Computer networking: project 2

Adrien Minne s154340
Delaunoy Arnaud s153059

1 Software architecture

1.1 connexion handling

1.1.1 WebServer

This is the main class. Its job is to manage the sockets; it accepts clients and assign a new Worker for handling the client's request. It thus handle the threadpool mechanism.
In addition to that it also keeps manage the cookies of all the clients providing a method for creating new cookies and getting cookie from their id. However, this class only handle the storing of the cookies and not the mechanisms that come with it.

1.1.2 Worker

This is a thread class that handle a client's request given its socket. To do so it calls the RequestHandler and close the socket when the Request is handled.

1.2 Request handling

1.2.1 RequestHandler

This class has a private constructor and is not meant to be instantiated. It provides a static method for handling request based on the client sockets.

1.2.2 HTTPRequest

This class represent a request of the HTTP protocol. The main job of this class if performed by its constructor as it takes a socket as argument and construct parsing what is given through this socket. It also performs some checks about the consistency of what's given though the socket. The pieces of information about this request are then easily accessible by getting the values of the variables of the HTTPRequest object.

1.2.3 HTTPReply

This class represent a reply of the HTTP protocol. It works as the opposite of the HTTPRequest. The constructor takes as arguments The different pieces of information of the reply and have a method to send it through a socket with the HTTP protocol. The HTML page contains references to png files representing the colored circles, we thus had to make sure that our server handled correctly requests for image file in addition to the request for the HTML page itself. It also provide method to send the reply to the client, after encoding the reply body in GZip, using chunk encoding.

1.2.4 HTTPRedirectionReply

This class inherit from HTTPReply and provides a constructor that create a reply for redirection taking only the location as argument; the other components being always the same for any redirection.

1.2.5 HTTPRErrorReply

This class inherit from HTTPReply and provides a constructor that create a reply for an HTTP error. It create a simple HTML page displaying the error message.

1.2.6 MethodExecutor

This is an abstract class. It has an abstract method called process that produce the appropriate HTTPReply object given a HTTPRequest object. This abstract method is then implemented by either the `GetMethodExecutor`, `PostMethodExecutor` or `HeadMethodExecutor` Which implement the logic of a get, post, head request. It also provides a method to manage cookies. It modifies the headers of a reply given the headers of the request. It also modifies the cookie associated to this Executor and the cookie list according to the situation.

1.3 client side

1.3.1 HTMLPage and HTMLErrorPage

These two classes generate HTML code. HTMLPage generate the HTML code of the main page of the game, and HTMLErrorPage generate a simple page where we display the HTTP error code. The HTMLPage is generated by reading a HTML file and inserting into it a list of combinations created in the Java code.

1.3.2 WebsiteFiles

We decided to create separates files for the CSS and JavaScript, which allows the client to cache them instead of loading them every time. The WebsiteFiles enumeration contains the path of those files, as well as a method to read these file and put them into a String that will be used as the body of the request.

1.4 Other classes

1.4.1 Colors

Enumeration representing a Color providing some basic method associated to it.

1.4.2 Combination

This class represent a Combination (Combination + result). It also provides a method parse a string containing the representation of a combination in order to set the combination of the object. It provides a method to set a random combination and a method to evaluate the combination setting the results based on a comparison combination given in argument.

1.4.3 Cookie

The cookies are represented in a Cookie class in Java. They are stored on the server in a static ArrayList in the Webserver class. Inside the Cookie class, we memorize all the tried combinations of the user, the correct combination, as well as the creation date of the cookies to be able to check whether it's expired or not. The class also contains a method to generate an HTMLPage from the combination tried by the user.

When handling a request, we check in the MethodExecutor class if the user did send a cookie. We then check if the cookies has expired and reset it if it is, then we use this cookie to generate the correct HTTP Reply body. If the client didn't send any cookie ID, we create a new cookie for it and send it with the reply. As some clients may discard the cookie expiration date, we don't send any. It means that a client will always use the same cookie ID for every game it plays, only the content of the cookie in the server will change and it's the server that will check if the cookie expired. It makes it simpler to manage cookie, but can induce minors problems discussed int the sections "limits" and "possible imrovements".

1.4.4 FileType

Enumeration that contain the types that can be sent through a request or reply. It contain a string indicating the header value associated to it and a boolean indicating if this type is represented by a string.

1.4.5 HTTP

This class holds some general pieces of information about the HTTP protocol such as the version and a method to get the server time.

1.4.6 HTTPOption

This is an enumeration containing all the possible headers. It contain a string indicating how this header is represented in the HTTP protocol and some basic methods associated to it.

1.4.7 ReturnCode

This is an enumeration holding the return codes that we have to use in this project. To each return code is associated its number and the corresponding status.

2 Multi-thread coordination

Having different cookie id for each user allow to totally separate The variables people are working on. However,a single user, by submitting several requests quickly, may lead requests to intefere. We thus add a synchronized block on the cookie object during the time actions are performed on this cookie.

3 Limits

- The cookies aren't send with an expiration date, a client will always keep the same cookie. The only problem with this is that if a client clear all it's cookies, the cookie it previously used will stay in the memory of the server and occupy memory space uselessly.

4 Possible improvements

- To prevent the accumulation of cookies on the server, we could've for example created another thread that would periodically search for expired cookies and delete them. The problem is that this thread could induce a lot of synchronization problems (e.g. it tries to clear a expired cookie, but at the same time the client using this cookies tries to connect to the server). These problems aren't trivial at all to solve, so we chose to ignore them. But as clearing cookies isn't something that people often does, we judged that trying to solve all the new synchronizations problems that would arise wasn't worth it.

- When the server reboot, if there's a ongoing game on the client side, the game will be in a incoherent state, as the current progress of the game will be lost on the server side. A simple solution would be to keep the cookies in a database on the disk, instead of on the RAM used by the server.