Lab 2 of CS 3211, 2013, Total 10 marks

Please submit in the IVLE workbin *Lab2* by 25th March, 9:59 PM.

Kindly note that there will be no extensions. If you are not finished by the deadline, please submit whatever partial answer you may have - this is better than not submitting at all. Only submissions in the IVLE Workbin will be graded. Submissions sent by e-mail, unfortunately, cannot be considered.

Upload one single zip file containing the two programs.

*Please submit your zip file to the IVLE Workbin folder **Lab2.***

Implementing a computer-controlled seat reservation system in Java. [6 marks]

A central computer connected to remote terminals is used to automate seat reservation for a concert hall. A booking clerk can display the current state of reservations on terminal screen. To book a seat, a client chooses a free seat and the clerk enters the number of the chosen seat, thereby issuing a ticket. A method is required to prevent double booking of the same seat by multiple clients from different terminals, while allowing clients free choice of the available seats. Note that a seat can appear to be free in a terminal, even though it is being booked by another clerk in another terminal.

Implement the seat reservation system in Java. Informally argue that your implementation prevents double-booking of the same seat. Your argument about why your Java program prevents double booking should appear as comments at the beginning of this file containing the Java program.

Simulating network printing [4 marks]

There are multiple client terminals that share one printer. Each client terminal maintains a queue of files to print. The shared printer can hold and print only one file each time. Thus, the printer should notify the terminals of the fact that the printer is available once the last printing job is finished. Afterwards, a terminal that occupies the printer through competition can request the printer to print the remaining file in its queue. The printer should be shared in a fair manner. Thus, one terminal should not keep occupying the printer.

Develop a Java implementation that simulates such a situation by filling in the handout files. More specifically, complete ClientImpl.java and PrinterImpl.java. The main method you can use is in Simulation.java that creates three threads for client terminals and another thread for a shared printer. Note that the printer thread does not terminate on purpose. After running this simulation, the following two requirements should be fulfilled.

1. All the files belonging to three client terminals must be printed out successfully.
2. No client should keep occupying the printer. Therefore, if the first client uses the printer, then the next time the second or the third client must use the printer unless the first client is the only client trying to use the printer and the rest of the clients already finished their jobs.