La programmation par évènements

Exercice I : Observer des objets (*Extrait du médian P09*)

Java fourni des classes et interfaces qui permettent de surveiller d'autres objets. La classe Observable permet d'indiquer qu'un objet qui en hérite peut être observé par d'autres objets. Les observateurs quant à eux sont des objets qui implémentent l'interface Observer.

Etant données cette classe et cette interface, le fonctionnement d'observation est le suivant. La classe Observable permet d'ajouter et supprimer un observateur (méthodes `addObserver` et `deleteObserver`). Elle permet aussi de notifier ses observateurs (méthode `notifyObservers`). Pour un observateur, la notification d'un changement sera faire par le biais d'un appel à sa méthode `update`.

I. Mise en œuvre basique de l'observation

1. Donner le code source de l'interface Observer.
2. Donner le code source de la classe Observable.

II. Application de l'observation à la classe Lampe

3. Donner le code source d'une Lampe observable.
4. Donner le code source d'un observateur de Lampe.

III. Extension du principe d'observation

5. La classe Observable possède une méthode `setChanged()` qui permet d'indiquer qu'un changement a effectivement eu lieu dans l'objet. Un appel à `notifyObservers` notifira alors les observateurs si et seulement si un changement a été signalé par le biais d'un appel préalable à `setChanged()`. Redonner le code de classe Observable avec ces nouveaux paramètres (inutile de redonner le code des méthodes qui ne changent pas).
6. Il possible de passer des informations relatives au changement - quelle que soit sa nature - d'un objet observé à ses observateurs. Pour cela, la méthode `notifyObservers` reçoit cet objet en paramètre et le transmet à la méthode `update` dans un paramètre aussi. Donner le code source de la méthode `notifyObservers` et la définition de la méthode `update` pour le support d'indications sur le changement.

IV Analyse et critique de la conception

7. Expliquer pourquoi utiliser une interface pour représenter un observateur ?
8. Expliquer pourquoi utiliser une classe pour représenter un objet observable ?
9. Pourquoi utiliser une méthode `setChanged` en plus de la méthode `notifyObservers` ?
10. Donner toutes les limites de cette conception.

Exercice 2 : Vue graphique d'un compteur

Implanter les classes permettant d'implanter un compteur graphique tel qu'il a été vu en cours (avec le modèle MVC). Etudier l'impact des threads sur le fonctionnement de l'interface.