The Factory Problem – CA4006 Assessment 1 v1.01

Problem Specification:

You have been hired by an aeroplane manufacturer to build a robot production line controller in the for a 10 robot factory building which is used by aircraft going from robot to robot. Robots access stored supplies of parts to install on the aircraft and take variable amounts of time to do their work. The production line can be represented using threads. Each robot can also be represented by a thread. You must implement the methods called by the robot production line controller to move aircraft along the line, instruct the robot on what tasks to do and what parts to install on what aircraft. When each job is completed (e.g. a method ArrivingGoingFromTo(int atRobot, int toRobot) should wake up an production line and tell it which robot to go to). The production line speed is fast but not instantaneous, taking only 100 ticks to go from one robot to the adjacent one. A vanilla option would be to assume the following:

- There is only one type of robot and it has a maximum work capacity; obviously this capacity is affected by the parts it installs.
- When the work to be done on an aircraft exceeds the maximum number of parts available, the system will display an error and remain stationary or until sufficient parts are loaded at the appropriate part of the production line.
- The robots can detect the identify of the aircraft parked beside them.
- There is only one aircraft entering the production line at a time.
- When not in service, the production line waits and 'sleeps'.

You have to come up with a way of storing the requests to the robots e.g. Robot 6 needs to install 4 parts on aircraft A and 3 parts on aircraft B. In other words there is a queue. Also if the production line receives work requests for aircraft that have already passed their robot, you need a way to minimise the work done.

These specifications are for the vanilla option of the implementation. Any form of creativity that you feel like putting in that will add interest to my marking of the project (multiple production lines, just in time supplies of parts, Nice GUIs, reporting on the number of threads executing/number of jobs being performed, 'smart' controllers with AI, random events such as faulty lines etc) is encouraged and marks awarded for the project will reflect this. Making use of java large-scale thread management support would be essential.

Hint: One Approach to the Problem Might be the following:

The program will contain two classes “aircraft” and “robot”. Each "aircraft" should be generated with a random arrival time, arrival floor and workplan and have a unique ID for identification purposes.

In order to compile your java code, I should just be able to do so with java compiler with “javac <filename>“ and after it gets compiled execute it with “java <filename(executable)>”. **Please do not rely on my using an IDE for the execution of your code.**

Output file:

The format for an output file should be something like following:

“Controller (Thread ID) # makes request to robot # at time # for aircraft #.”

There should be as many lines printed as are the number of requests. You should also generate an output file output.dat to store your output.

Note: You must assume that you the starting time for the first request is 0.

Deliverables and instructions for submission:

- The assignment is a two person project, worth 15% of the overall module mark and has to be done in Java (it's up to you if you want to do it individually, but marking will be the same).
- Please also write a short (4 page max) design documentation which may get you some partial credit if I cannot get your program to execute as it should. The report should (among other things) describe how you divided the work between you, how to compile and run the code, how your solution addresses issues like fairness, prevention of starvation etc.
- You have to submit all your source files, documentation and also all the generated class files for the linux platform.
- Please submit through Loop with a completed plagiarism declaration form which is signed by both of you.
- Deadline for submission is the Friday 20th March at 5.00pm. Penalties will apply for late submissions.
- All projects will have to give a demo on Monday 23rd March in class. You should be prepared to answer questions on your system.
- There will be a short intermediate presentation phase whereby you present your approach to the class. Every group must prepare 1 slide on their architecture/approach for the lecture on the 10th March (ungraded).

A Word on Resit Continuous Assessment:

- A resit is available for all components of this module.
- Students who fail the module on the first sitting must resit each component (CA or exam) that they failed. In particular, students who fail the CA must resit the CA.
- A student may not resit a component that has been passed and their marks for the component are carried forward
- Marks for failed components will be set to zero in ITS before the resit.

Good luck!