

Le patron *Singleton*Exercice 1 – Codage d'un Logger

Soit le code suivant (al-tp7-fichiers/Appli.java) :

```
1  import myLib.* ;
2
3  public class Appli {
4      private LibStuff stuff ;
5
6      public void initApp() {
7          // Info : "Initializing App..."
8          // ... some code here
9          stuff = new LibStuff() ;
10         // Info : "Initialization done !"
11     }
12
13     public void closeApp() {
14         // Info : "Closing app..."
15         // ... some code here
16         // Info : "Resources freed !"
17     }
18
19     public void run() {
20         boolean errors = false ;
21         for (int i = 0; i < 10; ++i) {
22             int action = (int) (100.0 * Math.random()) ;
23             if (action < 50) {
24                 stuff.doSomething(action) ;
25             } else {
26                 // Error : "Invalide action : "+action
27                 errors = true ;
28             }
29         }
30         if (errors) {
31             // Warning : some errors occurred during processing.
32         }
33     }
34
35     public static void main(String[] args) {
36         Appli app = new Appli() ;
37         app.initApp() ;
38         app.run() ;
39         app.closeApp() ;
40     }
41 }
```

On souhaite pouvoir journaliser les différents aspects du cycle de vie de cette application à des fins de débogage (debugging). Pour cela, on souhaite utiliser un objet, représentant un journal des événements (*log*), sur lequel on pourra effectuer les actions suivantes :

- émettre un message d'information (*INFO*),
- émettre un message d'avertissement (*WARNING*),
- émettre un message d'erreur fatale (*ERROR*),

Chacune de ces actions sera réalisée par une méthode spécifique prenant en argument le message à émettre sous forme d'une chaîne de caractères.

On souhaite en outre que les messages soient préfixés par leur numéro (ordre d'émission), leur type (*INFO*, *WARNING* ou *ERROR*), et envoyés dans la console d'exécution (`System.out`).

Enfin, on note que les points d'émission et leur type sont marqués dans le code source sous forme de commentaires et que la bibliothèque `mylib/LibStuff.java` doit également faire l'objet d'émissions de messages.

Question 1.1 : Quelle solution proposez-vous de mettre en place ? Donnez son diagramme UML.

Le diagramme UML ne rend pas compte des détails d'implémentation du patron.

Question 1.2 : Qualifiez complètement l'implémentation de la solution que vous allez mettre en place, justifiez les et proposez des **tests logiciels** permettant de valider toutes ces caractéristiques.

Exercice 2 – Évolution du Logger

On reprend l'exercice précédent et on se propose de fournir un deuxième type journal d'événements qui enverra cette fois les messages dans un fichier portant le même nom que l'application¹ avec l'extension `.log`. De plus, pour ce type de logger, les messages ne seront plus numérotés.

Les deux loggers seront disponibles dans l'application et l'utilisateur pourra activer le second type en ajoutant le paramètre `filelog` sur la ligne de commande `java Appli filelog`, sinon c'est le premier logger qui sera activé.

Enfin, on souhaite ne pas avoir à modifier l'application cliente obtenue à l'exercice 1, mis à part l'initialisation.

Question 2.1 : Quelle solution allez-vous mettre en place ? Donnez son diagramme UML complet et détaillez son implémentation, en particulier l'initialisation.

Question 2.2 : Les modifications apportées violent-elle des principes ?

Question 2.3 : Ajoutez des tests logiciels permettant de valider son fonctionnement.

1. Astuce : on pourra utiliser la propriété système `sun.java.command`.