

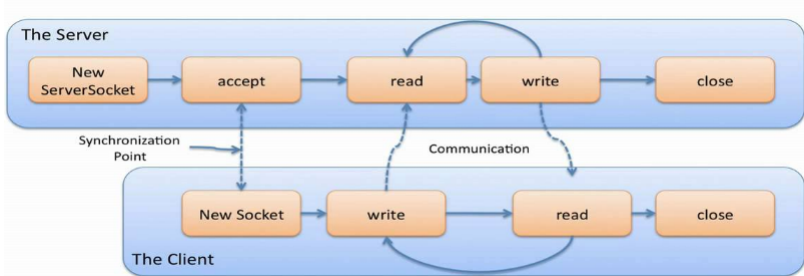
Komunikator Internetowy "Chatter"

Przemysław Jaroszkiewicz

Akademia Górniczo-Hutnicza im.Stanisława Staszica

- 1 Socket'y i komunikacja Client-Server
- 2 Komunikacja grupowa
- 3 MySQL Server
- 4 MySQL Database

Socket Client-Server



Stworzenie serwera:

```
ServerSocket serverSocket = new ServerSocket( port: 9999);  
Socket clientSocket = serverSocket.accept();  
InputStream input = clientSocket.getInputStream();  
this.output = clientSocket.getOutputStream();  
BufferedReader reader = new BufferedReader(new InputStreamReader(input));
```

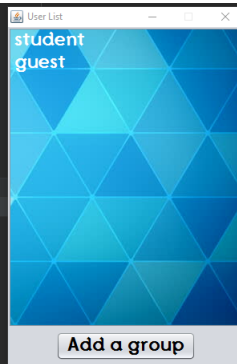
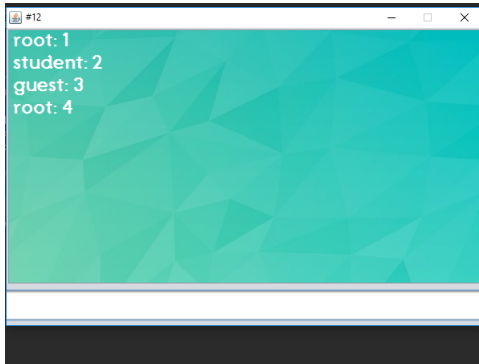
Wypisanie połączenia w konsoli:

```
System.out.println("Waiting for a new connection...");  
Socket clientSocket = serverSocket.accept();  
System.out.println("Client connected:" + clientSocket);
```

```
Waiting for a new connection...  
Client connected:Socket[addr=/192.168.244.233,port=52265,localport=9999]  
Waiting for a new connection...
```

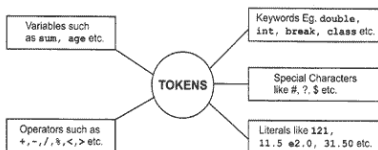
Przyjmowanie wielu połączeń

```
ServerWorker worker = new ServerWorker( server: this,clientSocket);  
workerList.add(worker);  
worker.start();
```



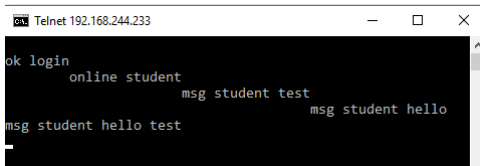
Tokeny i rozpoznawanie komend przez serwer

```
while ((line = reader.readLine()) != null) {  
    System.out.println("Pushed command: " + line);  
    String[] tokens = line.split(" ");  
    if (tokens != null && tokens.length > 0) {  
        String cmd = tokens[0];  
        if ("quit".equals(line)) {  
            handleLogout();  
            break;  
        } else if ("login".equalsIgnoreCase(cmd)) {  
            handleLogin(output, tokens);  
        } else if ("msg".equalsIgnoreCase(cmd)) {  
  
            String[] tokensMsg = StringUtils.split(line);  
            handleMsg(tokensMsg);  
        } else if ("leave".equalsIgnoreCase(cmd)) {  
            handleLeave(tokens);  
        }  
        else if ("join".equalsIgnoreCase(cmd)) {  
            handleJoin(tokens);  
        }  
        else {  
            String msg = "Unknown " + cmd + "\n";  
            output.write(msg.getBytes());  
        }  
    }  
}
```



Telnet

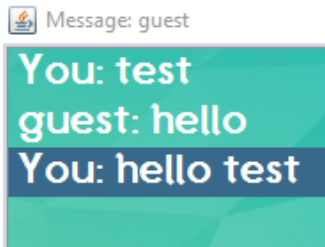
```
>  
>telnet 192.168.244.233 9999
```



A screenshot of a Telnet window titled "Telnet 192.168.244.233". The window shows a chat session with the following text:

```
ok login  
    online student  
        msg student test  
msg student hello test  
        msg student hello
```

```
Waiting for a new connection...  
Pushed command: login guest guest  
guest has logged in  
Pushed command: msg guest test  
test  
Pushed command: msg student hello  
hello  
Pushed command: msg guest hello test  
hello  
hello test
```



A screenshot of a "Chatter" window titled "Message: guest". The window shows a chat session with the following text:

```
You: test  
guest: hello  
You: hello test
```

Aplikacja - Client side

```
public boolean connect(){
    try {
        this.socket = new Socket(serverName,serverPort);
        System.out.println("Client port is " +socket.getLocalPort());
        this.serverOut = socket.getOutputStream();
        this.serverIn = socket.getInputStream();
        this.bufferedIn = new BufferedReader((new InputStreamReader(serverIn)));
        return true;
    } catch (IOException e) {
        e.printStackTrace();
    }
    return false;
}
```

```
private void startMessageReader(){
    Thread t = run() - { readMsgLoop(); };
    t.start();
}

private void readMsgLoop() {
    try {
        String line;
        while ((line = bufferedIn.readLine()) != null) {
            String[] tokens = StringUtils.split(line);
            if (tokens != null && tokens.length > 0) {
                String cmd = tokens[0];
                if ("online".equalsIgnoreCase(cmd)) {
                    handleOnline(tokens);
                }
            }
        }
    }
}
```

```
private void handleOnline(String[] tokens){
    String login = tokens[1];
    for(UserStatusListener listener:userStatusListeners){
        listener.online(login);
    }
}

public void handleOffline(String[] tokens){
    String login = tokens[1];
    for(UserStatusListener listener:userStatusListeners){
        listener.offline(login);
    }
}
```

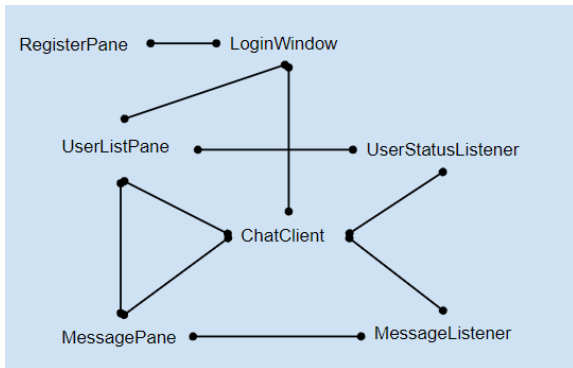

Aplikacja - Listeners

```
@Override
public void online(String login){
    userListModel.addElement(login);
}

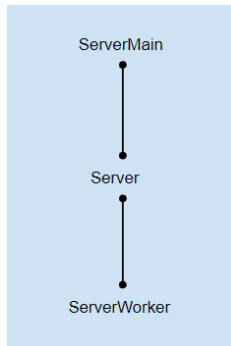
@Override
public void offline(String login){
    userListModel.removeElement(login);
}
```

```
@Override
public void onMessage(String fromLogin, String msgBody) {
    String line;
    if (login.equalsIgnoreCase(fromLogin)) {
        line = fromLogin + ": " + msgBody;
        userListModel.addElement(line);
        if(userlistModel.size()>36){
            userListModel.removeElementAt( index 0);
        }
    }
}

@Override
public void onGroupMessage(String groupName, String fromLogin, String msgBody) {
    String line;
    line = fromLogin + ": " + msgBody;
    userListModel.addElement(line);
    if(userlistModel.size()>36){
        userListModel.removeElementAt( index 0);
    }
}
```



Client



Server



MySQL Server

Query 1

Limit to 1000 rows

```
1 • create database myDatabase;  
2 • use myDatabase;  
3 • create table customer(LOGIN varchar(30) not null, PASS varchar(30) not null, EMAIL varchar(30) not null);  
4 • select * from customer;  
5 • drop table customer;  
6
```

<

Result Grid Filter Rows: Export: Wrap Cell Content:

	LOGIN	PASS	EMAIL
▶	student	student	student@gmail.com
	guest	guest	guest@gmail.com
	root	root	root

MySQL Client

```
Connection connection= DriverManager.getConnection( url,"jdbc:mysql://localhost:3306/myDatabase?useU
//Prepared Statement
PreparedStatement Pstatement=connection.prepareStatement( sql "insert into customer values(?, ?, ?)");
//Specifying the values of it's parameter
Pstatement.setString( parameterIndex 1,loginField.getText());
Pstatement.setString( parameterIndex 2,passwordField.getText());
Pstatement.setString( parameterIndex 3,emailField.getText());
Statement stm = connection.createStatement();
String query = "SELECT LOGIN, EMAIL FROM customer;";
stm.executeQuery(query);
ResultSet rs = stm.getResultSet();
while(rs.next()) {
    String dbUsername = rs.getString( columnIndex "LOGIN");
    String dbEmail = rs.getString( columnIndex "EMAIL");
    if(dbUsername.equals(loginField.getText()) || dbEmail.equals(emailField.getText())) {
        JOptionPane.showMessageDialog(panell, message: "Login or email is taken.");
        setRegister(0);
        break;
    }
}
if(register==1) {
    Pstatement.executeUpdate();
    frame.dispose();
}
```