Глебов Виктор
СБС-801-О-01

Криптографические протоколы
Лабораторная работа № 4
«Криптографические протоколы аутентификации»
Отчет по лабораторной работе:

8. Напишите приложение с графическим интерфейсом, реализующее криптопротокол **Neuman-Stubblebine**.

Криптопротокол 2.6.
Neuman-Stubblebine

Алиса и Трент используют общий секретный ключ K_A. Боб и Трент используют общий секретный ключ K_B.

1. Алиса объединяет свое имя и случайное число, и отправляет созданное сообщение Бобу: A, R_A.

2. Боб объединяет имя Алисы, ее случайное число и метку времени, шифрует созданное сообщение общим с Трентом ключом и результат посылает Тренту, добавляя свое имя и новое случайное число: $B, R_B, E_{K_B}(A, R_A, T_B)$.

3. Трент генерирует случайный сеансовый ключ. Затем он создает два сообщения. Первое включает имя Боба, случайное число Алисы, случайный сеансовый ключ, метку времени и шифруется ключом, общим для Трента и Алисы. Второе состоит из имени Алисы, сеансового ключа, метки времени и шифруется ключом, общим для Трента и Боба. Трент посылает оба сообщения Алисе вместе со случайным числом Боба: $E_{K_A}(B, R_A, K, T_B), E_{K_B}(A, K, T_B), R_B$

4. Алиса расшифровывает сообщение, зашифрованное ее ключом, извлекает K и убеждается, что R_A совпадает со значением, отправленным на этапе 1. Алиса посылает Бобу два сообщения. Одним является сообщение Трента, зашифрованное ключом Боба. Второе - это R_B, зашифрованное сеансовым ключом: $E_{K_B}(A, K, T_B), E_K(R_B)$.

5. Боб расшифровывает сообщение, зашифрованное его ключом, извлекает K и убеждается, что значения T_B и R_B те же, что и отправленные на этапе 2.

6. Алиса и Боб шифруют свои сообщения, используя сеансовый ключ K.

Описание работы программы:

Запускается Трент, Алиса, Боб -> Алиса создает сервер для Боба, Боб подключается к серверу Алисы и одновременно к Тренту, начинается обмен сообщениями и реализация протокола. Трент автоматически не убивается, чтобы можно было увидеть его логи.

Трент функции шифрования (дешифрование по аналогии):

```java
    public static String Encryption (String Text, int Key){
        String code = "";
        char m;
        for(int i = 0; i < Text.length(); i++){
            char t = Text.charAt(i);
            int n = t^Key;
            m = (char)Integer.parseInt(String.valueOf(n));
            code += m;
        }
        return code;
    }
    public static int Encryption_int (int Text, int Key){
        int code = 0;
        code = Text ^ Key;
        return code;
    }
    public static String Decryption (String Text, int Key){...}
    public static int Decryption_int (int Text, int Key){...}
}
```

```java
public class Main {
    private static ServerSocket ss = null;
    private static ServerSocket ss2 = null;
    private static Socket socket = null;
    private static Socket socket2 = null;
    private static int Port = 8560; // Порт Боба
    private static int Port2 = 8561; // Порт Алисы
    private static int Ka = (int)(Math.random()*((999-100)+1))+100; // Общий ключ Алисы и Трента
    private static int Kb = (int)(Math.random()*((999-100)+1))+100; // Общий ключ Боба и Трента
    private static int Session_Key = (int)(Math.random()*((3300-1300)+1))+1300; // Сеансовый ключ
    public String internal_username = "Trent";
```

```
22    public static void main(String[] args) {
23        try {
24            //Производим начальные вычисления
25            System.out.println("Ka: " + Ka);
26            System.out.println("Kb: " + Kb);
27            System.out.println("Session key: " + Session_Key);
28
29            ss = new ServerSocket(Port); // создаем сокет сервера и привязываем его к порту Боба
30            ss2 = new ServerSocket(Port2); // создаем сокет сервера и привязываем его к порту Алисы
31            System.out.println("Waiting Bob connection...");
32            socket = ss.accept(); // заставляем сервер ждать подключений и выводим сообщение когда кто-то связался с сервером
33            System.out.println("Bob has been connected to Trent!");
34
35            // Конвертируем потоки в другой тип. Берем входной и выходной потоки сокета, теперь можем получать и отсылать данные клиенту.
36            InputStream sin = socket.getInputStream();
37            OutputStream sout = socket.getOutputStream();
38            DataInputStream in = new DataInputStream(sin);
39            DataOutputStream out = new DataOutputStream(sout);
40
41            out.writeInt(Kb); //Отправляем Бобу общий с ним ключ
42
43            // Принимаем сообщение от Боба
44            String Bob_name = in.readUTF(); // Имя Боба
45            int Rb = in.readInt(); // Сгенерированное бобом число
46            String Alice_name = in.readUTF(); // Имя Алисы (шифр)
47            int Ra = in.readInt(); // Сгенерированное Алисой число (шифр)
48            int currenttime = in.readInt(); // Метка времени Боба (шифр)
49            System.out.println("Получено сообщение от Боба в формате [B,Rb,E(A,Ra,T)]");
50            System.out.println(Bob_name + " " + Rb + " (" + Alice_name + "," + Ra + "," + currenttime + ")");
51            System.out.println("Расшифровываем...");
52            System.out.println(Bob_name + " " + Rb + " (" + Decryption(Alice_name,Kb) + "," + Decryption_int(Ra,Kb) + "," + Decryption_int(currenttime,Kb) + ")
53
54            //Подключаем Алису
55            System.out.println("Waiting Alice connection...");
56            socket2 = ss2.accept(); // заставляем сервер ждать подключений и выводим сообщение когда кто-то связался с сервером
57            System.out.println("Alice has been connected!");
58
59            // Берем входной и выходной потоки сокета, теперь можем получать и отсылать данные клиенту.
60            OutputStream sout2 = socket2.getOutputStream();
61            DataOutputStream out2 = new DataOutputStream(sout2);
62
63            // Отправляем общий ключ Алисы и Трента ей
64            out2.writeInt(Ka);
65
```

```
            // Формируем первое сообщение для Алисы
            out2.writeUTF(Encryption(Bob_name,Ka));
            out2.writeInt(Encryption_int(Decryption_int(Ra,Kb),Ka));
            out2.writeInt(Encryption_int(Session_Key,Ka));
            out2.writeInt(Encryption_int(Decryption_int(currenttime,Kb),Ka));

            // Формируем Второе сообщение для Алисы (шифр)
            out2.writeUTF(Alice_name); // Уже зашифрованно Бобом (Kb)
            out2.writeInt(Encryption_int(Session_Key,Kb));
            out2.writeInt(currenttime); // Уже зашифрованно Бобом (Kb)

            // Случайное число Боба
            out2.writeInt(Rb);

            out.flush(); // заставляем поток закончить передачу данных.
            out2.flush(); // заставляем поток закончить передачу данных.

            String choice = "";
            while(true) {
                System.out.println("Трент завершил начальную инициализацию, закрыть программу? [Y/N]: ");
                Scanner sc = new Scanner(System.in);
                choice = sc.next();
                if (choice.equals("Y")) { System.exit( status: 9); }
                choice = "";
            }
        } catch(Exception x) { x.printStackTrace(); }
```

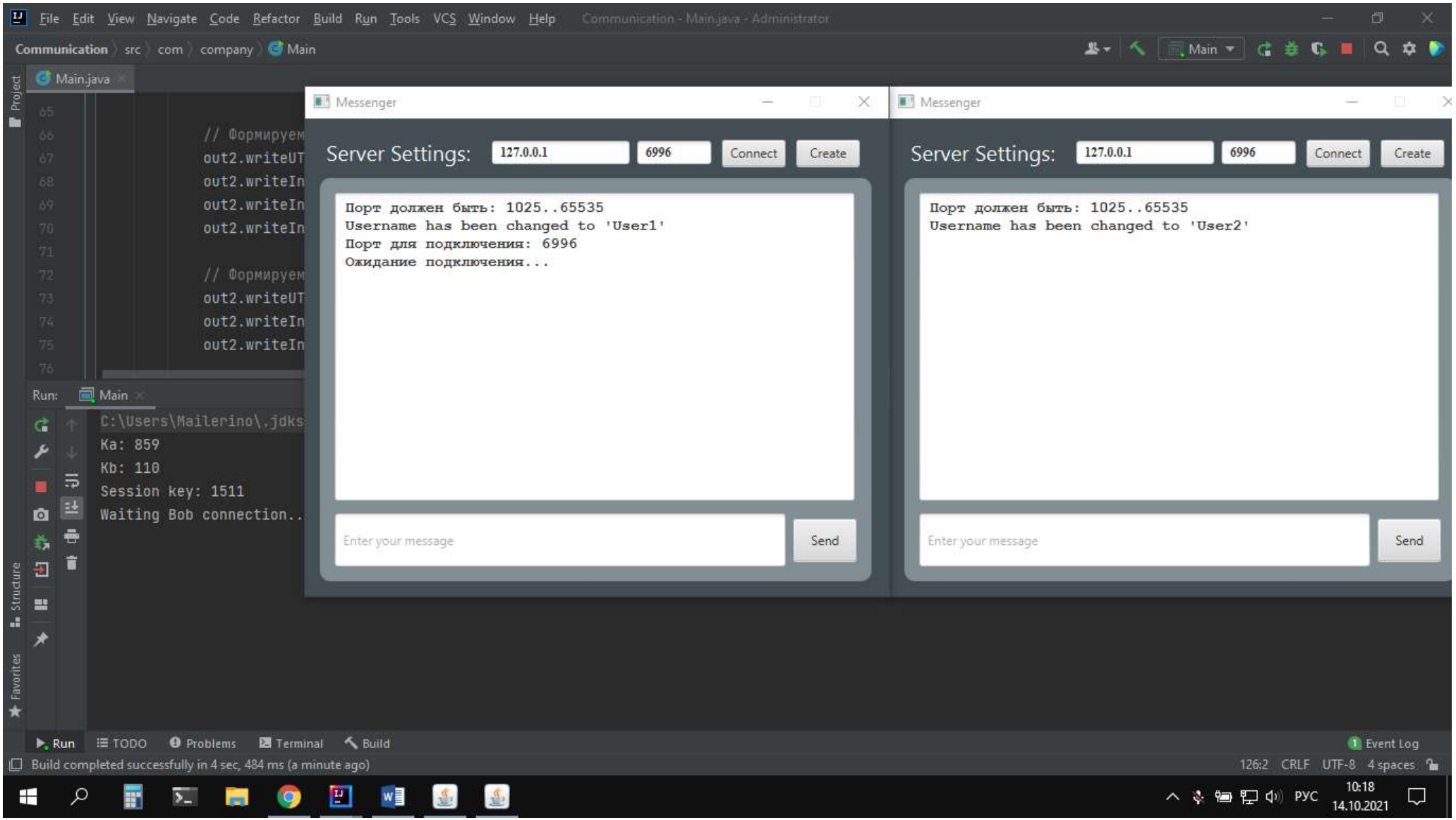
Клиент/Сервер:

```java
private class ReadMsgServer extends Thread {
    @Override
    public void run() {
        try {
            socket = ss.accept(); // заставляем сервер ждать подключений
            chat_field.appendText( s: "\nBob has been connected to Alice!");
            InputStream sin = socket.getInputStream(); // Конвертируем потоки в другой тип, чтоб легче обрабатывать текстовые сообщения.
            DataInputStream in = new DataInputStream(sin);
            OutputStream sout = socket.getOutputStream();
            DataOutputStream out = new DataOutputStream(sout);

            int Ra = (int)(Math.random()*((999-100)+1))+100;
            chat_field.appendText( s: "\nСгенерированное Ra: " + Ra);
            out.writeUTF(internal_username); // Отсылаем бобу Имя и Ra
            out.writeInt(Ra);

            socket2 = new Socket( host: "127.0.0.1",  port: 8561); // Подключаемся к тренту по специальному для Алисы и Трента порту
            InputStream sin2 = socket2.getInputStream();
            OutputStream sout2 = socket2.getOutputStream(); // Конвертируем потоки в другой тип, чтоб легче обрабатывать текстовые сообщения.
            DataInputStream in2 = new DataInputStream(sin2);
            DataOutputStream out2 = new DataOutputStream(sout2);

            Ka = in2.readInt(); // Считываем общий ключ Алисы и Трента
            chat_field.appendText( s: "\nОбщий ключ с Трентом: " + Ka);

            //Первое сообщение от трента для Алисы
            external_username = in2.readUTF();
            int newRa = in2.readInt();
            Session_Key = in2.readInt();
            int time_mark = in2.readInt();

            //Второе сообщение от трента для Боба (через Алису)
            String Ekb_A = in2.readUTF();
            int Ekb_K = in2.readInt();
            int Ekb_Tb = in2.readInt();
            int Rb = in2.readInt();

            //Расшифровываем первое сообщение
            external_username = Decryption(external_username,Ka);
            newRa = Decryption_int(newRa,Ka);
            Session_Key = Decryption_int(Session_Key,Ka);
            time_mark = Decryption_int(time_mark,Ka);

            // Проверяем совпадение new Ra с Ra на первом шаге
            if (Ra == newRa) {System.out.println("Ra Совпали!");}
            chat_field.appendText( s: "\nnew Ra: " + newRa + " Ra: " + Ra);
            chat_field.appendText( s: "\nSession Key: " + Session_Key);

            // Отправляем Бобу первое сообщение
            out.writeUTF(Ekb_A);
            out.writeInt(Ekb_K);
            out.writeInt(Ekb_Tb);

            // Отправляем Бобу второе сообщение
            out.writeInt(Encryption_int(Rb,Session_Key));


            String line = null; //Создаем пустую строку "буфер"
            while(true) {
                line = in.readUTF(); // ожидаем пока клиент пришлет строку текста.
                line = Decryption(line,Session_Key);
                chat_field.appendText( s: "\n[" + external_username + "]: " + line);
            }
        } catch (IOException e) {
            System.out.println("Исключение ReadMsgServer (Алисы)");
        }
    }
}
```

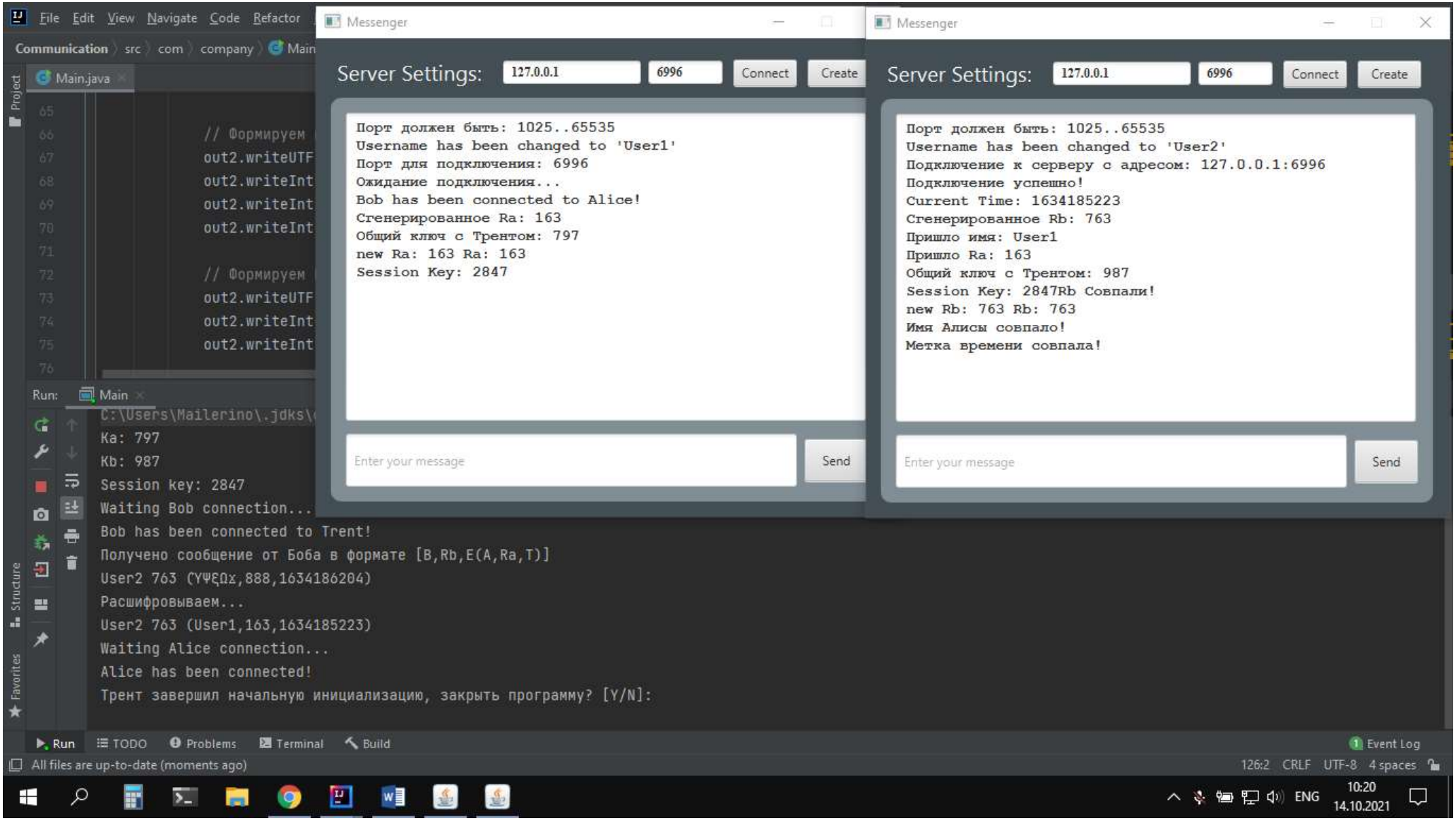
```java
private class ReadMsgClient extends Thread {
    @Override
    public void run() {
        try {
            InputStream sin = socket.getInputStream(); // Конвертируем потоки в другой тип.
            DataInputStream in = new DataInputStream(sin);
            OutputStream sout = socket.getOutputStream();
            DataOutputStream out = new DataOutputStream(sout);

            long unixTime = System.currentTimeMillis() / 1000L; // Вычисляем текущее время
            int Kb = 0, Ra = 0; // Подготавливаем переменные
            int Rb = (int)(Math.random()*((999-100)+1))+100; // Генерируем Rb
            chat_field.appendText( s: "\nCurrent Time: " + unixTime + "\nСгенерированное Rb: " + Rb);

            external_username = in.readUTF(); // Принимаем от Алисы её имя и Ra
            Ra = in.readInt();

            chat_field.appendText( s: "\nПришло имя: " + external_username + "\nПришло Ra: " + Ra);

            socket2 = new Socket( host: "127.0.0.1",  port: 8560); // Подключаемся к специальному порту Трента (предназначенному Бобу)
            OutputStream sout2 = socket2.getOutputStream();
            InputStream sin2 = socket2.getInputStream();
            DataOutputStream out2 = new DataOutputStream(sout2);
            DataInputStream in2 = new DataInputStream(sin2);

            Kb = in2.readInt(); // Считываем общий ключ Трента и Боба
            chat_field.appendText( s: "\nОбщий ключ с Трентом: " + Kb);

            String shAlice = Encryption(external_username,Kb);
            int shRa = Encryption_int(Ra,Kb);
            int shTime = Encryption_int(((int)unixTime),Kb);

            System.out.println("myName: " + internal_username);
            System.out.println("Rb: " + Rb);
            System.out.println("shAlice: " + shAlice);
            System.out.println("shRa: " + shRa);
            System.out.println("shCurTime: " + shTime);

            out2.writeUTF(internal_username); // Имя Боба
            out2.writeInt(Rb); // Число Боба
            out2.writeUTF(shAlice); // Имя алисы (шифр)
            out2.writeInt(shRa); // Ra (шифр)
            out2.writeInt(shTime); // Метка времени (шифр)

            //Второе сообщение от трента для Боба отправленное Алисой
            String Ekb_A = in.readUTF();
            int Ekb_K = in.readInt();
            int Ekb_Tb = in.readInt();

            // Rb от Алисы, закодированное сессионным ключом
            int newRb = in.readInt();

            // Расшифровываем
            Ekb_A = Decryption(Ekb_A, Kb);
            Ekb_K = Decryption_int(Ekb_K, Kb);
            Ekb_Tb = Decryption_int(Ekb_Tb, Kb);

            Session_Key = Ekb_K;
            newRb = Decryption_int(newRb,Session_Key);
            chat_field.appendText( s: "\nSession Key: " + Session_Key);

            // Проверяем совпадение new Rb с Rb
            if (Rb == newRb) {chat_field.appendText( s: "Rb Совпали!");}
            chat_field.appendText( s: "\nnew Rb: " + newRb + " Rb: " + Rb);
            if (Ekb_A.equals(external_username)) {chat_field.appendText( s: "\nИмя Алисы совпало!");}
            if (Ekb_Tb == (int)unixTime) {chat_field.appendText( s: "\nМетка времени совпала!");}
            if (Ekb_Tb < ((int)currentTime - 300)) {chat_field.appendText( s: "\nМетка времени ПРОСРОЧЕНА МИНИМУМ НА 5 МИНУТ!");}
            if (Ekb_Tb > ((int)currentTime + 300)) {chat_field.appendText( s: "\nМетка времени СПЕШИТ МИНИМУМ НА 5 МИНУТ!");}

            String line = null; //Создаем пустую строку "буфер"
            while(true) {
                line = in.readUTF(); // ожидаем пока клиент пришлет строку текста.
                line = Decryption(line,Session_Key);
                chat_field.appendText( s: "\n[" + external_username + "]: " + line);
            }
        } catch (IOException e) {
            System.out.println("Исключение ReadMsgClient (Боб)");
        }
    }
```

Тесты:

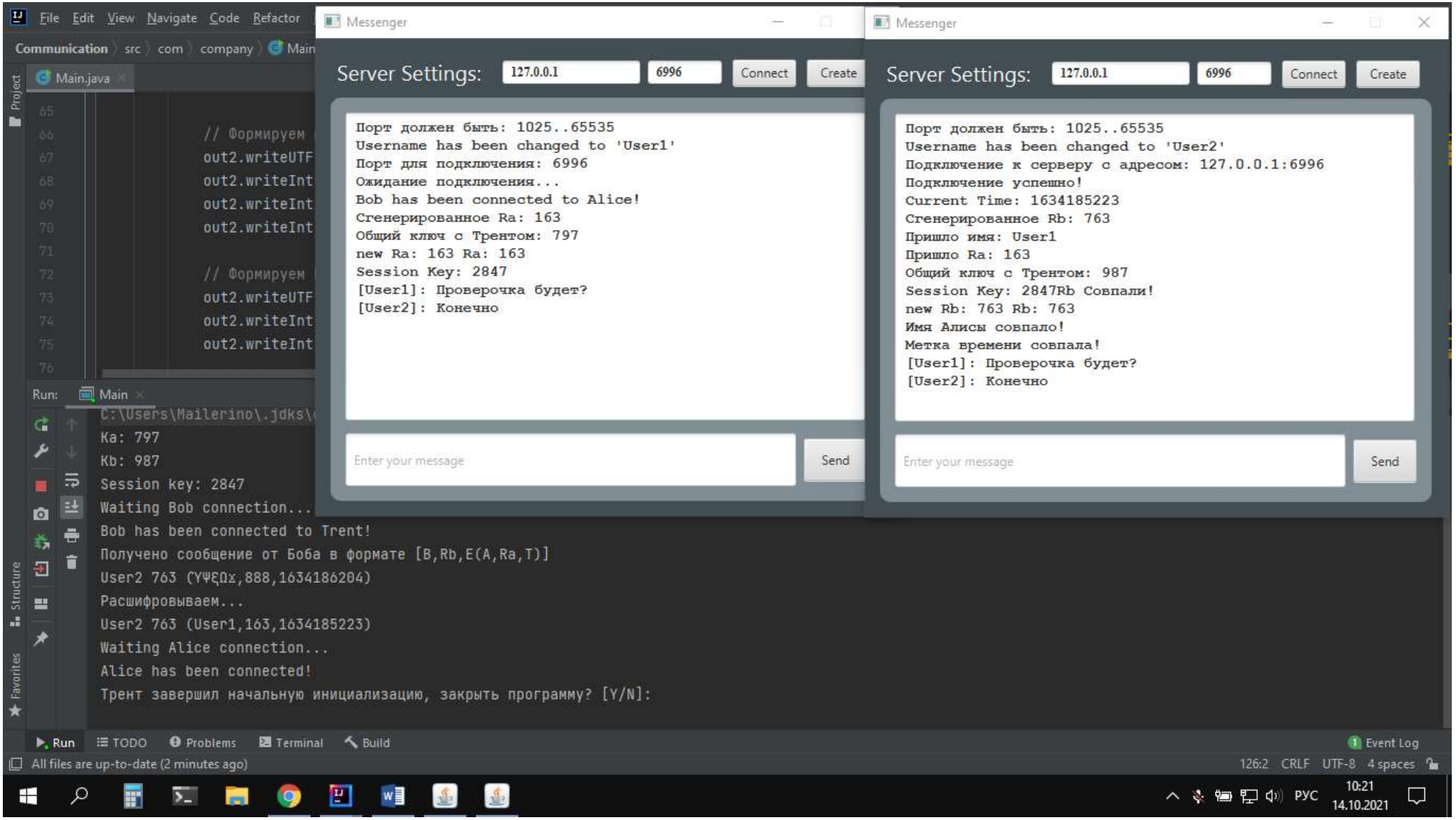
Запускаем Трент, Алису и Боба, создаем на стороне Алисы (User1) Сервер:



Подключаемся к серверу Алисы чтобы инициировать работу протокола

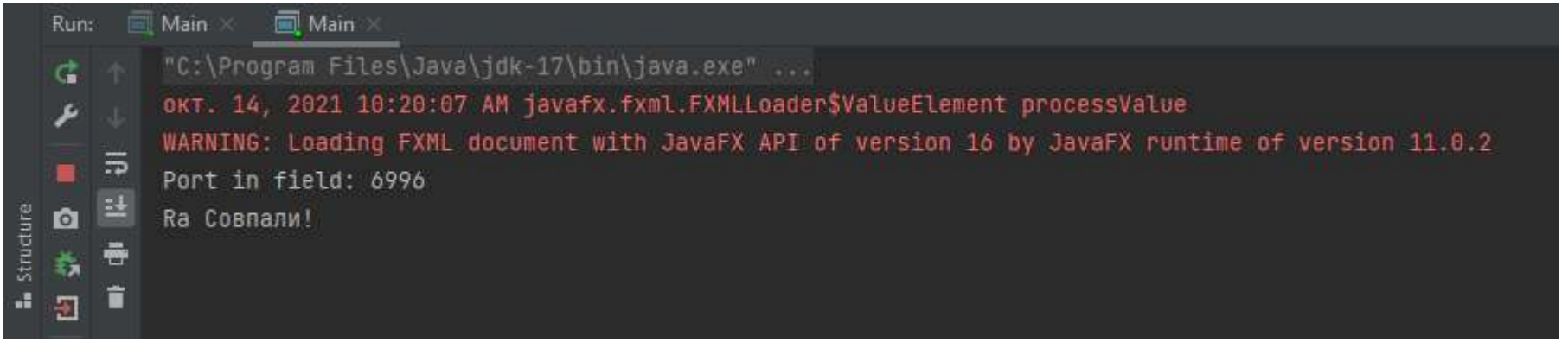


Трент завершил свою работу, протокол реализован, сессионные ключи совпали, можно общаться:

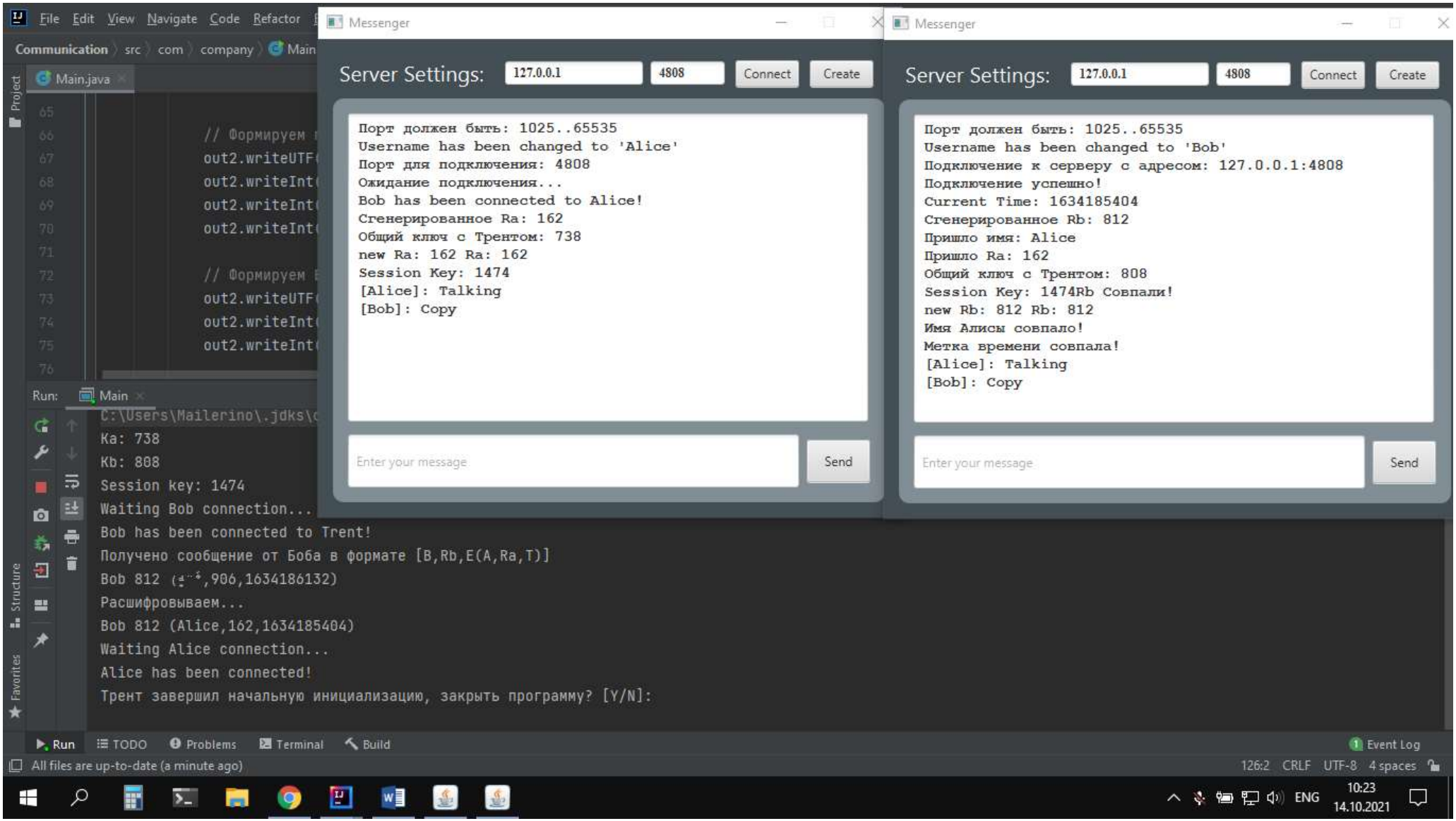


Дополнительные логи на Клиенте/Сервере:


Run:
Main
Main
"C:\Program Files\Java\jdk-17\bin\java.exe" ...
окт. 14, 2021 10:20:11 AM javafx.fxml.FXMLLoader$ValueElement processValue
WARNING: Loading FXML document with JavaFX API of version 16 by JavaFX runtime of version 11.0.2
myName: User2
Rb: 763
shAlice: ϒΨϱΩx
shRa: 888
shCurTime: 1634186204
Structure
Favorites


Run:
Main
Main
"C:\Program Files\Java\jdk-17\bin\java.exe" ...
окт. 14, 2021 10:20:07 AM javafx.fxml.FXMLLoader$ValueElement processValue
WARNING: Loading FXML document with JavaFX API of version 16 by JavaFX runtime of version 11.0.2
Port in field: 6996
Ra Совпали!
Structure

Тест 2:


Messenger
Server Settings:
127.0.0.1
4808
Connect
Create
Порт должен быть: 1025..65535
Username has been changed to 'Alice'
Порт для подключения: 4808
Ожидание подключения...
Bob has been connected to Alice!
Сгенерированное Ra: 162
Общий ключ с Трентом: 738
new Ra: 162 Ra: 162
Session Key: 1474
[Alice]: Talking
[Bob]: Copy
Enter your message
Send
Messenger
Server Settings:
127.0.0.1
4808
Connect
Create
Порт должен быть: 1025..65535
Username has been changed to 'Bob'
Подключение к серверу с адресом: 127.0.0.1:4808
Подключение успешно!
Current Time: 1634185404
Сгенерированное Rb: 812
Пришло имя: Alice
Пришло Ra: 162
Общий ключ с Трентом: 808
Session Key: 1474Rb Совпали!
new Rb: 812 Rb: 812
Имя Алисы совпало!
Метка времени совпала!
[Alice]: Talking
[Bob]: Copy
Enter your message
Send
Communication
src
com
company
Main
Main.java
Run:
Main
Ka: 738
Kb: 808
Session key: 1474
Waiting Bob connection...
Bob has been connected to Trent!
Получено сообщение от Боба в формате [B,Rb,E(A,Ra,T)]
Bob 812 (,906,1634186132)
Расшифровываем...
Bob 812 (Alice,162,1634185404)
Waiting Alice connection...
Alice has been connected!
Трент завершил начальную инициализацию, закрыть программу? [Y/N]:
Run
TODO
Problems
Terminal
Build
Event Log
All files are up-to-date (a minute ago)
126:2
CRLF
UTF-8
4 spaces
ENG
10:23
14.10.2021