University of Stirling, Computing Science

Test – Concurrent Systems

Time Allowance: 1:50 h + 5 mins to upload results

You are permitted to use any practical solutions you may have.
Internet use is **NOT** permitted,
except when you will be downloading the provided incomplete solution initially,
and uploading your solutions after the end of the test.

The Fishmonger Problem

A fishmonger is alone in the shop and either serves customers, if there are any in the shop, or rests if there are no customers. The shop is not a big one and only a number of clients can be in the shop at any given time, let's say four. Customers are served one at the time in the order they entered the shop, and once served, the fishmonger says goodbye and they leave the shop.
If the shop is empty and the fishmonger is resting, on arrival a new customer rings the bell and wakes the fishmonger up. If the shop is full, customers cannot enter and they leave, renouncing to buy fish that day.

Task 1 (up to 40%)

You are provided with a class implementing an incomplete solution.

0. Take enough time to read the code and understand in detail how the proposed solution works. You want to have a clear understanding of the provided code before devising your own solutions, as requested below.

The code you are given does not contain threads and synchronisation.

1. Decide which objects should be threads and amend the code accordingly. You may need to rename some methods.
2. To implement the fish shop, the Java class ArrayList is used. According to the Java API, this class is not thread-safe. Amend the code in such a way that all critical sections are properly managed. (You must use ArrayList)
3. Add a main class to instantiate the fishmonger, customers and shop classes. Ensure the programme executes in an orderly manner, according to the specification above.

Before you continue to Task 2 below, create in safe destination a folder named

<your student id number>_v6_test

containing three folders named Task1**,** Task2 **and** Task3**. Copy your code to the folder** Task1**.**

continued over page>

Task 2 (up to 30%)

Amend the fishmonger shop's behaviour in such a way that customers don't simply walk away if the shop is full but wait (outside the shop) for a place to become available in the shop when a customer leaves.
Similarly, ensure that at this stage, the implementation of the fishmonger guarantees that in case of no customers the fishmonger goes to sleep and gets woken up only by the arrival of a new customer in the shop.
Make sure your code outputs appropriate messages to the console screen, so that progress in the execution can be followed and any special cases are logged in your output. Make sure these outputs would be meaningful to a marker! You may want to add suitable delays to facilitate the reading of messages during program execution.

Before you continue to Task 3, copy your code to the folder Task2**, which has been created earlier on.**

Task 3 (up to 30%)

Extend your implementation of the fishmonger shop to having two neighbouring (but separate) shops. Customers can go into either shop to get service. For an initial solution have the customers decide at random which shop they visit. Two fishmongers for the two shops will do.
For a more advanced implementation, customers may visit one shop first, and if that is full go over to the other. Similarly, for the fishmongers, if their respective shop is empty, the fishmonger changes over to the other shop to help out. Only if both shops are empty the fishmongers sleep.

Please upload a zipped version of the folder containing your solutions as a single zip file to Canvas.