Comp3331 Assignment: Nicholas Mangos z5417382

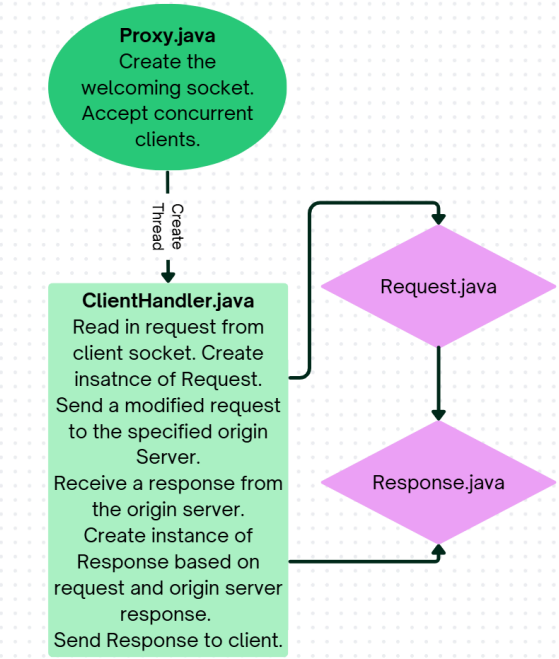
Programming Language and Code Organisation

The HTTP Proxy server code was written in **Java**.
The key files of the project include:

- Makefile: a file used to compile the Java code into executables. The code can be compiled by running the following command in the directory:
  `make`
- Proxy.java: This file provides the executable interface for the HTTP proxy in its main function. The Proxy can be initiated with the following command:
  `java Proxy <port> <timeout> <max_object_size> <max_cache_size>`

High-Level Design

The diagram summarises the main interactions.



Expanding upon interactions:

- Instance of Cache stored in Proxy.
- Cache stores successful responses of GET requests.
- If request answer is already in the Cache, Proxy will not contact the origin server and will send the cached response.
- The overview displayed will work for exceptions communicated by the origin server.
- For exceptions detected by the Proxy, new instances of Response are generated with HTML files based on the exception. These are sent to the client.

Class communications (not above)

- Response and Request are subclasses of the abstract class Message; they share several methods.
- Subclasses of Message store an instance of the class Header, an object for their header values.
- ClientHandler may create instances of ResponseFile to generate HTML files for exception responses.

Data Structures

Cache

Cache is the key data structure of the Proxy cache. Internally, it contains a HashMap mapping URL keys to Responses and a LinkedList of URL keys. It also contains a MaxObjectSize and MaxCacheSize assigned in the arguments for Proxy. Cache uses the linked list to determine the least recently used (LRU) response in the cache. Each time a

response is accessed from the cache, or inserted into the cache, its entry in the linked list is moved to the front.

Header
Header is the key data structure of the headers of HTTP messages. It contains a HashMap that maps header field names to field values. Field names are stored in lowercase as they are case-insensitive; however, field values are not converted to lowercase.

Message
Message is the parent class of Request and Response. The main variables it contains are the Header and message body. The message body only contains the content of a message and is stored as an array of bytes. It was chosen to be stored as an array of bytes (as opposed to a string) to ensure that encoded files are not corrupted.

Key data structures of ClientHandler
ClientHandler is a class that implements Runnable, meaning it can be used to create concurrent threads for each client. It uses many data structures, such as the following:

- ClientHandler uses a buffer of 8192 bytes (private constant BUFFER_SIZE = 8192) for reading information from input streams. This buffer size was chosen as the assignment specification set the constraint that the maximum header size for testing would be 8192 bytes, including the final CRLF.
- ClientHandler uses a ByteArrayOutputStream to store the content of messages read. After determining the end of the header, as communicated by a double CRLF, bytes will be added to the ByteArrayOutputStream directly.

Request
The Request class stores and processes the initial string of information provided by the ClientHandler. It determines the host and port of the origin server, the original value of the Connection header, the time the request was received and many other important pieces of information.

Limitations
To my knowledge, my implementation of an HTTP proxy server covers all the requirements of the assignment. However, there are some features of HTTP/1.1 that it does not support:

- It does not support any HTTP methods other than GET, HEAD, POST and CONNECT. It also does not send a '501 Not Implemented' response if other HTTP methods are encountered.
- It does not support persistent connections with transfer-encoded Requests. In the implementation, for messages that are encoded, the proxy will read all information until the input stream is shut down. This would block the following requests from going through.

Acknowledgments
No code was copied directly; however, some sources on the internet guided my implementation.

Kaj (2011, June 14). *create an ArrayList of bytes.* StackOverflow.

https://stackoverflow.com/questions/6340999/create-an-arraylist-of-bytes
This forum post was useful when I was creating ClientHandler, specifically for handling byte arrays.

dreamcrash (2021, April 7). *Java: How to make thread loop until condition met.* StackOverflow. https://stackoverflow.com/questions/66981916/java-how-to-make-thread-loop-until-condition-met
This forum post was useful when I was implementing the HTTP method CONNECT. Specifically, for creating threads for concurrent communication between the client and the origin server, and then waiting until both close.

Perception (2013, March 12). *java socket timeout behaviour.* StackOverflow. https://stackoverflow.com/questions/15362484/java-socket-timeout-behaviour
This forum post was useful when I was facing a bug with handleOriginServer. I did not realise that timeouts were specific to the resources socket, and therefore only their input/output streams would throw exceptions.