

**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»

Выполнил: студент группы

БПИ2401

Старков Дмитрий Константинович

Проверил:

Харрасов Камиль Раисович

Москва, 2025

Содержание

Цель работы	2
Ход работы.....	2
Вывод.....	5
Ответы на контрольные вопросы	5

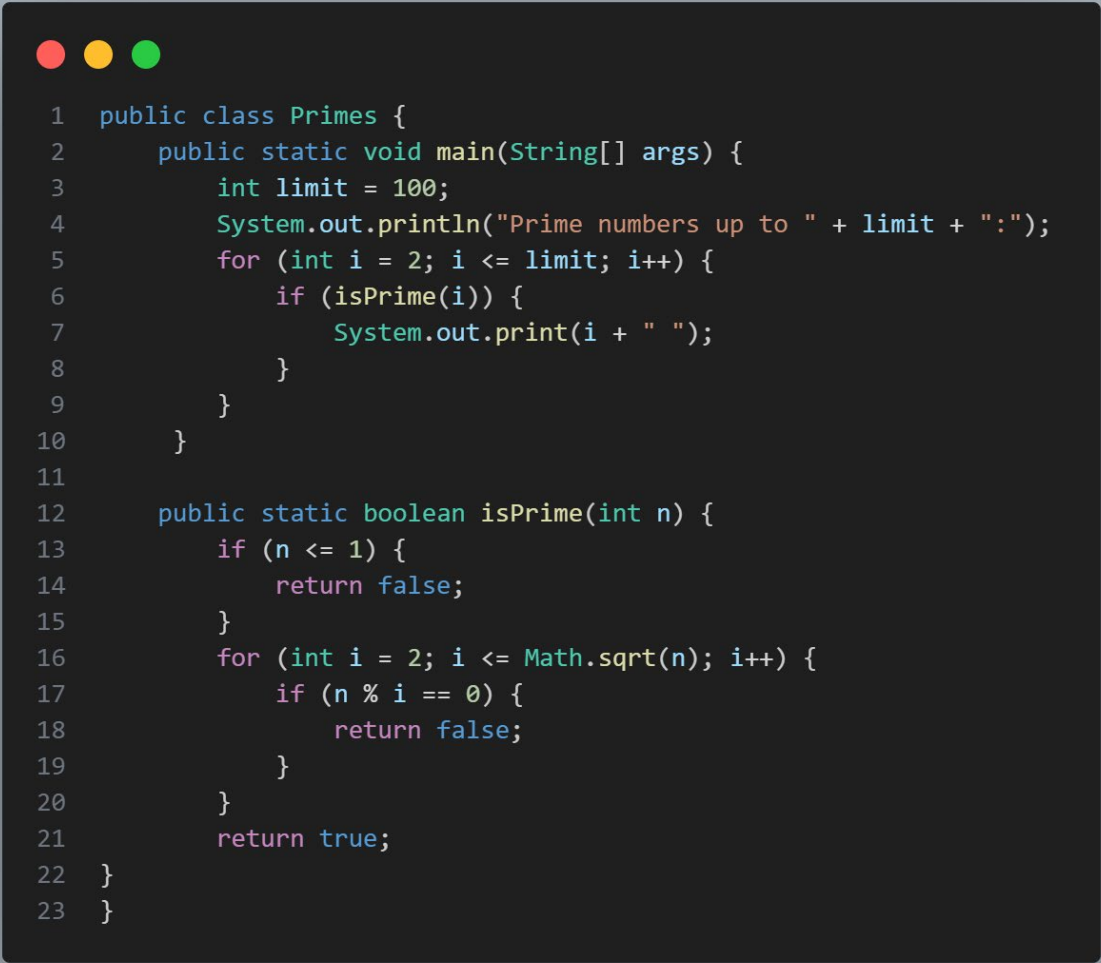
Цель работы

Освоить основы языка Java на примере простых программ для определения простых чисел и палиндромов

Ход работы

После необходимых подготовительных мероприятий, а именно установки java и jdk на компьютер, а также последующей их настройки, я приступил к выполнению первого задания.

Код для первого задания выглядит следующим образом



```
1 public class Primes {
2     public static void main(String[] args) {
3         int limit = 100;
4         System.out.println("Prime numbers up to " + limit + ":");
5         for (int i = 2; i <= limit; i++) {
6             if (isPrime(i)) {
7                 System.out.print(i + " ");
8             }
9         }
10    }
11
12    public static boolean isPrime(int n) {
13        if (n <= 1) {
14            return false;
15        }
16        for (int i = 2; i <= Math.sqrt(n); i++) {
17            if (n % i == 0) {
18                return false;
19            }
20        }
21        return true;
22    }
23 }
```

Для начала я создал класс Primes, внутри которого инициализировал точку входа в программу – функцию main, внутри которой я установил лимит

верхней границы диапазона чисел, чтобы сделать код более гибким и настраиваемым. Далее мной был установлен шаблон вывода в консоль информации о простых числах. После чего уже с помощью цикла идет перебор в заданном диапазоне, при котором идёт вызов отдельной функции `isPrime`.

Сама функция `isPrime` устроена довольно просто. В начале я от себя добавил проверку числа `n`, если оно меньше 2, то функция автоматически возвращает `False`. Затем идет стандартный алгоритм проверки числа на простоту, и если во время проверки число `n` ни разу не делилось на `i` без остатка, то в конечном итоге функция возвращает `True`, и в `main` на этапе вызова `isPrime` передается `True`, соответственно текущее число выводится в консоль.

Вот вывод после выполнения данной программы:

```
PS C:\Study\ИТИП\lab1> java Primes
Prime numbers up to 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
```

Вывод корректный.

Переходим ко второму заданию с программой для определения палиндромов
Вот код:

```

1  public class Palindrome {
2      public static void main(String[] args) {
3          for (String s : args) {
4              if (isPalindrome(s)) {
5                  System.out.println(s + " is a palindrome.");
6              } else {
7                  System.out.println(s + " is not a palindrome.");
8              }
9          }
10     }

11
12     public static String reverseString(String s) {
13         String reversed = "";
14         for (int i = s.length() - 1; i >= 0; i--) {
15             reversed += s.charAt(i);
16         }
17         return reversed;
18     }

19
20     public static boolean isPalindrome(String s) {
21         String reversed = reverseString(s);
22         return s.equals(reversed);
23     }
24 }

```

Теперь кратко объясню его функционал.

Начало стандартное – инициализация класса и точки входа в программу. Далее идет цикл, перебирающий все аргументы, переданные при запуске программы. После проверки через вызов функции `isPalindrome` в консоль выводится либо утвердительный либо отрицательный ответ. Что касается самой функции `isPalindrome`, она в свою очередь обращается к уже другой функции при инициализации реверсивной строки и последующим сравнением с оригинальной строкой. Функция `reverseString`, на которую и происходит ссылка, задаёт пустую строку, после чего, через цикл с убывающей переменной, обращается к каждому символу исходной строки по его индексу, начиная с конца, и добавляет этот символ в конец строки.

Теперь посмотри на вывод с предложенными в задании словами:

```
PS C:\Study\ИТИП\lab1> java Palindrome madam racecar apple kayak song noon
madam is a palindrome.
racecar is a palindrome.
apple is not a palindrome.
kayak is a palindrome.
song is not a palindrome.
noon is a palindrome.
```

Вывод корректный

Вывод

В ходе лабораторной работы я освоил основы языка Java, понял его логику и принцип работы.

Ответы на контрольные вопросы

1. Java является одновременно компилируемым и интерпретируемым языком. Сначала исходный код компилируется в байт-код с помощью компилятора, а затем этот байт-код выполняется виртуальной машиной Java, которая интерпретирует его или дополнительно компилирует в машинный код с помощью JIT.
2. JVM — это виртуальная машина Java, предназначенная для выполнения байт-кода. Она обеспечивает платформенную независимость, управление памятью, безопасность и поддержку многопоточности.
3. Жизненный цикл программы на Java состоит из написания исходного кода, компиляции его в байт-код, загрузки классов в JVM, проверки байт-кода, выполнения программы интерпретатором или JIT-компилятором и завершения работы с освобождением ресурсов.
4. В языке Java есть два вида типов данных: примитивные и ссылочные. Примитивные типы включают восемь базовых видов — целые, числа с плавающей точкой, символы и логический тип. Ссылочные типы — это классы, объекты, массивы и строки.

5. Прimitives типы данных отличаются от ссылочных тем, что хранят непосредственно значение, тогда как ссылочные содержат адрес объекта в памяти.
6. Преобразование примитивных типов в Java может происходить автоматически при расширении диапазона, например при переводе `int` в `long`, или выполняться явно через приведение типов при сужении, например при переводе `double` в `int`. Кроме того, есть механизм автоупаковки и распаковки, когда примитивы преобразуются в соответствующие объектные обёртки и обратно.
7. Байт-код — это промежуточное представление программы, которое создаётся компилятором и исполняется JVM. Он важен, потому что делает Java платформенно независимой: один и тот же байт-код можно запустить на любой системе, где есть виртуальная машина.
8. Для хранения символов в Java используется тип `char`. Он занимает 16 бит и представляет символы в кодировке UTF-16, что позволяет хранить широкий диапазон символов.
9. Литералы в Java — это фиксированные значения, записанные прямо в коде программы. Примеры литералов: целые числа вроде `10` или `0xFF`, вещественные числа вроде `3.14`, символьные значения вроде `'A'`, строковые значения вроде `"fdsfsdfsdfas"`, логические литералы `true` и `false`, а также специальный литерал `null` для ссылочных типов.
10. Java считается строго типизированным языком, потому что каждая переменная и объект имеют строго определённый тип, и нельзя использовать их вне рамок этого типа без явного преобразования. Это снижает вероятность ошибок и повышает надёжность кода.
11. Проблемы при использовании неявного преобразования типов могут быть связаны с потерей данных, например при переводе `double` в `int`, с переполнением, когда число не помещается в меньший тип, и с

ошибками логики, если результат преобразования оказался неожиданным.

Ссылка на гит - <https://github.com/BestStarProgramer/IT-P/tree/main/lab1>