



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____ «Информатика и системы управления»

КАФЕДРА _____ «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 0
по курсу «Языки и методы программирования»
«Приобретение опыта работы с VDS-сервером под управлением
ОС Linux»

Студент группы ИУ9-22Б Лавров Р. Д.

Преподаватель Посевин Д. П.

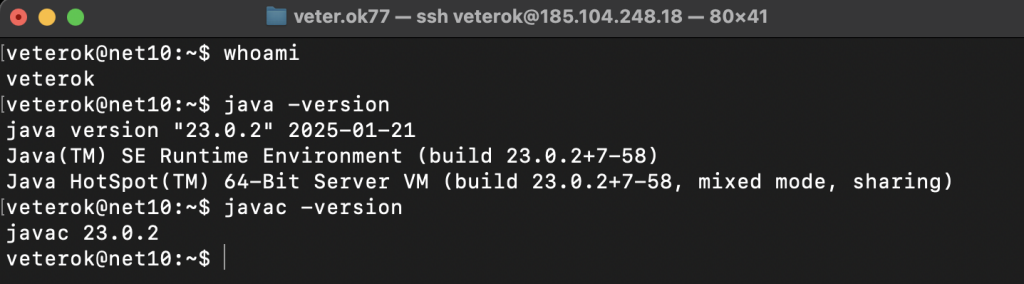
Москва 2025

1 Задание

1. По инструкции настроить учетную запись на удаленном VDS-сервере.
2. В своей учетной записи настроить окружение для удобного запуска компилятора Java.
3. Запустить любой простейший пример приведенный на лекции 1 или на лабораторной работе 1.
4. Запустить веб-сервер из примера на своем порту. Каждый студент резервирует свой собственный порт. Рекомендуется нумерацию портов использовать по следующему правилу: резервировать номер порта "800n где n - это порядковый номер студента в Электронном университете, другимим словами, если у студента в Электронном университете порядковый номер 3, то порт веб-сервера данного студента будет 8003.
5. Научиться запускать веб-сервер через утилиту screen.
6. Объединить простейший пример с веб-сервером, т.е. результаты работы программы простейшего прмера выводить через веб-сервер.
7. Запустить работу веб-сервера, выполненного в пункте 6 на своем порту.
8. Как только пункты 1-7 выполнены, выслать строго в телеграм-канал в комментарий к условию данной лабораторной работы: URL-страницы результата работы вашего веб-сервера.

2 Результаты

Результат создания учётной записи и настройки окружения



```
veter.ok77 — ssh veterok@185.104.248.18 — 80x41
veterok@net10:~$ whoami
veterok
veterok@net10:~$ java -version
java version "23.0.2" 2025-01-21
Java(TM) SE Runtime Environment (build 23.0.2+7-58)
Java HotSpot(TM) 64-Bit Server VM (build 23.0.2+7-58, mixed mode, sharing)
veterok@net10:~$ javac -version
javac 23.0.2
veterok@net10:~$ |
```

Рис. 1 — Результат

Исходный код простейшей программы приведенный на лекциях ??– ??– ??.

Листинг 1 — Пример реализации класса

```
1 public class Dog {
2     private String name;
3     int age;
4     String color;
5     int amoutoftails;
6
7     public Dog() {
8         System.out.println("The dog was born");
9         this.name = "no name";
10    }
11
12    public Dog(String name) {
13        System.out.println("The dog was born"+name);
14        this.name = name;
15    }
16
17    public Dog(String name, int age) {
18        System.out.println("The dog was born"+name);
19        this.name = name;
20        this.age = age;
21    }
22    void parse() {}
23    void prog() {}
24    void getName(String aaa) {
25        System.out.println("The dog was born"+aaa);
26    }
27    void changeName(String bbb){
28        this.name = bbb;
29    }
30
31    String getName(){
32        return "("+this.name+";";
33    }
34 }
```

Листинг 2 — Пример создания экземпляра класса

```
1 public class Main {
2     public static void main(String[] args) {
3         Dog a = new Dog("Bobik",40000);
4         Dog b = new Dog("Gychka");
5         System.out.println("->" + a.getName());
6         a.changeName("Innokentiy 2");
7         System.out.println("1->" + a.getName());
8         a.getName("ssxcxdfvrsvdr");
9         System.out.println("age->" + a.age);
10        System.out.println("color->" + a.color);
11        System.out.println("amoutoftails->" + a.amoutoftails);
12    }
13 }
```

Листинг 3 — Сервер из примера с модификацией

```
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStreamReader;
4 import java.io.PrintWriter;
5 import java.net.ServerSocket;
6 import java.net.Socket;
7 import java.nio.charset.StandardCharsets;
8
9 public class HttpServer {
10
11     public static void main(String[] args) {
12         try (ServerSocket serverSocket = new ServerSocket(8294)) {
13             System.out.println("Server started!");
14             Factorial fact = new Factorial();
15             while (true) {
16                 Socket socket = serverSocket.accept();
17                 System.out.println("Client connected!");
18                 BufferedReader input = new BufferedReader(
19                     new InputStreamReader(socket.getInputStream(),
20                     StandardCharsets.UTF_8));
21                 try (PrintWriter output = new PrintWriter(socket.getOutputStream())) {
22                     String requestLine = input.readLine();
23                     String response = "<p>Incorrect request</p>";
24                     output.println("HTTP/1.1 200 OK");
25                     System.out.println(requestLine);
26                     if (requestLine != null && requestLine.startsWith("GET ")) {
27                         String[] parts = requestLine.split(" ");
28                         if (parts.length > 1) {
29                             String path = parts[1].substring(1);
30                             if (isNumber(path) && path != "") {
31                                 int number = Integer.parseInt(path);
32                                 response = "<p>fact(" + number + ") = " + fact.count(number) + "</p>";
33                             }
34                         }
35                     }
36                     output.println("HTTP/1.1 200 OK");
37                     output.println("Content-Type: text/html; charset=utf-8");
38                     output.println();
39                     output.println(response);
40                     output.flush();
41
42                     System.out.println("Client disconnected!");
43                 }
44             }
45         } catch (IOException ex) {
46             ex.printStackTrace();
47         }
48     }
49     private static boolean isNumber(String s) {
50         for (int i = 0; i < s.length(); i++) {
51             if (!Character.isDigit(s.charAt(i))) {
52                 return false;
53             }
54         }
55         return true;
56     }
57 }
```

```
veterok@net10:~/lab0$ ls
Dog.java Factorial.java HttpServer.java Main.java
veterok@net10:~/lab0$ javac Main.java
veterok@net10:~/lab0$ java Main.java
The dog was bornBobik
The dog was bornGychka
->(Bobik)
1->(Innokentiy 2)
The dog was bornssxcxdfvrsvdr
age->40000
color->null
amoutoftails->0
veterok@net10:~/lab0$ |
```

Рис. 2 — Результат запуска Main.java

```
veterok@net10:~/lab0$ ls
Dog.class Dog.java Factorial.java HttpServer.java Main.class Main.java
veterok@net10:~/lab0$ javac HttpServer.java
veterok@net10:~/lab0$ screen -S veterok-server
[detached from 265599.veterok-server]
veterok@net10:~/lab0$ screen -ls
There is a screen on:
      265599.veterok-server      (21.02.2025 14:35:31)      (Detached)
1 Socket in /run/screen/S-veterok.
veterok@net10:~/lab0$ |
```

Рис. 3 — Создание screen сессии

```
veterok@net10:~/lab0$ java HttpServer.java
Server started!
Client connected!
GET /12 HTTP/1.1
Client disconnected!
Client connected!
GET /3 HTTP/1.1
Client disconnected!
Client connected!
GET /6 HTTP/1.1
Client disconnected!
```

Рис. 4 — Логи работы веб-сервера

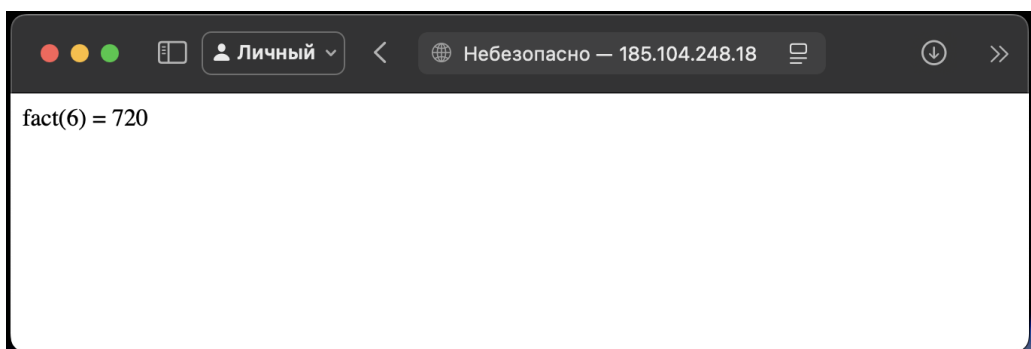


Рис. 5 — Результат работы веб-сервера