

Dokumentacja:

Projekt został opracowany w środowisku programistycznym NetBeans IDE8.2.

- programowanie obiektowe;
- wielowątkowość;
- strumienie we / wy;
- współpraca z siecią.
- Interfejs

- JFrame

Projekt jest rozdzielony na 4

pliki: Klient.java, Server.java, TCPConnection.java, ConnectionCheck.java

Zmienni v projekci:

Klient.java:

```
private TCPConnection connection;

private String IP_ADDR;

private int PORT;

char c;

char vchar;

String text;

private javax.swing.JTextField IP;

private javax.swing.JTextField Port;

private javax.swing.JButton btnConnect;

private javax.swing.JButton btnDisconnect;

private javax.swing.JTextField fieldInput;

private javax.swing.JTextField fieldNickName;

private javax.swing.JButton jButton1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;
```

```
private javax.swing.JPanel jPanel1;  
private javax.swing.JPanel jPanel2;  
private javax.swing.JPanel jPanel3;  
private javax.swing.JScrollPane jScrollPane1;  
private javax.swing.JLabel lbIP;  
private javax.swing.JLabel lbName;  
private javax.swing.JLabel lbPort;  
private javax.swing.JTextArea log;
```

ServerGrafika.java:

```
private final ArrayList<TCPConnection> Listconnections = new ArrayList<>();  
private Thread thread;  
  
int Port;  
  
char vchar;  
  
private javax.swing.JTextArea AreaLogsChat;  
private javax.swing.JButton btnCreateServer;  
private javax.swing.JLabel jLabel1;  
private javax.swing.JLabel jLabel3;  
private javax.swing.JPanel jPanel1;  
private javax.swing.JPanel jPanel2;  
private javax.swing.JPanel jPanel3;  
private javax.swing.JScrollPane jScrollPane1;  
private javax.swing.JLabel lbPort;  
private javax.swing.JTextField txtPORT;
```

TCPConnection.java:

```
private Socket socket;  
  
String msg;  
  
private Thread receiveThread;
```

private BufferedReader in;//BufferedReader to klasa Java, która odczytuje tekst ze strumienia wejściowego.

private BufferedWriter out;//BufferedWriter zapisuje tekst w strumieniu wyjściowym

private ConnectionCheck statusListener;//słuchacz zdarzeń

Funkcji:

Server.java:

Klient.java:

```
private void IPKeyTyped(java.awt.event.KeyEvent evt) {  
    // TODO add your handling code here:  
    char c = evt.getKeyChar();  
    if((Character.isLetter(c)) //liczby i znaki ,./  
    {  
        evt.consume();  
    }  
}
```

```
private void PortKeyTyped(java.awt.event.KeyEvent evt) {  
    // TODO add your handling code here:  
    char vchar = evt.getKeyChar();  
    if(!(Character.isDigit(vchar)) || Port.getText().length() >= 5){ //tylko liczby  
        evt.consume();  
    }  
}
```

```
}
```

```
private void fieldNickNameKeyReleased(java.awt.event.KeyEvent evt) {  
    // TODO add your handling code here:  
    lbName.setText(""); //oczyszczamy  
}
```

```
private void IPKeyReleased(java.awt.event.KeyEvent evt) {  
    // TODO add your handling code here:  
    lbIP.setText(""); //oczyszczamy label  
  
}
```

```
private void PortKeyReleased(java.awt.event.KeyEvent evt) {  
    lbPort.setText(""); //oczyszczamy label  
    // TODO add your handling code here:  
}
```

```

private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(fieldNickName.getText().trim().isEmpty() && IP.getText().trim().isEmpty() && Port.getText().trim().isEmpty()) {
        //sprawdzamy NickName, Port, IP
        lbPort.setVisible(Port.getText().trim().isEmpty());
        lbName.setVisible(fieldNickName.getText().trim().isEmpty());
        lbIP.setVisible(IP.getText().trim().isEmpty());
    }
    else if
        (fieldNickName.getText().equals("") || IP.getText().trim().isEmpty() || Port.getText().trim().isEmpty()) {
        JOptionPane.showMessageDialog(this, "Fill all empty fields");
    }
    else{
        fieldNickName.setEnabled(false);
        lbIP.setVisible(false);
        lbName.setVisible(false);
        lbPort.setVisible(false);
        IP_ADDR = IP.getText();
        String text = Port.getText();
        PORT = Integer.parseInt(text);
        if(PORT>65535)
        {
            JOptionPane.showMessageDialog(this, "Maximum TCP Port is 65535");
        }
    }
}

```

```

        JOptionPane.showMessageDialog(this, "Maximum TCP Port is 65535");
    }
    try {
        connection = new TCPConnection(this, IP_ADDR, PORT); //przekazuje IP, PORT, EVENT LISTENER
    } catch (IOException e) {
        printMsg("Connection exception:" + e);
    }
}
}

```

```

private void btnDisconnectActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    connection.Disconnect();
}

```

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    new Klient().setVisible(true);
}

```

```

private void fieldNickNameKeyTyped(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here
    char vchar = evt.getKeyChar();
    if(!(Character.isLetter(vchar)) || (fieldNickName.getText().length() >= 8))
    {
    }
}

```

```

private void fieldNickNameKeyTyped(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here
    char vchar = evt.getKeyChar();
    if(! (Character.isLetter(vchar) || (fieldNickName.getText().length() >= 8)))
    {
        evt.consume();
    }
}

Date dateNow = new Date();
SimpleDateFormat formatForDateNow = new SimpleDateFormat("hh:mm a");//zmienniamy format
private void SendMsg()
{
    String msg = fieldInput.getText();
    if(msg.equals("quit")) connection.Disconnect();
    if(msg.equals("")) return;
    fieldInput.setText("");
    if(msg.length() > 300)
    {
        JOptionPane.showMessageDialog(this, "Maximum length of text is 50 chars");
    }
    else{
        connection.sendMsg(formatForDateNow.format(dateNow) + " " + fieldNickName.getText() + ":" + msg);
    }
}
}

```

```

@Override
public void actionPerformed(ActionEvent ae) { //po naciśnięciu buttona przekazac string
    SendMsg();
}

private synchronized void printMsg(String msg)
{
    SwingUtilities.invokeLater(() -> { //bo metod będzie wyzywany z roznych watkow
        //i zeby ponizej rzeczy byli spelnioni w formie potrzebujemy ten metod
        log.append(msg + "\n");
        log.setEditable(false);
        log.setLineWrap(true);//zawijaj linie, które są zbyt długie dla obszaru wyświetlania.

        log.setCaretPosition(log.getDocument().getLength());
    });
}

@Override
public void IfConnectionReady(TCPConnection tcpConnection) {
    printMsg("Connection is ready");
}

@Override
public void GetMsg(TCPConnection tcpConnection, String value) {
    printMsg(value);
}
}

```

```

@Override
public void IfDisconnect(TCPConnection tcpConnection) {
    printMsg("Connection is close \n");
}

@Override
public void IfException(TCPConnection tcpConnection, Exception e) {
    printMsg("Exception" + e);
}
}

```

ConnectionCheck.java:

```

-   */
    public interface ConnectionCheck {
        void IfConnectionReady(TCPConnection tcpConnection);
        void GetMsg(TCPConnection tcpConnection, String value);
        void IfDisconnect(TCPConnection tcpConnection);
        void IfException(TCPConnection tcpConnection, Exception e);
    }

```

TCPConnection.java

```

public TCPConnection(ConnectionCheck statusListener, String ipAddr, int port) throws IOException {
    //stworzyc
    //wewnetrznyj socket
    this(statusListener, new Socket(ipAddr, port)); //od jednego konstruktora wywołanie innego
}

public TCPConnection(ConnectionCheck statusListener, Socket socket)
    throws IOException // zaakceptuje gotowy obiekt socketa i tym socketom
    //utworzy dla nas połączenie, throws IOException, bo metoda socket.getInputStream() i socket.ge
    //generuje Exception.
{
    this.statusListener = statusListener;
    this.socket = socket;
    in = new BufferedReader(new InputStreamReader(socket.getInputStream(), Charset.forName("UTF-8")));
    out = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream(), Charset.forName("UTF-8")));
    receiveThread = new Thread(() -> { //nowy strumień, który będzie służył wszystkim nadchodzącym
        try {
            statusListener.IfConnectionReady(TCPConnection.this); //TCPConnection
            while(!receiveThread.isInterrupted())
                statusListener.GetMsg(TCPConnection.this, in.readLine());
            String msg = in.readLine(); //Dostajemy string
        } catch (IOException e) {
            statusListener.IfException(TCPConnection.this, e);
        } finally {
            statusListener.IfDisconnect(TCPConnection.this);
        }
    });
    receiveThread.start();
}
}

```

```

@Override
public String toString() { // dla log
    return "Client with port " + socket.getPort();
}

public synchronized void sendMsg(String value)
{
    try {
        out.write(value + "\r\n");//generuje try,catch
        out.flush();//flush all buffers
    } catch (IOException e) {
        statusListener.IfException(TCPConnection.this,e);//nie udalo sie odprawic,
        Disconnect();
    }
}

public synchronized void Disconnect()//zeby przerwac connect
{
    receiveThread.isInterrupted();
    try {
        socket.close();//generuje try,catch
        statusListener.IfDisconnect(this);
    } catch (IOException e) {
        statusListener.IfException(TCPConnection.this,e);
    }
}
}

```

server.java:

```

private void btnOnlineActionPerformed(java.awt.event.ActionEvent evt) {

    txtOnline.setVisible(true);
    lbOnline.setVisible(true);
    txtOnline.invalidate();
    txtOnline.setText("" + ListConnections.size());
    // TODO add your handling code here:
}

```

```

private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
    // TODO add your handling code here:
}

```

```

private {
    // TODO add your handling code here:
    if(txtPORT.getText().trim().isEmpty())//sprawdzamy czy pole jest puste
    {
        JOptionPane.showMessageDialog(this,"Fill empty field!");
    }
    Thread thread = new Thread() ->StartServerInThread();//tworzymy watek
    thread.start();
}

```

```

private void txtPORTKeyTyped(java.awt.event.KeyEvent evt) {

    char vchar = evt.getKeyChar();
    if(!(Character.isDigit(vchar)) || txtPORT.getText().length() >= 5)//sprawdzamy czy jest cyfra i dlugosc pola <= 5
    {
        evt.consume();
    }
}

```

```

private void txtPORTKeyReleased(java.awt.event.KeyEvent evt) {
    // TODO add your handling code here:
    lbPort.setText("");//wyczyścimy label
}

```

```

private void StartServerInThread()
{
    if(txtPORT.getText().trim().isEmpty())//sprawdzamy czy pole jest puste
    {
        getContentPane().requestFocusInWindow();//focus na panel
        lbPort.setVisible(true);//label jest widoczna
    }
    else{
        AreaLogsChat.append("Trying to start server.....");
        String text = txtPORT.getText();
        Port = Integer.parseInt(text);
        if(Port>65535)
        {
            JOptionPane.showMessageDialog(this,"Maximum TCP Port is 65535");
        }
        else
        {
            try (ServerSocket serverSocket = new ServerSocket(Port)) { //może służyć niektórym portom i akceptować
                //połączenia przychodzące
                AreaLogsChat.append("Server is started \n");
                while(true)
                {
                    try{
                        TCPConnection tcpConnection = new TCPConnection(this,serverSocket.accept());
                        //dla każdego nowego
                        //połączenia tworzymy nowe
                        //TCP połączenia
                    }catch(IOException e){//jeżeli coś przy przyłączeniu klienta

```

```

{
    try{
        TCPConnection tcpConnection = new TCPConnection(this,serverSocket.accept());
        //dla każdego nowego
        //połączenia tworzymy nowe
        //TCP połączenia
    }
}

```

```

private void SendToAllConnections(String value)

```

```

{
    if(value==null) return;
    AreaLogsChat.append("\r\n" + value);
    for(int i=0;i<ListConnections.size();i++)//wysyłamy wszystkim
    {
        ListConnections.get(i).sendMsg(value);
    }
}

```

```

@Override
public void IfConnectionReady(TCPConnection tcpConnection) {
    ListConnections.add(tcpConnection);//dodaję do listy
    txtOnline.setText("" + ListConnections.size());
    SendToAllConnections("Connected: " + tcpConnection);
}

@Override
public void GetMsg(TCPConnection tcpConnection, String value) {
    SendToAllConnections(value);
}

```

```

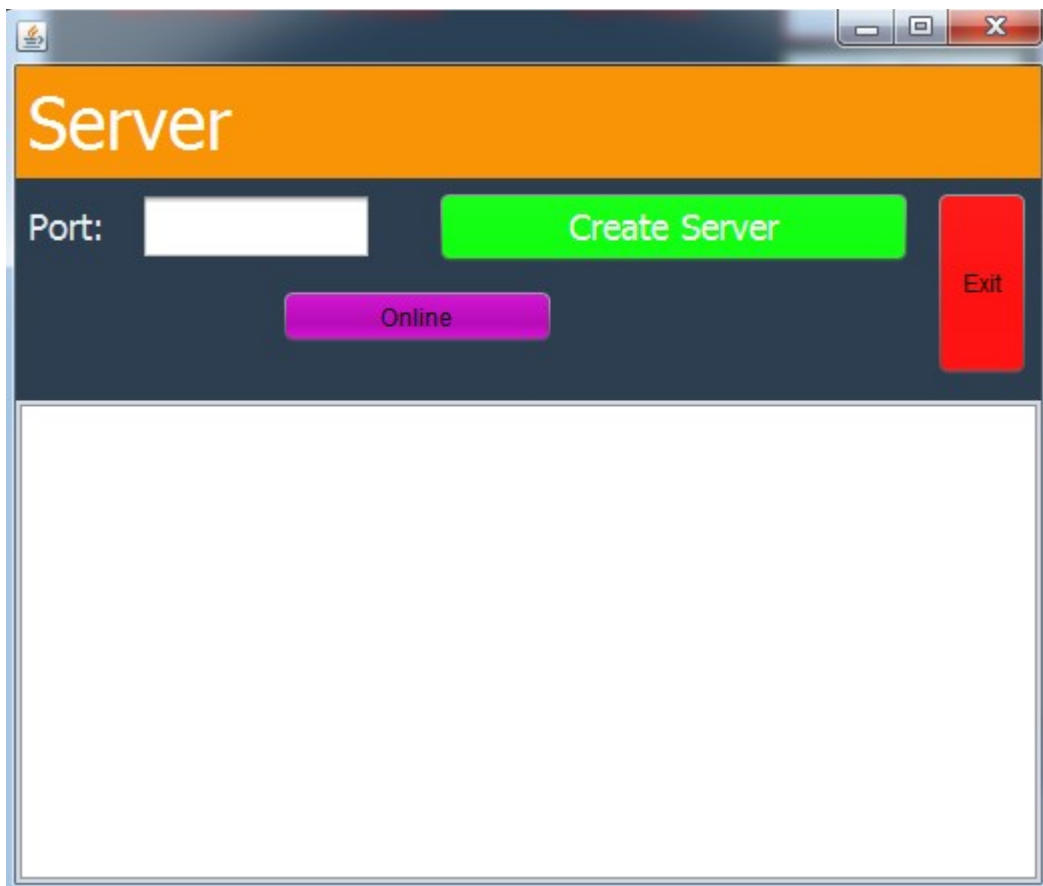
@Override
public void IfDisconnect(TCPConnection tcpConnection) {
    ListConnections.remove(tcpConnection);//usuwamy z listy
    txtOnline.setText("" + ListConnections.size());
    SendToAllConnections("Disconnected " + tcpConnection);
}

```

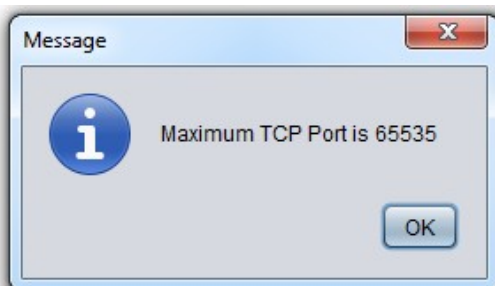


```
@Override
public void IfException(TCPConnection tcpConnection, Exception e) {
    ListConnections.remove(tcpConnection);//usuwamy z listy
    txtOnline.setText("" + ListConnections.size());
    AreaLogsChat.append("\n TCPConnection exception" + e);    }
```

Dokumentacja dla uzytkownika:
Server:



Użytkownik może wpisać Port długość którego jest nie większa niż 5 i liczba jest mniejsza równa 65535. Ponieważ maksymalny Port w TCP może być 65535. W przeciwnym przypadku użytkownik otrzymuje komunikat:



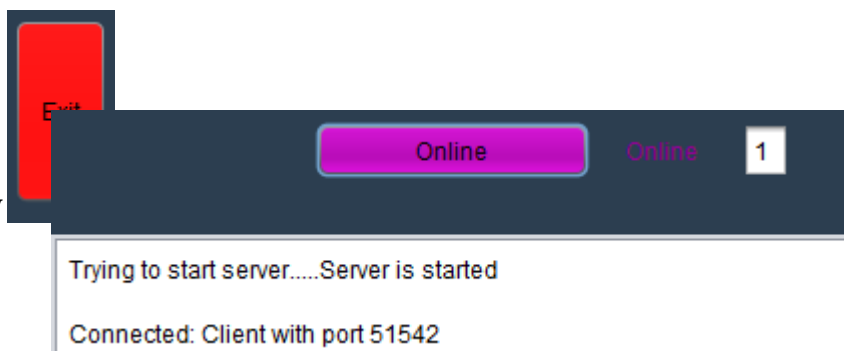
Jeżeli Port jest wpisany poprawnie otrzymujemy w oknie „Server is started” jeżeli nie tylko „Trying to start server....”

Teraz server jest stworzony. Żeby wyłączyć trzeba kliknąć na

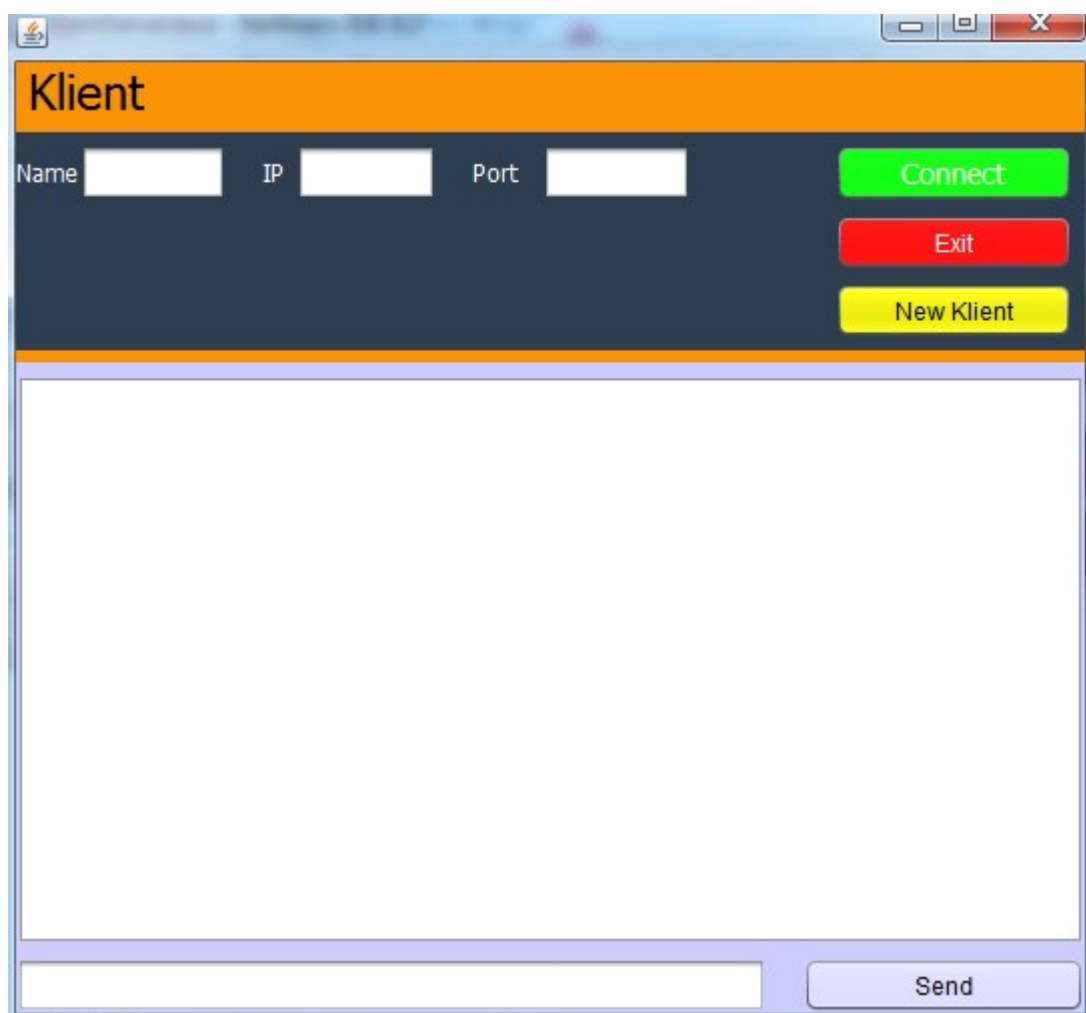


LUB przycisk

Jeżeli chcemy zobaczyć online , możemy zrobić przycisk „Online”



Dokumentacja Klienta:



Wpisać wszystkie pola jest warunek konieczny, czyli jeżeli spróbujemy nacisnąć na button Connect z pustymi polami otrzymujemy następuny wynik:

Klient

Name IP Port

Name is empty IP is empty Port is empty

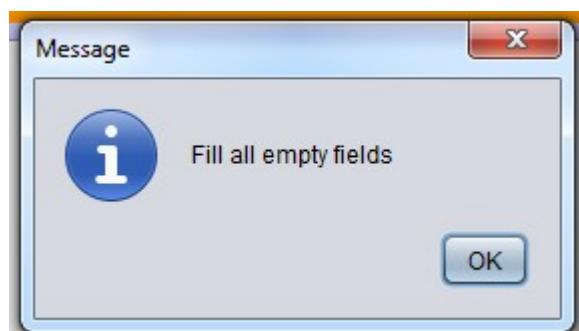
Connect Disconnect New Klient

W pole Name możemy wpisywać tylko Litery i nie większe niż 8
W pole IP możemy wpisywać tylko cyfry i znaki ”.”
W pole Port możemy wpisywać tylko cyfry.
Kiedy zaczynamy coś wpisywać w polu komunikat „Smth is empty” znika.

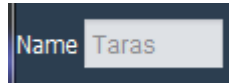
Name e IP Port 5

IP is empty

Jeżeli teraz naciśniemy
„Connect” to otrzymujemy
następny komunikat:



Kiedy wszystkie pola jest poprawnie wprowadziliśmy możemy nacisnąć „Connect” po naciśnięciu Name staje niemożliwe do zmian.

A dark-themed input field with the label 'Name' and the text 'Taras' inside.


Jeżeli nie możemy dołączyć się do servera dostajemy następny komunikat:

Connection exception: java.net.ConnectException: Connection refused: connect

Jeżeli połączyliśmy się do servera mamy następny nadpis w TextArea: Przycisk „Connect” staje się nie dostępny:

Connection is ready

Connected: Client with port 51324

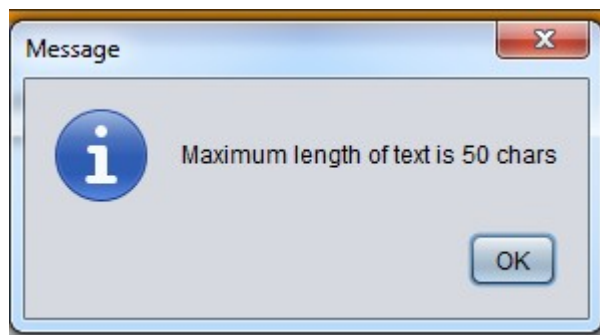
A dark-themed button with the text 'Connect' in white.

W tym polu możemy pisać wiadomość żeby odprawić możemy nacisnąć „Enter” lub przycisk „Send”

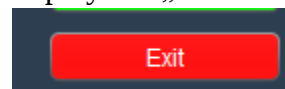
A light blue message input area with a white text field and a 'Send' button on the right.

11:48 PM Taras:привет

W polu wiadomości jest ograniczenia do 50 charów.



Żeby zamknąć program możemy w polu napisać „quit” albo przycisk „Exit” lub krzyżek.

A small input field containing the text 'quit'.A red button with the text 'Exit' in white.