TP : Robot de cuisine

Compétences visées :

- Mise en pratique des connaissances acquises en modélisation UML avec le logiciel StarUML.
- Mise en pratique des connaissances acquises en programmation Java.
- Mise en pratique des outils de développement logiciel (IDE Eclipse, Apache Ant, Apache Log4j2).

Dans les archives de l'école, nous avons retrouvé le code d'un robot de cuisine. Ce code est donné dans la Figure 1. Après une analyse du code, vous rendrez le code du robot fonctionnel et livrable puis vous lui ajouterez une nouvelle fonctionnalité.

Partie I : Analyse du robot de cuisine

```
1  package fr.esisar.cook;
2
3  import org.apache.logging.log4j.LogManager;
4  import org.apache.logging.log4j.Logger;
5
6  public class CookBot {
7
8        private static final Logger LOGGER = LogManager.getLogger(CookBot.class);
9
10       static final int ON = 0;
11       static final int OFF = 1;
12       static final int COOK = 2;
13
14       private int state;
15
16       public CookBot() {
17             super();
18             state = ON;
19       }
20
21       public void switchOn() {
22             if (state == ON) {
23                   LOGGER.info("CookBot is switched on...");
24             } else if (state == COOK) {
25                   state = ON;
26                   LOGGER.info("CookBot is switched on...");
27             } else if (state == OFF) {
28                   state = ON;
29                   LOGGER.info("CookBot is switched on...");
30             }
31       }
```

```
32
33      public void regularCook() {
34            if (state == ON) {
35                  state = COOK;
36                  LOGGER.info("CookBot is cooking...");
37            } else if (state == COOK) {
38                  LOGGER.info("CookBot is cooking...");
39            } else if (state == OFF) {
40                  LOGGER.warn("Cannot cook when it is switched off");
41            }
42      }
43
44      public void switchOff() {
45            if (state == ON) {
46                  state = OFF;
47                  LOGGER.info("CookBot is switched off");
48            } else if (state == COOK) {
49                  LOGGER.warn("Cannot be switched off while cooking");
50            } else if (state == OFF) {
51                  LOGGER.info("Already switch off");
52            }
53      }
54  }
```

Figure 1 : Code du robot de cuisine.

Question 1 : Réalisez, avec le logiciel StarUML, le diagramme de classes du robot de cuisine (Figure 1).

Question 2 : Réalisez, avec StarUML, le diagramme d'états-transitions du robot de cuisine.

Indice : Dans StarUML, lorsque vous cliquez sur un élément, vous retrouvez, en bas à droite de votre écran, un éditeur qui contient l'ensemble des caractéristiques de l'élément cliqué, que vous pouvez modifier.

Partie II : Et si on ajoutait une nouvelle fonctionnalité !

Question 3 : Créez un projet Eclipse, nommé Cook-v1. Récupérez le code du robot de cuisine et faites le fonctionner correctement avec la mise en place du logger et de l'automatisation des tâches (compilation, packaging et exécution).

Indices : Pour utiliser le logger Apache Log4j2, ajoutez dans un répertoire[1] lib/ de votre projet les deux bibliothèques log4-api-2.XX.XX.jar et log4j-core-2-XX.XX.jar[2]. Ces bibliothèques doivent être ajoutées au *classpath* d'Eclipse. De plus, vous devez ajouter à la racine de votre projet un fichier de configuration du logger nommé log4j2.xml, qui contient le code suivant pour activer l'affichage dans la console de tous les messages d'un niveau supérieur ou égal à trace :

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
  <Appenders>
    <Console name="Console">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} -
%msg%n"/>
    </Console>
```

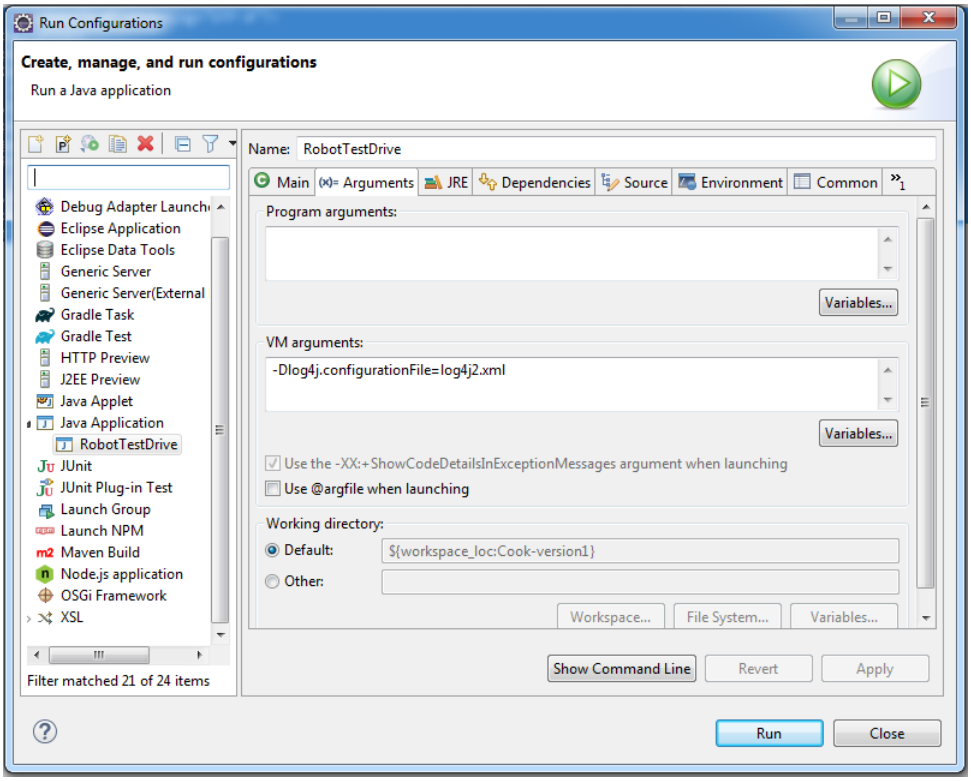
[1] Un répertoire de type *Folder* et non *Source Folder*.
[2] Où XX-XX est le numéro de version de Apache Log4j2.

```
  </Appenders>
  <Loggers>
    <Root level="trace">
      <AppenderRef ref="Console"/>
    </Root>
  </Loggers>
</Configuration>
```

Pour que ce fichier de configuration soit pris en considération à l'exécution, il faut le préciser :

- dans Eclipse, l'exécution du programme s'il est fait avec la commande Run, il faut préciser les options dans la fenêtre Run Configurations… (clic droit sur la classe principale, puis Run as > Run Configurations…). Dans l'onglet Arguments, dans la partie VM arguments, il faut ajouter :

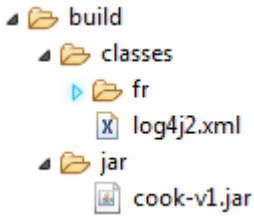
```
-Dlog4j.configurationFile=log4j2.xml
```



- avec Apache Ant, il faut préciser le *classpath* à la commande javac, c'est-à-dire préciser qu'il faut ajouter au *classpath* l'ensemble des bibliothèques disponibles dans le répertoire lib :

```
<classpath>
     <fileset dir="${lib}" includes="log4j*.jar"/>
</classpath>
```

Ensuite, pour la tâche de packaging, il faut ajouter une commande qui permet de copier le fichier de configuration du *logger* dans le répertoire où il y a les classes compilées :



Pour que le jar soit exécutable et qu'il contienne, par défaut, l'ensemble des bibliothèques qui lui sont nécessaires, il faut préciser à la commande jar les informations suivantes :

```
<zipgroupfileset dir="${lib}">
     <include name="**/log4j*.jar"/>
</zipgroupfileset>
```

Enfin, pour l'exécution de votre archive, vous devez passer en argument à la commande java le fichier de configuration log4j2.xml :

```
<arg value="-Dlog4j.configurationFile=log4j2.xml"/>
```

Le robot de cuisine est un super robot de cuisine qui permet actuellement seulement de cuisiner. Evidemment, il serait plus perfectionné (et donc plus cher) s'il était capable de cuire doucement de bons petits plats.

Question 4 : Complétez le diagramme d'états-transitions puis le diagramme de classes réalisés précédemment pour ajouter un nouveau mode de cuisson SLOW_COOK qui permette de faire cuire doucement les ingrédients. Lorsque le robot est dans l'état SLOW_COOK et que l'on souhaite :

- l'allumer (switchON), il passe dans l'état ON (*CookBot is switched on...*),
- l'éteindre (switchOFF), il reste dans l'état SLOW_COOK (*Cannot be switched off while slowly cooking*),
- passer à une cuisson classique (regularCook), il passe dans l'état COOK (*CookBot is cooking...*).

De plus, la fonction slowCook sera également ajoutée au robot :

- si le robot est dans l'état ON, il passera dans l'état SLOW_COOK (*CookBot is cooking at a lower temperature...*),
- si le robot est dans l'état OFF, il restera arrêté (*Cannot cook when it is switched off*),
- si le robot est dans l'état COOK, il passera dans l'état SLOW_COOK (*CookBot is cooking at a lower temperature...*),
- si le robot est dans l'état SLOW_COOK, il restera dans cet état (*CookBot is cooking at a lower temperature...*).

Question 5 : Complétez le code du robot pour prendre en considération la nouvelle fonctionnalité de cuisson douce.

Question 6 : Analysez la qualité du code du robot de cuisine autant d'un point de vue modélisation qu'implémentation ?