CP 372: Assignment 1 (10%)

Documentation due 10:30 pm, Wednesday, September 30, 2020
Code due 10:30 pm, Wednesday, October 7, 2020

In this assignment you will develop a client-server application using stream socket API of Java. Your application will support collective work of clients on a bibliography file.

Clients will:

- Request a connection with the server
- Send four types of messages through established connection:
    1. SUBMIT messages containing book description
    2. UPDATE messages containing an update to a particular book description
    3. GET messages containing requests for a particular book
    4. REMOVE messages containing requests to remove a book from the bibliography file

Server will:

- Accept requested connections and support multiple simultaneous connections
- Maintain an appropriate data structure that holds data received from all clients. The data stored on server is a "bibliography list" which is empty at startup. It is not persistent (if server shuts down and starts again – the list becomes empty).
- Process messages received from client:
    1. If message is SUBMIT, the server will update the bibliography list (if the entry is not a duplicate).
    2. If message is UPDATE, the server will update a relevant entry in the bibliography list.
    3. If message is GET, server will send to client a list of entries matching particular request.
    4. If message is REMOVE, server will remove the entry from the bibliography list.

In any case server has to react properly to the erroneous messages.

Simplified bibliography entry description has the following components: ISBN, TITLE, AUTHOR, PUBLISHER, YEAR.

Here are several examples of requests from a client in ASCII:
SUBMIT
ISBN 9783161484100
TITLE Modular Algorithms in Symbolic Summation and Symbolic Integration
AUTHOR Gerhard
PUBLISHER Mir

UPDATE
ISBN 9783161484100
YEAR 2004
PUBLISHER Springer

GET
TITLE Modular Algorithms in Symbolic Summation and Symbolic Integration
Client expects to receive records of all books in catalogue with this title

GET
AUTHOR Gerhard
Client expects to receive records of all books in the bibliography file with this author

GET
TITLE Modular Algorithms in Symbolic Summation and Symbolic Integration
AUTHOR Gerhard
Client expects to receive records of all books in the bibliography file with this title and this author

GET
ALL
Client expects to receive all entries of all books in the bibliography file

REMOVE
AUTHOR Gerhard
Client requests removal of all books in the bibliography file with this author

Here are some additional rules that simplify processing of the requests:

- ISBN field of an entry is a key field and it can not be updated by UPDATE request
- SUBMIT will not introduce a duplicated entry (duplicates are two entries with the same ISBN)
- REMOVE request requires confirmation from a client
- GET might have an option to display reply in 'bibtex' format (5% bonus mark for implementing this feature)
- Note, that ISBN is a part of every request, and it is responsibility of a client to verify that ISBN value is valid before request is sent to the server (i.e. contains 13 digits only and passes ISBN verification check; see ISBN-13 check digit calculation at https://en.wikipedia.org/wiki/International_Standard_Book_Number for details)

Code organization
Server is a **multithreaded** Java console application that starts with empty bibliography list and has port number as a command line argument, for example:

```
>java Server 4512
```

Client has to have a GUI. Minimal GUI requirements are:

- Text field to provide IP address of the server
- Text field to provide port number
- Connect/Disconnect button
- Text area to type in text to be sent to server and "Send" button
- Text area to display result of the request.

Any reasonable enhancement of the GUI is a subject of possible bonus.

Submissions:

1. Submit documentation (single PDF file) in form of a short RFC, containing description of your protocol, format of messages sent by client and server (do not forget to describe format of <response>, since it is not given here), synchronization policies, reactions of server and client to the errors, border-cases behavior etc.
   The name of your PDF file is to be A1xx.pdf where xx is your group number (for example, A192.pdf if your group number is 92, or A102.pdf if your group number is 02). Content of the zip-file – one file with your documentation in PDF format.
   Submit to "A1doc" drop box on MyLearningSpace.

2. Submit two applications – server and client in a single zip file, containing two directories – "Server" and "Client". Each directory **MUST** contain Java files only.
   The name of your zip file is to be A1xx.zip where xx is your group number (for example, A192.zip if your group number is 92, or A102.zip if your group number is 02).
   DO NOT SUBMIT ECLIPSE PROJECTS!!!!
   All Java files submitted have to use default package (**no package keyword**!!!).
   It should be possible to compile your Server and/or Client in a command line environment with "javac *.java".
   You can submit 1 page document with revision of your RFC along with the code.
   Submit to "A1" drop box on MyLearningSpace.

Testing suggestions:

1. Make sure that your Server and Client compile in command line environment.
2. Make sure that content persists on Server side between connections (start server, start client, submit a book, disconnect, start another client, request the book).
3. Make sure that Server supports multiple clients and does not crash.
4. Test for correctness of possible "empty" reply (start server, start client, do not submit anything just request something).
5. Make sure that Client does not crash or hang when server is not running.
6. Make sure that ISBN is verified by Client.
7. Make sure that concurrent requests work "atomically" (for example concurrent SUBMIT and GET, or concurrent SUBMIT and REMOVE): i.e. one client either sees all changes introduced by the other client(s), or does not see the change at all (depending on the order in which server threads handle the requests).
8. Test correctness of every kind of request, including correct behavior in case when request is erroneous (such as submit of a duplicate, update with non-existing ISBN number and so on).

NOTES:

1. **Violation of submission format, requirements and naming conventions will result in 0 grade for this assignment.**
2. **All submissions will be attested for similarity by MOSS.**