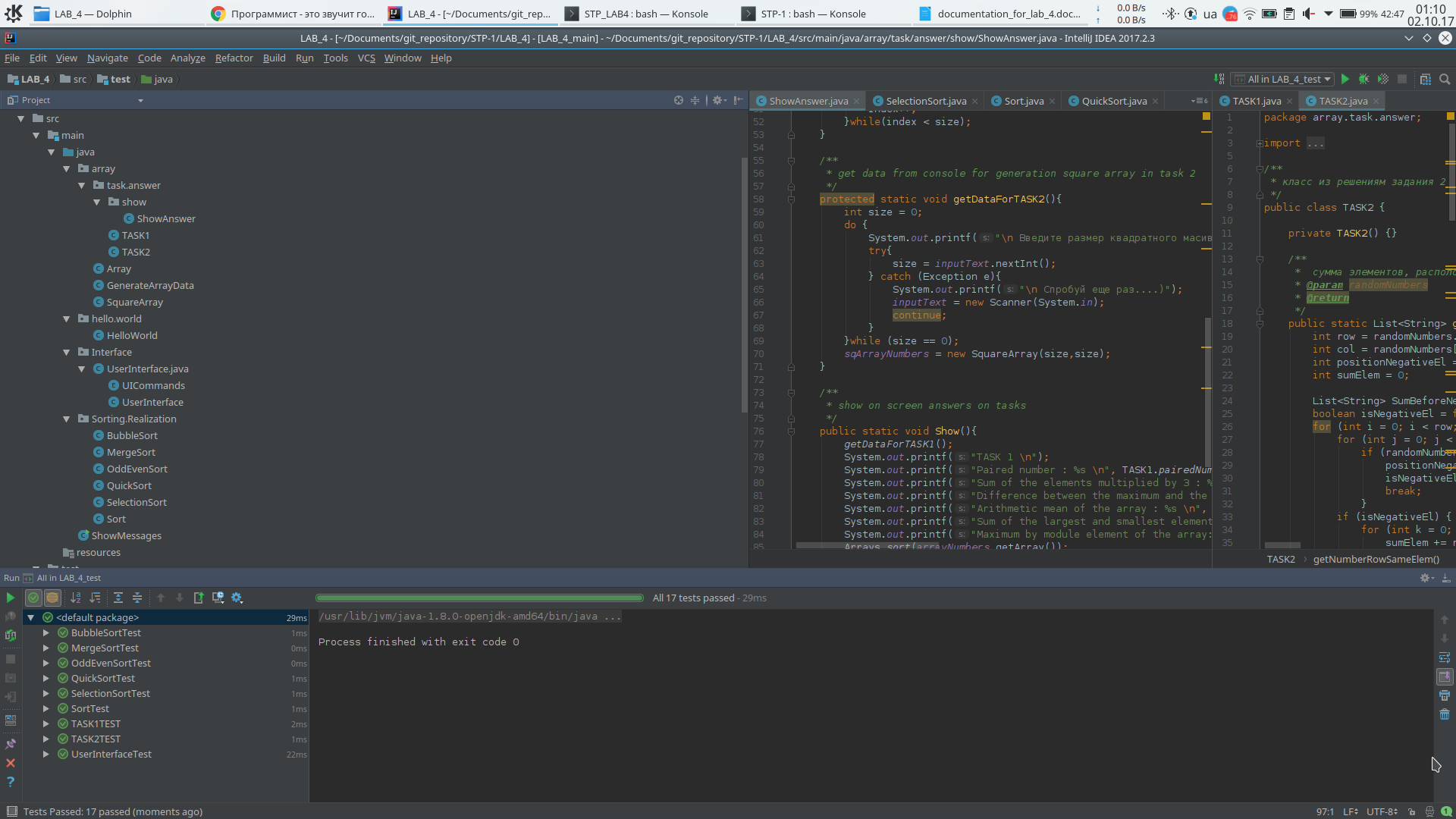
public void NumberRowsSameElTest(){

assertEquals("[]", numRowSameElem);

}

}

3. В результаті отримаю наступне:



Висновок: Під час лабораторної роботи я навчився створбвати unit тести для своєї програми і також навчився покриття коду тестами.