JAC444 Assignment 2 (Summer 2014)

Released in: week 9
Due on: August 3, 2014 23:59
Value: 10% of final mark
Late Penalty: 10% per day up to 7 days. A one-week delay will result in 0 mark.
Assignment type: optional group work of 2 students.

Part A

You are asked to build a multi-threaded server and a simple GUI-based client to search words in a text file.

1. The Client Application

A client program provides the user with a simple GUI to request the following services from a server program:

- append a paragraph to a text file,
- search the text file for all occurrences of a word.

The GUI has the following Swing components:

- an editable text area to enter a paragraph and display the result of a search,
- a text field for entering a search word,
- a label for displaying a simple message,
- three buttons labelled as "connect", "disconnect" and "append."

You may design the layout of these Swing components as long as it is user-friendly. You may also enhance the overall design of the GUI.

When a client program starts, the text field will display the default server name and the default port number such as "localhost 8075." The user must click the "connect" button in order to establish a connection with the server.

If a successful connection is made, a simple message (e.g. "file opened") is displayed at the bottom of the window. Otherwise another message (e.g. "connection failed") is displayed. The text area is blank at this moment.

In order to append a paragraph to a text file on the server side, the user must enter a paragraph at the text area and then click the "append" button. After receiving the paragraph, the server program will append the paragraph to the text file. It will then send a message "append done" back to the client program. This message will then be displayed at the bottom of the window.

In order to search the file, the user must enter a search word (without any blank space) at the text field. After the user has hit the ENTER key, the client program will send the search word to the server. The server program will then search the file for all occurrences of this word. It will send an array of sentences back to the client program. A sentence is always ended by the newline character ('\n'). The client program will then display all these sentences in the text area. The result of the search (i.e. number of sentences found) is displayed at the bottom of the window.

In order to disconnect the client from the server, the user will click the "disconnect" button. The server will send the message "disconnected" to the the client. This message is then displayed at the bottom of the window.

2. The Server Application

The server program is a multi-threaded application.

When the server program starts, it checks if a file exists or not. If the file does not exist, it will create a file with some text.

Then it displays the message "listening for a connection..." at the console window. When a client connection is established, it displays another message "connected to client at xxx.x.x.xx" at the console window. It then displays the message "listening for a connection..." again. Before a server thread is going to terminate a socket connection, it will also display a descriptive message at the console window.

Note: You must make sure that synchronized methods are used to search the file and append a paragraph to the file.

3. Submission Requirements for Part A

1) Use default packages.
2) Your assignment should be submitted as three JAR files.

The first JAR file is named as "<lastname.firstname >_PartA.jar."

It consists of the Java source files and the ReadMe file.

a) All the Java source files must be documented with brief comments.
b) In the ReadMe.txt file, you must list all the commands that should be used to run your client/server application.

(Warning: If I encounter errors in running your application, you will lose 20% immediately.)

The second JAR file is named as "Server.jar."

It consists of all the class files that must be used to run the server program.

The third JAR file is named as "Client.jar."

It consists of all the class files that must be used to run the client program.

NOTE: Make sure that you have executed "Server.jar" and "Client.jar" before your submission.

Part B

You are asked to build an RMI application. The RMI server creates a remote object to provide five services, namely *adding a new location*, *deleting an existing location*, *generating a report* on all the locations, *connecting* and *disconnecting* the remote database access. All the locations are stored in a MySQL database table. A location consists of an address (string) and a pair of coordinates (double).

You must declare five remote methods in a remote interface. ***The add method*** inserts the new location at a database table called LOCATION. As a result, it returns true. If the address of the location already exists in the table, the method will throw DuplicatedAddressException. ***The delete method*** removes an existing location from the database table. As a result, it returns true. If the location does not exist in the table, the method will throw InvalidLocationException. ***The createReport method*** returns an array of all locations that are stored in the table. ***The connect method*** causes the remote object to open a database connection and load all the data from the database table into an ArrayList. In order to have better performance and enforce data consistency, the add and delete methods will update both the ArrayList and the database table. ***The disconnect method*** causes the remote object to close the database connection. (Note: Do not forget about RemoteException.)

When the RMI server is started, it will send echo output to the console window. Whenever a database connection is opened or closed, echo output will be sent to the console window as well. IP address should be displayed whenever possible.

You must build a GUI-based RMI client to request five services from the remote object. You should use a button group of two radio buttons to connect and disconnect the remote database access. The RMI client must catch DuplicatedAddressExceptions, InvalidLocationExceptions and RemoteExceptions by displaying an error message on the GUI.

Submission Requirements for Part B

1) Use default packages.
2) Your assignment should be submitted as three JAR files.

<u>The first JAR file</u> is named as "<lastname.firstname >_PartB.jar."

It consists of the Java source files and the ReadMe file for part B.

a) All the Java source files must be documented with brief comments.
b) In the ReadMe.txt file, you must list all the commands that should be used to run your client/server application.

(Warning: If I encounter errors in running your application, you will lose 20% immediately.)

<u>The second JAR file</u> is named as "RMIserver.jar."

It consists of all the class files that must be used to run the server program.

<u>The third JAR file</u> is named as "RMIclient.jar."

It consists of all the class files that must be used to run the client program.

Make sure that you have executed "RMIserver.jar" and "RMIclient.jar" before you submit them.

Assignment 2 Submission

1) Create a text file named A2_submission_form.txt with the following content:

```
            JAC444 - Assignment 2 – Summer 2014

Assignment Submission Form
==========================================================================
I/we declare that the attached assignment is my/our own work in accordance
with Seneca Academic Policy.  No part of this assignment has been copied
manually or electronically from any other source (including web sites) or
distributed to other students.

Name(s)                    Student ID(s)                 Seneca LearnID(s)


--------------------------------------------------------------------------


--------------------------------------------------------------------------


--------------------------------------------------------------------------
```

2) Assignment 2 submission checklist

Part	File Name	Notes
Part A	<lastname.firstname>_PartA.jar	Contains ReadMe and source files
	Server.jar	Must be executable
	Client.jar	Must be executable
Part B	<lastname.firstname>_PartB.jar	Contains ReadMe and source files
	RMIserver.jar	Must be executable
	RMIclient.jar	Must be executable
	A2_submission_form.txt	

3) Select one student to submit the assignment. Compress all files on the checklist into a ZIP or 7z file and submit the compressed file onto the Blackboard (where the assignment is posted). The compressed file should be named as <lastname.firstname >_A2.zip or <lastname.firstname >_A2.7z.