# Разработка SMTP- клиента на языке Java

## Задание

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

## Основные возможности.

Приложение должно реализовывать следующие функции:

1. Создание нового письма, включающего такие поля, как From (от- правитель), To (получатель), Subject (тема), Body (текст)
2. Формирование всех необходимых заголовков письма, с тем, чтобы приёмная сторона не рассматривала данное письмо как спам.
3. Подключение к указанному SMTP-серверу и отсылка созданного письма

## Поддерживаемые команды

Разработанное приложение должно реализовывать следующие команды протокола SMTP:

- HELO – передача серверу информации о домене пользователя
- MAIL FROM – передача серверу адреса отправителя письма
- RCPT TO – передача серверу адреса получателя письма
- DATA – передача серверу тела письма
- QUIT – завершение сеанса связи

## Настройка приложения

Разработанное приложение должно обеспечивать настройку следующих параметров:

1. Собственное доменное имя для передачи в команде HELO
2. Адрес отправителя
3. IP-адрес или доменное имя сервера SMTP

## Методика тестирования

При проверки работоспособности приложения применялся браузер Google Chrome, Yandex почта. Средствами приложения производилась передача письма на указанный электронный адрес.

## Функциональность

Приложение общается с пользователем через консоль. Пользователь вводит адрес своего ящика, пароль и адрес получателя. После аутентификации пользователь вводит тему сообщения и непосредственно текст.

### Нефункциональные требования

На протяжении всего общения с сервером на консоль выводятся сообщения, присылаемые в ответ сервером. Для обычного пользователя они не нужны, но в ходе разработки помогают определить, в каком месте общение пошло не так.

SMTP сервер яндекса требует защищенную передачу данных, поэтому используется протокол SSL - уровень защищенных сокетов.

Накладываемые ограничения: Пользователь может ввести сообщение любой длины, но оно должно заканчиваться точкой на новой строке, чтобы программа поняла, что это действительно последний символ.

## Анализ задачи

SMTP - client представляет сервис клиенту для общения по протоколу SMTP для передачи почты между отправителем и получателем, используя сервер-посредник.

Клиентская часть - является компьютером, получающим возможность общения с компьютером на сервере для осуществления необходимых операций по передаче почты.

## Команды прикладного протокола

| Имя | Действие | Длина |
|---|---|---|
| Connect | Инициализация соединения | 0 |
| HELO | Приветствие от клиента | 2048 |
| AUTH | логин и пароль клиента | 2048 |
| MAIL_FROM | адрес отправителя | 2048 |
| RCPT_TO | адрес получателя | 2048 |
| DATA | Тело сообщения | 2048 |
| QUIT | завершение общения | 2048 |

Клиент отправляет команды на smtp сервер яндекса, который проверяет корректность и присылает ответы.

### Архитектура приложения

Для реализации данного приложения было разработано несколько классов: Mail - класс, реализующий составление письма (тема, сообщение) SMTPClient - класс, реализующий часть общения с клиентом, реализацию вспомогательных функций. ServerConnection - класс, реализующий сетевую часть приложение (создание сокета, общение с сервером, отправка письма) Utils - класс, для вспомогательных функций.

### Возможные улучшения приложения

Основное возможное улучшение - произвести рефакторинг кода, так как при реализации

архитектура приложения довольно сильно изменялась, что в итоге привело к не самому лучшему виду.

## Тестирование приложения

В ходе разработке применялось ручное тестирование, когда смотрись ответы от сервера и анализировались результаты, также использовался браузер для просмотра приходящих сообщений.

Также применялось модульное тестирование, когда модули программы тестировались специальными программными способами. Тестовое покрытие составило около 40% программы, многие функции было трудно протестировать из-за плохой архитектуры приложения.

## Исходный код приложения

Файл Mail.java

```
package smtpclient;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author nikita
 */
public class Mail {
    Utils util;
    String BODY = "";
    String SUBJ;
    String XMAILER = "X-MAILER: NIKMailer\r\n";
    String MIME = "'MIME-Version: 1.0\r\n";
    String CONTENT_TYPE = "CONTENT-TYPE: text/plain\r\n";
    String END = "\r\n.\r\n";
    public String mailSubj() {
        util = new Utils();
        return util.enterSubj();
    }
    public String mailBody(String body) {
       BODY=XMAILER+MIME+CONTENT_TYPE+"\r\n"+body+ END;
       return BODY;
    }

}
```

SMTPClient.java:

```
package smtpclient;

/**
 *
 * @author nikita
 */
import java.io.*;
import java.net.*;
import java.util.Base64;
import java.util.Scanner;
import javax.net.SocketFactory;
import javax.net.ssl.SSLSocketFactory;

public class SMTPClient {

    static PrintStream ps = null;
    static DataInputStream dis = null;
    static Socket smtp;
    static SocketFactory factory;
    static String SMTPServer;

    SMTPClient() throws IOException {
        factory = SSLSocketFactory.getDefault();

        smtp = factory.createSocket("smtp.yandex.ru", 465);
        OutputStream os = smtp.getOutputStream();
        ps = new PrintStream(os);
        InputStream is = smtp.getInputStream();
        dis = new DataInputStream(is);
    }

    public static String encodeString(String str) {
        byte[] b = str.getBytes();
        String encodedStr = Base64.getEncoder().encodeToString(b);
        return encodedStr;
    }

    public static boolean isAddressCorrect(String addr) {
        if (addr.length() < 6) {
            return false;
        }
            boolean b=addr.indexOf('@') == -1 || addr.indexOf('.') == -1;
        if (b) {
            return false;
        }
        String domain = addr.substring(addr.indexOf('.') + 1);
        for (int i = 0; i < domain.length(); i++) {
            if (domain.charAt(i) >= 'a' && domain.charAt(i) <= 'z' ||
domain.charAt(i) >= 'A'
                    && domain.charAt(i) <= 'Z') {
                continue;
            } else {
                return false;
            }
        }

        if ((addr.indexOf('@') == 0
```

```
                || addr.indexOf('@') == (addr.indexOf('.') + 1)
                || addr.indexOf('@') == (addr.indexOf('.') - 1))) {
            return false;
        }

        return true;
    }

    public static void main(String args[]) throws UnsupportedEncodingException,
IOException {

        new SMTPClient();
        ServerConnection con = new ServerConnection();

        Scanner scan = new Scanner(System.in);
        System.out.println("Enter dest");
        con.dest = scan.nextLine();
        if (!isAddressCorrect(con.dest)) {
            return;
        }
        System.out.println("Enter login");
        con.source = scan.nextLine();
        if (!isAddressCorrect(con.source)) {
            return;
        }
        System.out.println("Enter password");
        con.pass = scan.nextLine();

        con.sendMail(encodeString(con.pass), encodeString(con.source), ps, dis);
    }


}
```

ServerCoonection.java:

```
package smtpclient;

import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.PrintStream;
import java.net.Socket;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.net.SocketFactory;
import javax.net.ssl.SSLSocketFactory;

/**
 *
 * @author nikita
 */
public class ServerConnection {
```

```
 //   PrintStream ps = null;
//    DataInputStream dis = null;
    Socket smtp;
    SocketFactory factory;
    String SMTPServer;
    int PORT;
    Mail mail;
    String pass;
    String source;
    String dest;

    String HELO = "HELO ";
    String MAIL_FROM = "MAIL FROM: ";
    String RCPT_TO = "RCPT TO: ";

    String DATA = "DATA";
    String AUTH = "AUTH LOGIN";
    String SUBJECT = "SUBJECT: ";
    String TO = "TO: " + dest;
    String QUIT = "QUIT";
    String loc;

    public ServerConnection() {
        mail = new Mail();

   //     factory = SSLSocketFactory.getDefault();
      //  SMTPServer = serv;
      //  PORT = port;
      /*  try {
            smtp = smtp = factory.createSocket(SMTPServer, PORT);
            OutputStream os = smtp.getOutputStream();
            ps = new PrintStream(os);
            InputStream is = smtp.getInputStream();
            dis = new DataInputStream(is);
        } catch (IOException ex) {
            Logger.getLogger(ServerConnection.class.getName()).log(Level.SEVERE,
null, ex);
        }*/
    }

    public void send(String str, PrintStream ps1) {
        ps1.println(str);      // посылка строки на SMTP
        System.out.println("sent: " + str);
    }
    /*
     public String receive() throws IOException {
     String readstr;
     readstr = dis.readLine(); // получение ответа от SMTP
     //   System.out.println("SMTP respons: " + readstr);
     return readstr;
     }
     */

    public void checkInput(DataInputStream dis) throws IOException {
        String readstr;

        readstr = dis.readLine();
```

```
        System.out.println("SMTP respons: " + readstr);
        if (readstr.charAt(0) == '5' || readstr.charAt(0) == '4') {
            return;
        }

    }

    public String enterMessage() {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter message");
        String tmp;
        String body = "";
        while (true) {
            tmp = scan.nextLine();
            if (tmp.equals(".")) {
                break;
            }
            body = body + " " + tmp;
        }
        return body;
    }

    public void sendMail(String pass, String login, PrintStream ps1,DataInputStream
dis1 ) {
        try {
            String tmp;
            send(HELO + loc, ps1);
            checkInput(dis1);          // получение ответа SMTP
            send(AUTH, ps1);
            checkInput(dis1);
            send(login, ps1);
            checkInput(dis1);
            send(pass, ps1);
            checkInput(dis1);
            send(MAIL_FROM + source, ps1);
            checkInput(dis1);
            send(RCPT_TO + dest, ps1);
            checkInput(dis1);
            send(DATA, ps1);
            checkInput(dis1);

            send("FROM: " + source, ps1);

            send(SUBJECT + mail.mailSubj(), ps1);

            send("TO: " + dest, ps1);
            checkInput(dis1);

            tmp = enterMessage();
            //    System.out.println(tmp);
            send(mail.mailBody(tmp), ps1);
            checkInput(dis1);
            send(QUIT, ps1);
        } catch (IOException e) {
            System.out.println("Error sending: " + e);
        }
```

```
    }
}
```

Utils.java:

```
package smtpclient;

import java.util.Scanner;

/**
 *
 * @author nikita
 */
public class Utils {
     public String enterSubj(){
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter subject");
        return scan.nextLine();
    }

}
```

# Сервер протокола FTP, функционирующий в активном режиме

## Задание

Разработать приложение для операционных систем семейства Windows или Linux, обеспечивающее функции FTP-сервера.

## Анализ задачи

Разработанное приложение реализует функции сервера, после запуска к серверу может подключиться клиент (на 21й порт), после соединения начинается общение, клиент отправляет команды, которые обрабатывает сервер. В случае, когда команда требует передачи информации, клиент отправляет номер порта, на который сервер устанавливает соединение и по этому каналу происходит передача данных.

## Функциональные требования:

1. Хранение идентификационной и аутентификационной информации нескольких пользователей
2. Поддержка анонимного входа (пользователь anonymous)
3. Обработка подключения клиента
4. Выдача по запросу клиента содержимого каталога
5. Навигация по системе каталогов
6. Создание нового каталога

7. Удаление каталога
8. Посылка по запросу клиента содержимого указанного файла
9. Приём по запросу клиента содержимого указанного файла
10. Удаление указанного файла
11. Протоколирование соединения сервера с клиентом

## Поддерживаемые команды

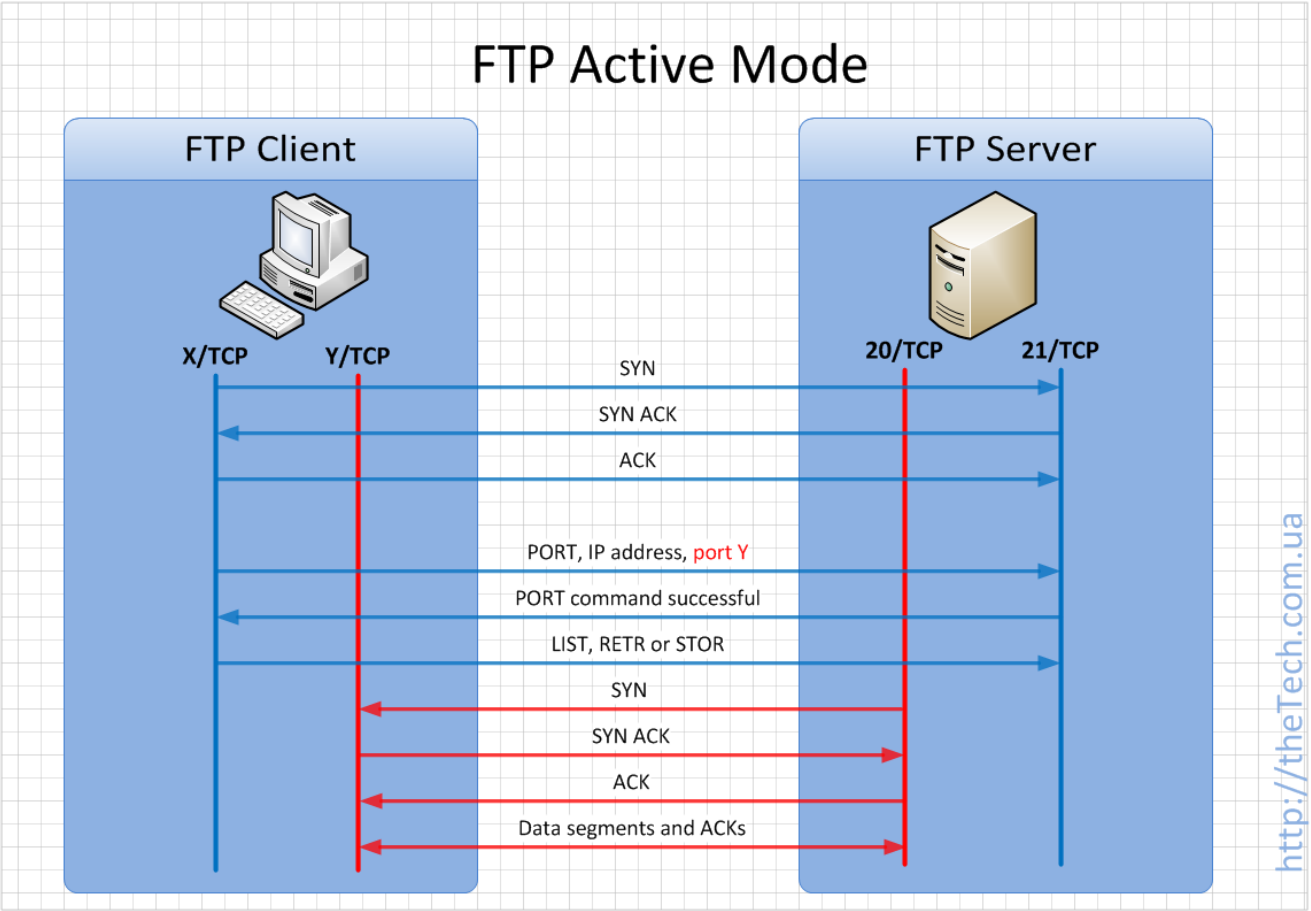
Разработанное приложение реализовывает следующие команды протокола FTP: * USER – получение от клиента идентификационной информации пользователя * PASS – получение от клиента пароля пользователя * LIST – отправка клиенту расширенной информации о списке файлов каталога * NLST – отправка клиенту сокращённой информации о списке фай- лов каталога * CWD – смена текущего каталога сервера * MKD – создание каталога * RMD – удаление каталога * DELE – удаление файла на сервере * PORT – получение параметров сокета клиента (адреса и порта), осуществляющего приём и передачу данных * RETR – посылка файла клиенту * STOR – запись полученного от клиента файла * DELE – удаление файла * QUIT – завершение сеанса

## Взаимодействие сервера и клиента

Сервер создает прослушивающий сокет на 21 порту, туда подключается клиент. По этому каналу производится обмен командами. Когда клиент отправляет команду, например, ls - то сначала серверу приходит команда:

```
EPRT |2|::1|5282|
```

Где сначала номер протокола, затем адрес и номер порта, куда должен подключиться сервер для передачи данных. После передачи сервер отключается от временного порта и ждет новых команд. —




## Функционирование сервера

Настоящее приложение поддерживает базовые команды протокола ftp. Команда ls - выводит список файлов, в текущем каталоге. dir - выводит список файлов с подробной информацией: полный путь к файлу на сервере, дату и время последней модификации, размер. Команда put- позволяет загрузить на сервер указанный файл с указанным именем. Get - позволяет скачать файл с сервера в указанную директорию с произвольным именем. PWD - выводит текущую директорию сервера, в которой находится пользователь. Команда cd - позволяет перемещаться по каталогам. Выше корневого каталога пользователь подняться не может. Команда удаления директории производит удаление указанного каталога, даже если в нем содержаться файлы или другие подкаталоги. Также есть команда удаление отельного файла.

## Архитектура приложения

Для реализации данного приложения было разработано несколько классов: FTPServer - класс, в котором реализована сетевая часть, происходит создание сокета и создание объекта нового класса, реализующий сетевой диалог непосредственно с подключившимся клиентом (класс Transferfile). Transferfile реализует подключение к указанному клиентом порту, читает и записывает команды и данные в 2 сокета (для данных и для команд), В зависимости от того, какая команда пришла от клиента вызывается соотетствующий метод из класса WorkWithDirectory. WorkWithDirectory - класс, реализующий работу с каталогами (удаление, загрузку и т.п.). Основная бизнес-логика работы сервера реализована именно в этом классе.

Класс Utils - класс, реализующий вспомогательные задания, например перевод числа типа long в дату и время.

### Тестирование приложения

Для взаимодействия с сервером использовалась стандартная утилита ftp.exe. Данная утилита не позволяет отправлять команды серверу, которые не поддерживаются протоколом, поэтому она очень хорошо подходит на роль метода проверки приложения на соответствие. Тестирование осуществлялось в ручном режиме, когда производилось наблюдение отправляются ли файлы или нет, зависает ли клиент или нет. Также для тестирования был организован промежуточный вывод команд, которые приходят от клиента.

### Сложности при разработке

Основной проблемой явилось то, что по-началу было трудно понять, что приходит от клиента, так как приходил набор символов, который сходу было трудно интерпретировать. Решить проблему помогло чтение RFC, благодаря чему я понял формат команд. Второй проблемой стало соблюдение последовательности ответов от сервера (преимущественно, когда идет общение по порту данных и порту команд), разобравшись с последовательностью, приведенной на картинке выше, разработка пошла хорошо.

### Возможные улучшения

Главным направлением улучшения приложения является реализация новых команд протокола ftp. Также можно реализовать работу сервера в пассивном режиме, однако это уже выходит за рамки текущего задания, направленного на ознакомление с протоколом. Также можно рассмотреть улучшение кода как возможное улучшение. Данная реализация - первый опыт разработки ftp сервера, поэтому при кодировании были использованы не лучшие решения, которые, безусловно, можно улучшить.

### Исходный код

FTPServer1.java:

```
package ftpserver1;

/**
 *
 * @author nikita
 */
//FtpServer.java
import java.net.*;
import java.io.*;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;

public class FTPServer1 {
```

```
    public static void main(String args[]) throws Exception {
        ServerSocket soc = new ServerSocket(21);
        System.out.println("FTP Server Started on Port Number 21");
        while (true) {
            System.out.println("Waiting for Connection ...");
            transferfile t = new transferfile(soc.accept());

        }
    }
}

class transferfile extends Thread {

    Socket ClientSoc;
    String curDir;
    DataInputStream din, din2;
    DataOutputStream dout, dout2;
    WorkWithDirectory direct;

    transferfile(Socket soc) {
        try {
            // dataSoc= null;
            direct = new WorkWithDirectory();
            curDir = "C:\\FTP SERVER DIRECTORY";
            ClientSoc = soc;
            din = new DataInputStream(ClientSoc.getInputStream());
            dout = new DataOutputStream(ClientSoc.getOutputStream());
            System.out.println("FTP Client Connected ...");
            dout.writeBytes("220 FTP server ready\n");
            start();

        } catch (Exception ex) {
        }
    }

    Socket dataSoc;

    void connectNewPort(int port) {
        try {
            dataSoc = new Socket();
            //  dataSoc = new Socket("127.0.0.1", port,
InetAddress.getByName(null), 20);
            dataSoc.bind(new InetSocketAddress(20));
            dataSoc.connect(new InetSocketAddress("::1", port));
            din2 = new DataInputStream(dataSoc.getInputStream());
            dout2 = new DataOutputStream(dataSoc.getOutputStream());

        } catch (IOException ex) {
            Logger.getLogger(transferfile.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    public void listCommand(int port) throws IOException {
        dout.writeBytes("150 ok, it`s xlist command\n");
        connectNewPort(port);
        direct.fileParams(curDir);
```

```
        for (int i = 0; i < direct.fileParams.size(); i++) {
            dout2.writeBytes(direct.fileParams.get(i) + "\n");
        }
        direct.fileParams.clear();
        dataSoc.close();
        dout.writeBytes("226 Transfer complete.\n");
    }

    public void xlistCommand(int port) throws IOException {
        dout.writeBytes("150 ok, it`s list command\n");
        connectNewPort(port);

        direct.getFileNames(curDir);
        for (int i = 0; i < direct.fileNames.size(); i++) {
            dout2.writeBytes(direct.fileNames.get(i) + "\n");
        }
        direct.fileNames.clear();
        dataSoc.close();
        dout.writeBytes("226 Transfer complete.\n");
    }

    public void sendCommand(int port, String fileName) throws IOException {
        dout.writeBytes("150 ok, it`s get command\n");
        connectNewPort(port);
        direct.sendFile(curDir + "\\" + fileName, dout2);
        dout2.close();
        dout.writeBytes("226 Transfer complete.\n");
    }

    public void putCommand(int port, String fileName) throws IOException {
        dout.writeBytes("150 ok, it`s send command\n");
        connectNewPort(port);
        direct.receiveFile(curDir + "\\" + fileName, din2,
dataSoc.getReceiveBufferSize());
        din2.close();
        dout.writeBytes("226 Receiving complete.\n");
    }

    public void mkdirCommand(String dirName) throws IOException {
        //    dout.writeBytes("150 ok, it`s mkdir command\n");
        //      connectNewPort(port);
        direct.mkDir(curDir + "\\" + dirName);
        //      din2.close();
        dout.writeBytes("226 new directory ok\n");
    }

    public void deleteDirCommand(String dirName) throws IOException {
        File dir = new File(curDir + "\\" + dirName);
        direct.deleteDir(dir);
        dout.writeBytes("226 dir deleted\n");
    }

    public void deleteFileCommand(String fileName) throws IOException {
        direct.delereFile(curDir + "\\" + fileName);
        dout.writeBytes("226 file deleted\n");
    }
    public void cwdCommand(String dirName) throws IOException{
```

```
        curDir= direct.cwd(dirName, curDir);
        dout.writeBytes("226 directory changed\n");
    }
    public void xpwdCommand() throws IOException{
       dout.writeBytes("220"+" "+curDir+"\n");
    }
    public void quitCommand() throws IOException{
         dout.writeBytes("bye\n");
    }
    public void run() {
        int tmpPort = 0;
        try {
            String str2;
            String str = din.readLine();
            System.out.println(str);
            if (str.compareTo("USER user ftp") == 0) {
                dout.writeBytes("230 Login successful.\n");
            }
            while (true) {
                str = din.readLine();
                System.out.println(str);
                if (str.substring(0, 4).equalsIgnoreCase("EPRT")) {
                    tmpPort = Integer.parseInt(str.substring(12, 17));
                    System.out.println(tmpPort);
                    dout.writeBytes("220 Port command successful\n");
                    str = din.readLine();
                    System.out.println(str);
                }

                if (str.substring(0, 4).compareToIgnoreCase("NLST") == 0) {
                    xlistCommand(tmpPort);
                    continue;
                }
                if (str.substring(0, 4).compareToIgnoreCase("RETR") == 0) {
                    sendCommand(tmpPort, str.substring(5));
                    continue;
                }
                if (str.substring(0, 4).compareToIgnoreCase("STOR") == 0) {
                    putCommand(tmpPort, str.substring(5));
                    continue;
                }
                if (str.substring(0, 4).compareToIgnoreCase("XMKD") == 0) {
                    mkdirCommand(str.substring(5));
                    continue;
                }
                if (str.substring(0, 4).compareToIgnoreCase("XRMD") == 0) {
                    deleteDirCommand(str.substring(5));
                    continue;
                }
                if (str.substring(0, 4).compareToIgnoreCase("DELE") == 0) {
                    deleteFileCommand(str.substring(5));
                    continue;
                }
                if (str.substring(0, 4).compareToIgnoreCase("LIST") == 0) {
                    listCommand(tmpPort);
                    continue;
                }
```

```
                if (str.substring(0, 3).compareToIgnoreCase("CWD") == 0) {
                    cwdCommand((str.substring(4)));
                    continue;
                }
                if (str.substring(0, 4).compareToIgnoreCase("XPWD") == 0) {
                    xpwdCommand();
                    continue;
                }
                if (str.substring(0, 4).compareToIgnoreCase("QUIT") == 0) {
                    quitCommand();
                    continue;
                }
            }

        } catch (IOException ex) {
            Logger.getLogger(transferfile.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
}
```

Utils.java:

```
package ftpserver1;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.TimeZone;

/**
 *
 * @author nikita
 */
public class Utils {
    public String getTime(final long timeInMillis)
{
    final SimpleDateFormat format = new SimpleDateFormat("dd.MM.yyyy, HH:mm:ss");
    final Calendar c = Calendar.getInstance();
    c.setTimeInMillis(timeInMillis);
    c.setTimeZone(TimeZone.getDefault());
    return format.format(c.getTime());
}
}
```

WorkWithDirectory.java:

```
package ftpserver1;

import java.io.BufferedOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.StringTokenizer;

/**
 *
 * @author nikita
 */
public class WorkWithDirectory {

    private File[] files;

    ArrayList<String> fileNames = new ArrayList();
    ArrayList<String> fileParams = new ArrayList();
    Utils util;
    String DIR;

    WorkWithDirectory() {
        DIR = "C:\\FTP SERVER DIRECTORY";
        util = new Utils();
    }

    private void getListFiles(String path) {
        File myFolder = new File(path);
        files = myFolder.listFiles();
    }

    public void fileParams(String path) {
        getListFiles(path);

        for (int i = 0; i < files.length; i++) {
            fileParams.add(files[i].toString() + " " +
util.getTime(files[i].lastModified()) + " "
                    + files[i].length());
        }

    }
 public String cwd(String dirName, String curDir) {
        StringTokenizer st;
        String tmpDir=null;
        int index;
        if (dirName.equalsIgnoreCase("..")) {
            if (curDir.equalsIgnoreCase(DIR)) {
                return DIR;
            }
            else {
                index=curDir.lastIndexOf("\\");
                return curDir.substring(0, index);
            }
        }
        else {
            return curDir+"\\"+dirName;
        }
    }
```

```
    public int getFileNames(String path) {
        getListFiles(path);
        String tmpStr;
        StringTokenizer st;
        for (int i = 0; i < files.length; i++) {

            tmpStr = files[i].toString();
            st = new StringTokenizer(tmpStr, "\\");
            while (true) {
                if (st.countTokens() != 1) {
                    st.nextElement();
                } else {
                    fileNames.add(st.nextToken());
                    break;
                }
            }

        }
        return files.length;
    }

    public void sendFile(String name, DataOutputStream dout) throws IOException {
        File f = new File(name);
        if (!f.exists()) {
            dout.writeBytes("500 can`t find file\n");
            return;
        } else {

            FileInputStream fin = new FileInputStream(f);
            int ch = fin.read();
            while (ch != -1) {
                dout.writeByte((ch));
                ch = fin.read();
            }

            fin.close();
        }
    }

    public void receiveFile(String name, DataInputStream din, int size) throws
FileNotFoundException, IOException {
        String filename = name;
        byte[] bytes = new byte[size];
        int count;
        File f = new File(name);
        FileOutputStream fos = new FileOutputStream(f);
        BufferedOutputStream bos = new BufferedOutputStream(fos);
        while ((count = din.read(bytes)) > 0) {
            bos.write(bytes, 0, count);
        }
        bos.close();
        fos.close();
    }

    public void mkDir(String dirName) {
        File f = new File(dirName);
        f.mkdir();
```

```
    }

    public void deleteDir(File dir) {
        if (dir.isDirectory()) {
            String[] children = dir.list();
            for (int i = 0; i < children.length; i++) {
                File f = new File(dir, children[i]);
                deleteDir(f);
            }
            dir.delete();
        } else {
            dir.delete();
        }
    }

    public void delereFile(String FileName) {
        File f = new File(FileName);
        f.delete();

    }


}
```

### Выводы:

В ходе работы было реализовано два приложения, работающие по протоколам прикладного уровня: SMTP клиент и FTP сервер. Данная работа помогла разобраться в алгоритмах данных протоколов. Научился реализовывать стандарты взаимодействия по данным протоколам.