JPWP DnDApp - Czat grupowy - Zadania

Anna Kłucjasz, Krzysztof Skoś

April 2020

Spis treści

1 Zadania

We wszytskich zadania zakładamy, że używane zmienne zostały zadeklarowane poprawnie.

1.1 Protokół komunikacyjny

Zaproponuj własną implementację przedstawionego w prezentacji protokołu komunikacyjnego oraz uzupełnij brakujące linijki oznaczone przez "???".

```
private void clientListeningSocket() {
    try {
        socket = new Socket (serverIP, port);
        out = new PrintWriter(socket.getOutputStream(), true);
        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        boolean loop = true;
        while(loop){
            final String incomingLine = in.readLine();
            final String displayLine = organizingProtocol(incomingLine);
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    chatMessages.append("\n" + displayLine);
                }
            });
        }
    } catch (IOException e){
        Log.d("tag", Objects.requireNonNull(e.getMessage()));
    }
}
public void onClick (final View view) {
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            switch (view.getId()) {
                case R.id.sendBtn:
                    String outgoingChatMessage = editMessage.getText().toString();
???
???
???
                    out.println(outgoingChatMessage);
                    editMessage.setText("");
                    break;
                case R.id.d20Btn:
                    String outgoingDiceRoll = Integer.toString(mkDiceRoll(20));
???
???
                    out.println(outgoingDiceRoll);
                    break;
                case R.id.d4Btn:
                    outgoingDiceRoll = Integer.toString(mkDiceRoll(4));
???
???
                    out.println(outgoingDiceRoll);
                    break;
            }
        }
    });
    thread.start();
```

```
}
private String organizingProtocol (String incomingLine) {

???
???
???
???
???
???
???
???
???
???
???
???
???
???
???
???
???
???
???
}
```

1.2 Stworzenie prostego klienta

Napisz prostą implementację klienta, który komunikuje się z serwerem w sieci lokalnej, mającym IP 192.168.1.4 i nasłuchującym portem 23745.
Ponadto klient powinien móc wysyłać krótkie wiadomości tekstowe serwerowi i czekać na odpowiedź.
Podpowiedź: możesz wykorzystać klasy Socket, BufferedReader, PrintWriter.

1.3 Stworzenie prostego serwera

Napisz prostą implementację serwera z pierwszego zadania. Załóż, że serwer ma ip 192.168.1.4 i powinien nasłuchiwać na porcie 23745. Jeśli uda się połączyć z klientem, to serwer powinien oczekiwać na wiadomość tekstową i odpisywać klientowi "Wiadomość została przyjęta."

1.4 Implementacja obsługi wielu klientów

Dostosuj serwer z poprzedniego zadania tak, aby serwer mógł rozmawiać równocześnie z wieloma klientami.

1.5 Czat grupowy

Uzupełnij kod w taki sposób, aby zapewnić komunikację pomiędzy kilkoma klientami połączonymi z serwerem.

```
public class CliWork implements Runnable {



CliWork () {
???
    }

    @Override
    public void run() {
        BufferedReader in = null;

        try {
            in = new BufferedReader(new InputStreamReader(client.getInputStream()));
        } catch (IOException e) {
            Log.d("tag", e.getMessage());
        }
        while (true) {
            try {
                assert in != null;
                String receivedFromClientLine = in.readLine();
???
???
???
???
                final String finalReceivedFromClientLine = receivedFromClientLine;
                chatActivity.runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        textArea.append("\n" + finalReceivedFromClientLine);
                    }
                });
            } catch (IOException e) {
                Log.d("tag", e.getMessage());
            }
        }
    }
}
```

```
public class ChatActivity extends AppCompatActivity{

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_chat);
       //^ android stuff
        Thread thread = new Thread(new Runnable() {
            @Override
            public void run() {
                if (isHost()) {
                    serverListeningSocket();
                } else {
                    clientListeningSocket();
                }
            }
        });
        thread.start();
    }
    //Socket dla servera
    private void serverListeningSocket() {
        try {
            server = new ServerSocket(port);
        } catch (IOException e) {
            Log.d("tag", Objects.requireNonNull(e.getMessage()));
        }
        while (true) {
            CliWork work;
            try {
                work = new CliWork(                                                    );
???
???
                Thread t = new Thread(work);
                t.start();
            } catch (IOException e) {
                Log.d("tag", Objects.requireNonNull(e.getMessage()));
            }
        }
    }
    //Socket dla klienta
    private void clientListeningSocket() {
        try {
            socket = new Socket (serverIP, port);
            out = new PrintWriter(socket.getOutputStream(), true);
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            boolean loop = true;
            while(loop){
                final String incomingLine = in.readLine();
                final String displayLine = organizingProtocol(incomingLine);
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        chatMessages.append("\n" + displayLine);
                    }
                });
            }
        } catch (IOException e){
```

```
            Log.d("tag", Objects.requireNonNull(e.getMessage()));
        }
    }
```

//onClick - co się dzieje po kliknięciu przycisków
//sendBtn: wysyłania wiadomości; d20/4Btn: rzutu kością

```
    public void onClick (final View view) {
        Thread thread = new Thread(new Runnable() {
            @Override
            public void run() {
                switch (view.getId()) {
                    case R.id.sendBtn:
                        String outgoingChatMessage = editMessage.getText().toString();
???
                        break;
                    case R.id.d20Btn:
???
                        break;
                    case R.id.d4Btn:
???
                        break;
                }
            }
        });
        thread.start();
    }

    private String organizingProtocol (String incomingLine) {
???         //Zadanie 1
    }
}
```