

**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  
История языка программирования Java. Типы данных

Выполнила: студентка группы БПИ2401  
Садовникова Евгения  
Проверил: Харрасов Камиль Раисович

Москва, 2025

## Оглавление

Цель работы .....	3
Задания .....	3
Ход работы: .....	3
Задание 1 .....	3
Код .....	3
Выходные данные: .....	3
Задание 2 .....	4
Результат: .....	5
Код .....	5
Выходные данные: .....	5
Задание 3 .....	6
Результат: .....	6
Код .....	6
Выходные данные: .....	7
Контрольные вопросы .....	9
Вывод.....	13

**Цель работы:** начать изучить язык программирования Java и типы данных.

## Задания

1. Первая программа «Hello World».
2. Создайте программу, которая находит и выводит все простые числа меньше 100.
3. Создайте программу, которая определяет, является ли введенная строка палиндромом.

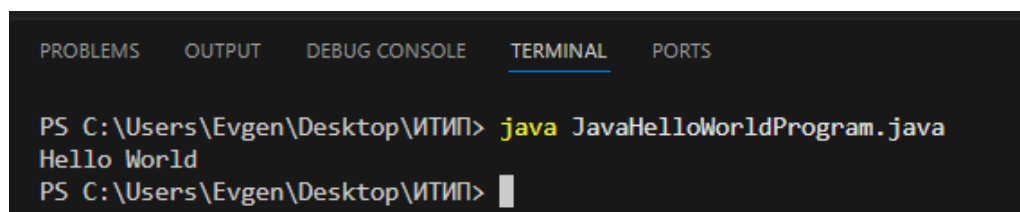
## Ход работы:

### Задание 1

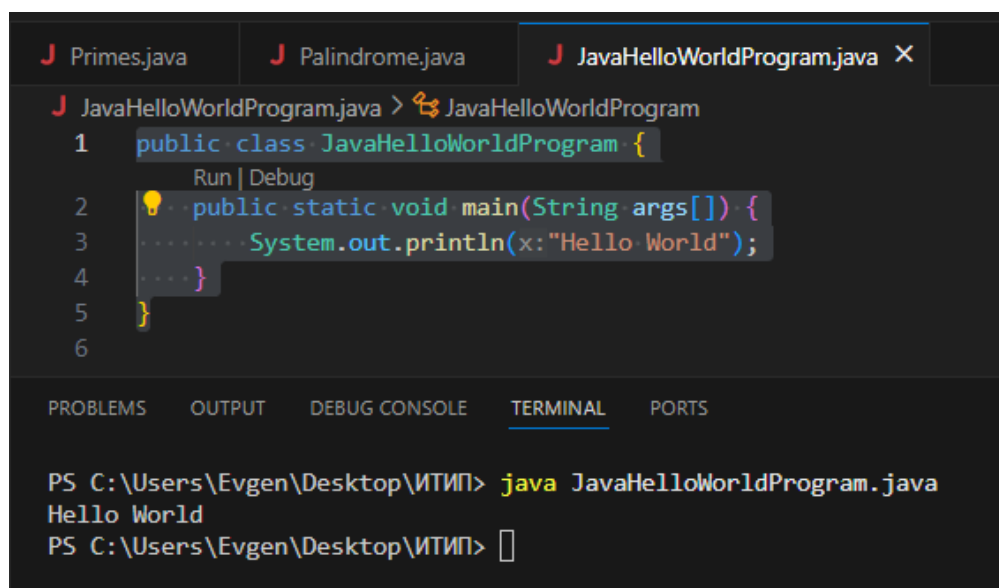
Код из файла JavaHelloWorldProgram.java

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

Выходные данные:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
  
PS C:\Users\Evgen\Desktop\ИТИП> java JavaHelloWorldProgram.java  
Hello World  
PS C:\Users\Evgen\Desktop\ИТИП>
```



```
J Primes.java  J Palindrome.java  J JavaHelloWorldProgram.java X  
J JavaHelloWorldProgram.java > JavaHelloWorldProgram  
1 public class JavaHelloWorldProgram {  
   Run | Debug  
2     public static void main(String args[]) {  
3         System.out.println(x:"Hello World");  
4     }  
5 }  
6  
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
  
PS C:\Users\Evgen\Desktop\ИТИП> java JavaHelloWorldProgram.java  
Hello World  
PS C:\Users\Evgen\Desktop\ИТИП>
```

## Задание 2

Создадим файл с именем Primes.java, в этом файле опишем следующий класс:

```
public class Primes {  
    public static void main(String[ ] args) {  
    }  
}
```

Внутри созданного класса после метода main() опишем метод isPrime (int n), который определяет, является ли аргумент простым числом или нет. Можно предположить, что входное значение n всегда будет больше 2. Полное описание метода будет выглядеть так:

```
public static boolean isPrime(int n) {  
}
```

Данный метод реализован с помощью использования цикла for. Данный цикл перебирает числа, начиная с 2 до (но не включая) корня из n, проверяя, существует ли какое-либо значение, делящееся на n без остатка. Для этого используется оператор остатка «%». Если какая-либо переменная полностью делится на аргумент, срабатывает оператор return false. Если же значение не делится на аргумент без остатка, то это простое число и срабатывает оператор return true.

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

Далее был заполнен метод main() другим циклом, который перебирает числа в диапазоне от 2 до 100 включительно. Выводятся на экран те значения, которые метод isPrime() посчитал простыми.

```
public static void main(String[] args) {  
    System.out.println("is Prime:");  
    for (int i = 1; i <= 100; i++) {  
        if (isPrime(i)) {  
            System.out.printf(i + " ");  
        }  
    }  
}
```

## Результат:

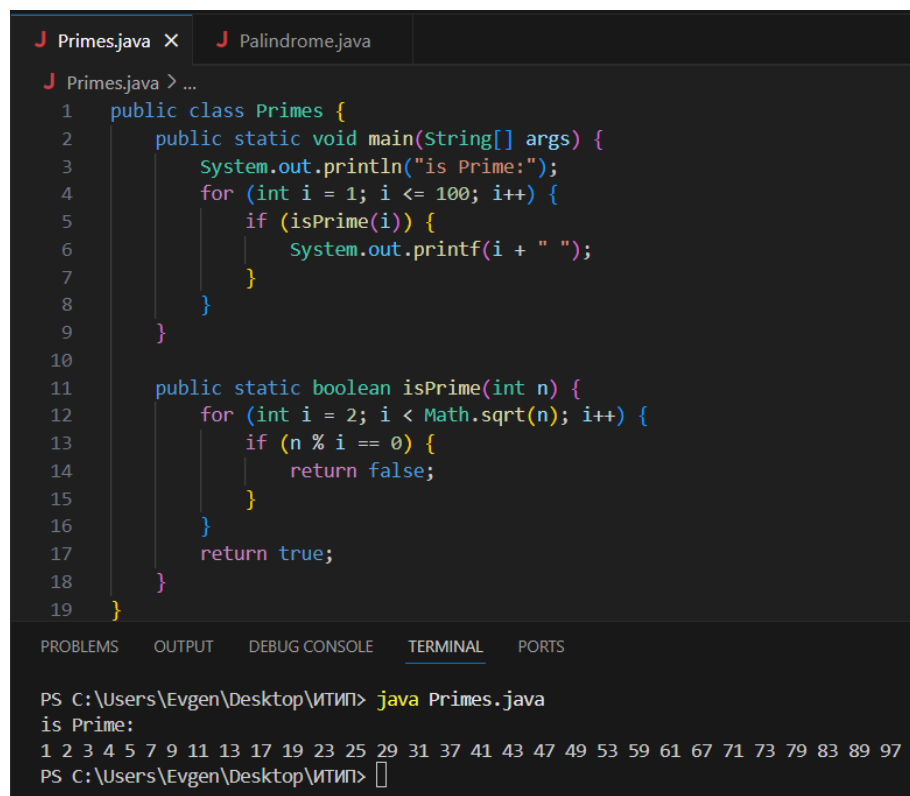
### Код из файла IsPrime.java

```
public class Primes {
    public static void main(String[] args) {
        System.out.println("is Prime:");
        for (int i = 1; i <= 100; i++) {
            if (isPrime(i)) {
                System.out.printf(i + " ");
            }
        }
    }

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

## Выходные данные:

```
PS C:\Users\Evgen\Desktop\ИТИП> java Primes.java
is Prime:
1 2 3 4 5 7 9 11 13 17 19 23 25 29 31 37 41 43 47 49 53 59 61 67 71 73 79 83 89 97
PS C:\Users\Evgen\Desktop\ИТИП> █
```



The screenshot shows an IDE with two tabs: 'Primes.java' and 'Palindrome.java'. The 'Primes.java' tab is active, displaying the following code:

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

Below the code editor, the 'TERMINAL' tab is active, showing the execution output:

```
PS C:\Users\Evgen\Desktop\ИТИП> java Primes.java
is Prime:
1 2 3 4 5 7 9 11 13 17 19 23 25 29 31 37 41 43 47 49 53 59 61 67 71 73 79 83 89 97
PS C:\Users\Evgen\Desktop\ИТИП> █
```

### Задание 3

Для этой программы создадим класс с именем Palindrome в файле с названием Palindrome.java. Воспользуемся следующим кодом:

```
public class Palindrome {
    public static void main(String[ ] args) {
        for (int i = 0; i < args.length; i++ ) {
            String s = args[i];
        }
    }
}
```

Создаим метод, позволяющий полностью изменить символы в строке.

Заголовок метода следующий:

```
public static String reverseString(String s)
```

Реализуем этот метод путем создания локальной переменной, которая является пустой строкой, а затем добавляем символы из входной строки в выходные данные в обратном порядке. Воспользуемся методом length(), который возвращает длину строки, и методом charAt(int index), который возвращает символ по указанному индексу.

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

Далее создадим еще один метод:

```
public static boolean isPalindrome(String s)
```

Этот метод поворачивает слово s, а затем сравнивает с первоначальными данными. Воспользуемся методом equals(Object obj) для проверки значения равенства

```
public static boolean isPalindrome(String s) {
    if (s.equals(reverseString(s))) {
        return true;
    }
    return false;
}
```

**Результат:**

**Код из файла Palindrome.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.printf(s + " - Palindrome\n");
            } else {
                System.out.printf(s + " - Not Palindrome\n");
            }
        }
    }

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

    public static boolean isPalindrome(String s) {
        if (s.equals(reverseString(s))) {
            return true;
        }
        return false;
    }
}

```

### Выходные данные:

```

PS C:\Users\Evgen\Desktop\ИТИП> java Palindrome.java madam racecar apple kayak song noon
madam - Palindrome
racecar - Palindrome
apple - Not Palindrome
kayak - Palindrome
song - Not Palindrome
noon - Palindrome
PS C:\Users\Evgen\Desktop\ИТИП>

```

```
J Primes.java J Palindrome.java X
J Palindrome.java > Palindrome > reverseString(String)
1 public class Palindrome {
    Run | Debug
2     public static void main(String[] args) {
3         for (int i = 0; i < args.length; i++) {
4             String s = args[i];
5             if (isPalindrome(s)) {
6                 System.out.printf(s + " - Palindrome\n");
7             } else {
8                 System.out.printf(s + " - Not Palindrome\n");
9             }
10        }
11    }
12 }
13
14 public static String reverseString(String s) {
15     String local_s = "";
16     for (int i = s.length() - 1; i >= 0; i--) {
17         local_s += s.charAt(i);
18     }
19     return local_s;
20 }
21
22 public static boolean isPalindrome(String s) {
23     if (s.equals(reverseString(s))) {
24         return true;
25     }
26     return false;
27 }
28 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Evgen\Desktop\ИТИП> java Palindrome.java madam racecar apple kayak song noon
madam - Palindrome
racecar - Palindrome
apple - Not Palindrome
kayak - Palindrome
song - Not Palindrome
noon - Palindrome
PS C:\Users\Evgen\Desktop\ИТИП> 
```



## Контрольные вопросы

1. Java является компилируемым или интерпретируемым языком?

### **Java и компилируемый и интерпретируемый язык**

Java разработан так, чтобы быть независимым от аппаратной платформы. Программы на Java **компилируются** в байт-код, который может выполняться на любой платформе с установленной JVM(где **интерпретирует** его построчно или применяет JIT-компиляцию). Это обеспечивает возможность запуска одного и того же кода на различных устройствах без необходимости его модификации.

2. Что такое JVM и для чего предназначается?

**JVM - Виртуальная машина Java** (Java Virtual Machine, JVM).

JVM интерпретирует байт-код в машинный код во время выполнения. Программа, написанная на Java, может быть скомпилирована в байт-код и затем выполнена на любой платформе, где установлена виртуальная машина Java (Java Virtual Machine, JVM), без необходимости в изменениях или повторной компиляции.

3. Каков жизненный цикл программы на языке Java?

1. **Написание кода** в текстовом редакторе или интегрированной среде разработки (IDE) с использованием синтаксиса Java.

2. **Компиляция**: исходный код (.java) компилируется в байткод (.class) с помощью компилятора javac. Байт-код является платформенно независимым и может выполняться на любой платформе, где установлена JVM.

3. **Интерпретация**: байт-код выполняется виртуальной машиной Java. JVM интерпретирует байт-код в машинный код во время выполнения. Для ускорения процесса интерпретации байт-кода в машинный код был введен механизм JIT-компиляции. Во время выполнения программы JIT-компилятор анализирует часто исполняемые участки кода и компилирует их в нативный код для конкретной платформы. Этот нативный код затем кешируется, чтобы

его можно было повторно использовать при последующих вызовах, при этом увеличивается количество потребляемой памяти.

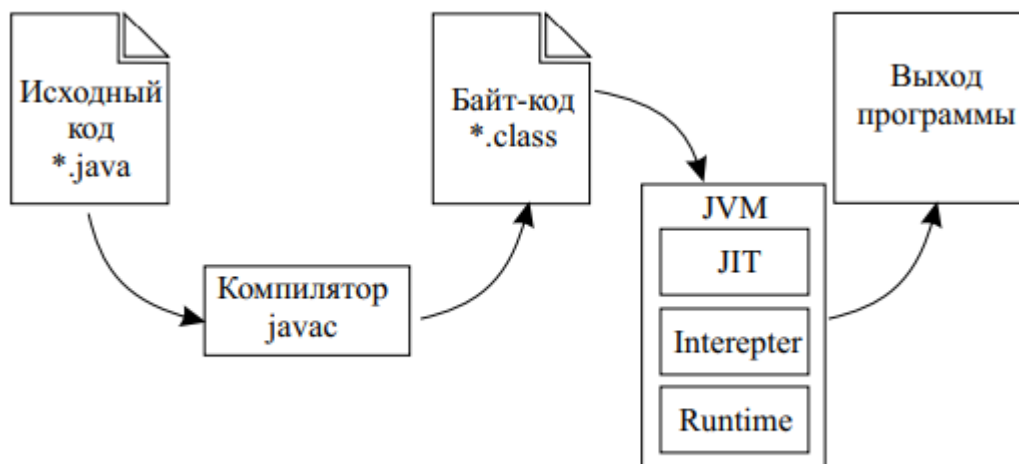
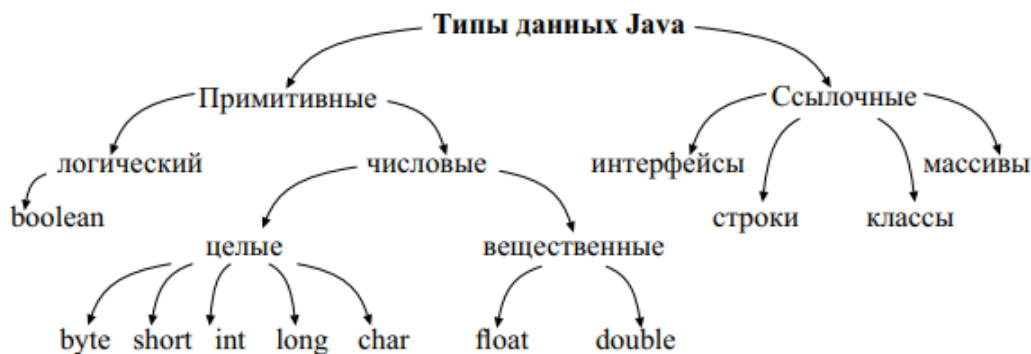


Рис. 1.1. Жизненный цикл программы на языке Java

4. Какие виды типов данных есть в языке Java?

В Java существует несколько типов данных, которые можно разделить на две основные категории: **примитивные типы** и **ссылочные типы**.



5. Чем примитивные типы данных отличаются от ссылочных?

**Ссылочные** типы представляют собой объекты и массивы. Они хранят ссылки на места в памяти, где находятся данные. К основным ссылочным типам относятся классы, интерфейсы, массивы, перечисления.

**Примитивные** типы данных хранят свои значения непосредственно в памяти. Они занимают фиксированное количество байтов (например, `int` занимает 4 байта). Примитивные переменные хранятся на стеке, что обеспечивает быструю доступность и освобождение памяти после завершения работы.

## 6. Как происходит преобразование примитивных типов в Java?

В Java существует несколько способов преобразования между примитивными типами данных. Это называется приведением (casting). В зависимости от направления приведения различают: **явное и неявное** преобразование типов. При неявном приведении меньший тип данных преобразуется в больший без потери информации. Например:

```
• byte → short;  
• short → int;  
• int → long;  
• float → double. 1.  
int a = 10; 2. long b = a; // Автоматически преобразуем int в long
```

Когда нужно преобразовать больший тип данных в меньший, необходимо использовать явное приведение, так как может произойти потеря точности

```
• int → short;  
• long → int;  
• double → float.  
long c = 1000000L;  
int d = (int)c; // Явное приведение long к int  
double e = 123.456;  
float f = (float)e; // Приведение double к float
```

Целые числа могут быть приведены к числам с плавающей точкой неявно. Для приведения чисел с плавающей точкой к целым числам используется явное приведение, при котором дробная часть отбрасывается. Логический тип `boolean` не может быть приведен ни к одному другому типу данных и наоборот. Попытка сделать такое приведение вызовет ошибку компиляции. При выполнении арифметических операции над значениями разных типов данных результат будет иметь тот же тип, который является наибольшим среди участвующих в выражении типов.

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

**Байт-код** в Java — это промежуточный код, в который компилируются Java-программы. Он независим от платформы и выполняется виртуальной машиной Java (JVM). Это позволяет запускать один и тот же код на разных

устройствах и в разных операционных системах, следуя принципу «пиши один раз, выполняй везде» (WORA).

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

В Java для хранения символов используется тип данных **char**. Символы в памяти представляются в формате **Unicode** (стандарт кодирования символов).

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

**Литерал** — это константное значение, записанное прямо в коде программы, которое не меняется во время выполнения. Литералы используются для инициализации переменных.

10. Почему Java считается строго типизированным языком?

Java считается строго типизированным языком, потому что каждая переменная и выражение имеют определённый тип, который нельзя менять произвольно, и компилятор строго проверяет соответствие типов во время компиляции.

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

При неявном преобразовании типов могут возникнуть потеря точности, неожиданные результаты арифметики, переполнение или несовместимость типов (например, для `boolean`).

## **Вывод**

В ходе лабораторной работы было начато изучение язык программирования Java. В ходе чего было выпалено 2 задания: создана программа, которая находит и выводит все простые числа меньше 100, и программа, которая определяет, является ли введенная строка палиндромом.