Lisbon Institute of Engineering

Degree in Informatics and Computer Engineering

Concurrent Programming

Summer 2022/2023, Third Series of Exercises

---

1. Make the **MessageQueue<T>** class for communicating messages between *coroutines*, through *first in first out message* queues, providing the methods:
    - **suspend fun enqueue(message:T): Unit**
    - **suspend fun dequeue(timeout: Duration): T**

   Both methods must support cancellation. The construction of the queues must support the definition of their maximum size.

2. Perform extension functions on the **AsynchronousServerSocketChannel** and **AsynchronousSocketChannel** classes to accept connections, writes and reads, without blocking the invoking *threads.* These functions must be **suspendable**, support cancellation and expose a synchronous interface.

3. Build a server system with a TCP/IP interface for exchanging messages between client systems, with the functionality shown below.
    - Client systems interact with the server system by sending text lines, which can be commands or messages. A command begins with '/', followed by the command name and zero or more arguments. A message is any line that does not start with '/'.
    - The server system is organized into rooms. Each client system can be in zero or one room. Once connected, the client systems are not in any room. There are commands to enter a room, leave a room and end the connection.
    - When a client system is in a room: a) it can send messages to that room; b) it receives all the messages sent by other clients in that room.

The commands that can be sent by a client over a TCP/IP connection are:

- **/enter <room-name>** - enters the room <room-name>, creating it if necessary.
- **/leave** - leaves the room you are in.
- **/exit** - terminates the connection to the server.

The system must also accept the following commands sent locally via *standard input*:

- **/shutdown timeout** - starts the server shutdown process, stopping accepting connections but waiting for all connections to finish. All clients should receive a message notifying them that the shutdown process has started. If there are still clients connected after **timeout** seconds, the server should abruptly terminate.
- **/exit** - abruptly terminates the server.

The system developed should use a number of *threads* appropriate to the computing capacity of the host machine, and should not be proportional to the number of clients connected.

For manual server tests, use any TCP/IP client, such as telnet, putty or netcat. The inclusion of automatic tests in the project is appreciated.

The delivery must be made by creating the **1.0.0** tag in each student's individual repository, or in the group repository created for this purpose. The delivery must also include a **README.md** file with the important design and implementation aspects.

The repository https://github.com/isel-leic-pc/s2223v-se3 has an implementation of an equivalent system, using two *threads* per connected client. Consider the organization described in this repository as a basis for the organization of the system to be realized.

Deadline: June 11, 2023 This series of exercises can be done in groups of up to three students.

ISEL, May 15, 2023