Declaration on Plagiarism

Assignment Submission Form

This form must be filled in and completed by the student(s) submitting an assignment

Name(s): Killian Byrne & Ala Al Din Afana

Programme: Concurrent and distributed programming
Module Code: CA4006
Assignment Title: *Assignment 1*
Submission Date: 23/03/2020
Module Coordinator: Dr Rob Brennan

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I/We have read and understood the Assignment Regulations. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

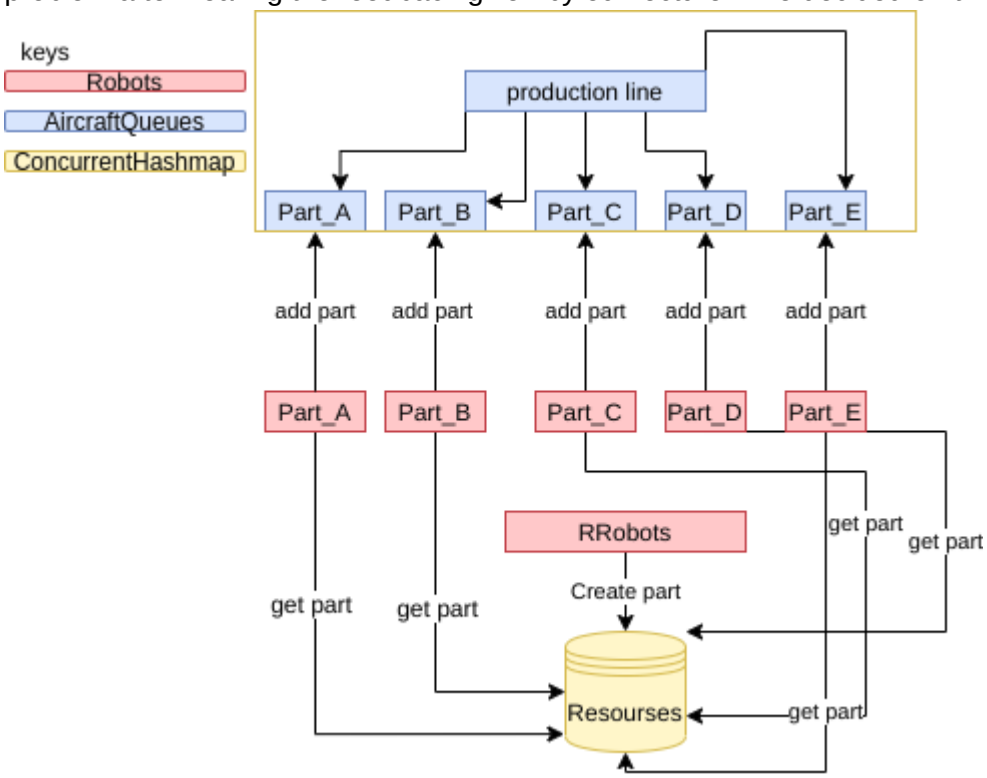
I/We have read and understood the referencing guidelines found at http://www.dcu.ie/info/regulations/plagiarism.shtml , https://www4.dcu.ie/students/az/plagiarism

and/or recommended in the assignment guidelines.

Name(s):Killian Byrne, Ala Al Din Afana, Date:23/03/2020

Student name: Ala Al Din Afana
Student number: 16395986
Student name: Killian Byrne
Student number: 16791985

When we first got this problem we were unsure on how to solve it we presented a real time systems approach during the class presentation and realised we were over complicating the problem after hearing the feedback given by our lecturer. We decided on this approach.



Given the problem we broke down our program in several classes. We first created an Aircraft class. We gave our aircraft many attributes such as id arrival time priority and workplan. At first it seemed important to use all these attributes but as we coded the assignment and dug deeper into the concurrent objects in java we realised the attribute priority is not important.

Aircraft.java
We then made a class called aircraft generator. This class is given a number of aircrafts to create each aircraft is given a random workplan with a random amount of parts to be installed.
For the purpose of this assignment we made five parts that need to be installed on the aircraft
{"part_A","part_B","part_C","part_D","part_E"} this could have been any amount of parts but the more parts added the harder it is to see the results in real time.

AircraftGenerator.java

All aircrafts generated by the AircraftGenerator are added to the Aircraft queue.
The aircraft queue is a class that creates a queue for the aircrafts to be stored in .
We decided to use the queue architecture because like this we ensure that starvation cannot occur to any aircraft and fairness.

We created a few functions to assist us to solve this problem. These functions include (add remove and get.
To ensure no two threads can't write to the queue at the same time we used the keyword synchronized for adding and removing to the queue this protects the section of code with a lock and Ensures no Race conditions occur when reading/writing to a shared resource.

Now that we have AircraftQueue with Aircrafts that need parts to be installed we had to decide on a way for the robots to concurrently add parts to different Aircrafts without any deadlocks or starvation. After some research we realized we can utilise the power of java and use ConcurrentHashMap . Unlike normal hashmaps ConcurrentHashMap are thread safe in nature, this allows multiple threads to work on it without any complications. Default concurrency-level of ConcurrentHashMap is 16 which is sufficient for the purpose of this assignment.
In ConcurrentHashMap, at a time any number of threads can perform retrieval operation but for updation in object, thread must lock the particular segment in which thread want to operate.This type of locking mechanism is known as Segment locking or bucket locking.Hence at a time 16 updation operations can be performed by threads.

The ConcurrentHashMap architecture was perfect for our design.
We added 6 keys to the ConcurrentHashMap, each key had a value of an AircraftQueue.
The keys included the production line queue (0) then a queue for each part which is initially empty. We will talk about how we utilised these queues later on in this report.

Main.java

The production line is represented by a thread called ProductionLine that takes in the concurrenthashmap containing keys mapped to each aircraft queue. The thread checks the first part that needs to be installed for each aircraft in the production line queue and sends the aircraft to the appropriate queue for that part in the hashmap.
For example:
{ProductionLine queue :{ Aircraft 1 , Aircraft 2 , Aircraft 3, Aircraft 4,})
Check the part on top of the queue for Aircraft 1 .Let's assume that top of the queue was Part A.
We add Aircraft 1 to AircraftQueue mapped to Part A.
{ Part A : { Aircraft 1 , ……}
this production line thread will keep on running as long as long as all AircraftQueues are not empty.
The Main class also calls 5 robots. Each robot is a thread that will be in charge of installing specific parts to the Aircrafts. We will discuss how the robot works in more detail below.

Rescourses.java
To make the problem more realistic we added a resource class that contains all the parts that need to be added to aircrafts. After discovering how effective ConcurrentHashMaps are we decided to store the resources in this architecture as well. each part was mapped to a queue of parts . the parts are represented by strings. Example:
{part_A : {part_A, part_A , part_A part_A, part_A ,part_A ,part_A, part_A, part_A}}.
We created an add function to add resources and a remove function ro remove them.
Both functions use the synchronized keyword and allow only one thread to manipulate them at a time. A resource robot (RRobot) is in charge of maintaining the resource queue and creates a new resource and adds it to the queue once a resource is taken by an “installation” robot.

Robot.java
The Robot thread is incharge of installing parts to aircrafts. The class takes in three attributes:

Resources resource, ConcurrentHashMap<String,AircraftQueue> queues String part

Each robot is incharge of a specific part queue. The robot checks the work queue for the part that needs to be added to the aircraft if it matches the parts it’s responsible for, the robot will go to resources, remove a part and add it to the aircraft. The robot then checks the top of the work queue again for the aircraft If the part that needs to be installed is not specific to the robot it sends it back to the production line queue (which will add the aircraft to the queue for the next part that needs to be installed). This way we ensure fairness across all aircrafts. The idea of FIFO eliminates any chance of starvation. By using this approach all robots will be installing different parts for different aircrafts concurrently.
The robot threads are started when the main execution thread begins and are eliminated when all the aircrafts are completely built (on exit code (0)).

RRobots.java
RRobots are resource robots. They are incharge of maintaining the resources. Each robot is delegated to one part and maintains its queue. The robots will monitor the supply of each part and as soon as a part is removed it creates a new part and adds it to its resource queue The robots will run when the main class runs and terminate on exit code (0)

All the threads are started by the main execution thread and are terminated on exit code (0)

Work division
When we designed the system we split our initial design in 8 classes. We were each incharge of 4 classes and we pair-programmed the main class. We communicated with each other throughout the development of the code and made group decisions on any new updates to the design.

How to run
- To Compile run ./compile.sh
- In linux run the run script “./run”