МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена трудового Красного Знамени федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра Математическая кибернетика и информационные технологии

Отчет по лабораторной работе №1

по дисциплине «Информационные технологии и программирование на языке Java»

Выполнил: студент группы БВТ2403

Подлуцкий Никита Сергеевич

Руководитель: _____

Москва, 2025

2. Цель работы

Цель работы — научиться применять базовые возможности языка Java для решения практических задач. В ходе работы необходимо было реализовать программы для поиска простых чисел и проверки строк на палиндромы, используя при этом циклы, условные операторы и отдельные статические методы, как было указано в задании.

3. Индивидуальное задание

1. Создать программу, которая находит и выводит на экран все простые числа в диапазоне от 2 до 100 включительно.
2. Создать программу, которая определяет, является ли введенная строка палиндромом.

4. Основная часть

4.1. Краткое описание реализации

Программа была разделена на три класса, чтобы отделить основную логику от вспомогательных функций. Такой подход был выбран для удобства и читаемости кода, а также для следования примеру из методических указаний, где предлагается выносить логику в отдельные методы.

- **Main.java**: основной класс, который содержит метод main и запускает выполнение обеих задач.
- **Primes.java**: содержит статический метод isPrime для проверки простоты числа. Для ускорения работы проверка делителей идет не до самого числа, а до его квадратного корня.
- **Palindrome.java**: содержит два статических метода, как требовалось в задании. Метод reverseString переворачивает строку, а метод isPalindrome использует его для сравнения исходной строки с перевернутой.

4.2. Листинг программы

```java
package com.example;

import java.util.List;

public class Main {
    Run | Debug
    public static void main(String[] args) {

        System.out.println(x:"Простые числа от 2 до 100:");
        runPrimeNumberFinder();
        System.out.println("-".repeat(count:15)); // Типо "string"*15 в python


        System.out.println(x:"Проверка палиндромов:");
        runPalindromeChecker();
    }

    public static void runPrimeNumberFinder() { // static это чтобы не создавать объект класса
        for (int i = 2; i <= 100; i++) { //Можно прибавить по 2, но тогда надо начинать с 3 и вывести 2 вручную
            if (Primes.isPrime(i)) {
                System.out.print(i+" ");

            }if (i % 10 == 0) {System.out.println(x:"");}
        }
    }

    public static void runPalindromeChecker() {
        List<String> words = List.of(e1:"мадам", e2:"racecar", e3:"Никита", e4:"level");
        words.forEach(word -> {
            Palindrome.isPalindrome(word);
            if (Palindrome.isPalindrome(word)) {
                System.out.println(word + " - это палиндром.");
            } else {
                System.out.println(word + " - это не палиндром.");
            }
        });
    }
}
```

```java
package com.example;

public class Primes {
    public static boolean isPrime(int number) {
        if (number < 2) {
            return false;
        }
        int lit = (int) Math.sqrt(number); // (int) это как int(), а корень ускорят процесс
        for (int i = 2; i <= lit; i++) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
    }
}
```

```java
package com.example;

public class Palindrome {

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

    public static boolean isPalindrome(String word) {
        String original = word.toLowerCase();
        String reversed = reverseString(original);

        return original.equals(reversed);
    }
}
```

4.3. Результаты работы программы

```
Простые числа от 2 до 100:
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
--------------
Проверка палиндромов:
мадам - это палиндром.
racecar - это палиндром.
Никита - это не палиндром.
level - это палиндром.
```

5. Заключение

В ходе выполнения лабораторной работы я успешно реализовал программы для поиска простых чисел и проверки палиндромов на языке Java. Я применил на практике циклы (for) и условные операторы (if), как было описано в теоретической части. Также я понял, как выносить код в отдельные статические методы, что делает основную программу проще и понятнее.

GitHub: https://github.com/Nikita-Podlutsky/ITIP/tree/main

Контрольные вопросы:

1. Java — это одновременно и компилируемый, и интерпретируемый язык. Исходный код сначала компилируется в специальный байт-код, а уже потом этот байт-код выполняется (интерпретируется) Виртуальной Машиной Java.
2. JVM (Виртуальная Машина Java) — это программа, которая выполняет скомпилированный Java-код. Она создает одинаковую среду на любой операционной системе, благодаря чему Java-программы могут работать везде без изменений.
3. Жизненный цикл программы состоит из трех основных этапов: написание исходного кода (файл .java), компиляция его в байт-код (файл .class) и выполнение этого байт-кода на JVM.
4. В Java есть два вида типов данных: примитивные (такие как int, double, boolean, char) и ссылочные (любые объекты и массивы, например, String).
5. Примитивные типы хранят само значение (например, число 42). Ссылочные типы хранят не сам объект, а ссылку (адрес в памяти), по которому этот объект можно найти.
6. Преобразование бывает неявным (автоматическим), когда мы присваиваем значение меньшего типа переменной большего типа (например, int в long). И бывает явным, когда мы вручную указываем тип в скобках, например (int)someDouble, что требуется при сужении типа.
7. Байт-код — это результат компиляции Java-кода, который не зависит от конкретного компьютера или операционной системы. Он важен для платформенной независимости, так как один и тот же байт-код может быть запущен на любой машине, где есть JVM.
8. Для хранения символов используется тип char. В памяти он занимает 2 байта и представляет собой символ в кодировке Unicode.

9. Литерал — это фиксированное значение, записанное прямо в коде.
   Примеры: 100 (литерал типа int), "Привет" (строковый литерал), true (логический литерал), 'C' (символьный литерал).

10. Java считается строго типизированным языком, потому что тип каждой переменной должен быть объявлен до ее использования, и компилятор строго следит за соответствием типов. Это позволяет находить ошибки на этапе компиляции, а не во время выполнения программы.

11. Проблемы могут возникнуть при явном преобразовании типов, когда больший тип преобразуется в меньший. Это может привести к потере точности или искажению значения, если оно не помещается в новый, меньший тип.