ENSE 370 Software Systems Design

Assignment A02

Design Patterns I

(Due date: Friday February 14th 2025, 23:59)

Instructions:

Design Diagrams

1. On separate pieces of paper or using a drawing software, draw the Use-Case Diagram and the Class Diagram for this assignment.
2. Scan or convert your drawings to Adobe PDF format.
3. Name your files as: **UseCaseDiagram.pdf**
   ClassDiagram.pdf
4. Copy your files to Snoopy and then submit them.
   (Type **~ense370/bin/submit A02 UseCaseDiagram.pdf ClassDiagram.pdf**)

Source codes

5. Login to Snoopy (**snoopy.engg.uregina.ca** / **142.3.105.92**).
6. Create your program using any text editor of your choice (e.g. **vim / emacs / nano**).
7. Name your files using the following convention:

Class definition files	Capitalize first letter of each word e.g. RobotFactory.java
Main program file	*"A"+number+username+".java"* (e.g. **A02jon123.java** if your username is jon123)

8. Compile your program and test that it works correctly.
   (e.g. **javac A02jon123.java RobotFactory.java [other files …]**)
   (and **java A02jon123**)
9. Submit all your java source code files.
   (Type **~ense370/bin/submit A02 A02jon123.java RobotFactory.java [other files …]**)

In this assignment, you will create a Java program to represent a factory robot controller that sends commands to multiple robots in an assembly line to create a specific object following a sequence of predetermined steps. There are three types of robot:

1. Cutting robot, which cuts parts from raw materials,
2. Drilling robot, which drills holes in a part, and
3. Assembly robot, that combines parts together (e.g., putting screws into the holes of one or more parts)

All robots have the following information and actions:

1. Type (i.e. Cutting, Drilling or Assembly)
2. Manufacturer (Cutting and Drilling robots are made by "Regina Machines" and the assembly robot is made by "SK Robotics")
3. Serial number (a unique 12-digit number)
4. A constructor with 3 parameters that will initialize the robot to the given values
5. An action fetchParts() that will fetch the appropriate part or raw material from storage
6. An action doTask() that performs the required task (of cutting, drilling or assembling)
7. An action storeParts() that will store the finished part or product back to storage

Note that there can be more than one instance of a cutting robot (e.g. there could be 6 cutting robots cutting one part each at the same time, so you will end up with 6 identical parts). Similarly, there can be more than one instance of a drilling robot (e.g. there can be 3 robots drilling holes in 3 of the parts at the same time, and then all 3 robots do the drilling again on the another 3 parts so as to make 6 drilled parts), but there can only be one assembly robot (because all the parts will come together to make only one product).

Design the system using design pattern(s) that you have learnt in class (you may use more than one design pattern at the same time). Note that in your design you **must** be able to easily add new robot classes in the future, do **not** repeat the same information in multiple objects of the same class, and there **must** be only one instance of the assembly robot.

In your main program file, create **two** instances of a cutting robot, **two** instances of a drilling robot and **one** instance of an assembly robot. You may initialize the robot to any meaningful initial values of your choice. Use the robot controller to send instructions to each robot to cut parts, drill holes and assemble the product. Assume that it takes **four** parts with holes in them to make one final product. Note that for each step the robot must first fetch the raw materials or parts from storage, and then store the finished part or product back to storage when that step is completed.

Print some meaningful messages to the screen to demonstrate that that robot controller is carrying out the process correctly. Please include comments to explain which design pattern(s) you are using and what are the classes in that design pattern.

Your output may (but not limited to) look as follows:

```
$ java A02jon123
Cutting Robot created
Regina Machines 100200300401
Cutting Robot created
Regina Machines 100200300402
Drilling Robot created
Regina Machines 200200300401
Drilling Robot created
ReginaMachines 200200300401
Assembly Robot created
SK Robotics 300200300401
Raw material fetched
Raw material fetched
Raw material cut
Raw material cut
Cut part sent to storage
Cut part sent to storage
Raw material fetched
Raw material fetched
Raw material cut
Raw material cut
Cut part sent to storage
Cut part sent to storage
Cut part fetched
Cut part fetched
Holes drilled
Holes drilled
Drilled part sent to storage
Drilled part sent to storage
Cut part fetched
Cut part fetched
Holes drilled
Holes drilled
Drilled part sent to storage
Drilled part sent to storage
Drilled part fetched
Drilled part fetched
Drilled part fetched
Drilled part fetched
Product Assembled
Product sent to storage
$
```

The End