

**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»

Отчет по лабораторной работе № 1.

по дисциплине «Информационные технологии и программирование»

по разделу:

«1. История языка программирования

Java. Типы данных»

Выполнила: студентка группы БПИ2401 Рябова Екатерина

Проверил: Харрасов Камиль Раисович

Москва

2025

1. Цель работы:

Научиться выполнять простые операции с основными типами данных – числовыми, строковыми и булевыми. Научиться использовать условные операторы, условные операторы и методы. Освоить простейший ввод и вывод данных.

2. Ход работы:

1. Подготовка к работе в Visual Studio Code.

Установим Java Development Kit, включающий в себя компилятор `javac`, среду выполнения Java с виртуальной машиной Java, стандартные библиотеки классов. Test Runner for Java от Microsoft и Java Development Packs, а также Java Platform Extension for Visual Studio Code.

2. Выполним Листинг 1.1. Класс `JavaHelloWorldProgram`.

Напишем программу из методички, создадим каталог для первой лабораторной работы и сохраним программу под именем `JavaHelloWorldProgram`. Код программы представлен на рисунке 1.

```
.vscode > java > firstlab > J JavaHelloWorldProgram.java > ...
1  package firstlab;
2  public class JavaHelloWorldProgram {
    Run main | Debug main | Run | Debug
3      public static void main(String args[ ]){
4          System.out.println("Hello World");
5      }
6  }
7  |
```

Рисунок 1. Код программы `JavaHelloWorldProgram.java`

Откроем терминал, перейдем в каталог первой лабораторной работы и выполним команду `java JavaHelloWorldProgram.java`. Результат выполнения команды показан на рисунке 2.

```

PS C:\Users\user\.vscode\mtuci\.vscode\java> cd firstlab
PS C:\Users\user\.vscode\mtuci\.vscode\java\firstlab> java JavaHelloWorldProgram.java
Hello World
PS C:\Users\user\.vscode\mtuci\.vscode\java\firstlab>

```

Рисунок 2. Результат выполнения команды `java JavaHelloWorldProgram.java`

После команды `javac JavaHelloWorldProgram.java` и компиляции у нас генерируется байт-код и сохраняется с тем же названием и расширением `.class` в том же каталоге (результат на рисунке 3). Далее можно подняться на уровень выше и выполнять в командной строке команду `java` каталог.имя. Результат представлен на рисунке 4:

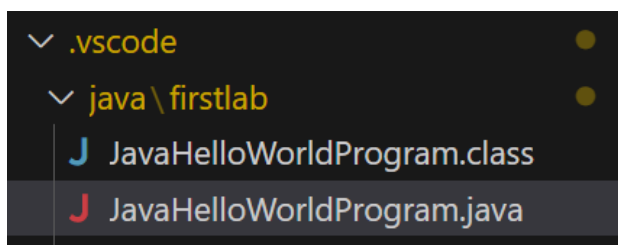


Рисунок 3. Результат выполнения команды `javac JavaHelloWorldProgram.java`

```

PS C:\Users\user\.vscode\mtuci\.vscode\java\firstlab> javac JavaHelloWorldProgram.java
PS C:\Users\user\.vscode\mtuci\.vscode\java\firstlab> cd ..
PS C:\Users\user\.vscode\mtuci\.vscode\java> java firstlab.JavaHelloWorldProgram
Hello World

```

Рисунок 4. Результат выполнения команды вида `java каталог.имя`.

3. Задание 1 для лабораторной работы:

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

Создадим файл с именем `Primes.java` и напишем следующий код, представленный на рисунке 5.

```
.vscode > java > firstlab > J Primes.java > Java > Primes
1  package firstlab;
2
3  public class Primes {
4      Run main | Debug main | Run | Debug
5      public static void main(String[] args) {
6          for (int i = 2; i <= 100; i++) {
7              if (isPrime(i)) {
8                  System.out.println(i);
9              }
10         }
11
12         public static boolean isPrime(int n) {
13             for (int i = 2; i <= Math.sqrt(n); i++) {
14                 if (n % i == 0) {
15                     return false;
16                 }
17             }
18             return true;
19         }
20     }
```

Рисунок 5. Код программы *Primes.java*

Vscode автоматически подставил имя пакета и название класса.

В у публичного класса *Primes* реализуем 2 публичных *static* метода (такие, что их можно вызвать без создания экземпляра класса): *main* – точка входа – и *isPrime* – проверка на простоту. Класс *main*, как и положено, возвращает ничего – *void*, а принимает на вход массив аргументов командной строки. Класс *isPrime* принимает на вход целое число и возвращает булево значение.

Далее в классе *main* реализуем проверку всех чисел от 2 до 100 (включительно) циклом, в котором мы инициализируем целое число *i*, равное 2, и прибавляет к нему 1 (оператором *++*) до тех пор, пока *i* не превзойдет 100. Каждое число *i* мы проверяем на простоту методом *is.Prime* и в случае, когда этот метод возвращает *true*, мы выводим это число на экран (используя стандартный

поток вывода и метода `println`, который печатает наше число с переносом на следующую строчку.

В классе `isPrime` мы реализуем стандартную оптимизированную проверку числа на простоту – ищем его делители от 1 до корня из этого числа. Для этого возьмем цикл от 2 (минимальный простой делитель) до `Math.sqrt()` с шагом 1. Для каждого предполагаемого делителя `i` посмотрим, чему равен остаток от деления `n` на `i`. Если он равен нулю, то число точно не может быть простым, и мы возвращаемся из метода со значением `false`. Если мы прошлись циклом по всем значениям до квадратного корня из `n` и нигде не ретёрнулись, значит ни на одно из значений `i` наше число не делится, и, следовательно, является простым. В этом случае возвращаемся со значением `true`.

Замечу сразу, что мы не рассматриваем случай, когда `n` у нас меньше 2, так как предполагается, что пока что мы используем этот метод в цикле `мэйна`, а значит, туда не могут попасть значения, выходящие за пределы диапазона `[2, 100]`.

Результат выполнения представлен на рисунке 6:

```
PS C:\Users\user\.vscode\mtuci\.vscode\java> java firstlab.Primes
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
```

Рисунок 6. Результат выполнения программы Primes.java

4. Задание 2 для лабораторной работы:

Создайте программу, которая определяет, является ли введенная строка палиндромом.

Для этой программы создадим класс в каталоге firstlab с именем Palindrome в файле с названием Palindrome.java.

Напишем следующую программу, представленную на рисунке 7:

```

.vscode > java > Firstlab > Palindrome.java > Language Support for Java(TM) by Red Hat > Palindrome
1  package firstlab;
2
3  public class Palindrome {
4      Run main | Debug main | Run | Debug
5      public static void main(String[] args) {
6          for (int i = 0; i < args.length; i++){
7              String s = args[i];
8              System.out.println(isPalindrome(s));
9          }
10     }
11     public static String reverseString(String s){
12         String r_string = "";
13         for (int i = s.length() - 1; i >= 0; i--){
14             r_string = r_string + s.charAt(i);
15         }
16         return r_string;
17     }
18     public static boolean isPalindrome(String s){
19         String s_reverse = reverseString(s);
20         if (s.equals(s_reverse)){
21             return true;
22         }
23         return false;
24     }
25 }

```

Рисунок 7. Код программы *Palindrome.java*

В ней мы также создаем публичные и статичные классы: `main` – принимает массив аргументов командой строки (введенные слова) и ничего не возвращает, `reverseString` - метод, который переворачивает принимаемую на вход строку и возвращает строку задом-наперед, и `isPalindrome` – который сравнивает принимаемую на вход строку `s` с ее реверсивной версией и возвращает `true` или `false`.

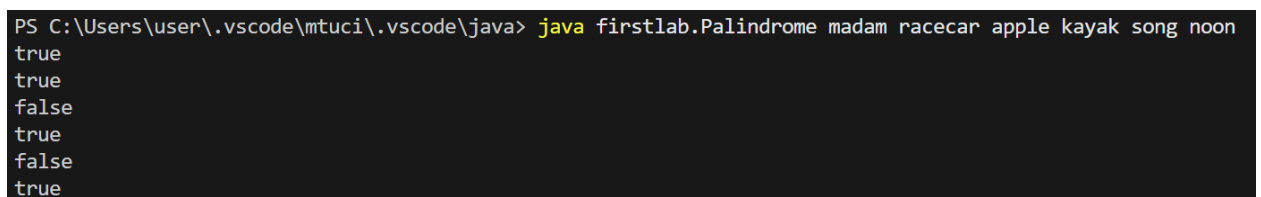
В методе `main` мы проходимся циклом по массиву слов – `args`. Инициализируем переменную-индекс `i`, которая может принимать значения от 0 до длины массива `args` (верхняя граница не включительно, так как у массива длины 1 только один индекс – 0). Проходимся по всем индексам с шагом 1. Внутри цикла объявляем переменную `s` и присваиваем ей значение `i`-того слова

из args. Это слово проверяем на палиндромность методом isPalindrome и полученный результат (true или false) выводим обратно в командную строку.

В методе reverseString создаем новую пустую переменную-строку s_reverse. Дальше как бы производим распаковку в обратном порядке для входной строки, используя цикл, в котором мы локальную переменную индексов i инициализируем со значением s.length – 1 (так как наибольший индекс строки на один меньше ее длины, ведь индексация начинается с 0), затем отнимаем по единице за раз оператором - - до тех пор, пока индекс не станет равным 0. Каждый символ, получаемый из строки методом charAt – методом извлечения символа по индексу, мы прибавляем к s_reverse, то есть в той самой переменной, которая была пустой, по-тихоньку собираются символы изначальной строки, собранные в обратном порядке. После прохождения цикла возвращаем s_reverse.

В методе isPalindrome мы принимаем на вход строку s. Сразу создаем ее перевернутую копию уже написанным методом – reverseString. В условном операторе проверяем, одинаковы ли эти две строки (проверяем методом equals, который применяется к одной из строк и принимает в аргументы другую строку. Если строки одинаковы, то мы проваливаемся внутрь условного оператора, откуда возвращаемся назад в main с результатом true, если разные – то проходим мимо и идем дальше, откуда возвращаемся со значением false.

Результат выполнения представлен на рисунке 8:



```
PS C:\Users\user\.vscode\mtuci\.vscode\java> java firstlab.Palindrome madam racecar apple kayak song noon
true
true
false
true
false
true
```

Рисунок 8. Результат выполнения программы Palindrome.java

Работа загружена на гитхаб по ссылке:

https://github.com/Kateriabova/Riaboava_Kate_BPI2401_IT2.git

5. Ответы на контрольные вопросы:

1. Java является компилируемым или интерпретируемым языком?

И то, и то. Сначала программа компилируется в байт-код, который потом интерпретируется JVM.

2. Что такое JVM и для чего предназначается?

Это Java Virtual Machine, виртуальная машина для выполнения байт-кода; обеспечивает кроссплатформенность и управление памятью.

3. Каков жизненный цикл программы на языке Java?

Сначала происходит JIT-компиляция (Compiler javac) в байт-код (с расширением .class), затем этот байт-код интерпретируется в JVM и выполняется.

4. Какие виды типов данных есть в языке Java?

Примитивные (числовые: целые (int, long, short, byte, char) и нецелые (float и double), булевые) и ссылочные (массивы, классы, строки, интерфейсы)

5. Чем примитивные типы данных отличаются от ссылочных?

Примитивные типы хранят сами объекты в стеках, а ссылочные – лишь ссылки на объекты.

6. Как происходит преобразование примитивных типов в Java?

Можно приводить одни типы к другим, например, нецелые к целым, или расширять (неявно), например, int к long, где можно хранить в два раза больше информации.

7. Что такое байт-код в Java, и почему он важен для платформенной независимости?

Байт-код - промежуточный код для JVM; обеспечивает независимость от операционной системы и железа.

8. Какой тип данных используется для хранения символов в Java? Как представляются символы в памяти?

Используется символ `char` – 16-битный Unicode. Символы хранятся как беззнаковые целые.

9. Что такое литералы в Java? Приведите примеры литералов для разных типов данных.

Литералы - способ записи данных, фиксированное, неизменяемое значение, которое непосредственно записывается в исходный код программы.

Например, литералы для целых чисел: `0b0101` (бинарная запись). Для нецелых чисел: `3.14F` (float), `"string"` для строк, `true` или `false` для boolean.

10. Почему Java считается строго типизированным языком?

Потому что Java требует обязательного объявления типа данных для каждой переменной и проверяет их на этапе компиляции, к тому же все неявные преобразования ограничены.

11. Какие проблемы могут возникнуть при использовании неявного преобразования типов?

Потеря данных и точности (при преобразовании большего типа к меньшему) и ошибки переполнения в том же случае.

3. Вывод:

В ходе работы я освоила взаимодействие с основными элементами каждого языка – условными операторами, циклами, методами. Поработала с основными типами данных и поняла, какие методы и операции с ними допустимы, а какие – нет. Так же в ходе лабораторной удалось на практике

разобраться с частью жизненного цикла программы — компиляцией программы в байт-код.