**University of Stirling, Computing Science**

## Test – Concurrent Systems

Time Allowance: 2h

You are permitted to use any practical solutions you may have.   
Internet use is **NOT** permitted.

**Sleeping Barber Problem**

The problem is analogous to that of keeping a barber working when there are customers, and resting when there are none and doing so in an orderly manner.

The analogy is based upon a hypothetical barber shop with one barber. The barber has one barber chair and a waiting room with a number of chairs in it. When the barber finishes cutting a customer's hair, he dismisses the customer and then goes to the waiting room to see if there are other customers waiting. If there are, he brings one of them back to the chair and cuts his hair. If there are no other customers waiting, he returns to his chair and sleeps in it.

Each customer, when he arrives, looks to see what the barber is doing. If the barber is sleeping, then the customer wakes him up and sits in the chair. If the barber is cutting hair, then the customer goes to the waiting room. If there is a free chair in the waiting room, the customer sits in it and waits his turn. If there is no free chair, then the customer leaves.

**Task 1 (40%)**

You are provided with a class implementing an incomplete solution to the Sleeping Barber Problem.

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 barbershop, 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.
3. Add a main class to instantiate the barber, customers and shop classes. Ensure the programme executes in an orderly manner.

**Before you continue to Task 2, copy your code to a safe destination.**

**Task 2 (30%)**

Amend the barbershop’s behaviour in such a way that customers don’t simply walk away, but wait outside the shop for a seat in the waiting area to become available.

Similarly, the barber is rather active at present (if there are no customers in the shop, he naps briefly but then checks for customers again). Change the code, so that the barber goes to sleep and gets woken up if a new customer has arrived 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.

**Before you continue to Task 3, copy your code to a safe destination.**

**Task 3 (30%)**

Extend your implementation of the barbershop application 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. You will also need at least two barbers for the two shops. To start you off, have one barber for each shop.

For a more advanced implementation, customers may visit one shop first, and if that is full go over to the other. Similarly, for the barbers, if their respective shop is empty, the barber changes over to the other shop to help out. Only if both shops are empty the barbers sleep.

**Please upload your solutions as a single zip file to Canvas.**