МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Математическая Кибернетика и Информационные технологии»

Отчёт по Лабораторной работе №1

По дисциплине: «Информационные технологии и программирование»

«Знакомство с языком Java. Типы данных.»

Выполнила: студентка группы БПИ2401

Алексеева Татьяна Игоревна

Проверил: Харрасов Камиль Раисович

Москва

2025

Цель работы

Целью лабораторной работы является знакомство с историей языка программирования Java, его особенностями, достоинствами, типами данных, а также изучение базового синтаксиса, освоение процесса написания, компиляции и запуска простейших программ.

Ход работы

При изучении нового языка программирования первой программой, которую пишет человек, является вывод “Hello, World!”. После установки всего необходимого для работы (Java Development Kit, Visual Studio Code и Extension Pack for Java) напишем и запустим программу на Java, которая выведет приветствие:

```
public class JavaHelloWorldProgram {

public static void main(String args[]) {
    System.out.println("Hello, World!");
}
}
```

Запустим её и получим результат:

```
PS C:\Users\taale\OneDrive\Desktop\ITandP\LabWorkOne> java JavaHelloWorldProgram.java
Hello, World!
PS C:\Users\taale\OneDrive\Desktop\ITandP\LabWorkOne>
```

Поподробнее разберём код.
Программа начинается с объявления публичного класса JavaHelloWorldProgram, имя которого соответствует имени файла. Точкой входа в программу является метод main, который объявлен как public class void. Модификатор public обеспечивает доступность метода для JVM, static позволяет вызывать метод без создания экземпляра класса, а void указывает на отсутствие возвращаемого значения. Параметр (String args[]) предназначен для передачи аргументов командной строки. Для вывода сообщения в консоль используется конструкция System.out.println(“Hello, World!”), где System – системный класс, out – стандартный поток вывода, а println() – метод вывода данных с переходом строки.

Задание 1. Создать программу, которая находит и выводит все простые числа меньше 100.

```
public class Primes {
    public static void main(String[] args) {
        for (int i = 2; i <= 100; i++) {
            if (isPrime(i)) {
                System.out.print(i + " ");
```

```
            }
        }
        System.out.println();
    }

    public static boolean isPrime(int n) {
        for (int i = 2; i <= Math.sqrt(n); i++) {
            if (n % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

Результат запуска программы:

```
PS C:\Users\taale\OneDrive\Desktop\ITandP\LabWorkOne> java Primes.java
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
PS C:\Users\taale\OneDrive\Desktop\ITandP\LabWorkOne>
```

Программа состоит из двух основных методов: главного метода main и вспомогательного метода isPrime. В методе main реализован цикл, последовательно перебирающий числа от 2 до 100 включительно. Для каждого числа вызывается метод isPrime, который возвращает логическое значение true или false в зависимости от того, является ли число простым.

Алгоритм проверки на простоту в методе isPrime оптимизирован за счёт проверки делителей только до квадратного корня из проверяемого числа, что значительно сокращает вычислительную сложность. Для вычисления квадратного корня используется статические метод Math.sqrt() из стандартной библиотеки Java. Оператор остатка от деления % позволяет определить, делится ли число нацело на проверяемый делитель.

При обнаружении простого числа в методе main оно выводится в консоль с помощью метода system.out.print(), а пробел после каждого числа обеспечивает читаемое форматирование вывода. После завершения цикла вызов System.out.println() без параметров осуществляет переход на новую строку.

Данная программа демонстрирует основные принципы Java: модульность кода (разделение на методы), использование стандартных библиотечных функций, работу с примитивными типами и базовые управляющие конструкции.

Задание 2. Создать программу, которая определяет, является ли введенная строка палиндромом.

```java
public class Palindrome {
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            String s = args[i];
            if (isPalindrome(s)) {
                System.out.println(s + "- это палиндром");
            } else {
                System.out.println(s + "- это не палиндром");
            }

        }
    }

    public static String reverseString(String s) {
        String reversed = "";
        for (int i = s.length() - 1; i >= 0; i--) {
            reversed += s.charAt(i);
        }
    return reversed;
    }

    public static boolean isPalindrome(String s) {
        String reversed = reverseString(s);
        return s.equals(reversed);
    }
}
```

После запуска программы видим:

```
PS C:\Users\taale\OneDrive\Desktop\ITandP\LabWorkOne> java Palindrome.java madam racecar apple kayak song noon
madam- это палиндром
racecar- это палиндром
apple- это не палиндром
kayak- это палиндром
song- это не палиндром
noon- это палиндром
PS C:\Users\taale\OneDrive\Desktop\ITandP\LabWorkOne>
```

Программа состоит из трех методов: главного метода main и двух вспомогательных методов reverseString и isPalindrome. В методе main реализован цикл, который обрабатывает все аргументы командной строки, переданные при запуске программы.

Метод reverseString реализует алгоритм переворота строки путём последовательного добавления символов исходной строки в обратном порядке в новую строку. Для этого используется цикл, который начинается с последнего символа (индекс s.length() – 1) и движется к первому символу (индекс 0).

Метод isPalindrome использует функциональность reverseString для получения перевернутой версии строки и сравнения её с оригиналом с помощью метода equals(). Важно отметить, что для сравнения строк используется именно

метод equals(), а не оператор “==”, так как первый сравнивает содержимое строк, а второй – ссылки на объекты в памяти.

Программа демонстрирует ключевые аспекты работы со строками в Java, обработку аргументов командной строки, а также принцип модульности кода через разделение функциональности на отдельные методы.

Ответы на контрольные вопросы

1. Java является и компилируемым, и интерпретируемым языком программирования. Исходный код компилируется в байт-код, затем интерпретируется JVM. Для ускорения работы используется JIT-компиляция.
2. JVM (Java Virtual Machine) – это виртуальная машина, которая исполняет байт-код Java. Её основное назначение – обеспечение кроссплатформенности и управление памятью.
3. Написание кода с использованием синтаксиса Java → Компиляция (исходный код (.java) компилируется в байт код (.class) с помощью компилятора javac) → Интерпретация (байт-код выполняется виртуальной машиной Java. JVM интерпретирует байт-код в машинный код во время выполнения) → JIT-компиляция «горячих» участков кода → Выход программы.
4. В языке Java есть примитивные (int, float, double, boolean и т.д.) и ссылочные (массивы, классы, интерфейсы, строки) типы данных.
5. Примитивные типы данных хранят значение непосредственно в переменной, а ссылочные – хранят адрес объекта в памяти (ссылку). Примитивные типы не являются объектами и не имеют методов.
6. Преобразование примитивных типов бывает двух видов: неявное преобразование, происходит автоматически, когда меньший тип преобразуется в больший, оно не приводит к потери данных (byte → short → int → long → float → double). Есть явное преобразование, которое требует указания целого типа в скобках, оно может привести к потере точности или данных (double → float → long → int → shot → byte).
7. Байт-код – это промежуточный код, который компилируется из исходного Java-кода. Он представляет собой набор инструкций для JVM. Его важность заключается в: едином формате (исходный код компилируется в универсальный байт-код, который не зависит от конкретной операционной системы или архитектуры процессора), Write Once, Run Anywhere (программа, скомпилированная в байт-код, может выполняться на любой платформе, где установлена JVM), JVM интерпретирует байт-код в

машинные команды конкретной платформы, оптимизации (JIT-оптимизация преобразует часто используемые участки байт-кода в нативный код для ускорения выполнения).

8. Для хранения символов в Java используется примитивный тип char. 16-битный символ Unicode. Используется для представления одиночных символов.
9. Литералы – это явно заданные значения в исходном коде программы. (int – 42, -100, 0; double – 3.14, -7.6; boolean – true, false).
10. Java считается строго типизированным языком потому, что он требует обязательного объявления типа для каждой переменной и выполняется строгую проверку типов на этапе компиляции. Компилятор запрещает несовместимые операции с типами данных и разрешает неявные преобразования только в направлении от меньшего к большему, в то время как все явные преобразования требуют явного указания целого типа.
11. Проблемы, которые могут возникнуть при использовании явного преобразования типов: потеря точности при преобразовании чисел с плавающей точкой к целым, переполнение при преобразовании больших значений к меньшим типам, неожиданные результаты арифметических операций, ошибки округления при работе с дробными числами, проблемы производительности из-за неявных преобразований в выражениях со смешанными типами.

Ссылка на репозиторий: https://github.com/T0tyana/Labworks_ITIP.git

Вывод

В ходе лабораторной работы были освоены основы программирования на Java, включая структуру программы, работу с примитивными типами данных, создание и вызов методов, а также процессы компиляции и выполнения кода. Приобретены практические навыки написания алгоритмов проверки чисел на простоту и определения палиндромов, что позволило закрепить понимание ключевых принципов языка, таких как строгая типизация, работа со строками и обработка аргументов командной строкой.