**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Дисциплина «Введение в разработку программного обеспечения»

ОТЧЕТ

Лабораторная работа № 3

Моделирование и алгоритмизация как средства проектирования программного обеспечения

Выполнил студент

Группы ИТД-21

Чайдаков И.М.

Принял преподаватель

Малиновский И. Л.

Гомель 2024

Моделирование и алгоритмизация как средства проектирования программного обеспечения

**Цель работы:** научиться разрабатывать графические схемы для примитивных алгоритмов и работать с системой модульных тестов на языке *Java*.

**Задание:** Создать блок-схему алгоритма сортировки пузырьком и реализовать его на языке *Java*. Протестировать разработанный алгоритм системой модульных тестов.

**Ход выполнения задания:**

**1.** Описываем графическую схему предстоящего алгоритма (Рисунок 1).

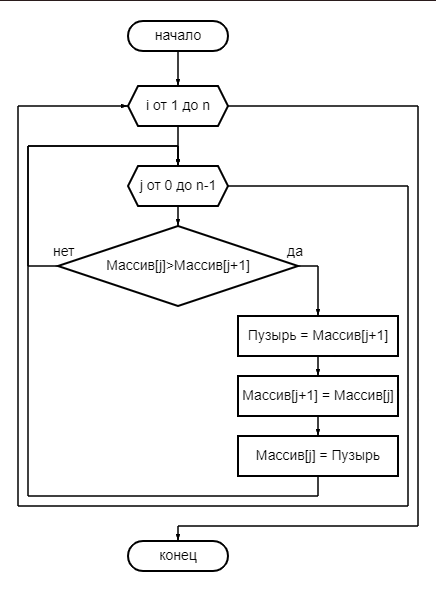


Рисунок 1 — Графическая схема алгоритма.

**2.** Описываем класс *BubbleSort* который будет предоставлять нам метод для сортировки (см. Приложение А).

**3.** Описываем консольное приложение для демонстрации работы алгоритма (см. Приложение B).

**4.** Запускаем исполняемый файл проверяя выполнение поставленных задач (Рисунок 2).

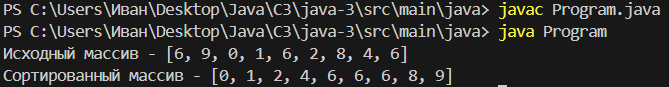


Рисунок 2 — Консольный вывод.

**5.** Описываем модульные тесты для проверки всех методов класса BubbleSort (см. Приложение C).

**6.** Запускаем разработанные тесты для верификации работы класса (Рисунок 3).

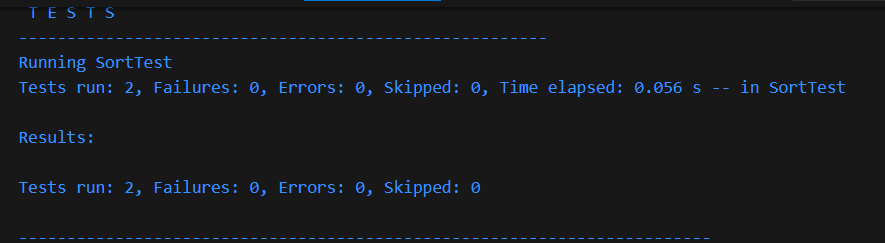


Рисунок 3 — Тестирование класса.

**Вывод:** Графические схемы позволяют в явной форме изобразить работу алгоритма что улучшает не только личное понимание работы программы, но и повышает эффективность командной работы разработчиков из-за улучшения коммуникации. Модульные тесты позволяют явно проверить исполнение функций класс без необходимости разработки отдельное консольного приложения.

**ПРИЛОЖЕНИЕ А**

**Класс *BubbleSort***

package Library;

public class BubbleSort {

    public int[] massive;

    public BubbleSort(int[] mass) {

        massive = mass;

    }

    public int[] sort() {

        int bubble;

        for(int i = 0;i < massive.length;i++) {

            for(int j = 0;j < massive.length - 1;j++) {

                if(massive[j] > massive[j+1]) {

                    bubble = massive[j+1];

                    massive[j+1] = massive[j];

                    massive[j] = bubble;

                }

            }

        }

        return massive;

    }

}

**ПРИЛОЖЕНИЕ B**

**Консольное приложение**

import Library.BubbleSort;

import java.util.Arrays;

public class Program {

    public static void main(String Args[]) {

        int[] massive = new int[] {6, 9, 0, 1, 6, 2, 8, 4, 6};

        System.out.println("Исходный массив - " + Arrays.toString(massive));

        BubbleSort sorter = new BubbleSort(massive);

        System.out.print("Сортированный массив - " + Arrays.toString(sorter.sort()));

    }

}

**ПРИЛОЖЕНИЕ С**

**Модульные тесты**

import static org.junit.jupiter.api.Assertions.assertArrayEquals;

import org.junit.jupiter.api.Test;

import Library.BubbleSort;

public class SortTest {

    @Test

    public void creationTest() {

        int[] massive = {2, 6, 2, 8, 4, 0, 2, 4, 6, 2, 7};

        BubbleSort sorter = new BubbleSort(massive);

        int[] result = sorter.massive;

        assertArrayEquals(massive, result, "Проверка создания массива");

    }

    @Test

    public void sortTest() {

        int[] massive = {2, 6, 2, 8, 4, 0, 2, 4, 6, 2, 7};

        BubbleSort sorter = new BubbleSort(massive);

        int[] result = sorter.sort();

        assertArrayEquals(new int[]{0, 2, 2, 2, 2, 4, 4, 6, 6, 7, 8}, result, "Проверка сортировки");

    }

}