

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ  
 НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного аналізу  
Кафедра системного проектування**

**Звіт**

**про виконання лабораторної роботи №2  
з дисципліни «Програмування на мові Java»**

Виконав:  
студент III курсу, групи ДА-22  
Жадько Микита Сергійович

Київ – 2024

ХІД РОБОТИ

1. **№9. Сформувати масив b з вихідного одновимірного масиву a за наступним алгоритмом: спочатку йдуть елементи масиву a з парними значеннями в порядку їх зростання, потім елементи з непарними значеннями в порядку їх зменшення. Для визначення кількості парних елементів використовуйте оператор взяття модуля "%".**

Лістинг коду програми:

public class Task9 {

public int[] formArrayB(int[] a) {

int[] b = new int[a.length];

int evenIndex = 0;

int oddIndex = a.length - 1;

for (int value : a) {

if (value % 2 == 0) {

insertSorted(b, evenIndex++, value, true);

} else {

insertSorted(b, oddIndex--, value, false);

}

}

return b;

}

private void insertSorted(int[] b, int index, int value, boolean isEven) {

if (isEven) {

int i = index - 1;

while (i >= 0 && b[i] > value) {

b[i + 1] = b[i];

i--;

}

b[i + 1] = value;

} else {

int i = index + 1;

while (i < b.length && b[i] > value) {

b[i - 1] = b[i];

i++;

}

b[i - 1] = value;

}

}

}

Лістинг коду тесту:

package org.example;

import java.util.Arrays;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

class Task9Test {

private Task9 task9 = new Task9();

private int[] \_testArray;

@BeforeEach

void setUp() {

\_testArray = new int[] {-1, 2, 3, -4, 0, 6, 7, 8, 9, 10};

}

@Test

void formArrayB() {

int[] result = task9.formArrayB(\_testArray);

int[] expected = new int[] {-4, 0, 2, 6, 8, 10, 9, 7, 3, -1};

assertTrue(Arrays.equals(expected, result));

}

}

Результати виконання тесту:

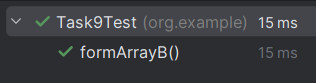


Рисунок 1 — Результат виконання тесту програми №9.

1. **№14. Перевірте, чи є елементи масиву a множиною (для цього серед елементів масиву не повинно бути двох елементів з однаковим значенням).**

Лістинг коду програми:

package org.example;

import java.util.HashSet;

import java.util.Set;

public class Task14 {

public boolean isSet(int[] a) {

Set<Integer> set = new HashSet<>();

for (int value : a) {

if (set.contains(value)) {

return false;

}

set.add(value);

}

return true;

}

}

Лістинг коду тесту:

package org.example;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

class Task14Test {

private Task14 task14 = new Task14();

private int[] \_testArray;

@BeforeEach

void setUp() {

\_testArray = new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 12, 13, 14, 15, 16};

}

@Test

void isSet() {

boolean result = task14.isSet(\_testArray);

boolean expected = false;

assertEquals(expected, result);

}

}

Результати виконання тесту:

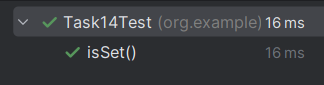


Рисунок 2 — Результат виконання тесту програми №14.

1. **№19. Виведіть на дисплей значення і індекси тільки тих елементів масиву a, значення яких не дорівнюють значенням інших елементів, тобто унікальних елементів масиву.**

Лістинг коду програми:

package org.example;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

public class Task19 {

public int[] getUniqueElements(int[] a) {

Set<Integer> allElements = new HashSet<>();

Set<Integer> duplicates = new HashSet<>();

List<Integer> list = new ArrayList<>();

for (int value : a) {

if (!allElements.add(value)) {

duplicates.add(value);

}

}

for (int number : a) {

if (!duplicates.contains(number)) {

list.add(number);

}

}

int[] uniqueArray = new int[list.size()];

for (int i = 0; i < list.size(); i++) {

uniqueArray[i] = list.get(i);

}

return uniqueArray;

}

}

Лістинг коду тесту:

package org.example;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

import java.util.Arrays;

class Task19Test {

private Task19 task19 = new Task19();

private int[] \_testArray;

@BeforeEach

void setUp() {

\_testArray = new int[]{1, 12, 13, 14, 15, 6, 7, 8, 9, 10, 10, 12, 13, 14, 15, 16};

}

@Test

void getUniqueElements() {

int[] result = task19.getUniqueElements(\_testArray);

int[] expected = {1, 6, 7, 8, 9, 16};

assertTrue(Arrays.equals(expected, result));

}

}

Результати виконання тесту:

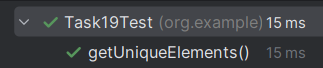


Рисунок 3 — Результат виконання тесту програми №19.

1. **№24. Визначте номер дня в році по заданому номеру дня в місяці і номеру місяця (вводяться як аргументи при виклику програми). Ознака, чи є рік високосним, задається як булевська змінна. Вказівка: кількість днів до початку цього місяця (НЕ високосний рік): январь 0, лютий - 31, березень - 59, квітень - 90, травень -120, червень - 151, липень - 181, серпень - 212, вересень - 243, жовтень - 273, листопад - 314, грудень - 334 задати у 26 вигляді масиву. У високосному році, починаючи з березня, до кількості днів додається 1.**

Лістинг коду програми:

package org.example;

public class Task24 {

public int getDayOfYear(int day, int month, boolean isLeapYear) {

int[] daysBeforeMonth = {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334};

int dayNumber = daysBeforeMonth[month - 1] + day;

if (isLeapYear && month > 2) {

dayNumber++;

}

return dayNumber;

}

}

Лістинг коду тесту:

package org.example;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

class Task24Test {

private Task24 task24 = new Task24();

private int \_testDay;

private int \_testMonth;

private boolean \_testIsLeapYear;

@BeforeEach

void setUp() {

\_testDay = 15;

\_testMonth = 3;

\_testIsLeapYear = true;

}

@Test

void getDayOfYear() {

int result = task24.getDayOfYear(\_testDay, \_testMonth, \_testIsLeapYear);

int expected = 75;

assertEquals(expected, result);

}

}

Результати виконання тесту:

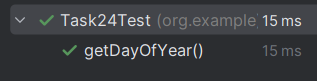


Рисунок 4 — Результат виконання тесту програми №24.

1. **№29. Сформувати масив b з масиву a наступним чином: масив b складається з тих елементів масиву a, які повторюються в масиві (по одному значенню для однакових елементів), наприклад, для масиву a: 3 7 4 3 8 7 5, масив b матиме вигляд: 3 7.**

Лістинг коду програми:

package org.example;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

public class Task29 {

public int[] getNonUniqueElements(int[] a) {

Set<Integer> allElements = new HashSet<>();

Set<Integer> duplicates = new HashSet<>();

List<Integer> list = new ArrayList<>();

for (int value : a) {

if (!allElements.add(value)) {

duplicates.add(value);

}

}

for (int number : duplicates) {

list.add(number);

}

int[] uniqueArray = new int[list.size()];

for (int i = 0; i < list.size(); i++) {

uniqueArray[i] = list.get(i);

}

return uniqueArray;

}

}

Лістинг коду тесту:

package org.example;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

import java.util.Arrays;

class Task29Test {

private Task29 task29 = new Task29();

private int[] \_testArray;

@BeforeEach

void setUp() {

\_testArray = new int[]{1, 12, 13, 14, 15, 6, 7, 8, 9, 10, 10, 12, 13, 14, 15, 16};

}

@Test

void getNonUniqueElements() {

int[] result = task29.getNonUniqueElements(\_testArray);

int[] expected = {10, 12, 13, 14, 15};

assertTrue(Arrays.equals(expected, result));

}

}

Результати виконання тесту:

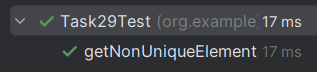


Рисунок 5 — Результат виконання тесту програми №29.

ВИСНОВКИ

У процесі виконання лабораторної роботи я навчився працювати в середовищі розробки IntelliJ IDEA, встановлювати проекти, зокрема Maven, а також створювати та організовувати класи та тести для перевірки коректності коду. Я ознайомився з процесом написання програм, що реалізують різні алгоритми для роботи з одновимірними масивами, використовуючи мову програмування Java.

Особливу увагу було приділено таким завданням, як обробка елементів масиву на основі певних умов (негативні, позитивні або нульові значення), знаходження максимальних елементів, робота з однаковими елементами, циклічні зсуви, перевірка впорядкованості масиву, і визначення локальних мінімумів. Це дозволило закріпити знання з обробки масивів, використання умовних конструкцій і циклів, а також розширити навички написання ефективних програм.

Крім того, я дізнався про важливість юніт-тестування для перевірки функціональності та надійності програмного коду. Використання бібліотеки JUnit дало можливість перевірити правильність реалізації кожної з задач та виявити помилки, що сприяло більш ефективному налагодженню програм.

Лабораторна робота допомогла вдосконалити навички програмування на Java та показала важливість систематичного підходу до тестування коду за допомогою JUnit, що є ключовою складовою професійної розробки програмного забезпечення.