Lab 11: File IO / Sockets / Exceptions

Due date: By the end of your lab session of the week of Monday, Nov 7th

Goals:

- Learn how to read from a file, write to a file and handle related exceptions
- Learn how to use Stream Sockets to communicate between Client and Server (Network Communication)
- Learn how to catch and handle java defined Exceptions
- Learn how to write, throw and catch a custom Exception

Description:

In this lab, you will be implement a simple Client/Server paradigm that allows for a user to request user information from a file in the server and receive feedback from the server upon a successful request.

We will use **localhost** as our host in this lab. But in real life, the host could be a different one from the localhost. This means that your server and client can be in different machines. Since we are using same machine, to depict this situation we will use different folders for server and client, this simulates the server and client in different machines.

For this lab, you are expected to write two programs:

1. **Server.java** which is a Program that implements the server side of the communication.
2. **Client.java** which is a Program that implements the client side of the communication.

You will need two packages for this lab:

1. java.io.* (We will only need the packages related to File)
2. java.net.*

Task 1: Implementing the Server

In this task you will be implementing the Server side of the program. You will need to read messages from the Client, find user information based on the request from the client, and send informations via messages back to the Client.

Steps:

1. Create the file **database.txt** from which you will read the user information from-
    a. Open the terminal and change the current directory to the **"Desktop"** as follows

```
$ cd Desktop
```

    b. Create a directory named **"cs180"** using the **mkdir** command and change the current directory to point to it as follows

```
$ mkdir cs180
$ cd cs180
```

c. Create a subdirectory of "cs180" and name it **"lab12"**. Change the current directory to point to it.

```
```

d. Create a subdirectory of "lab12" and name it **"Server"**. Change the current directory to point to it.

```
$ mkdir Server
$ cd Server
```

e. Create a new file named ***"database.txt"*** using the touch command.

```
```

f. Open "database.txt" using Emacs editor

```
$ emacs database.txt
```

g. Write the following informations in the file which will be formatted as follows (Each entry is separated by **semicolon**):

```
#;Username;Lastname;Firstname;Marks
1;ksolaima;Solaiman;KMA;50
2;mashbeck;Ashbeck;Matthew;50
3;chambe41;Chambers;Joshua;45
4;tsiah;Siah;TseQin;42
5;sflanner;Flannery;Sean;46
6;rush24;Rush;Madeline;43
```

2. You will write a program named **Server.java**
   a. The server will create a *ServerSocket* on any port (say, x) and will wait for any client to connect. If the creation of serversocket is successful, the server will print the following message - **"Server Waiting for Connection"**. Otherwise, any kind of exception should be handled.
   b. If a client tries to connect, accept the connection and print the message - **"Connection is successful and waiting for commands"**.
   c. The server will then receive **the filename and the username** from the client via stream message. The server should print information received from the client in the following format:

      Filename: <filename>
      Username: <username>

d. Then it will try to open the file which should be situated in the **"~/Desktop/cs180/lab12/Server"** folder .
e. If the file is not found, the server will catch the "**FileNotFoundException"** exception. If the server catches this exception, as the fail-safe measure it will send the **message "FileNotFoundException"** to the client.
f. Otherwise, if the file is found, the server will **open the file and read its contents** line by line. All information in the file will be separated by semicolon. Use this information to separate the information fields. Handle any exceptions that you might face while reading the file contents.
g. The server will then read the username from the client side and attempt to find the username in "database.txt". If the username is found then it sends the remaining information back to client as a message in the following format -

   LastName: ###
   FirstName: ###
   Marks: ##

h. Otherwise, if the username is not found in the file, the server must send back a **message "InfoNotFoundException"**. This is not a Java-defined Exception. It does not need to be caught in the Server side.
i. Close all the files and Stream connections.

Example:

Server.java Demo

Task 2: Implementing the Client

In this task you will be implementing the client side of the program. You will need to request data from the Server and receive messages back from the server. You will also need to create and handle an Exception for when user information is not found in the server.

Steps:

1. Create the folder ***Client*** in which you will create the folder ***info.txt*** and you will write the user information into info.txt.
    a. Open the terminal and change the current directory to the **"~/Desktop/cs180/lab12/"** as follows

    b. Create a subdirectory of "lab12" and name it **"Client"**. Change the current directory to point to it.

```
$ mkdir Client
$ cd Client
```

2. You will write a program named **Client.java**
    a. The client will try to connect to a host (in this case, **localhost**) via some port (x). Any exception while doing this should be caught.
    b. The client then **prompts the user** for a Filename and a Username.
    c. The client will **send this information to the server** via stream sockets using messages.
    d. The client will **receive a message back from the server** and depending on the message, it will complete different tasks.

Message	Task Details
FileNotFoundException	Prints the error message in the console - **File does not exist**
InfoNotFoundException	**Creates a custom exception** (Refer to Step 3)**,** throws and catches it. When you catch it print the error.
Otherwise	**Writes the messages in a file** named which will be created in the **"~/Desktop/cs180/lab12/Client"** folder.

e. Close all the files and connections.

3. Create an exception class named **InfoNotFoundException** with error message **"Your Information is not in our file".**

Example:

Client.java Demo (Username and Filename Exists)

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_111.jdk/Contents/Home/bin/java ...
Enter Filename:
database.txt
Enter Username:
ksolaima

Process finished with exit code 0
```

Client.java Demo (Only Filename Exists)

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_111.jdk/Contents/Home/bin/java ...
Enter Filename:
database.txt
Enter Username:
mkakodkar
InfoNotFoundException: Your Information is not in our file

Process finished with exit code 0
```

Client.java Demo (File does not exist)

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_111.jdk/Contents/Home/bin/java ...
Enter Filename:
a.txt
Enter Username:
ksolaima
File does not exist

Process finished with exit code 0
```

Running Your Work

When testing your programs, run Server.main() at first. Then run Client.main(). This order must be maintained because the server needs to running waiting for the client connection.

Turning in Your Work

Turn in your work by navigating to the directory of your lab11 folder and using the following command:

```
$ turnin -c cs180=COMMON -p lab11 lab11
```

Rubric

- 10pts: Creating ServerSocket & Accepting Socket from Client in Server
- 8 pts: Creating Socket in Client
- 6 pts: Parsing file properly in Server Side and sending correct messages
- 6 pts: Opening, reading and closing file in Server
- 6 pts: Opening, closing and writing in file in Client
- 6 pts: Handling FileNotFoundException
- 8 pts: Creating InfoNotFoundException and handling it