Behavioural Types for Mainstream PLs: assignments

António Ravara
NOVA School of Science and Technology
Lisbon, Portugal

July 28, 2023

Languages

Use one of the following three languages + checkers:

- JaTyC: `https://github.com/jdmota/java-typestate-checker/tree/master`
- Rusty Typestates: `https://docs.rs/typestate/latest/typestate/`
  Useful documentation: `https://rustype.github.io/notes/notes/rust-typestate-series/rust-typestate-index`
- FreeST (a functional language with Session Types): `http://rss.di.fc.ul.pt/tryit/FreeST`

Problems

Descriptions:

- JaTyC and Rusty Typestates: see next slide
  (you may use channels for communication, instead of sockets)
- FreeST: the Digital Locker, with channels and Session Types
  `https://github.com/Azure-Samples/blockchain/tree/master/blockchain-workbench/application-and-smart-contract-samples/digital-locker`

You may find useful to draw/check your typestates with the typestate-editor:

`github.com/typestate-editor/typestate-editor.github.io`

All feedback most welcome!

Project description

A File Server

The aim is that clients can request files from it.

The communication will be implemented using java.net.Socket or java.nio.channels classes and will follow a specific byte protocol:

1. the client sends the string “REQUEST\n” to start a request;
2. the client sends the name of the file followed by “\n”;
3. if the file exists, the server responds by sending to the client each one of the bytes of the file;
4. if the file does not exist, or when the end of the file is reached, the server sends the 0 byte;
5. after the client receives the 0 byte, it can start a new request (see steps 1 and 2), or send the string “CLOSE\n” to finish the protocol.

Assignment

Part one

1. Design protocols for classes FileClient and FileServer. correct usages of the protocols imply that both parties will communicate according to the description above.
2. The server's protocol should be specified so that each byte of a file is sent at a time (i.e. creating a method that accepts a string with the full contents of the file at once is not the assignment).
3. In the same manner, the client's protocol should have methods to retrieve the contents of the file one by one.

After specifying the typestates, implement the classes (complete the code skeletons provided, assuming socket communication is always stable – no exceptions from reading and writing to input or output streams).

Assignment

Part two (only for JaTyC)

Create a FileClient2 class that extends FileClient with a method to retrieve the file contents line by line instead of byte by byte.

For this part, you need to create a new method and protocol for FileClient2, correctly extending the protocol of FileClient. The FileServer class should remain as implemented in Part 1.

Evaluation

Compile your code; if you're using JaTyC, run it (no output is expected, meaning that no errors were found). Then, execute:

1. FileServer to start the server and wait for client requests;
2. FileClient to start a client communicating with the server.

After the client gracefully terminates, you may terminate the process by hitting CTRL+C.

Practical Aspects

Develop by yourself or in groups of two

Feel free to ask for support

Deadline

August 10, 2023

Submission

- By email to aravara@fct.unl.pt
- Send either a zip with (only!) source files (all needed to compile the project), or a link to a shared repository/folder