

Министерство науки и высшего образования РФ
ФГБОУ ВО «Омский государственный технический университет»
Кафедра «Автоматизированные системы обработки информации и
управления»

ОТЧЁТ

по Лабораторной работе №5

По дисциплине: Объектно-ориентированное программирование
студента Охтеня Валерия Алексеевича группы ПИН-212

Пояснительная записка

Шифр работы От-2068998-43-ПИН-212-2 ПЗ

Специальность 09.03.04

Старший преподаватель

А.А. Кабанов

Студент

В.А. Охтеня

Омск 2022

Задание

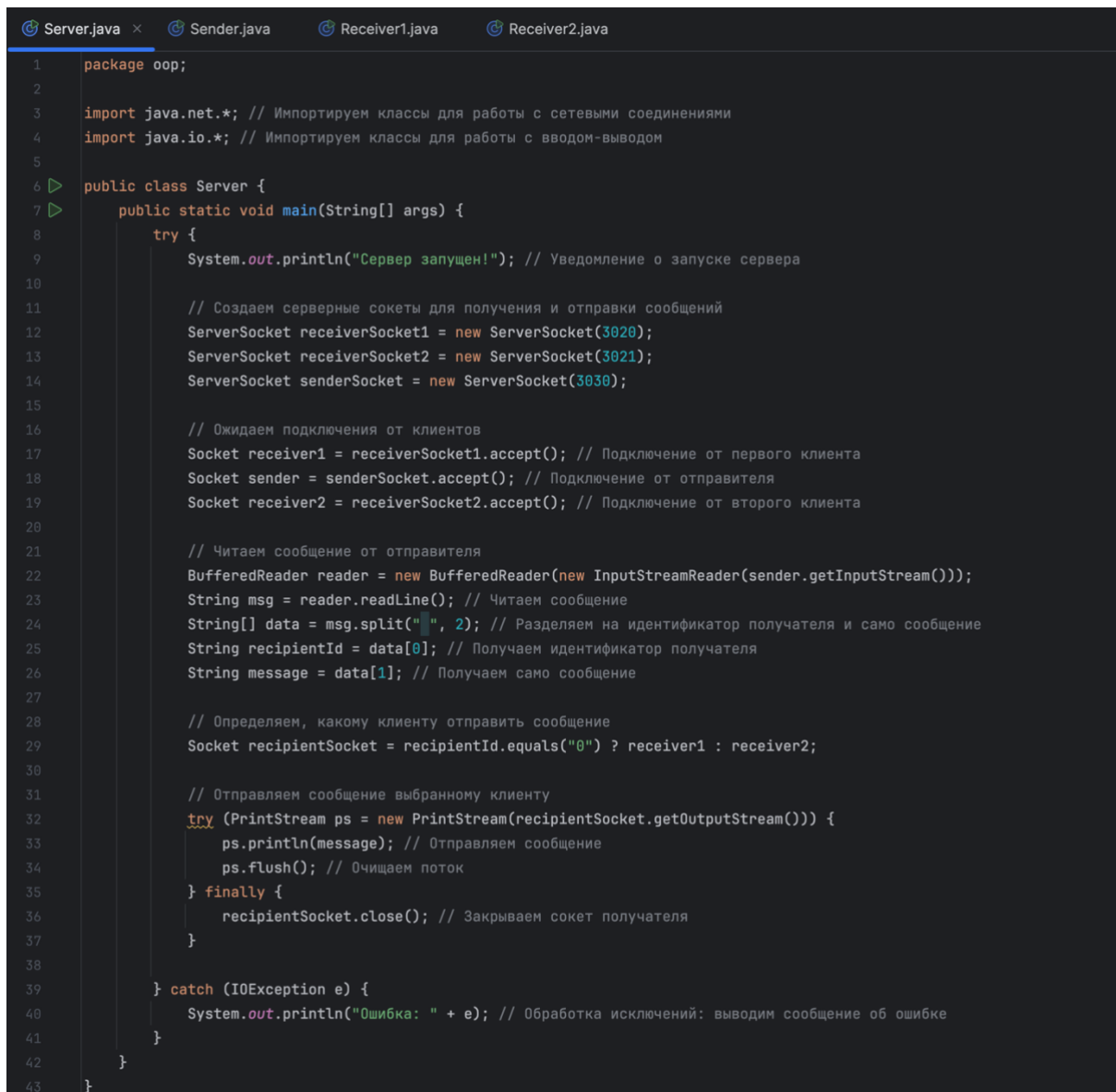
Клиент посылает через сервер сообщение другому клиенту, выбранному из списка.

Цель работы

Реализовать сетевое приложение, использующее протоколы стека TCP/IP.

Ход работы

Написал код программы (рис 1-5).



```
1 package oop;
2
3 import java.net.*; // Импортируем классы для работы с сетевыми соединениями
4 import java.io.*; // Импортируем классы для работы с вводом-выводом
5
6 public class Server {
7     public static void main(String[] args) {
8         try {
9             System.out.println("Сервер запущен!"); // Уведомление о запуске сервера
10
11             // Создаем серверные сокеты для получения и отправки сообщений
12             ServerSocket receiverSocket1 = new ServerSocket(3020);
13             ServerSocket receiverSocket2 = new ServerSocket(3021);
14             ServerSocket senderSocket = new ServerSocket(3030);
15
16             // Ожидаем подключения от клиентов
17             Socket receiver1 = receiverSocket1.accept(); // Подключение от первого клиента
18             Socket sender = senderSocket.accept(); // Подключение от отправителя
19             Socket receiver2 = receiverSocket2.accept(); // Подключение от второго клиента
20
21             // Читаем сообщение от отправителя
22             BufferedReader reader = new BufferedReader(new InputStreamReader(sender.getInputStream()));
23             String msg = reader.readLine(); // Читаем сообщение
24             String[] data = msg.split(" ", 2); // Разделяем на идентификатор получателя и само сообщение
25             String recipientId = data[0]; // Получаем идентификатор получателя
26             String message = data[1]; // Получаем само сообщение
27
28             // Определяем, какому клиенту отправить сообщение
29             Socket recipientSocket = recipientId.equals("0") ? receiver1 : receiver2;
30
31             // Отправляем сообщение выбранному клиенту
32             try (PrintStream ps = new PrintStream(recipientSocket.getOutputStream())) {
33                 ps.println(message); // Отправляем сообщение
34                 ps.flush(); // Очищаем поток
35             } finally {
36                 recipientSocket.close(); // Закрываем сокет получателя
37             }
38
39         } catch (IOException e) {
40             System.out.println("Ошибка: " + e); // Обработка исключений: выводим сообщение об ошибке
41         }
42     }
43 }
```

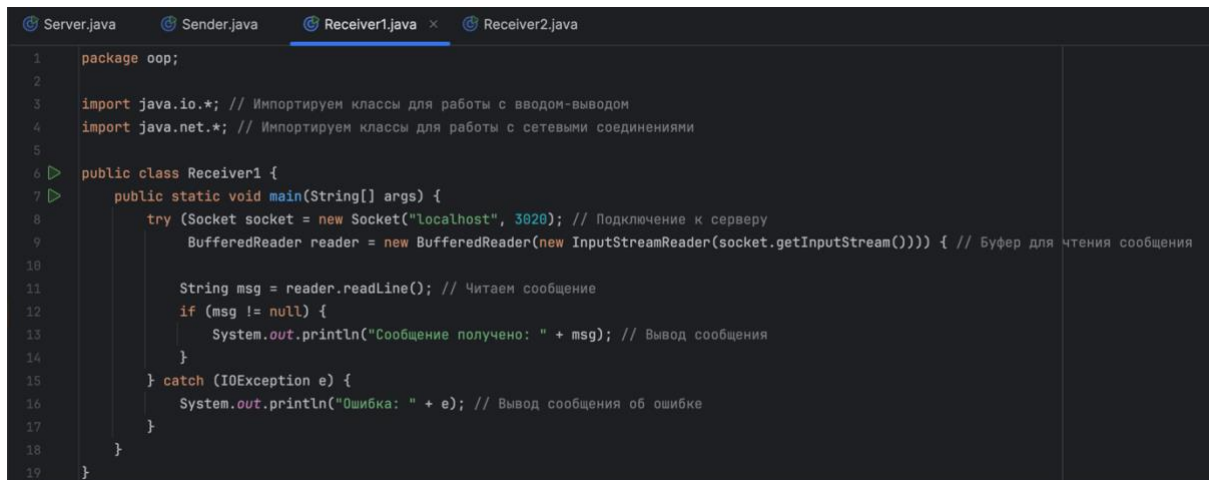
Рисунок 1 – Код сервер

```
Server.java Sender.java x Receiver1.java Receiver2.java
1 package oop;
2
3 import java.util.Scanner; // Импортируем класс для работы с вводом данных
4 import java.io.*; // Импортируем классы для работы с вводом-выводом
5 import java.net.*; // Импортируем классы для работы с сетевыми соединениями
6 import java.util.ArrayList; // Импортируем класс для работы с динамическими массивами
7
8 public class Sender {
9     public static void main(String[] args) {
10         Scanner scan = new Scanner(System.in); // Создаем объект Scanner для считывания ввода с клавиатуры
11         Socket socket = null; // Сокет для подключения к серверу
12
13         try {
14             // Список доступных получателей
15             ArrayList<String> receiverNames = new ArrayList<>();
16             receiverNames.add("Receiver1");
17             receiverNames.add("Receiver2");
18
19             // Вывод доступных получателей
20             System.out.println("Доступные получатели: ");
21             for (int i = 0; i < receiverNames.size(); i++) {
22                 System.out.println(i + " - " + receiverNames.get(i));
23             }
24
25             // Запрашиваем у пользователя номер получателя
26             System.out.print("Введите номер получателя: ");
27             String receiverIndex = scan.nextLine(); // Получаем номер получателя
28
29             // Запрашиваем у пользователя сообщение
30             System.out.print("Введите сообщение: ");
31             String message = scan.nextLine(); // Получаем сообщение
32
33             // Устанавливаем соединение с сервером
34             socket = new Socket(InetAddress.getLocalHost(), 3030);
35             System.out.println("Сообщение отправлено!"); // Уведомление об отправке сообщения
36
37             // Отправляем сообщение
38             try (PrintStream ps = new PrintStream(socket.getOutputStream())) {
39                 ps.println(receiverIndex + " " + message); // Форматируем строку с номером получателя и сообщением
40                 ps.flush(); // Очищаем поток
41             }
42         } catch (IOException e) {
43             System.out.println("Ошибка: " + e); // Вывод сообщения об ошибке
44         } finally {
45             // Закрываем сокеты, если они были успешно открыты
```

Рисунок 2 – Код отправитель (Часть 1)

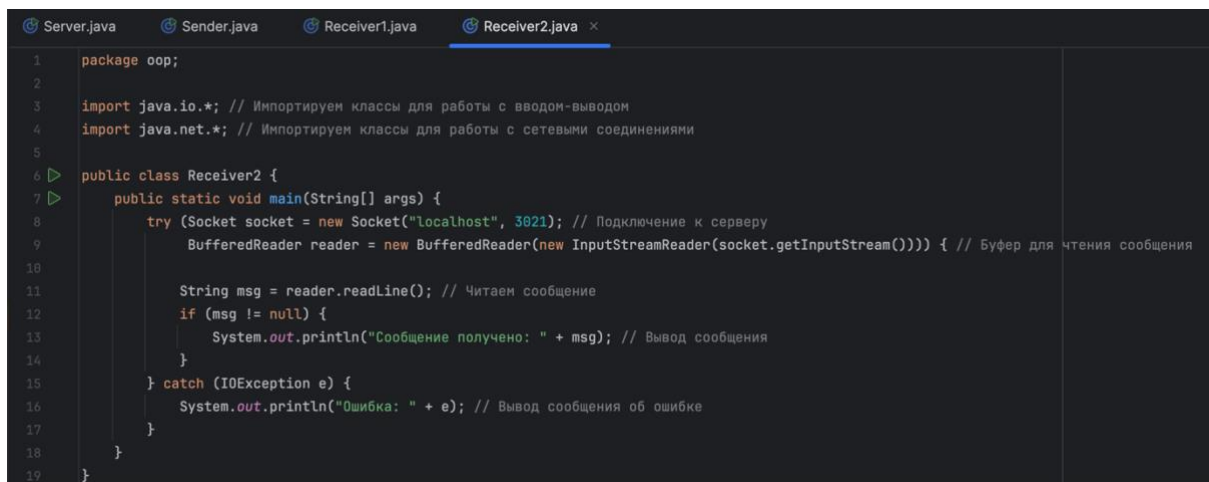
```
46         if (socket != null) {
47             try {
48                 socket.close(); // Отключение от сервера
49             } catch (IOException e) {
50                 System.out.println("Ошибка при закрытии сокета: " + e);
51             }
52         }
53         scan.close(); // Закрываем Scanner
54     }
55 }
56 }
```

Рисунок 3 – Код отправитель (Часть 2)



```
1 package oop;
2
3 import java.io.*; // Импортируем классы для работы с вводом-выводом
4 import java.net.*; // Импортируем классы для работы с сетевыми соединениями
5
6 public class Receiver1 {
7     public static void main(String[] args) {
8         try (Socket socket = new Socket("localhost", 3020); // Подключение к серверу
9             BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream())) { // Буфер для чтения сообщения
10
11             String msg = reader.readLine(); // Читаем сообщение
12             if (msg != null) {
13                 System.out.println("Сообщение получено: " + msg); // Вывод сообщения
14             }
15         } catch (IOException e) {
16             System.out.println("Ошибка: " + e); // Вывод сообщения об ошибке
17         }
18     }
19 }
```

Рисунок 4 – Код получатель1



```
1 package oop;
2
3 import java.io.*; // Импортируем классы для работы с вводом-выводом
4 import java.net.*; // Импортируем классы для работы с сетевыми соединениями
5
6 public class Receiver2 {
7     public static void main(String[] args) {
8         try (Socket socket = new Socket("localhost", 3021); // Подключение к серверу
9             BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream())) { // Буфер для чтения сообщения
10
11             String msg = reader.readLine(); // Читаем сообщение
12             if (msg != null) {
13                 System.out.println("Сообщение получено: " + msg); // Вывод сообщения
14             }
15         } catch (IOException e) {
16             System.out.println("Ошибка: " + e); // Вывод сообщения об ошибке
17         }
18     }
19 }
```

Рисунок 5 – Код получатель2

Далее запустил и протестировал программу (рис 6-11).

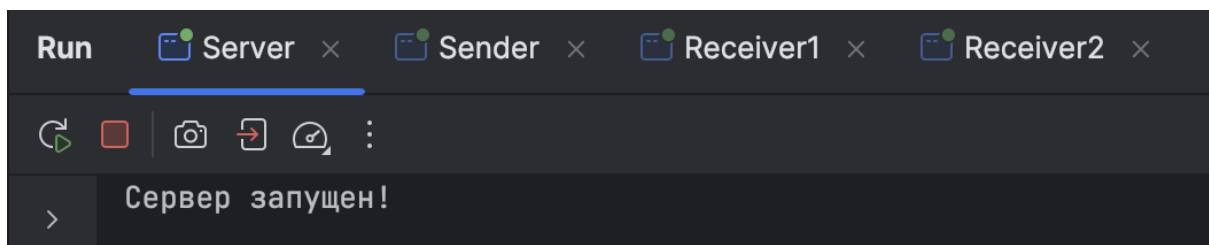


Рисунок 6 – Тест1 (Сервер запущен)

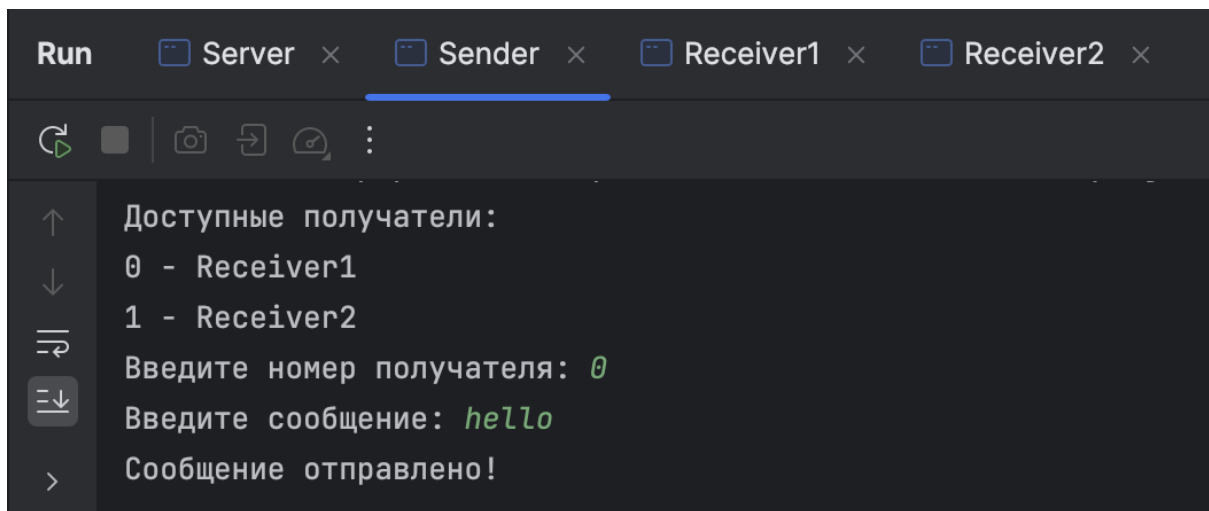


Рисунок 7 – Тест1 (Отправитель отправляет сообщение к Получатель1)

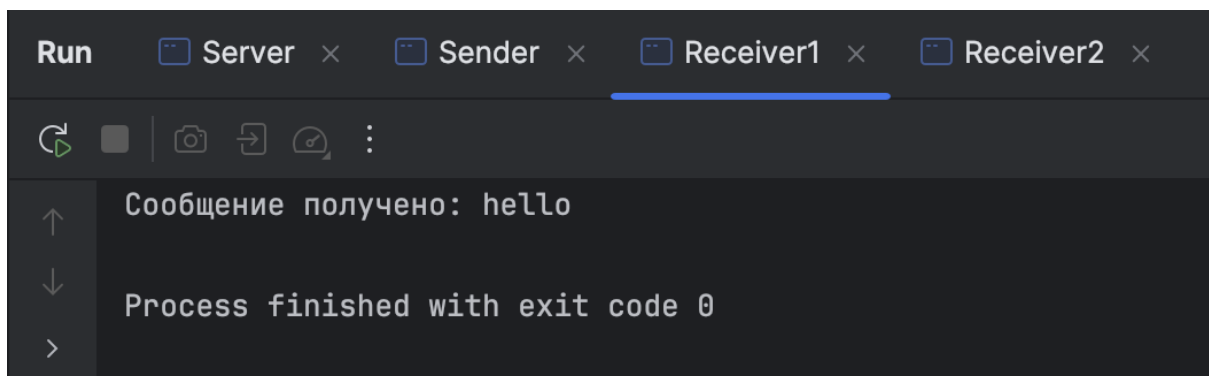


Рисунок 8 – Тест1 (Получатель1 получил сообщение от Отправитель)

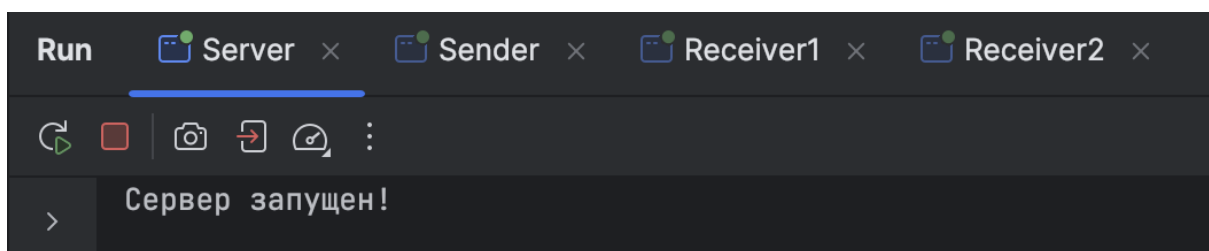


Рисунок 9 – Тест2 (Сервер запущен)

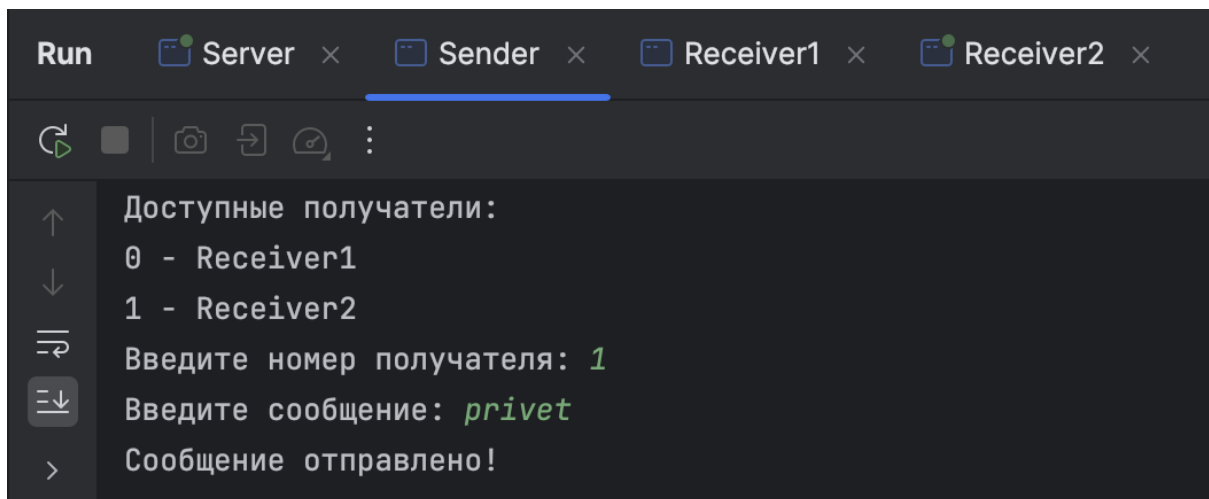


Рисунок 10 – Тест2 (Отправитель отправляет сообщение к Получатель2)

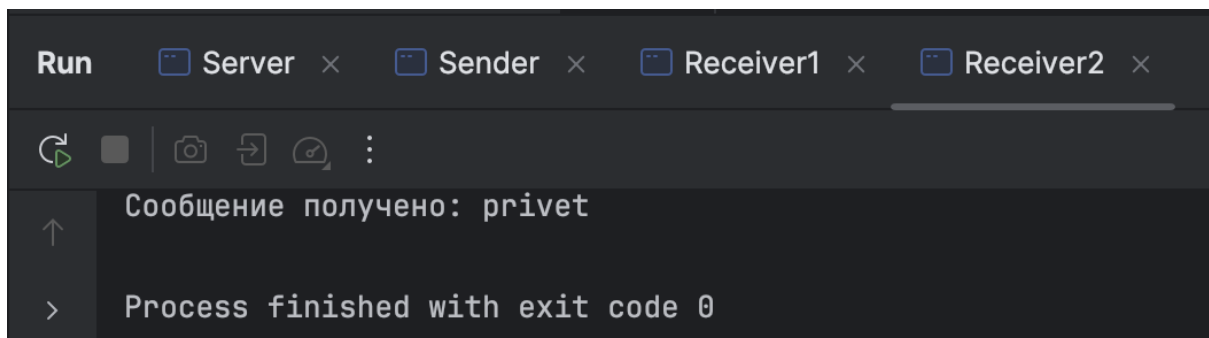


Рисунок 11 – Тест2 (Получатель2 получил сообщение от Отправитель)

Вывод

Реализовать сетевое приложение, использующее протоколы стека TCP/IP.