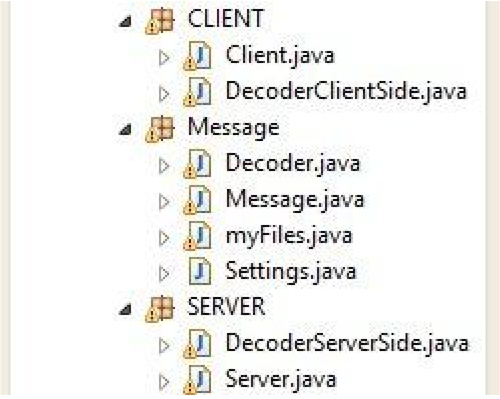
Hamid Hooshmandi

Assignment 3

Thursday, March 7, 2013

# Program Description:

The program contains following packages and classes:



- CLIENT
  - Client.java
  - DecoderClientSide.java
- Message
  - Decoder.java
  - Message.java
  - myFiles.java
  - Settings.java
- SERVER
  - DecoderServerSide.java
  - Server.java

## Server Package:

DecoderServerSide.java
Decode messages coming from the client

## Client Package:

DecoderClientSide.java
Decode messages coming from the server

## Message Package:

### Decoder:

**getMessage function:**

- gets the InputStream and returns a Message object containing message byte array
- it also return back the inputstream, to be later used for reading messages with body contents

```
public static Message getMessage(InputStream in, boolean containsBody) throws
IOException{
    .
    .
    .
    .
        msg.setOutputStm(messageBuffer.toByteArray());
        return msg;
}
```

# Synchronization:

to synchronze files ,3 arraylists of myFile type is created

ArrayList<myFiles> filesListSnapshot
ArrayList<myFiles> filesListClient
ArrayList<myFiles> fileListServer

filled with files of client, server, and snapshot files,(Snapshot file is an xml file. Representing files on the client computer
) these files then compared together in the **compareFileLists** function.
The function does the following

**compareFileLists function:**

- gets 3 arraylists of myFile type and compare them together,
- it returns another arraylist of myFile type, with a code corresponding to each filename, indicating what has to be done to the file
- example: 001 means file exists on the server but doesn't exist on the snapshot or the client

| snapshot | Client | Server | | |
|---|---|---|---|---|
| 1 | 1 | 0 | Delete file on the client | |
| 1 | 0 | 1 | Delete file on the server | |
| 1 | 1 | 1 | Compare | |
| 1 | 0 | 0 | | File was deleted on both client and server |
| | | | | |
| 0 | 1 | 0 | Put file on the server | |
| 0 | 1 | 1 | | File was created on both client and server |
| | | | | |
| 0 | 0 | 1 | Get file from the server | |
| | | | | |

# Test cases:

## Login test

I tested clients and server on "localhost"

run the server with following command line arguments:
-p 20112

Run the client with the following command line arguments:
-d ./CLIENT_SHARE_DIR -p 20112 -u username -q password

Results:

Client

```
-------SERVER------
DROP/1.0 LOG 200 ok
```

Server

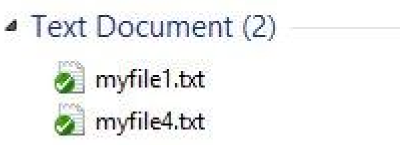
```
-------Client------
LOG DROP/1.0
Authorization:username:password
```

## Synchronization Test

Synchronization is done manually with an "s" command from the client

Snapshot file

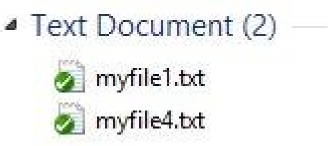
Client



Text Document (2)

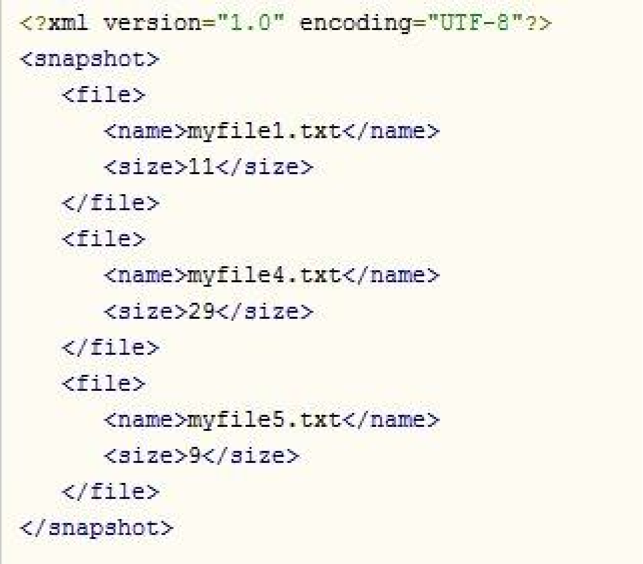
myfile1.txt
myfile4.txt

Server



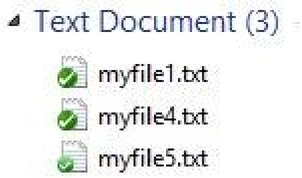
Text Document (2)

myfile1.txt
myfile4.txt



```
<?xml version="1.0" encoding="UTF-8"?>
<snapshot>
   <file>
      <name>myfile1.txt</name>
      <size>11</size>
   </file>
   <file>
      <name>myfile4.txt</name>
      <size>29</size>
   </file>
   <file>
      <name>myfile5.txt</name>
      <size>9</size>
   </file>
</snapshot>
```

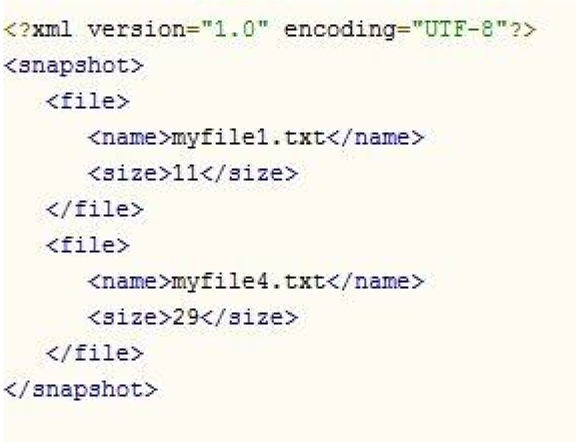
Next I created a new file on the client and issued the synch command



Text Document (3)

myfile1.txt
myfile4.txt
myfile5.txt

Results:

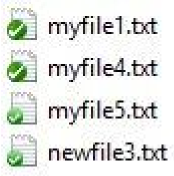


```
<?xml version="1.0" encoding="UTF-8"?>
<snapshot>
   <file>
      <name>myfile1.txt</name>
      <size>11</size>
   </file>
   <file>
      <name>myfile4.txt</name>
      <size>29</size>
   </file>
</snapshot>
```

```
enter command:s
-------SERVER------
DROP/1.0 DIR 200 null
size:32
lines:2

myfile1.txt:11
myfile4.txt:29

-------SERVER------
DROP/1.0 PUT 200 ok
```

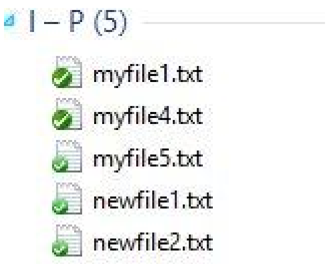
```
-------Client------
DIR DROP/1.0


-------Client------
PUT myfile5.txt DROP/1.0
Bytes:9

some file
```

Next I created 2 new files on the server and 1 new file on the client



myfile1.txt
myfile4.txt
myfile5.txt
newfile3.txt

Created newfile3.txt

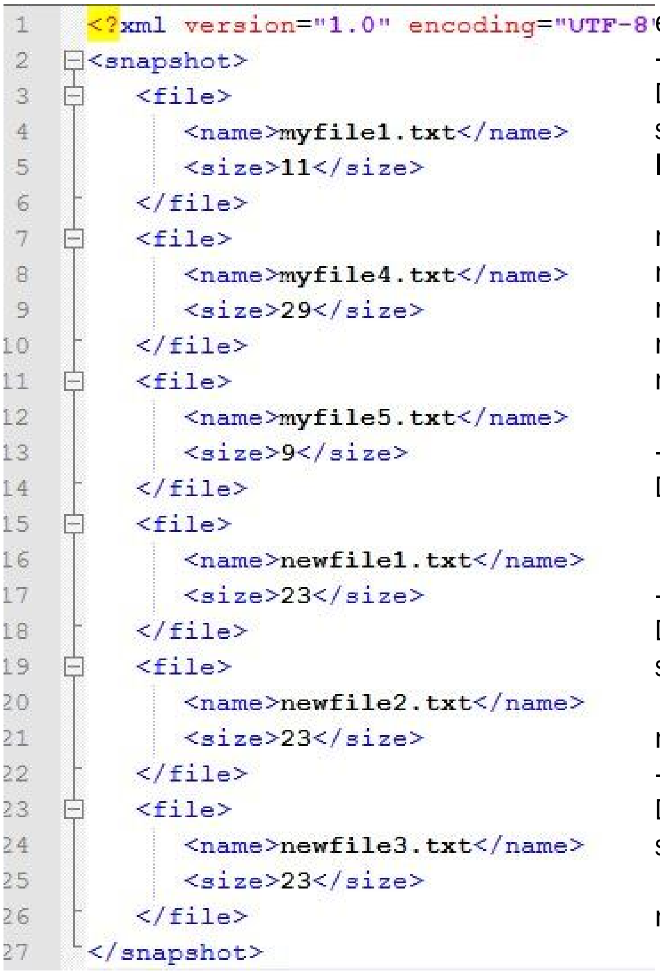


I – P (5)

myfile1.txt
myfile4.txt
myfile5.txt
newfile1.txt
newfile2.txt

Created newfile1.txt
Created newfile2.txt

Result:



```
<?xml version="1.0" encoding="UTF-8
<snapshot>
    <file>
        <name>myfile1.txt</name>
        <size>11</size>
    </file>
    <file>
        <name>myfile4.txt</name>
        <size>29</size>
    </file>
    <file>
        <name>myfile5.txt</name>
        <size>9</size>
    </file>
    <file>
        <name>newfile1.txt</name>
        <size>23</size>
    </file>
    <file>
        <name>newfile2.txt</name>
        <size>23</size>
    </file>
    <file>
        <name>newfile3.txt</name>
        <size>23</size>
    </file>
</snapshot>
```

```
enter command:s
-------SERVER------
DROP/1.0 DIR 200 null
size:81
lines:5

myfile1.txt:11
myfile4.txt:29
myfile5.txt:9
newfile1.txt:23
newfile2.txt:23

-------SERVER------
DROP/1.0 PUT 200 ok


-------SERVER------
DROP/1.0 GET 200 null
size:23

new file 1 text content
-------SERVER------
DROP/1.0 GET 200 null
size:23

new file 2 text content
```

```
-------Client------
DIR DROP/1.0


-------Client------
PUT newfile3.txt DROP/1.0
Bytes:23

new file 3 text content
-------Client------
GET newfile1.txt DROP/1.0


-------Client------
GET newfile2.txt DROP/1.0
```

Snapshot file was updated by the client