G52APR Coursework 3 (2014/15)

Please also refer to the specification document for Parts 1 & 2 of the coursework.

Part 3 – HTTP Server with TransactionRegister

This document should be read in conjunction with the earlier documents specifying parts 1 and 2 of this coursework.

You should by now have a server that can send valid responses to the requests it receives from clients. This part of the coursework is to allow your server to be able to save a simple history of the server status and the requests received and responded to.

You will create a transactionRegister that will save to a file. If you develop this using a good choice of design (eg patterns) you will find this coursework easier to complete than if you simply start coding giving no consideration to how to design your solution. Bear in mind we may wish to easily add more transactionRegisters later (e.g. a database transactionRegister) and you should design for this.

Note 1: a significant part of the marks will be awarded for the design of the software.

HTTPServer

Your main method in the server (G52APRServer) should handle some extra parameters which will specify whether to save data or not and whether to reset (empty) the register file before starting registering.

A new parameter “-r” [without the quotes] indicates registering should be turned on (this is in addition to the parameters from part 2). An alternative parameter “-R” [without the quotes] indicates registering should be turned on and the register file reset (emptied). A final parameter should indicate the location and name of the register file as a path. Parameters should be in the order given in the following examples:

eg 1 - run the server, on port 4444, with root C:/files/wwwroot/, with registering into file C:/tmp/register.txt :

```
java –cp <classpath> G52APRServer 4444 C:/files/wwwroot/ –r C:/tmp/register.txt
```

eg 2 - run the server with registering **and** reset the register file:

```
java –cp <classpath> G52APRServer 4444 C:/files/wwwroot/ –R C:/tmp/register.txt
```

You may find it easier to test by setting these start parameters in Eclipse. If you use java at the command line you will need to deal with any libraries you have used.

File TransactionRegister

The transactionRegister will be responsible for saving the following three pieces of information using one line for each occurrence.

1. *Start Details*

This should consist of 3 comma separated items: Current date/time, “$”, “starting registering” [without the quotes]

2. *Request details*
   This should consist of 4 comma separated items: Request date/time, “>”, the requested file, the request line [without the quotes around >]
3. *Response details*
   This should consist of 4 comma separated items: Response date/time, “<”, the requested file, the status line [without the quotes around the <]

Request Line – This is the request line, not including any body or headers.

Status Line – This is the first line of the response sent to the client.

Examples:

```
Wed, 17 Mar 2004 18:00:45 GMT,  $, starting registering

Wed, 17 Mar 2004 18:00:49 GMT,  >, index.html, GET index.html HTTP/1.0

Wed, 17 Mar 2004 18:00:51 GMT,  <, index.html, HTTP/1.0 200 OK
```

A good design would allow different types of transactionRegister to be easily created in the future (e.g. a database transactionRegister or a file transactionRegister to a different file name) and would allow multiple transactionRegisters to be easily used simultaneously.