

**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

Выполнил: студент группы БПИ2401

Трухина Анастасия Александровна

Проверил: Харрасов Камиль Раисович

Москва,

2025

Оглавление

Цель работы:	3
Задание:	3
Основная часть.....	4
Ответы на контрольные вопросы:	6
Заключение	9

Цель работы:

Цель работы — освоить базовые принципы языка Java, включая структуру программы, типы данных, методы и работу со строками, на практике через реализацию двух задач: поиска простых чисел и проверки палиндромов. В ходе выполнения закрепляются навыки написания читаемого, модульного кода, корректного использования циклов, условий, сравнения строк и передачи аргументов через командную строку.

Задание:


- 1) Необходимо написать программу, которая находит и выводит все простые числа меньше 100. Программа должна быть реализована в файле с именем `Primes.java`, содержащем класс `public class Primes`. Внутри этого класса необходимо объявить метод `public static void main(String[] args)`, а также дополнительный статический метод `public static boolean isPrime(int n)`, который определяет, является ли переданное целое число `n` простым. Считается, что входное значение `n` всегда больше 2. Метод `isPrime` должен проверять делимость числа `n` на все целые числа от 2 до \sqrt{n} включительно с использованием оператора остатка `%`. Если хотя бы одно из этих чисел делит `n` без остатка, метод должен вернуть `false`; если ни одно не делит — вернуть `true`. В методе `main()` необходимо организовать цикл, перебирающий целые числа от 2 до 100 включительно, вызывать для каждого числа метод `isPrime(n)` и выводить на экран те значения, для которых метод возвращает `true`.
- 2) Необходимо написать программу, которая определяет, является ли каждая из переданных через аргументы командной строки строк палиндромом. Программа должна быть реализована в файле с именем `Palindrome.java`, содержащем класс `public class Palindrome`. Внутри этого класса необходимо объявить метод `public static void main(String[] args)`, а также два дополнительных статических метода: первый — `public static String reverseString(String s)`, который принимает строку `s` и возвращает её символы в обратном порядке, используя методы `s.length()` и `s.charAt(int index)` для посимвольного прохода с конца строки; второй — `public static boolean isPalindrome(String s)`, который использует метод `reverseString(s)` для получения перевёрнутой версии строки и сравнивает её с оригиналом с помощью метода `.equals()`, при этом нельзя использовать оператор `==` для сравнения строк. В методе `main()`

необходимо обработать все аргументы командной строки (args), для каждого из них вызвать метод isPalindrome(s) и вывести результат в формате: ««строка» — палиндром» или ««строка» — не палиндром». Программу необходимо запускать через командную строку следующим образом: java Palindrome madam racecar apple kayak song noon.

Основная часть

Задание 1.

Код полученной программы:



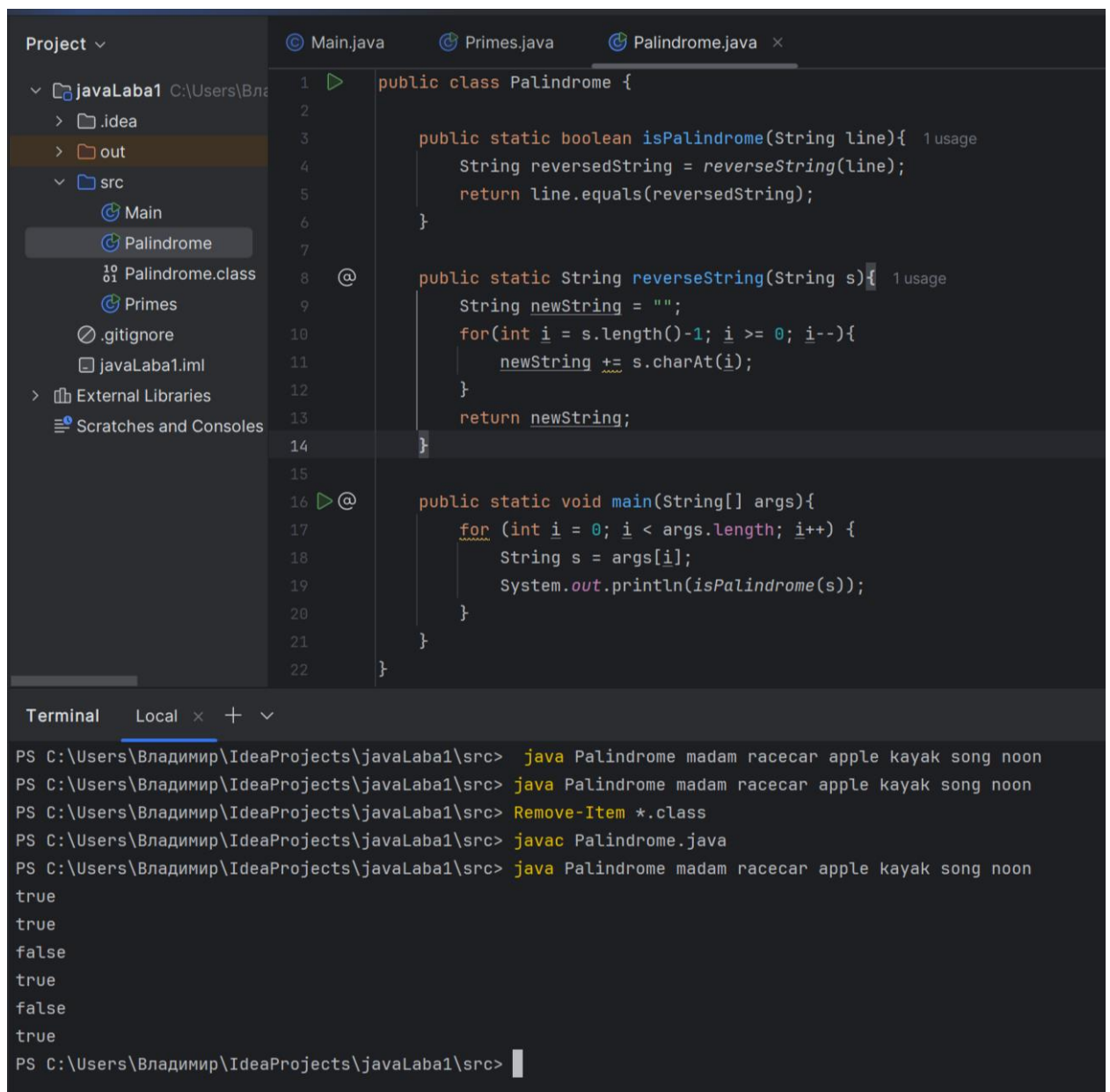
```
© Main.java  Primes.java x
1  public class Primes {
2
3      public static boolean isPrime(int n){ 1 usage
4
5          for(int d = 2; d <= Math.ceil(Math.sqrt(n)); d++){
6              if(n % d == 0)
7                  return false;
8          }
9          return true;
10     }
11
12     public static void main(String[] args){
13         for(int i = 2; i <= 100; i++){
14             if(isPrime(i) == true){
15                 System.out.print(i + " ");
16             }
17         }
18     }
19 }
20

n  Primes x
:
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program File
3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
Process finished with exit code 0
```

Объяснение кода:

В публичном классе Primes объявляем публичный метод isPrime, который возвращает True или False. Вычисляются простые числа по следующему принципу: у числа нет делителей, кроме 1 и самого числа. Реализуется перебор чисел от 2 до корня из самого числа, для поиска возможных делителей. В случае, если есть хотя бы 1 делитель, метод возвращает сразу же False. В точке входа в программу (метод main) реализуется цикл, в котором ищутся простые числа от 2 до 100 и выводятся в консоли.

Задание 2



```
Project ▾
  ▾ javaLaba1 C:\Users\Владимир\IdeaProjects\javaLaba1
    ▾ .idea
    ▾ out
    ▾ src
      Main
      Palindrome
      Palindrome.class
      Primes
      .gitignore
      javaLaba1.iml
    ▾ External Libraries
    ▾ Scratches and Consoles

Main.java  Primes.java  Palindrome.java ×
1  public class Palindrome {
2
3      public static boolean isPalindrome(String line){ 1 usage
4          String reversedString = reverseString(line);
5          return line.equals(reversedString);
6      }
7
8      public static String reverseString(String s){ 1 usage
9          String newString = "";
10         for(int i = s.length()-1; i >= 0; i--){
11             newString += s.charAt(i);
12         }
13         return newString;
14     }
15
16     public static void main(String[] args){
17         for (int i = 0; i < args.length; i++) {
18             String s = args[i];
19             System.out.println(isPalindrome(s));
20         }
21     }
22 }
```

```
Terminal  Local × + ▾
PS C:\Users\Владимир\IdeaProjects\javaLaba1\src> java Palindrome madam racecar apple kayak song noon
PS C:\Users\Владимир\IdeaProjects\javaLaba1\src> java Palindrome madam racecar apple kayak song noon
PS C:\Users\Владимир\IdeaProjects\javaLaba1\src> Remove-Item *.class
PS C:\Users\Владимир\IdeaProjects\javaLaba1\src> javac Palindrome.java
PS C:\Users\Владимир\IdeaProjects\javaLaba1\src> java Palindrome madam racecar apple kayak song noon
true
true
false
true
false
true
PS C:\Users\Владимир\IdeaProjects\javaLaba1\src>
```

Объяснение кода:

Было прописано 2 метода:

В 1ом проверяется равны ли значения объектов по их ссылкам. Во 2ом

строка возвращается в обратном порядке. В методе `main` происходит ввод значений из терминала (командной строки) через цикл, пока идет сам непосредственный ввод.

`javac Palindrome.java` – пропишем команду в терминале в папке `src`, чтобы сформировать файл с байт-кодом с расширением `.class`, что позволяет далее ввести строку через терминал

Ответы на контрольные вопросы:

1. Java является языком, сочетающим черты компилируемого и интерпретируемого: сначала исходный код (`.java`) компилируется компилятором `javac` в платформенно-независимый байт-код (`.class`), который затем выполняется виртуальной машиной Java (JVM). На этапе выполнения JVM интерпретирует байт-код, но для повышения производительности активно использует механизм JIT-компиляции (Just-In-Time), динамически компилируя часто используемые участки байт-кода в нативный машинный код целевой платформы. Таким образом, Java не является чисто компилируемым или чисто интерпретируемым языком — она использует гибридный подход, обеспечивающий как переносимость, так и приемлемую производительность.

2. JVM (Java Virtual Machine) — это виртуальная машина, предназначенная для выполнения байт-кода Java. Она обеспечивает загрузку, верификацию и выполнение скомпилированных Java-классов, управляет памятью (включая автоматическую сборку мусора), обеспечивает безопасность выполнения кода и абстрагирует программу от особенностей конкретной аппаратной платформы и операционной системы. Благодаря JVM один и тот же байт-код может выполняться на любом устройстве, где установлена совместимая реализация JVM, что лежит в основе принципа «Write Once, Run Anywhere» (WORA).

3. Жизненный цикл программы на языке Java включает следующие этапы: сначала разработчик пишет исходный код на языке Java и сохраняет его в файл с расширением `.java`; затем компилятор `javac` преобразует этот исходный код в байт-код, сохраняя результат в файл с расширением `.class`; после этого при запуске программы с помощью команды `java` JVM загружает

байт-код, проверяет его на корректность и безопасность, а затем выполняет — сначала интерпретируя инструкции, а затем, при необходимости, используя JIT-компиляцию для преобразования часто выполняемых участков в нативный машинный код с целью оптимизации производительности; программа завершает работу после выполнения метода `main` или при возникновении необработанного исключения.

4. В языке Java существует два основных вида типов данных: примитивные и ссылочные. Примитивные типы включают восемь встроенных типов: логический (`boolean`), символьный (`char`), а также числовые — целочисленные (`byte`, `short`, `int`, `long`) и вещественные (`float`, `double`). Ссылочные типы включают классы (в том числе класс `String`), интерфейсы, массивы, перечисления и аннотации; они хранят не само значение, а ссылку на объект, размещённый в куче (`heap`).

5. Примитивные типы данных отличаются от ссылочных тем, что они хранят непосредственное значение (например, число или символ) и не имеют методов, тогда как ссылочные типы хранят адрес (ссылку) на объект в памяти, а сами объекты содержат данные и методы для их обработки. Примитивные переменные обычно размещаются в стеке и имеют фиксированный размер (например, `int` всегда занимает 4 байта), в то время как объекты ссылочных типов создаются в куче, их размер динамичен, и по умолчанию такие переменные инициализируются значением `null`, в отличие от примитивов, которые получают значения по умолчанию (`0`, `false` и т.д.).

6. Преобразование примитивных типов в Java происходит двумя способами: неявно (автоматически) и явно (с использованием приведения типов). Неявное преобразование (`widening conversion`) применяется, когда значение меньшего типа присваивается переменной большего типа (например, `byte` → `int` или `int` → `double`), и при этом не происходит потери данных. Явное преобразование (`narrowing conversion`) требуется при присваивании большего типа меньшему (например, `double` → `int` или `long` → `short`) и записывается в виде (тип)выражение; в этом случае возможна потеря точности или переполнение, так как программист берёт на себя ответственность за корректность преобразования. Логический тип `boolean` нельзя преобразовывать ни в какой другой тип и наоборот.

7. Байт-код в Java — это промежуточное представление программы, генерируемое компилятором `javac` из исходного кода и сохраняемое в файлах с расширением `.class`. Он состоит из инструкций, понятных только JVM, и не зависит от архитектуры процессора или операционной системы. Байт-код важен для платформенной независимости, потому что именно он позволяет одной и той же скомпилированной программе выполняться на любой платформе, где установлена совместимая JVM, без необходимости повторной компиляции или модификации исходного кода, что и реализует ключевой принцип Java — «Write Once, Run Anywhere».

8. Для хранения символов в Java используется примитивный тип данных `char`, который занимает 16 бит (2 байта) и основан на стандарте Unicode. Каждый символ представляется в памяти как целое число, соответствующее его коду в таблице Unicode (в диапазоне от `\u0000` до `\uffff`, то есть от 0 до 65535). Это позволяет Java корректно работать с символами различных языков мира, включая кириллицу, иероглифы и специальные символы.

9. Литералы в Java — это фиксированные значения, записанные непосредственно в исходном коде. Примеры литералов: целочисленные — `42`, `100L` (для `long`), `0xFF` (шестнадцатеричный); с плавающей точкой — `3.14` (`double` по умолчанию), `2.5f` (`float`); символьные — `'A'`, `'\n'`, `'\u03A9'`; строковые — `"Hello, World!"`; логические — `true` и `false`; а также `null` для ссылочных типов. Литералы позволяют инициализировать переменные конкретными значениями без вычислений.

10. Java считается строго типизированным языком, потому что каждая переменная должна быть явно объявлена с указанием её типа, и этот тип нельзя изменить в ходе выполнения программы. Компилятор строго проверяет совместимость типов при всех операциях — присваивании, передаче аргументов, возврате значений и арифметических вычислениях. Любая попытка использовать несовместимые типы без явного приведения приводит к ошибке на этапе компиляции. Такой подход предотвращает множество ошибок времени выполнения, повышает надёжность и читаемость кода, а также упрощает его анализ и отладку.

11. При использовании неявного преобразования типов (автоматического расширения) могут возникнуть скрытые проблемы, связанные с потерей точности или неожиданным поведением программы. Например, при неявном преобразовании целого числа в `float` или `double` возможна потеря значащих цифр из-за ограниченной точности представления вещественных чисел. Также при смешивании типов в арифметических выражениях результат может иметь тип, не соответствующий ожиданиям программиста (например, деление двух целых чисел даёт целый результат, даже если он присваивается переменной `double`). Хотя неявное преобразование само по себе безопасно и не вызывает ошибок компиляции, оно может маскировать логические ошибки, особенно при работе с большими числами или при последующем явном приведении к меньшему типу.

Заключение

В ходе лабораторной работы освоены базовые принципы Java: структура программы, работа с методами, строками и аргументами командной строки. Реализованы две задачи — поиск простых чисел до 100 и проверка палиндромов — с корректным использованием циклов, условий и сравнения строк через `.equals()`. Программы успешно компилируются и запускаются из командной строки, выводя ожидаемый результат. Работа закрепила навыки модульного программирования и понимание ключевых особенностей языка Java.

Ссылка на ГитХаб с файлами кода: [NT-005-TN/ITiP_LabWorks_Trukhina](https://github.com/NT-005-TN/ITiP_LabWorks_Trukhina)