Middle East Technical University Department of Computer Engineering

CENG 443

Introduction to Object-Oriented Programming Languages and Systems

Fall 2020 - 2021

Homework 2 - Industry 4.0
version 1.0

Due date: 25 December 2020, 23:59

1 Introduction

In this assignment, you will build an animation about a smart factory where different types of robots work in coordination to produce new robots, some of which also join them to speed up the production. When building the animation, you will employ the object-oriented design principles you have learned during the classes. Most of the code, including the GUI, is already completed for you, so you will only focus on robot logic.

2 Robot Logic

A robot consists of four **Part**s: a **Base**, an **Arm**, a **Payload**, and a **Logic** chip. However, only the following payload - logic combinations are possible: **Gripper - Supplier**, **Welder - Builder**, **Camera - Inspector**, and **MaintenanceKit - Fixer**. The logic chips work as follows:

1. **Supplier**: If the production line is not full, the supplier randomly generates a robot part and places it at the next available location on the production line. If the production line is full, the supplier removes a randomly selected part on the production line. After completing either action, the supplier notifies the builders.
2. **Builder**: The builder checks all parts on the production line to see if there is a possibility to fulfill one production step. A production step might be to take an arm and attach it to a base, or to take a payload and attach it to a base-arm, or to finish a robot by adding the correct logic chip, or to pick up a completed robot and add it alongside of worker robots (to speed up the production), or to store the completed robot if there is enough storage space. If there is no possibility of fulfilling a production step, then

the builder waits. The builder also signals the factory to stop production once the storage space is full.

3. **Inspector**: The inspector checks each worker robot to see if it has lost one of its part (due to wear and tear). If so, the inspector puts that robot in a broken robots list and notifies the fixers.

4. **Fixer**: The fixer fetches the first broken robot from the broken robots list. Then, it fixes and wakes up the broken robot. If there are no broken robots in the list, then the fixer waits.

3 Design Issues and Grading Rubric

- You will only complete parts where **TODO** comments are.
- You will only use threading primitives: synchronized, wait, notify, notifyAll. Do not use high level concurrency constructs.
- When implemented properly, the animation should not throw any exceptions during its runtime.
- Encountering runtime problems: -25 pts
- Commenting your code: 5 pts
- Supplier works as expected: 10 pts
- Inspector works as expected: 10 pts
- Fixer works as expected: 10 pts
- Builder works as expected: 45 pts
  - Attaches correct parts to a robot: 10 pts
  - Places the completed robot at worker robots: 10 pts
  - Places the completed robot at robot storage: 10 pts
  - Signals the factory to stop production: 15 pts
- You will practice reflection for accessing/modifying private attributes of a class (Check TODO comments in **Common**): 10 pts
- You will apply Factory and Abstract Factory patterns for creating robot parts (Check TODO comments in **Factory**): 10 pts
- If you need additional functionalities, define them in **Common**.

4 Submission

Submission will be done via ODTUClass. You will submit a zip file called **hw2.zip** that contains the following files: **Supplier.java**, **Builder.java**, **Inspector.java**, **Fixer.java**, **Factory.java**, and **Common.java**. A penalty of **5 x LateDay x LateDay** will be applied for submissions that are late at most 3 days.