Exercices de Patterns (10)

Question 1

Vous venez de rejoindre l'équipe de développement du programme IPLCommander, un puissant gestionnaire de fichiers écrit en Java.

- L'interface core.Fichier décrit les opérations principales de notre programme : lister le contenu d'un répertoire, lire/écrire/effacer un fichier.
- La classe core.FichierImpl implémente cette interface, en restreignant l'accès à un répertoire particulier qui servira de racine aux chemins gérés par la classe.
- La classe gui.GUI gère l'interface homme-machine. Elle utilise Swing pour offrir une interface simpliste.
- La classe gui.IPLCommander est le programme principal : il instancie un FichierImpl et le GUI qui l'utilise puis rend la fenêtre visible.

Le programme fonctionne bien, mais votre client suspecte que certains utilisateurs l'utilisent mal. Il souhaiterait donc avoir la capacité de facilement configurer une version qui loggerait toutes les actions entreprises par les utilisateurs (par exemple via System.out.println). Quel pattern va vous aider à réaliser cela ? Implémentez-le.

Participants théoriques	*Noms utilisés dans le code*

Nom théorique des méthodes	*Nom utilisé dans le code*

Grâce aux logs, votre client détecte le problème : certains utilisateurs modifient les fichiers alors qu'ils ne devraient pas pouvoir le faire. Votre client vous demande d'implémenter un système permettant de facilement bloquer la lecture et/ou la modification et/ou l'effacement des fichiers. Quel pattern va vous aider à réaliser cela ? Implémentez-le.

Participants théoriques	*Noms utilisés dans le code*

Nom théorique des méthodes	*Nom utilisé dans le code*

Finalement votre client souhaiterait que tout le monde travaille avec les fichiers configurés sur son serveur plutôt qu'avec les fichiers locaux. Pour permettre cet usage à distance, vous avez :

- La classe APIServlet qui implémente une servlet supportant des requêtes HTTP POST et qui appellent les fonctionnalités de core.Fichier. Consultez le code pour voir comment y envoyer une requête (les paramètres attendus, l'encodage souhaité, etc).
- La classe WebServer qui démarre un serveur Jetty avec cette servlet.
- La classe HTTPUtils qui simplifie l'émission d'une requête, confer fin de ce document.

Cette tâche relève du « Distributed computing [1]». Les patrons que nous avons vus ne recouvrent pas les architectures multi-processus qui communiquent via un réseau. Cependant, si on généralise un de ces patrons, on peut bel et bien proposer une solution pour cet exercice. Quel est ce patron ? Implémentez-le.

[1] https://en.wikipedia.org/wiki/Distributed_computing

HTTPUtils

Les classes fournies nativement par Java pour effectuer des requêtes Web ne sont pas des plus simples à utiliser. Vous trouverez sur la plateforme pédagogique du cours la classe HTTPUtils. Vous y trouverez notamment :

String utils.HTTPUtils.performPostCall(String url, Map<String,String> params)

- Envoie l'équivalent d'un appel Ajax à l'URL url.
- Les paramètres sont spécifiés dans la map params (null si aucun).
- Renvoie soit :
  - La réponse si le code HTTP est OK (200)
  - Un utils.HTTPUtils.HTTPException si le code HTTP<>200 ; getMessage() donne la réponse renvoyée et getHTTPStatus() le code HTTP.
  - Un utils.HTTPUtils.HTTPNetworkException si l'URL est mal formée ou qu'il y a un problème avec le réseau ; getCause() renvoie l'exception qui a réellement été jetée.

Il y a aussi performGetCall qui est similaire.

Attention : ces méthodes sont synchrones, elles bloquent en attendant la réponse. A vous de rendre ces appels asynchrones à l'aide d'une AsyncTask.

Question 2

Examen Janvier 2015

Soit le code suivant :

```
public class Logger {
   public static void main(String[] args) {
      Logger logger = ...;
      logger.logMessage(Logger.INFO, "This is an information.");
      logger.logMessage(Logger.DEBUG, "This is a debug level information.");
      logger.logMessage(Logger.ERROR, "This is an error information.");
   }
}
```

Tous les messages doivent s'afficher à la console (System.out.println), peu importe leur niveau. Les messages de niveau Logger.DEBUG et Logger.ERROR doivent en plus être écrits dans un fichier appelé out.log. Finalement les messages de niveau Logger.ERROR doivent en plus s'afficher sur l'erreur standard (System.err.println). Quel pattern recommandez-vous pour réaliser cela ? Implémentez-le.