**University of Stirling, Computing Science**

**The Printer Queue**

An organisation has set up a spooled printing system: applications add printer jobs to a queue, and the printer removes the items when it is ready to print them. If the queue is empty the printer is idle. The queue is not infinite, but only provides limited capacity, let say, for simplicity, not more than four jobs at any given time.

**Task 1 (up to 40%)**

You are provided with a class implementing an incomplete solution. In this first implementation, done for testing purposes, jobs that arrive when the queue is full will be lost.

1. 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 printer queue, the Java class ArrayList is used. According to the Java API, this class is not thread-safe, i.e., informally speaking, when used with threads critical sections are not guaranteed to work properly (e.g. methods accessing the ArrayList may be not synchronised). Amend the code in such a way that all critical sections are properly managed. (Keep using an ArrayList)
3. Add a main class to instantiate the applications, printer and printer queue 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**.**

**Task 2 (up to 30%)**

Amend the printer queue’s behaviour in such a way that applications' print jobs are not lost if the printer queue is full, but are held until a place in the queue becomes available (when a job is printed).

Similarly, if there are no print jobs in the print queue, the printer waits until a print job is added to the printer queue. So the printer should not constantly check the printer queue.

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%)**

Now extend the printer in a series of small steps:

1. Extend your implementation to have three printers. Assume that they are all the same type of printer.
2. Once operating successfully, have one colour printer and two black and white printers. The applications should choose at random whether a print job is colour, or black and white. If a printer queue is full, use the other instead. Have a strategy if both are full. Say in your comments what the strategy is.
3. Once operating successfully, change the inherent first come first saved discipline for the colour printer queue to shortest job first. You will need the application to include print job size information (defined as a number 1-10) with a job that are passed to the printer queue.

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