**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

по дисциплине

«Введение в разработка программного обеспечения»

**Моделирование и алгоритмизация как средства**

**Проектирования программного обеспечения**

Выполнил студент группы ИТП-21

Дешев А. П.

Проверил преподаватель

Башаримов Ю. С.

Гомель 2024

Цель работы: исследовать развитие представлений о разработке

программного обеспечения, сосредоточив внимание на объектно-ориентированном программировании (ООП) в *Java*.

**Задание.**

Подготовить отчет, который должен содержать:

* Название работы.
* Цель работы.
* Условие задачи.
* Разработать *UML*‒диаграмму.
* Листинг программы.
* Вывод.

**Ход работы**

**Задание:**

Составить и блок‒схему алгоритма и программу для вычисления значений переменных в соответствии с условием. Вариант задания приведён в таблице 1. Результат выполнения задания представлен на рисунке 1, *UML*‒диаграмма на рисунке 2. Код программы представлен в приложении А. Графическая схема основного алгоритма представлена на рисунке 5.

Таблица 1 – Условие к заданию (Вариант 7)

|  |  |
| --- | --- |
| Вариант | Условие задачи |
| 7 | Алгоритм сортировки пузырьком |

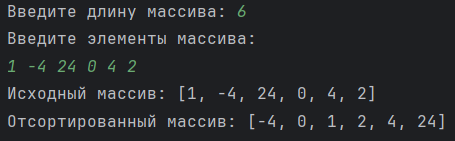




Рисунок 1 ‒ Результат выполнения задания



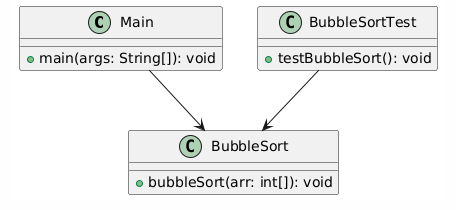


Рисунок 2 ‒ *UML*‒диаграмма

Результат тестирования программы в *idea* представлен на рисунке 3, в *leetcode* на рисунке 4.

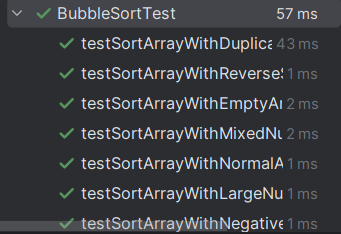


Рисунок 3 ‒ Результат тестирования программы в *idea*



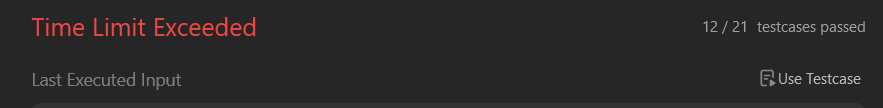


Рисунок 4 ‒ Результат тестирования программы в *leetcode*



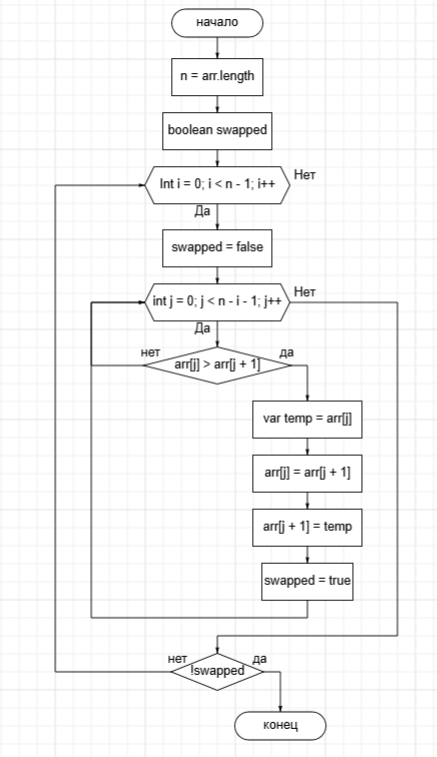
**

Рисунок 5 ‒ Графическая схема основного алгоритма



Вывод: в результате выполнения лабораторной работы изучили развитие

представлений о разработке программного обеспечения, сосредоточив внимание на объектно-ориентированном программировании (ООП) в *Java*.

**ПРИЛОЖЕНИЕ А**

**Текст программы**

**main.java:**

*import java.util.Scanner;  
import java.util.Arrays;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 System.out.print("*Введите длину массива: *");  
 var n = scanner.nextInt();  
  
 int[] arr = new int[n];  
 System.out.println("*Введите элементы массива*:");  
 for (int i = 0; i < n; i++) {  
 arr[i] = scanner.nextInt();  
 }  
  
 System.out.println("*Исходный массив*: " + Arrays.toString(arr));  
  
 BubbleSort.bubbleSort(arr);  
  
 System.out.println("*Отсортированный массив: *" + Arrays.toString(arr));  
  
 scanner.close();  
 }  
}*

**BubbleSortTest.java:**

*import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.assertArrayEquals;  
  
public class BubbleSortTest {  
  
 @Test  
 public void testSortArrayWithNormalArray() {  
 int[] arr = new int[]{64, 34, 25, 12, 22, 11, 90};  
 int[] expectedArr = new int[]{11, 12, 22, 25, 34, 64, 90};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithEmptyArray() {  
 int[] arr = new int[]{};  
 int[] expectedArr = new int[]{};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithOneElement() {  
 int[] arr = new int[]{5};  
 int[] expectedArr = new int[]{5};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithAlreadySortedArray() {  
 int[] arr = new int[]{1, 2, 3, 4, 5};  
 int[] expectedArr = new int[]{1, 2, 3, 4, 5};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithReverseSortedArray() {  
 int[] arr = new int[]{5, 4, 3, 2, 1};  
 int[] expectedArr = new int[]{1, 2, 3, 4, 5};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithAllElementsEqual() {  
 int[] arr = new int[]{2, 2, 2, 2, 2};  
 int[] expectedArr = new int[]{2, 2, 2, 2, 2};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithNegativeNumbers() {  
 int[] arr = new int[]{-1, -3, -2, 0, 1};  
 int[] expectedArr = new int[]{-3, -2, -1, 0, 1};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithMixedNumbers() {  
 int[] arr = new int[]{3, -1, 4, 1, -2, 0};  
 int[] expectedArr = new int[]{-2, -1, 0, 1, 3, 4};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithLargeNumbers() {  
 int[] arr = new int[]{1000, 500, 100, 250, 750};  
 int[] expectedArr = new int[]{100, 250, 500, 750, 1000};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithDuplicates() {  
 int[] arr = new int[]{1, 3, 2, 2, 4, 1};  
 int[] expectedArr = new int[]{1, 1, 2, 2, 3, 4};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithLargeArray() {  
 int[] arr = new int[]{10, 4, 3, 5, 1, 2, 8, 6, 7, 9};  
 int[] expectedArr = new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithSingleNegativeNumber() {  
 int[] arr = new int[]{-5};  
 int[] expectedArr = new int[]{-5};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithTwoElementsUnordered() {  
 int[] arr = new int[]{2, 1};  
 int[] expectedArr = new int[]{1, 2};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithTwoElementsOrdered() {  
 int[] arr = new int[]{1, 2};  
 int[] expectedArr = new int[]{1, 2};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
  
 @Test  
 public void testSortArrayWithThreeElements() {  
 int[] arr = new int[]{3, 1, 2};  
 int[] expectedArr = new int[]{1, 2, 3};  
 Main.sortArray(arr);  
 assertArrayEquals(expectedArr, arr);  
 }  
}*

**BubbleSort.java:** *import java.util.Arrays;  
  
public class BubbleSort {  
  
 public static void bubbleSort(int[] arr) {  
 int n;  
 n = arr.length;  
 boolean swapped;  
  
 for (int i = 0; i < n - 1; i++) {  
 swapped = false;  
  
 for (int j = 0; j < n - i - 1; j++) {  
 if (arr[j] > arr[j + 1]) {  
  
 var temp = arr[j];  
 arr[j] = arr[j + 1];  
 arr[j + 1] = temp;  
  
 swapped = true;  
 }  
 }  
 if (!swapped) {  
 break;  
 }  
 }  
 }  
}*