TD6 : Command, Observer (et Factory)

Exercice 1 - Automate

Une société de fabrication d'automates programmables souhaite développer sa nouvelle gamme de produits basés sur la technologie Java.

Le cœur du moteur de commandes doit être facilement extensible à de nouvelles fonctionnalités.

Le Robot est composé d'un bras articulé muni d'une pince pouvant se déplacer linéairement dans les trois directions: X, Y et Z. Pour cela il dispose d'un moteur pas à pas pour chaque axe.

Chaque moteur peut se déplacer en avant (positif) ou en arrière (négatif) d'un nombre de pas fixé,

sur les trois axes du robot: X,Y et Z. La méthode move(int steps) permet de donner le nombre entier de pas du déplacement sur le moteur correspondant.
La consigne light(boolean on) permet d'allumer ou éteindre la lumière au bout du bras. Enfin les consignes open() et close() permettant d'ouvrir ou fermer la pince.

A terme, on souhaite pouvoir :

- rendre asynchrone l'exécution des actions sur le robot
- pouvoir effectuer un arrêt d'urgence
- pouvoir visualiser l'état d'avancement des commandes de plusieurs manières :
  - affichage texte dans une console
  - affichage graphique des mouvements
- remettre le robot dans son état initial après une série d'actions

1/ Rappeler le schéma du Patern Command

2/ Modifier le code donné (http://igm.univ-mlv.fr/ens/Master/M1/2012-2013/POO/automate0.zip) pour mettre en place le pattern Command.
Transformer le main() en conséquence.

3/ créer une Factory de Command's et utilisez la dans le code du main().

4/ On veut découpler réellement les demandes d'actions au Robot et leur exécution : ajouter la gestion d'une pile de commandes, ainsi que l'exécution asynchrone

note : vous pouvez vous aider des implémentations de BlockingQueue.

5/ On souhaite surveiller les mouvement des axes en implémentant des messages indiquant si l'axe démarre son déplacement ou s'il stoppe le mouvement.

Pour l'instant, la surveillance se traduira simplement par des affichages « sysout ». Mais la structure implémentée devra permettre de « brancher » une interface graphique qui montrera l'évolution des actions sur le Robot.

- Rappeler le schéma du Pattern Observer.
- On implémentera cette surveillance en utilisant le Pattern Observer. N'utilisez pas les Observable du JDK.
- Attention, réfléchissez aux objets qui doivent être observés. Ce sot les objets applicatifs et pas les commandes.

Ajouter un observer qui loggera dans un fichier l'historique de toutes les actions demandées

6/ Ajouter la gestion de priorité d'une commande : une commande Abort, de priorité maximum, doit pouvoir s'exécuter avant toutes les autres commandes en attente (on ne s'intéressera pas ici à l'interruption de la commande en cours d'exécution).

note : vous pouvez vous aider de PriorityBlockingQueue.

note : après l'exécution de Abort, vider la liste des commandes en attente

7/ On veut pouvoir facilement remettre le robot dans son état initial. Par exemple :

```
controller.startSession()
…. envoi de commandes … (… controller.submit(...) ...)
// "annule" toutes les commandes envoyées et
// remet donc le robot à sa position initiale, avec la pince et la lumière dans l'état initial
controller.undoSession() ;
```