

Санкт-Петербургский Политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчёт о лабораторной работе

Дисциплина: Сети ЭВМ и телекоммуникации

Тема: Клиент протокола FTP, работающий в пассивном режиме

Выполнил студент гр. 43501/3

(подпись) Никитенко А.П.

Руководитель

(подпись) Вылегжанина К.Д.

“ ____ ” _____ 2016 г.

Санкт-Петербург

2016

1. Индивидуальное задание

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции клиента протокола FTP.

2. Цель работы

Изучить протокол FTP прикладного уровня стека TCP/IP.

3. Описание протокола FTP

Протокол FTP (RFC 959) предназначен для передачи файлов в сети Интернет. Протокол использует двухканальную схему передачи:

1. Управляющий канал

Канал использует порт 21 и TCP в качестве транспортного протокола. Данное соединение существует все время обмена.

2. Канал данных

Используется TCP-порт 20 или непривилегированный порт. Данный канал предназначен для передачи файлов и списков файлов каталога.

В соответствии с протоколом существуют два режима работы сервера:

- 1) активный — когда сервер инициализирует соединение для передачи файлов;
- 2) пассивный — когда клиент инициализирует соединение для передачи файлов.

Пассивный режим предназначен для решения проблемы, возникающей при работе сервера с клиентом, использующим NAT. Т.к. NAT не позволяет устанавливать входящие соединения, сервер не может открыть канал данных, что не позволяет использовать протокол для передачи файлов и получения списка файлов.

4. FTP-клиент

4.1. Описание приложения

Разработанный FTP-клиент реализует следующие функции:

- 1) Подключение к указанному серверу
- 2) Получение списка файлов в каталоге
- 3) Навигация по системе каталогов
- 4) Копирование файла на сервер
- 5) Копирование файла с сервера
- 6) Создание и удаление каталогов
- 7) Удаление файлов
- 8) Обеспечение работы со ссылками на файлы и каталоги
- 9) Поддержка передачи в бинарном и текстовом режиме
- 10) Протоколирование соединения сервера с клиентом

Используемый режим работы FTP – пассивный.

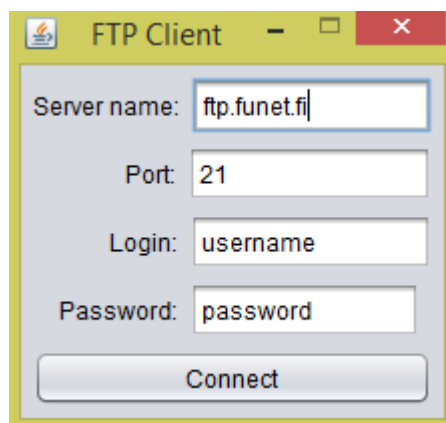
Реализованные команды протокола:

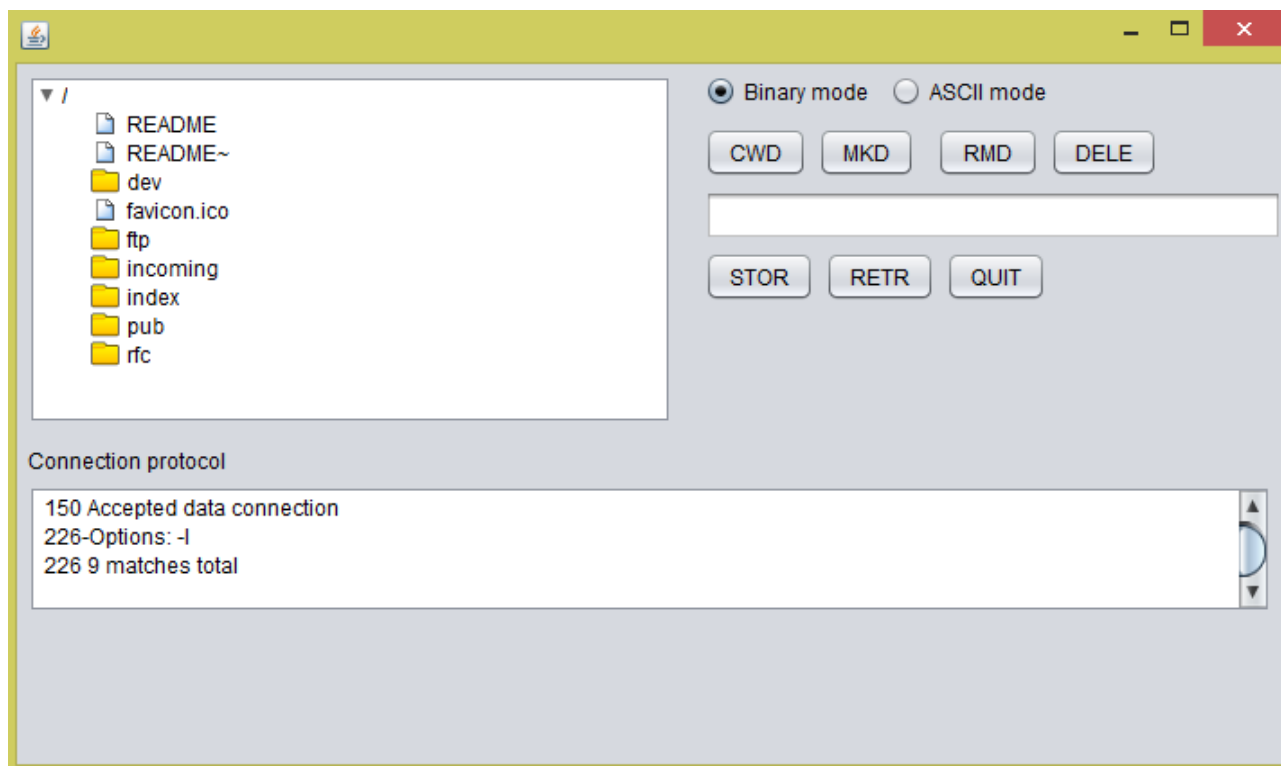
- 1) USER + PASS;
- 2) MKD/RMD;
- 3) DELE;
- 4) CWD;
- 5) PASV;
- 6) LIST;
- 7) TYPE;
- 8) RETR/STOR;
- 9) QUIT.

Разрабатываемое приложение обеспечивает настройку следующих параметров:

- IP-адрес или доменное имя файлового сервера
- Порт подключения
- Имя пользователя
- Пароль пользователя
- Режим передачи: текстовый/бинарный

Приложение имеет графический интерфейс пользователя. Пользователь имеет возможность задать адрес сервера, порт подключения, имя пользователя и пароль.





Лог подключения отображается в текстовом окне с полосой прокрутки внизу формы.

Содержание текущего каталога отображается в текстовом окне в верхней части формы.

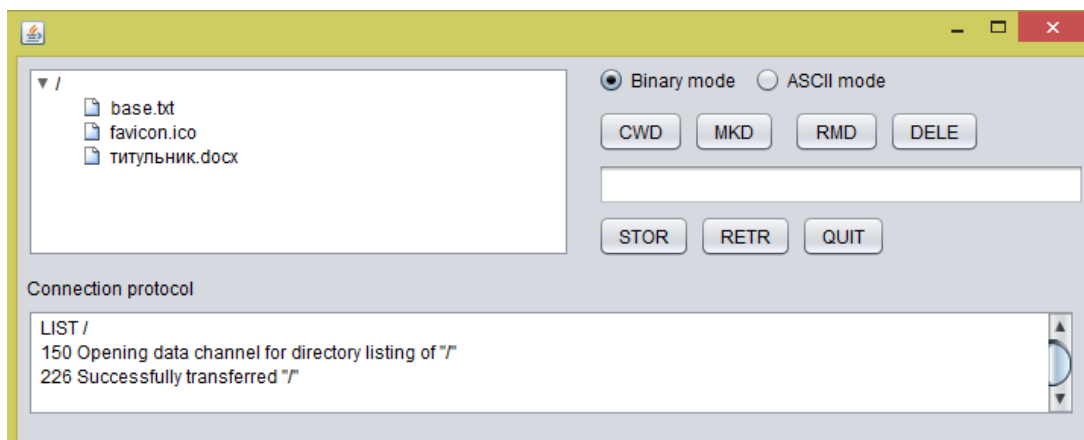
Для

- удаления директории – выбрать директорию на сервере с помощью дерева ФС и нажать кнопку RMD;
- создания директории – ввести имя новой директории и нажать кнопку MKD;
- загрузки файла на локальный диск – выбрать файл с помощью дерева ФС и нажать на кнопку RETR, указать директорию, в которую требуется сохранить файл;
- отправки файла на FTP-сервер – нажать на кнопку STOR и выбрать файл на локальном диске;
- смены текущего каталога сервера - выбрать директорию на сервере с помощью дерева ФС и нажать кнопку CWD;
- удаления файла - выбрать файл с помощью дерева ФС и нажать кнопку DELE;
- выхода из программы - нажать кнопку QUIT.

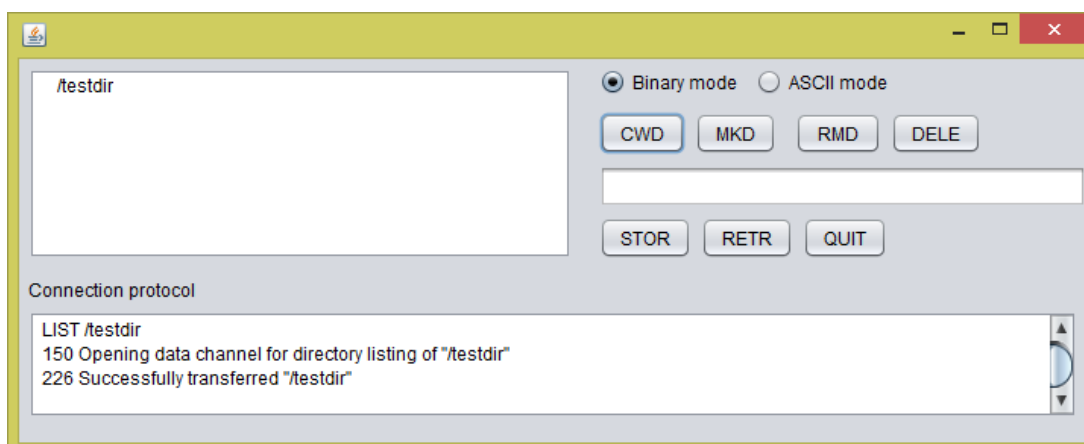
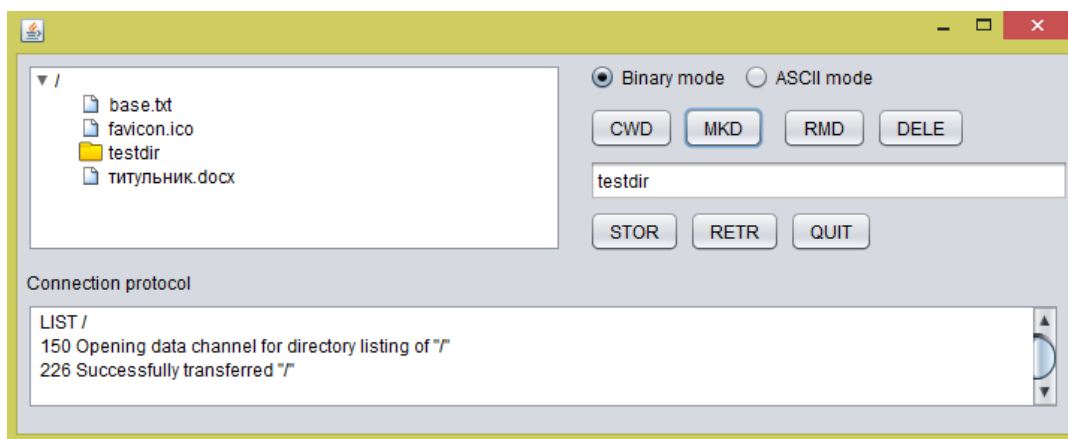
Исходный код программы представлен в приложении.

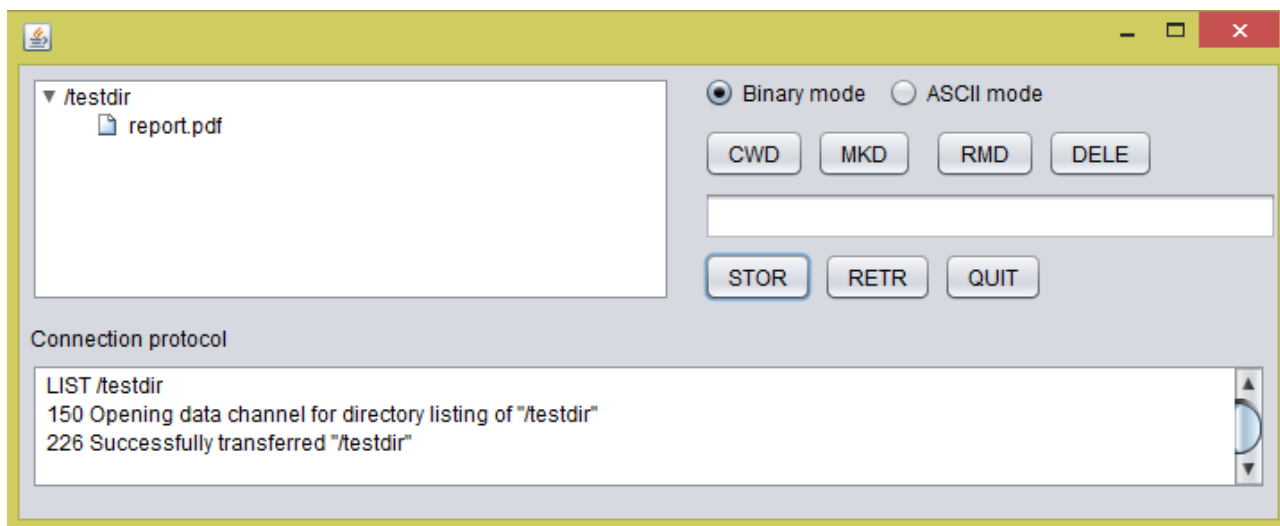
4.2. Демонстрация работы приложения

Файловые серверы, имеющиеся в сети Internet не позволяют продемонстрировать все возможности программы, поэтому демонстрация работы программы будет происходить на локальном сервере.

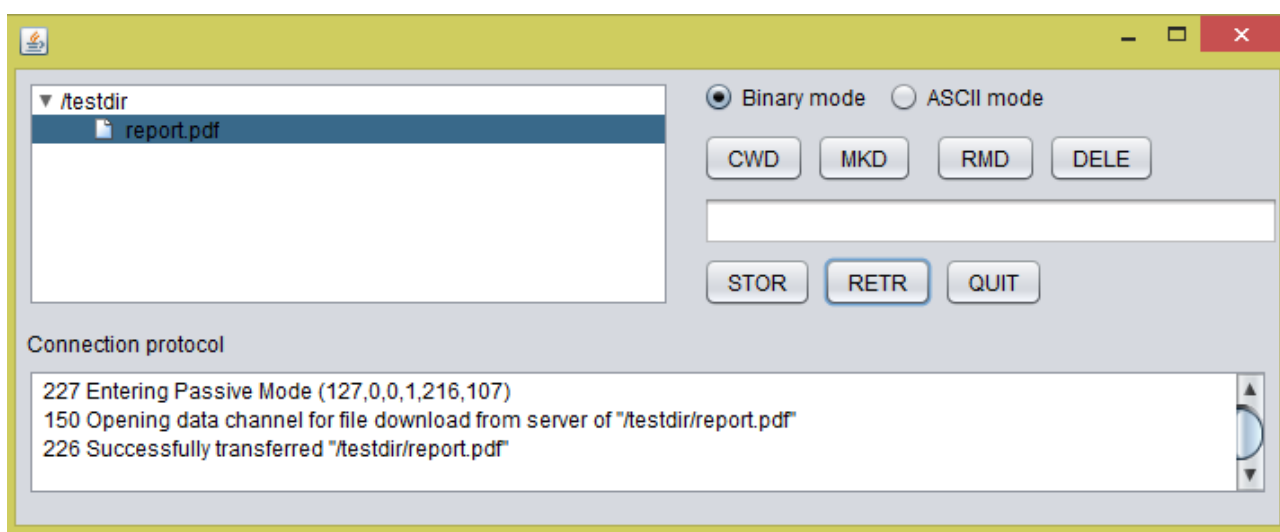


Попробуем для начала добавить новую директорию, перейти в нее и отправить туда какой-нибудь файл.



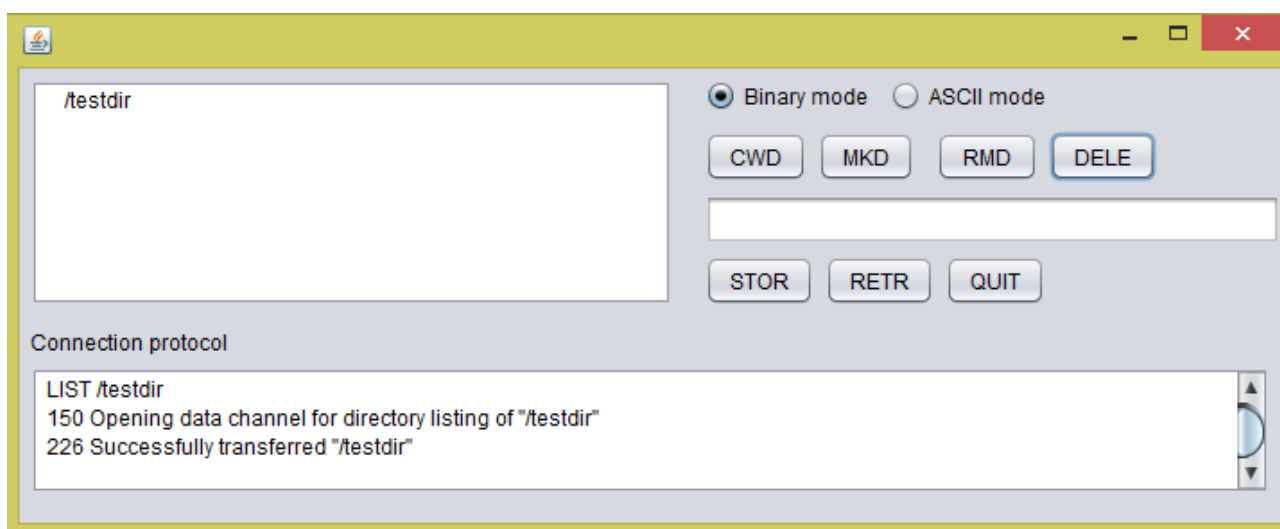


Теперь загрузим этот файл к себе.

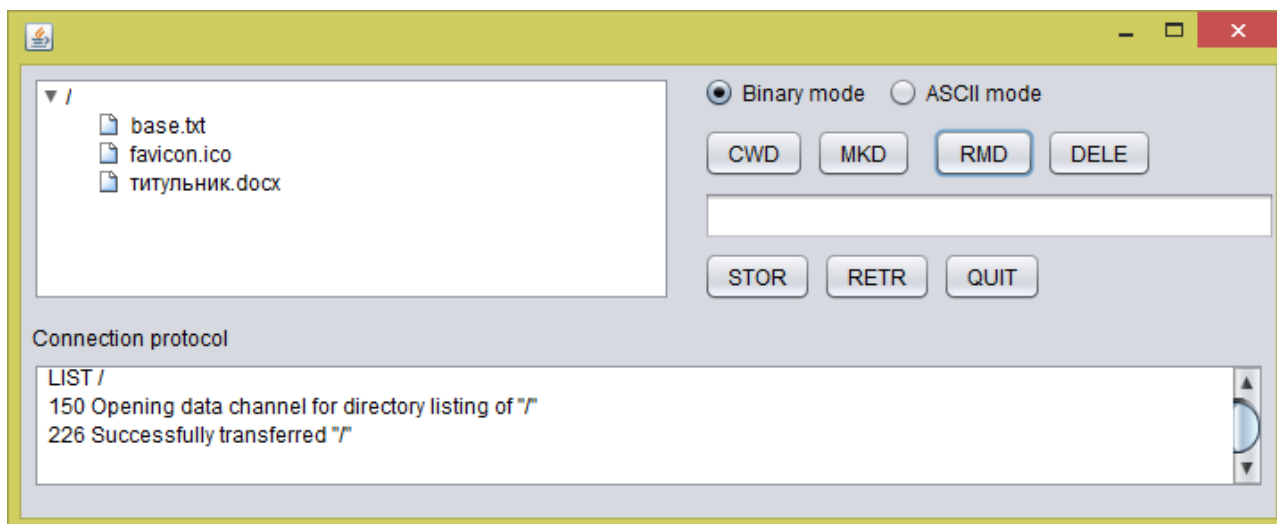


Файл был загружен на рабочий стол.

Теперь удалим файл report.pdf из этого каталога, а затем удалим и сам каталог.



Файл удален.



Как мы видим, директория была удалена. Для выхода из программы используем кнопку "QUIT"

4.3. Архитектура приложения

В состав FTP клиента входят следующие классы:

- 1) FTP_GUI – содержит точку входа в приложение, описание и логику пользовательского интерфейса формы создания нового соединения;
- 2) Connection – содержит описание и логику пользовательского интерфейса формы ftp-клиента;
- 3) SimpleFTP – описывает логику работы FTP-клиента.

Выводы

В ходе выполнения работы были изучены особенности протокола FTP, а также получен навык реализации клиента прикладного протокола на языке программирования Java.

Протокол FTP имеет ряд достоинств и недостатков.

Достоинства:

1. Простота. Отдельное соединение для передачи команд. Служебная информация в текстовом виде.
2. Эффективность. Обычно передается очень мало служебной информации.
3. Возможность контроля прав доступа и их отображения.

Недостатки:

1. Нет поддержки шифрования. Имя пользователя, пароль и файлы передаются в незашифрованном виде.

- 2 . Для передачи и получения каждого файла или списка файлов необходимо устанавливать новое соединение из-за чего при передаче множества файлов снижается эффективность.

Приложение

SimpleFTP.java

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.StringTokenizer;

public class SimpleFTP {

    /**
     * Create an instance of SimpleFTP.
     */
    public SimpleFTP() {

    }

    /**
     * Connects to an FTP server and logs in with the supplied username and
     * password.
     */
    public synchronized String connect(String host, int port, String user,
        String pass) throws IOException {

        String buffer = "";

        if (socket != null) {
            throw new IOException("SimpleFTP is already connected. Disconnect first.");
        }
        socket = new Socket(host, port);
        reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        writer = new BufferedWriter(
            new OutputStreamWriter(socket.getOutputStream()));

        String response = readLine();
        buffer = buffer.concat(response + "\n");
        if (!response.startsWith("220")) {
            throw new IOException(
                "SimpleFTP received an unknown response when connecting to the FTP server: "
                + response);
        }
    }
}
```

```

buffer = buffer.concat(readAll());
buffer = buffer.concat("USER " + user + "\n");
sendLine("USER " + user);

response = readLine();
buffer = buffer.concat(response + "\n");
if (!response.startsWith("331")) {
    throw new IOException(
        "SimpleFTP received an unknown response after sending the user: "
        + response);
}
buffer = buffer.concat(readAll());

buffer = buffer.concat("PASS " + pass + "\n");
sendLine("PASS " + pass);
response = readLine();
buffer = buffer.concat(response + "\n");
if (!response.startsWith("230")) {
    throw new IOException(
        "SimpleFTP was unable to log in with the supplied password: "
        + response);
}
buffer = buffer.concat(readAll());
// Now logged in.
return buffer;
}

/**
 * Disconnects from the FTP server.
 */
public synchronized void disconnect() throws IOException {
    try {
        sendLine("QUIT");
    } finally {
        socket = null;
    }
}

public synchronized String pwdResponse() throws IOException {
    String buffer = "";
    String command = "PWD";
    sendLine(command);
    buffer = buffer.concat(command + "\n");
    buffer = buffer.concat(readLine());
    //buffer = buffer.concat("aaaaaaaaaaaa");
    return buffer.concat(readAll());
}

/**
 * Returns the working directory of the FTP server it is connected to.

```

```

*/
public synchronized String pwd(String pwdResponse) throws IOException {
    String[] parts = pwdResponse.split("\n");
    if ((parts[1].startsWith("257 ") || (parts[1].startsWith("250 "))) {
        int firstQuote = pwdResponse.indexOf("\"");
        int secondQuote = pwdResponse.indexOf("\"", firstQuote + 1);
        if (secondQuote > 0) {
            return pwdResponse.substring(firstQuote + 1, secondQuote);
        }
    }
    return null;
}

public synchronized String mkd(String dir) throws IOException {
    String command = "MKD " + dir;
    String buffer = "";
    buffer = buffer.concat(command + "\n");
    sendLine(command);
    String response = readLine();
    buffer = buffer.concat(response + "\n");
    buffer = buffer.concat(readAll());
    return buffer;
}

public synchronized String rmd(String dir) throws IOException {
    String command = "RMD " + dir;
    String buffer = "";
    buffer = buffer.concat(command + "\n");
    sendLine(command);
    String response = readLine();
    buffer = buffer.concat(response + "\n");
    buffer = buffer.concat(readAll());
    return buffer;
}

public synchronized String dele(String file) throws IOException {
    String command = "DELE " + file;
    String buffer = "";
    buffer = buffer.concat(command + "\n");
    sendLine(command);
    String response = readLine();
    buffer = buffer.concat(response + "\n");
    buffer = buffer.concat(readAll());
    return buffer;
}

/**
 * Changes the working directory (like cd).
 */
public synchronized String cwdResponse(String dir) throws IOException {
    String buffer = "";
    String command = "CWD " + dir;

```

```

        sendLine(command);
        buffer = buffer.concat(command + "\n");
        String response = readLine();
        buffer = buffer.concat(response + "\n");
        buffer = buffer.concat(readAll());
        return buffer;
    }

    public synchronized boolean cwd(String cwdResponse) throws IOException {
        String[] parts = cwdResponse.split("\n");
        if ((parts[1].startsWith("257 ") || (parts[1].startsWith("250 "))) {
            return true;
        }
        return false;
    }

    public synchronized String stor(InputStream inputStream, String filename)
        throws IOException {

        BufferedInputStream input = new BufferedInputStream(inputStream);
        String buf = "";
        String command = "PASV";
        sendLine(command);
        buf = buf.concat(command + "\n");
        String response = readLine();
        buf = buf.concat(response + "\n");
        buf = buf.concat(readAll());

        String ip = null;
        int port = -1;
        int opening = response.indexOf('(');
        int closing = response.indexOf(')', opening + 1);
        if (closing > 0) {
            String dataLink = response.substring(opening + 1, closing);
            StringTokenizer tokenizer = new StringTokenizer(dataLink, ",");
            try {
                ip = tokenizer.nextToken() + "." + tokenizer.nextToken() + "."
                    + tokenizer.nextToken() + "." + tokenizer.nextToken();
                port = Integer.parseInt(tokenizer.nextToken()) * 256
                    + Integer.parseInt(tokenizer.nextToken());
            } catch (Exception e) {
                throw new IOException("SimpleFTP received bad data link information: "
                    + response);
            }
        }

        command = "STOR " + filename;
        buf = buf.concat(command + "\n");
        sendLine(command);
        Socket dataSocket = new Socket(ip, port);
        response = readLine();
        buf = buf.concat(response + "\n");
    }

```

```

if (!response.startsWith("150 ")) {
    buf = buf.concat(readAll());
    return buf;
}
buf = buf.concat(readAll());

BufferedOutputStream output = new BufferedOutputStream(dataSocket
    .getOutputStream());
byte[] buffer = new byte[4096];
int bytesRead = 0;
while ((bytesRead = input.read(buffer)) != -1) {
    output.write(buffer, 0, bytesRead);
}
output.flush();
output.close();
input.close();

response = readLine();
buf = buf.concat(response + "\n");
buf = buf.concat(readAll());
return buf;
}

public synchronized String LIST(String dir)
    throws IOException {
    String buffer = "";
    String command = "PASV";
    sendLine(command);
    buffer = buffer.concat(command + "\n");
    String response = readLine();
    buffer = buffer.concat(response + "\n");
    if (!response.startsWith("227 ")) {
        throw new IOException("SimpleFTP could not request passive mode: "
            + response);
    }
    buffer = buffer.concat(readAll());

    String ip = null;
    int port = -1;
    int opening = response.indexOf('(');
    int closing = response.indexOf(')', opening + 1);
    if (closing > 0) {
        String dataLink = response.substring(opening + 1, closing);
        StringTokenizer tokenizer = new StringTokenizer(dataLink, ",");
        try {
            ip = tokenizer.nextToken() + "." + tokenizer.nextToken() + "."
                + tokenizer.nextToken() + "." + tokenizer.nextToken();
            port = Integer.parseInt(tokenizer.nextToken()) * 256
                + Integer.parseInt(tokenizer.nextToken());
        } catch (Exception e) {
            throw new IOException("SimpleFTP received bad data link information: "
                + response);
        }
    }
}

```

```

    }
}

command = "LIST " + dir;
sendLine(command);
buffer = buffer.concat(command + "\n");
Socket dataSocket = new Socket(ip, port);
response = readLine();
buffer = buffer.concat(response + "\n");
buffer = buffer.concat(readAll());

BufferedInputStream dataInput = new BufferedInputStream(dataSocket
    .getInputStream());
byte b[] = new byte[BLOCK_SIZE];
int amount;
StringBuffer sb = new StringBuffer();
// Read the data into the StringBuffer
while ((amount = dataInput.read(b)) > 0) {
    sb.append(new String(b, 0, amount));
}

String fileSystem = sb.toString();
dataInput.close();

response = readLine();
if (response.startsWith("226")) {
    buffer = buffer.concat(response + "\n");
    buffer = buffer.concat(readAll());
}
buffer = buffer.concat("%" + fileSystem + "\n");

return buffer;
}

/**
 * Sends a file to be stored on the FTP server. Returns true if the file
 * transfer was successful. The file is sent in passive mode to avoid NAT or
 * firewall problems at the client end.
 */
public synchronized String retr(OutputStream outputStream, String filename)
    throws IOException {
    String buf = "";
    BufferedOutputStream outData = new BufferedOutputStream(outputStream);
    String command;
    command = "PASV";
    buf = buf.concat(command + "\n");
    sendLine("PASV");
    String response = readLine();
    buf = buf.concat(response + "\n");
    if (!response.startsWith("227 ")) {
        throw new IOException("SimpleFTP could not request passive mode: "
            + response);
    }
}

```

```

    }
    buf = buf.concat(readAll());

    String ip = null;
    int port = -1;
    int opening = response.indexOf('(');
    int closing = response.indexOf(')', opening + 1);
    if (closing > 0) {
        String dataLink = response.substring(opening + 1, closing);
        StringTokenizer tokenizer = new StringTokenizer(dataLink, ",");
        try {
            ip = tokenizer.nextToken() + "." + tokenizer.nextToken() + "."
                + tokenizer.nextToken() + "." + tokenizer.nextToken();
            port = Integer.parseInt(tokenizer.nextToken()) * 256
                + Integer.parseInt(tokenizer.nextToken());
        } catch (Exception e) {
            throw new IOException("SimpleFTP received bad data link information: "
                + response);
        }
    }

    command = "RETR " + filename;
    sendLine(command + "\n");
    Socket dataSocket = new Socket(ip, port);
    response = readLine();
    buf = buf.concat(response + "\n");
    if (!response.startsWith("150")) {
        throw new IOException("SimpleFTP was not allowed to send the file: "
            + response);
    }
    buf = buf.concat(readAll());
    BufferedInputStream dataInput = new BufferedInputStream(dataSocket
        .getInputStream());
    byte[] buffer = new byte[4096];
    int bytesRead = 0;
    while ((bytesRead = dataInput.read(buffer)) != -1) {
        outData.write(buffer, 0, bytesRead);
    }
    outData.flush();
    outData.close();
    dataInput.close();

    response = readLine();
    buf = buf.concat(response + "\n");
    if (response.startsWith("226")) {
        buf = buf.concat(readAll());
        return buf;
    }
    return buf;
}

/**

```

```

    * Enter binary mode for sending binary files.
    */
    public synchronized String bin() throws IOException {
        String buffer = "";
        String command = "TYPE I";
        sendLine(command);
        buffer = buffer.concat(command + "\n");
        buffer = buffer.concat(readLine() + "\n");
        return buffer.concat(readAll());
    }

    /**
     * Enter ASCII mode for sending text files. This is usually the default
     * mode. Make sure you use binary mode if you are sending images or other
     * binary data, as ASCII mode is likely to corrupt them.
     */
    public synchronized String ascii() throws IOException {
        String buffer = "";
        String command = "TYPE A";
        sendLine(command);
        buffer = buffer.concat(command + "\n");
        buffer = buffer.concat(readLine() + "\n");
        return buffer.concat(readAll());
    }

    /**
     * Sends a raw command to the FTP server.
     */
    private void sendLine(String line) throws IOException {
        if (socket == null) {
            throw new IOException("SimpleFTP is not connected.");
        }
        try {
            writer.write(line + "\r\n");
            writer.flush();
        } catch (IOException e) {
            socket = null;
            throw e;
        }
    }

    private String readLine() throws IOException {
        String line = reader.readLine();
        return line;
    }

    String readAll() throws IOException {
        String response;
        String buffer = "";
        while ((reader.ready()) && ((response = readLine()) != null)) {
            buffer = buffer.concat(response + "\n");
        }
    }

```



```

    return buffer;
}

private Socket socket = null;
/**
 * The offset at which we resume a file transfer.
 */

private BufferedReader reader = null;

private BufferedWriter writer = null;

/**
 * The socket output stream.
 */
private PrintStream outputStream = null;
/**
 * The socket input stream.
 */
private BufferedReader inputStream = null;
private static int BLOCK_SIZE = 4096;
}

```

Connection.java

```

import java.awt.Frame;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.event.TreeSelectionEvent;
import javax.swing.event.TreeSelectionListener;
import javax.swing.text.BadLocationException;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeModel;
import javax.swing.tree.TreeSelectionModel;

public class Connection extends javax.swing.JFrame {

    String currentDir;
    String selectedItem;

    private void addToProtocol(String str) throws Exception {

myTextPane_protocol.getStyledDocument().insertString(myTextPane_protocol.getStyledDocument
().getLength(), str, null);
    }

    private String reloadPWD() throws Exception {

```

```

String pwdResponse = FTP_GUI.ftp.pwdResponse();
addToProtocol(pwdResponse + "\n");
currentDir = FTP_GUI.ftp.pwd(pwdResponse);
return pwdResponse;
}

private void reloadLIST() throws Exception {
    String[] parts = (FTP_GUI.ftp.LIST(currentDir)).split("%");
    addToProtocol(parts[0]);
    String real = parts[1].replaceAll(" {2,}", " ");
    String[] lines = real.split("\r\n");

    DefaultMutableTreeNode root = new DefaultMutableTreeNode(currentDir);
    DefaultTreeModel model = (DefaultTreeModel) tree.getModel();
    model.setRoot(root);

    for (int i = 0; i < lines.length - 1; i++) {
        String[] part = lines[i].split(" ");
        String path;
        if (part[0].startsWith("-")) {
            path = "/flags/file.png";
        } else {
            path = "/flags/folder.png";
        }
        DefaultMutableTreeNode node = new DefaultMutableTreeNode(new TreePosition(part[8],
path));
        root.add(node);
    }
    model.reload();
}

private void goToRoot() throws Exception {
    int endIndex = currentDir.lastIndexOf("/");
    if (currentDir.length() > 1) {
        String cwdResponse = FTP_GUI.ftp.cwdResponse(currentDir.substring(0, endIndex + 1));
        addToProtocol(cwdResponse + "\n");
        if (FTP_GUI.ftp.cwd(cwdResponse)) {
            reloadPWD();
            reloadLIST();
        }
    }
}

/**
 * Creates new form Connection
 */
public Connection() throws Exception {
    initComponents();

    addToProtocol(FTP_GUI.start);

    connectionType.add(connectionBinary);

```

```

        connectionType.add(connectionASCII);
        connectionType.clearSelection();
        connectionType.setSelected(connectionBinary.getModel(), true);
        addToProtocol(FTP_GUI.ftp.bin());

        reloadPWD();

tree.getSelectionModel().setSelectionMode(TreeSelectionMode.SINGLE_TREE_SELECTION);
        tree.addTreeSelectionListener(new TreeSelectionListener() {
            @Override
            public void valueChanged(TreeSelectionEvent e) {
                DefaultMutableTreeNode node = (DefaultMutableTreeNode)
tree.getLastSelectedPathComponent();

                if (node == null) //Nothing is selected.
                {
                    return;
                }

                Object nodeInfo = node.getUserObject();
                try {
                    if (node.isLeaf()) {
                        TreePosition pst = (TreePosition) nodeInfo;
                        selectedItem = pst.getName();
                    } else {
                        goToRoot();
                    }
                } catch (Exception ex) {
                    try {
                        goToRoot();
                    } catch (Exception ex1) {
                    }
                }
            }
        });

        tree.setCellRenderer(new CountryTreeCellRenderer());
        reloadLIST();
    }

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    connectionType = new javax.swing.ButtonGroup();

```

```

connectionBinary = new javax.swing.JRadioButton();
connectionASCII = new javax.swing.JRadioButton();
jLabel2 = new javax.swing.JLabel();
jScrollPane2 = new javax.swing.JScrollPane();
myTextPane_protocol = new javax.swing.JTextPane();
jButton1 = new javax.swing.JButton();
myButton_MKD = new javax.swing.JButton();
myButton_RMD = new javax.swing.JButton();
myButton_DELE = new javax.swing.JButton();
myButton_QUIT = new javax.swing.JButton();
myTextField_path = new javax.swing.JTextField();
myButton_STOR = new javax.swing.JButton();
myButton_RETR = new javax.swing.JButton();
jScrollPane1 = new javax.swing.JScrollPane();
tree = new javax.swing.JTree();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

connectionBinary.setText("Binary mode");
connectionBinary.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        connectionBinaryActionPerformed(evt);
    }
});

connectionASCII.setText("ASCII mode");
connectionASCII.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        connectionASCIIActionPerformed(evt);
    }
});

jLabel2.setText("Connection protocol");

jScrollPane2.setViewportView(myTextPane_protocol);

jButton1.setText("CWD");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

myButton_MKD.setText("MKD");
myButton_MKD.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        myButton_MKDActionPerformed(evt);
    }
});

myButton_RMD.setText("RMD");
myButton_RMD.addActionListener(new java.awt.event.ActionListener() {

```



```

        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
358, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(myTextField_path)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jButton1)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(myButton_MKD)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(myButton_RMD,
javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(myButton_DELE)
            .addContainerGap(69, Short.MAX_VALUE))
        .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createSequentialGroup()
                    .addComponent(connectionBinary)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addComponent(connectionASCII)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(myButton_STOR)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(myButton_RETR)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(myButton_QUIT))))
                .addGap(0, 0, Short.MAX_VALUE))))))
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 117,
Short.MAX_VALUE)
                .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(connectionBinary)
                    .addComponent(connectionASCII)
                    .addGap(11, 11, 11)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(myButton_RMD)

```

```

        .addComponent(jButton1)
        .addComponent(myButton_MKD)
        .addComponent(myButton_DELE))
    .addGap(7, 7, 7)
    .addComponent(myTextField_path, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(myButton_STOR)
    .addComponent(myButton_RETR)
    .addComponent(myButton_QUIT))))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jLabel2)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addContainerGap(16, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

private void connectionBinaryActionPerformed(java.awt.event.ActionEvent evt) {
    try {

myTextPane_protocol.getStyledDocument().insertString(myTextPane_protocol.getStyledDocument
().getLength(), FTP_GUI.ftp.bin(), null);
    } catch (BadLocationException | IOException ex) {
        System.out.println("binary");
    }

}

private void connectionASCIIActionPerformed(java.awt.event.ActionEvent evt) {
    try {

myTextPane_protocol.getStyledDocument().insertString(myTextPane_protocol.getStyledDocument
().getLength(), FTP_GUI.ftp.ascii(), null);
    } catch (BadLocationException | IOException ex) {
        System.out.println("ascii");
    }

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {

        String cwdResponse = FTP_GUI.ftp.cwdResponse(selectedItem);

        addToProtocol(cwdResponse);
        if (FTP_GUI.ftp.cwd(cwdResponse)) {
            reloadPWD();

```

```

        reloadLIST();
    }
} catch (Exception ex) {
}
}

private void myButton_MKDActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String mkdResponse = FTP_GUI.ftp.mkd(myTextField_path.getText());
        addToProtocol(mkdResponse);
        reloadLIST();
    } catch (Exception ex) {
    }
}

private void myButton_QUITActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        FTP_GUI.ftp.disconnect();
    } catch (IOException ex) {
    }
    Frame[] open;
    open = FTP_GUI.getFrames();
    System.exit(0);
}

private void myButton_RMDEActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String rmdResponse = FTP_GUI.ftp.rmd(selectedItem);
        addToProtocol(rmdResponse);
        reloadLIST();
    } catch (Exception ex) {
    }
}

private void myButton_DELEActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String deleResponse = FTP_GUI.ftp.dele(selectedItem);
        addToProtocol(deleResponse);
        reloadLIST();
    } catch (Exception ex) {
    }
}

private void myButton_STORActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        JFrame parentFrame = new JFrame();
        JFileChooser fileChooser = new JFileChooser();
        if (fileChooser.showOpenDialog(parentFrame) == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            String storResponse = FTP_GUI.ftp.stor(new FileInputStream(file), file.getName());

```



```

        addToProtocol(storResponse);
        reloadLIST();
    }
} catch (Exception ex) {

}

}

private void myButton_RETRActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setSelectedFile(new File(selectedItem));
        JFrame parentFrame = new JFrame();
        if (fileChooser.showSaveDialog(parentFrame) == JFileChooser.APPROVE_OPTION) {

            File fileToSave = fileChooser.getSelectedFile();
            String retrResponse = FTP_GUI.ftp.retr(new FileOutputStream(fileToSave),
selectedItem);

            addToProtocol(retrResponse);

        }
        } catch (Exception ex) {
        }
    }

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (Exception ex) {

    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {

```

```

        try {
            new Connection().setVisible(true);
        } catch (Exception ex) {
            Logger.getLogger(Connection.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
});
}

// Variables declaration - do not modify
private javax.swing.JRadioButton connectionASCII;
private javax.swing.JRadioButton connectionBinary;
private javax.swing.ButtonGroup connectionType;
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JButton myButton_DELETE;
private javax.swing.JButton myButton_MKD;
private javax.swing.JButton myButton_QUIT;
private javax.swing.JButton myButton_RETR;
private javax.swing.JButton myButton_RMD;
private javax.swing.JButton myButton_STOR;
private javax.swing.JTextField myTextField_path;
private javax.swing.JTextPane myTextPane_protocol;
private javax.swing.JTree tree;
// End of variables declaration
}

```

FTP_GUI.java

```

import java.awt.Frame;
import javax.swing.JOptionPane;

public class FTP_GUI extends javax.swing.JFrame {

    public static SimpleFTP ftp;
    public static String start;

    /**
     * Creates new form FTP_GUI
     */
    public FTP_GUI() {
        initComponents();
    }

    private void showMessageDialog() {
        int reply = JOptionPane.showConfirmDialog(null, "Создать новое подключение?",
"Connection Error", JOptionPane.YES_NO_OPTION);
        // В зависимости от решения пользователя:
        if (reply == JOptionPane.NO_OPTION) {
            Frame[] open;

```

```

        open = FTP_GUI.getFrames();
        open[0].dispose();
    }
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    myLabel_serverName = new javax.swing.JLabel();
    myLabel_login = new javax.swing.JLabel();
    myLabel_password = new javax.swing.JLabel();
    myTextField_serverName = new javax.swing.JTextField();
    myTextField_login = new javax.swing.JTextField();
    myTextField_password = new javax.swing.JTextField();
    myButton_connect = new javax.swing.JButton();
    myLabel_port = new javax.swing.JLabel();
    myTextField_port = new javax.swing.JTextField();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("FTP Client");
    setResizable(false);

    myLabel_serverName.setText("Server name:");

    myLabel_login.setText("Login:");

    myLabel_password.setText("Password:");

    myTextField_serverName.setText("ftp.funet.fi");

    myTextField_login.setText("username");

    myTextField_password.setText("password");

    myButton_connect.setText("Connect");
    myButton_connect.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            myButton_connectActionPerformed(evt);
        }
    });

    myLabel_port.setText("Port:");

    myTextField_port.setText("21");

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

```

```

getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
false)
            .addComponent(myButton_connect, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGroup(layout.createSequentialGroup()
                .addComponent(myLabel_serverName)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(myTextField_serverName,
javax.swing.GroupLayout.PREFERRED_SIZE, 120,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(52, 52, 52)
                        .addComponent(myLabel_port)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(myTextField_port))
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                        .addGap(19, 19, 19)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(myLabel_password)
                    .addComponent(myLabel_login)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(myTextField_login)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(myTextField_password)
                        .addGap(6, 6, 6))))
                    .addContainerGap())
                );
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(myLabel_serverName)
            .addComponent(myTextField_serverName,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(myLabel_port)

```

```

        .addComponent(myTextField_port, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(myLabel_login)
        .addComponent(myTextField_login, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(myLabel_password)
        .addComponent(myTextField_password,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(myButton_connect)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

private void myButton_connectActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        ftp = new SimpleFTP();

        start = ftp.connect("localhost", 21, "username", "password");
        //start = ftp.connect(myTextField_serverName.getText(),
Integer.parseInt(myTextField_port.getText()), myTextField_login.getText(),
myTextField_password.getText());

        this.setVisible(false);
        Connection myConnection = new Connection();
        myConnection.setVisible(true);
    } catch (Exception ex) {
        showMessageDialog();
    }
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {

```

```

        javax.swing.UIManager.setLookAndFeel(info.getClassName());
        break;
    }
}
} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(FTP_GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(FTP_GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(FTP_GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(FTP_GUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new FTP_GUI().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton myButton_connect;
private javax.swing.JLabel myLabel_login;
private javax.swing.JLabel myLabel_password;
private javax.swing.JLabel myLabel_port;
private javax.swing.JLabel myLabel_serverName;
private javax.swing.JTextField myTextField_login;
private javax.swing.JTextField myTextField_password;
private javax.swing.JTextField myTextField_port;
private javax.swing.JTextField myTextField_serverName;
// End of variables declaration
}

```