Practical Assignment I (Hostel)

The practical assignment I is focused on an architecture where concurrency (processes and threads), inter-process communications (sockets) and GUI are key aspects.

1 Description

The project is about a simulation of the management process of Hostels sharing some main characteristics:

- Ground floor:
  - Reception:
    - for check-in and check-out
    - 3 receptionists
    - a leaving room
    - a porter to open and close the main door
  - meal room:
    - a waiter to attend customers
    - one table with several seats
    - first-in first-out service
- other floors:
  - 3 bedrooms
  - each bedroom:
    - has 3 single beds
    - a bathroom for one client only at a time
    - customers of both sexes (simultaneously), even if they do not know each other

The main door is always closed except when the check-in and check-out processes are taking place. Before check-in, clients are walking outside the Hostel. After check-out, clients leave the Hostel for a walk. The Hostel does not provide any meal but breakfast.

Check-in:

- porter opens the door and he calls customers to enter the Hostel
- at most 6 customers can enter and stand in the reception queue (fifo) and then the Porter closes the door
- if there are no more customers, the check-in process ends
- if there are more customers:
  - after the last customer has entered, each receptionist calls one client at a time for check-in until the reception queue becomes empty
  - the check-in process for each client can take from 0-1000ms [tci]

    - after checking-in, the customer goes to the room/floor indicated by the receptionist
    - after the last customer checks in, there is a new check-in iteration

Check-out (process managed by floor)

- customers are all awoken (bottom-up)
- customers prepare themselves in the bathroom, one at a time
- each client can stay in the bathroom from 0-1000ms [tbr]
- when a customer of a room leaves the bathroom, he heads to the meal room
- the meal room has a table with 9 seats
- after the arrival of all customers of a floor to the meal room, the waiter hands a breakfast bag to each customer
- customers are served on a first-come, first-serve basis
- each customer can take 0 to 1000ms to have breakfast [tbf]
- after having the meal, each customer head to leaving hall
- the porter opens the door after the arrival of all customers
- after the exit of all customers, the porter closes the door and a new check-out iteration can start

2 General Requirements and Constraints

- Two processes:
    - the CCP (Control Centre Process): where configuration and supervision take place;
    - HCP (Hostel Control Process): it is the process where the Hostel exists
    - each process has its own GUI (do not forget that a process starts in the “public static void main(String[] args” method)
- Technologies:
    - programming language: Java
    - Java Processes
    - Java Threads
    - Java Concurrency: implicit and explicit monitors
    - SWING for GUI
    - The communication between CCP and HCP is based on sockets and at the server side it must support multiple-clients simultaneously
    - It is not allowed to resort to standard java classes to implement any type of queues
    - Do not use Maven.

3 Control Centre Process

A process with a GUI where configuration and supervision take place:

Configuration:

- Number of customers in a simulation: [3, 27], increments of 1, default value 9
- tci, tbr and tbf:
    - [0-1000]ms, increments of 100ms, default value 100ms
    - for each Thread, a random value between 0 and the chosen value

Supervision:

- Start button:
    - to start a new simulation
    - simulation:
        - leaves the “idle” state
        - enters the “running” state
    - Customers wait outside to enter the Hostel
- Check-in button:
    - To start the check-in process
    - Only one group at a time (at most 6 customers)
    - Valid if still there are customers to enter the Hostel
- Check-out button:
    - To start the next floor check-out process
    - Only one floor at a time
    - Valid:
        - after all customers checked-in
        - at least one floor with customers has not checked-out
- Suspend button:
    - to suspend the running simulation
    - Simulation:
        - Leaves the “running” state
        - Enters the “suspended” state
    - Valid if simulation is in the “running” state
    - Customers suspend their movement
- Resume button:
    - to resume the “suspended” simulation
    - Simulation:
        - Leaves the “suspended” state
        - Enters the “running” state
    - Customers resume their normal activity
- End button:

    - the two processes end immediately
- Operating mode:
    - When simulation is “running” it can operate in two different modes: manual or auto. In auto mode, simulation runs autonomously meaning that the Active Entities can interact with each other without the intervention of any supervision. In manual mode, Customers can only evolve to the next state under supervision.
    - option = {manual, auto} buttons
    - step button: to authorize Customers to evolve to the next state (on running state and manual mode only)

When customers leave the Hostel, they enter the “idle” state (walking around outside the Hostel)

4 Hostel Control Process

A process with a GUI where the Hostel is represented. Its representation must include the ground floor and the first floor. The remaining floors representation is optional.

4.1 Monitors

At least the following monitors must be implemented:

MOH: Outside Hall – where all Customers stay outside the Hostel

MCI: Check-in – where Customers wait in queue and do the check-in

MBR: Bed Room – where 3 Customers sleep, including the bathroom

MMR: Meal Room – where Customers have their meal

MLH: Leaving Hall – where Customers wait to leave the Hostel

MLG: logger – where the log of the simulation is registered

Each Monitor must be represented in the HCP GUI (except MOH and MLG) where the identification and state of the intervening Active Entities (Threads) are shown.

4.2 Active Entity

At least the following types of Active Entities are needed in the HCP: Customer (TCustomer), Porter (TPorter), Receptionist (TReceptionist), Waiter (TWaiter) and ControlCentreProxy (TControlCentreProxy).

5 Project Organization

You must follow the next guidelines:

- One NetBeans project only
- Two root Packages:
    - HCP for all Hostel Centre Packages and files
    - CCP for all Control Centre Packages and files
- Project name:
    - PA1GXX
    - Where: XX -> group number
- All Thread Classes must start with the letter T, such as TCustomer
- All Monitors must start with the letter M, such as MFifo
- Each Monitor must be an individual java class
- All Interfaces must start with the letter I, such as ICustomer
- The access to Monitors must be through Interfaces provided their methods
- Threads (Active Entities) that access Monitors must receive as argument the correspondent interfaces
- All variables must be private (if not, a justification must be provided)
- Whenever possible, Constructors must be declared as private (use a *getInstance* method to instantiate classes and return an interface)
- Variables whose initial value does not change must be declared as final
- ReentrantLock is much more flexible than synchronized methods and provide additional functionalities. The usage of ReentrantLock is mandatory. Nevertheless, **you must know** how synchronized methods work.
- Methods' and variables' names must be semantically oriented whenever convenient

6 Simulation Logger

The state of the simulation must be printed in a text file (LOG.TXT) and also in the console. The logger must be implemented as Monitor. The simulation state is monitored at the HCP only.

7 Evaluation

The key aspects for the evaluation are:

- Requirements implementation
- Usability: user friendly GUI
- JavaDoc (including the packages)
- final report:
    - what has not been implemented and/or not correctly implemented
    - all interfaces by Monitor and the methods of each interface
    - contribution of each student in %