Middle East Technical University Department of Computer Engineering

CENG 443

Introduction to Object-Oriented Programming Languages and Systems

Fall 2023-2024

Lab 2 Preliminary - Keeping The Assembly Line Rolling
(Ver 1.0)

1 Introduction

Keywords: *Concurrency*

In this assignment, you are tasked to write a program to manage production and assembly lines for a car factory.

2 Overview

Several components are needed to produce a car. For the simplicity of the assignment, we will assume the necessary parts are:

- 1 car engine,
- 4 car wheels,
- 6 car glasses.

These components are produced in their respective production lines and joined together on the assembly line. But due to supply and demand, keeping the production lines running non-stop is not economic. So instead, each production line follows the steps below continuously:

- Start the production line
- Produce just enough components to produce a single car and not more
- Stop the production line
- Wait until a car is produced

Similarly, the assembly line should:

- Wait until all necessary components are produced
- Start the assembly line
- Assemble a car
- Stop the assembly line

In other words, the production lines should wait for the assembly line to assemble a car to re-start production and the assembly line should wait for all three production lines to produce enough components to assemble.

3 Example Output

Due to indeterminism, your output will likely be different from the one below. What is important is that your output should not violate the specification e.g. produced cars should be assembled after all necessary components are produced or, production lines should start producing only after it is started.

First few lines of an output of the program can be seen below. Note that your app should run **indefinitely** until terminated:

```
Started the Wheel Production Line
Started the Engine Production Line
Started the Glass Production Line
Produced a Wheel
Produced a Glass
Produced an Engine
Stopped the Engine Production Line
Produced a Wheel
Produced a Glass
Produced a Glass
Produced a Wheel
Produced a Glass
Produced a Wheel
Stopped the Wheel Production Line
Produced a Glass
Produced a Glass
Stopped the Glass Production Line
Started the Assembly Line
Assembled a Car
Started the Glass Production Line
Stopped the Assembly Line
Started the Wheel Production Line
Started the Engine Production Line
Produced a Wheel
Produced a Glass
Produced a Glass
Produced an Engine
Produced a Wheel
Stopped the Engine Production Line
Produced a Glass
Produced a Glass
Produced a Wheel
Produced a Glass
Produced a Wheel
Produced a Glass
Stopped the Wheel Production Line
Stopped the Glass Production Line
Started the Assembly Line
Assembled a Car
Started the Glass Production Line
Stopped the Assembly Line
Started the Wheel Production Line
....
```

4 Some Remarks

Although your code will not be graded, and you have freedom in your design, your code should utilize concurrency nonetheless since the upcoming lab will be graded accordingly.

- Your main class name should be **CarFactory**.
- All three production lines and the assembly line should run as a separate thread and act independent of each other.
- For every action (starting or stopping a line, production of a component or assembly of a car), you should use the corresponding action method in **Actions** class.
- When ensuring coordination between production and assembly lines you can use any synchronization primitive other than atomics.
- You should not make any assumptions based on timings of actions since different timing intervals may be used during the evaluation of the upcoming quiz.
- You should make sure that no deadlock, no starvation,no busy-wait or no unnecessary wait will occur in runtime.
- To expand the remarks above, solution models below (but not limited to) are not valid solutions:

  Running all logic on a single thread

  Production lines waiting for each other to start or stop

  Threads sleeping and waking up periodically to poll a variable

  Threads waking up more than necessary

  Threads causing other threads to block while sleeping
- Atomicity of the output depends on implementation. Hence, you may need to add synchronization between print statements in **Actions** Class.
- Use Java 8 or a newer version