Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

ОТЧЕТ

к лабораторной работе №7

**«Введение в разработку с использованием платформы Java»**

|  |  |
| --- | --- |
|  | |
|  |
|  |
|  |
|  |  | | |
|  | | |  | |  |
|  | | |  | |  |
| Студент гр. 324402 | | |  | | Цевелюк А.И. |
| Проверила | | |  | | Купрейчик А.С. |

Минск 2024

**1 Цель работы**

Цель работы:изучить ряд основных языковых средств языка Java.

**2 Теоретические сведения по лабораторной работе**

Введение в язык программирования Java

Java — это объектно-ориентированный язык программирования, который разработан для создания надежных, масштабируемых и платформонезависимых приложений. Одной из ключевых особенностей Java является его принцип «Write Once, Run Anywhere» — код, написанный на Java, может быть запущен на любой платформе, которая поддерживает виртуальную машину Java (JVM).

Основные концепции Java включают:

- Объектно-ориентированное программирование (ООП), которое включает в себя принципы инкапсуляции, наследования и полиморфизма.

- Строгая типизация, которая обеспечивает контроль типов на уровне компиляции.

- Байт-код — промежуточное представление программы, выполняемое JVM, что делает Java платформонезависимой.

Объектно-ориентированное программирование в Java

ООП в Java основано на четырёх основных принципах:

- Инкапсуляция — механизм, позволяющий скрывать внутренние детали реализации класса и предоставлять доступ к данным только через методы класса.

- Наследование — механизм, который позволяет создавать новые классы на основе существующих, что способствует повторному использованию кода.

- Полиморфизм — способность объекта использовать методы производного класса через ссылку на базовый класс.

- Абстракция — процесс выделения общих характеристик объектов без указания их конкретной реализации.

- Каждая программа на Java состоит из классов и объектов. Класс — это шаблон, описывающий поведение и состояние объекта, а объект — это экземпляр класса.

Работа с классами и объектами

Класс является основной структурной единицей в Java. Он объединяет переменные (поля) и методы, которые реализуют поведение объекта. Пример объявления класса:

public class Example {

private int value; // Поле класса

// Конструктор

public Example(int value) {

this.value = value;

}

// Метод класса

public int getValue() {

return value;

}

}

Для создания объекта класса используется ключевое слово new, которое вызывает конструктор класса:

Example obj = new Example(10);

Массивы в Java

Массивы в Java — это объекты, которые позволяют хранить несколько значений одного типа в одной переменной. Массивы фиксированы по размеру и индексируются с 0. Пример создания массива:

int[] numbers = new int[5]; // Создаем массив из 5 элементов

Каждый элемент массива может быть доступен с помощью его индекса:

numbers[0] = 10; // Присваиваем значение первому элементу массива

System.out.println(numbers[0]); // Выводим значение первого элемента

Методы ввода/вывода в Java

Java предоставляет несколько способов взаимодействия с пользователем и файлами для ввода и вывода данных.

Ввод с клавиатуры. Для считывания данных с клавиатуры используется класс Scanner:

Scanner scanner = new Scanner(System.in);

int input = scanner.nextInt();

Вывод данных на экран. Для вывода данных на экран используется метод System.out.println():

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

Работа с файлами. Для чтения данных из файла используется класс Scanner, а для записи в файл — класс FileWriter:

// Чтение данных из файла

Scanner fileScanner = new Scanner(new File("input.txt"));

int data = fileScanner.nextInt();

// Запись данных в файл

FileWriter writer = new FileWriter("output.txt");

writer.write("Hello, File!");

writer.close();

Обработка исключений

В Java ошибки во время выполнения программы обрабатываются с помощью механизма исключений. Для этого используется конструкция try-catch. Пример обработки исключения:

try {

int result = 10 / 0; // Это вызовет исключение деления на ноль

} catch (ArithmeticException e) {

System.out.println("Ошибка: Деление на ноль!");

}

Циклы и операторы управления

Циклы позволяют выполнять блок кода многократно. Основные циклы в Java:

- Цикл for:

for (int i = 0; i < 5; i++) {

System.out.println(i);

}

- Цикл while:

int i = 0;

while (i < 5) {

System.out.println(i);

i++;

}

- Цикл do-while:

int i = 0;

do {

System.out.println(i);

i++;

} while (i < 5);

Условные операторы:

- Оператор if-else позволяет выполнять различные действия в зависимости от условий:

if (x > 0) {

System.out.println("x положительное");

} else {

System.out.println("x отрицательное");

}

- Оператор switch используется для выбора одного из нескольких возможных вариантов выполнения:

switch (day) {

case 1:

System.out.println("Понедельник");

break;

case 2:

System.out.println("Вторник");

break;

default:

System.out.println("Другой день");

}

Многомерные массивы

В Java также поддерживаются многомерные массивы. Пример создания двумерного массива:

int[][] matrix = new int[3][3]; // Создаем матрицу 3x3

Элементы двумерного массива можно изменять или выводить с помощью двойного индекса:

matrix[0][0] = 1;

System.out.println(matrix[0][0]);

Работа с диапазонами значений

Для работы с числами из определённого диапазона можно использовать условные операторы. Например, чтобы вывести числа, попадающие в заданный диапазон:

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

if (arr[i] >= lowerBound && arr[i] <= upperBound) {

System.out.println(arr[i]);

}

}

Запись результатов в файл

После выполнения программы, результаты могут быть записаны в файл для последующего использования или анализа. Пример записи:

FileWriter writer = new FileWriter("output.txt");

writer.write("Результаты программы");

writer.close();

**3 Формулировка индивидуального задания**

Создать класс Array для хранения массива размером size (полученного от пользователя). Позволить заполнить массив случайными числами; числами, введенными с клавиатуры; числами, считанными из файла. Вывести на экран и в файл все числа массива из указанного диапазона и их количество.

**4 Код решения индивидуального задания**

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Random;

import java.util.Scanner;

public class Array {

private final int[] arr;

private final int size;

public Array(int size) {

this.size = size;

this.arr = new int[size];

}

public void fillWithRandom() {

Random random = new Random();

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

arr[i] = random.nextInt(100);

}

}

public void fillFromKeyboard() {

Scanner scanner = new Scanner(System.in);

System.out.println("Введите " + size + " чисел:");

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

arr[i] = scanner.nextInt();

}

}

public void fillFromFile(String filename) throws IOException {

Scanner fileScanner = new Scanner(new File(filename));

for (int i = 0; i < size && fileScanner.hasNextInt(); i++) {

arr[i] = fileScanner.nextInt();

}

fileScanner.close();

}

public void print() {

System.out.print("Массив: ");

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

System.out.print(arr[i] + " ");

}

System.out.print('\n');

}

public void printAndCountInRange(int lowerBound, int upperBound, String outputFilename) throws IOException {

int count = 0;

FileWriter writer = new FileWriter(outputFilename);

System.out.println("Числа в диапазоне [" + lowerBound + ", " + upperBound + "]:");

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

if (arr[i] >= lowerBound && arr[i] <= upperBound) {

System.out.print(arr[i] + " ");

writer.write(arr[i] + " ");

count++;

}

}

System.out.print('\n');

writer.write('\n');

System.out.println("Количество чисел в диапазоне: " + count);

writer.write("Количество чисел в диапазоне: " + count + "\n");

writer.close();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Введите размер массива: ");

int size = scanner.nextInt();

Array array = new Array(size);

System.out.println("Выберите метод заполнения массива:");

System.out.println("1. Случайные числа");

System.out.println("2. Ввод с клавиатуры");

System.out.println("3. Чтение из файла");

int choice = scanner.nextInt();

try {

switch (choice) {

case 1:

array.fillWithRandom();

break;

case 2:

array.fillFromKeyboard();

break;

case 3:

System.out.print("Введите имя файла: ");

String filename = scanner.next();

array.fillFromFile(filename);

break;

default:

System.out.println("Неверный выбор.");

return;

}

array.print();

System.out.print("Введите нижнюю границу диапазона: ");

int lowerBound = scanner.nextInt();

System.out.print("Введите верхнюю границу диапазона: ");

int upperBound = scanner.nextInt();

array.printAndCountInRange(lowerBound, upperBound, "output.txt");

} catch (IOException e) {

System.out.println("Ошибка при работе с файлом: " + e.getMessage());

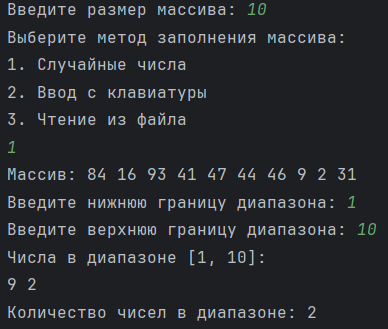
}

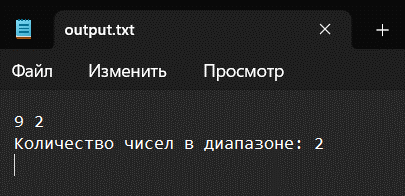
scanner.close();

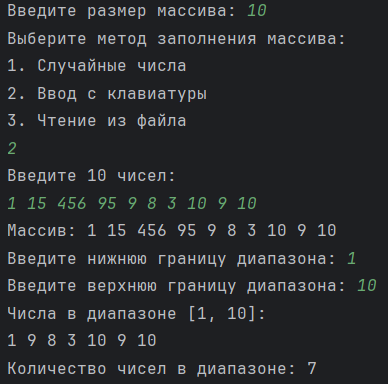
}

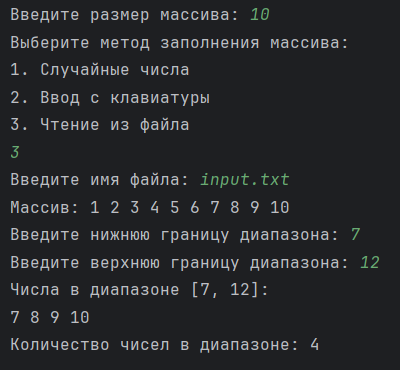
}

**5 Скриншоты выполнения индивидуального задания**









**6 Ответы на контрольные вопросы**

1. Покажите два способа объявления одномерного массива, состоящего из 12 элементов типа double.

В Java существует несколько способов объявления массива. Вот два из них:

- Первый способ: Объявление массива с последующей инициализацией.

double[] array = new double[12]; // Создается массив на 12 элементов

- Второй способ: Объявление и инициализация массива сразу.

double[] array = {0.0, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 10.10, 11.11};

2. Покажите все варианты, которыми можно воспользоваться для инициализации элементов одномерного массива целочисленными значениями от 1 до 5.

Для инициализации массива целыми числами от 1 до 5 можно использовать несколько подходов:

- Вариант 1: Инициализация массива вручную:

int[] array = {1, 2, 3, 4, 5};

- Вариант 2: Использование цикла для автоматической инициализации:

int[] array = new int[5];

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

array[i] = i + 1; // Заполнение значениями от 1 до 5

}

- Вариант 3: Инициализация с помощью метода Arrays.fill():

int[] array = new int[5];

Arrays.fill(array, 1); // Все элементы массива будут равны 1

3. Напишите программу, в которой массив используется для нахождения среднего арифметического десяти значений типа double. Используйте любые десять чисел.

Пример программы:

public class AverageCalculator {

public static void main(String[] args) {

double[] numbers = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 10.0};

double sum = 0;

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

sum += numbers[i];

}

double average = sum / numbers.length;

System.out.println("Среднее арифметическое: " + average);

}

}

4. Напишите программу, которая сортирует массив символьных строк в лексикографическом порядке по убыванию и по возрастанию. Продемонстрируйте её работу.

Пример программы:

import java.util.Arrays;

import java.util.Collections;

public class StringSorter {

public static void main(String[] args) {

String[] strings = {"apple", "banana", "orange", "kiwi"};

// Сортировка по возрастанию

Arrays.sort(strings);

System.out.println("Сортировка по возрастанию: " + Arrays.toString(strings));

// Сортировка по убыванию

Arrays.sort(strings, Collections.reverseOrder());

System.out.println("Сортировка по убыванию: " + Arrays.toString(strings));

}

}

5. В чем состоит разница между методами indexOf() и lastIndexOf() класса String?

- indexOf() возвращает индекс первого вхождения указанной подстроки или символа в строке.

- lastIndexOf() возвращает индекс последнего вхождения указанной подстроки или символа в строке.

Пример:

String text = "hello world hello";

System.out.println(text.indexOf("hello")); // Выведет 0

System.out.println(text.lastIndexOf("hello")); // Выведет 12

6. Перепишите приведенную ниже последовательность операторов, воспользовавшись тернарным оператором.

Исходный код:

if (x < 0) {

y = 10;

} else {

y = 20;

}

Решение с использованием тернарного оператора:

y = (x < 0) ? 10 : 20;

8. Является ли ошибкой превышение верхней границы массива?

Да, в Java превышение верхней границы массива является ошибкой времени выполнения, которая приводит к исключению ArrayIndexOutOfBoundsException.

Пример:

int[] array = new int[5];

array[5] = 10; // Ошибка: ArrayIndexOutOfBoundsException

9. Является ли ошибкой использование отрицательных значений индекса для доступа к элементам массива?

Да, использование отрицательных индексов для доступа к элементам массива также вызывает исключение ArrayIndexOutOfBoundsException.

Пример:

int[] array = new int[5];

array[-1] = 10; // Ошибка: ArrayIndexOutOfBoundsException

10. Можно ли управлять оператором switch с помощью объектов типа String? Если да, то напишите пример такого использования.

Да, начиная с Java 7, оператор switch поддерживает работу с объектами типа String. Вот пример:

public class SwitchExample {

public static void main(String[] args) {

String day = "Monday";

switch (day) {

case "Monday":

System.out.println("Это понедельник");

break;

case "Tuesday":

System.out.println("Это вторник");

break;

default:

System.out.println("Другой день недели");

break;

}

}

}

Этот код выведет: "Это понедельник".

**7 Выводы по лабораторной работе**

Вывод: я изучил ряд основных языковых средств языка Java.