Développement informatique avancé orienté application - TP

L'architecture MVC

Virginie Van den Schrieck

Dans le cadre de ce TP, nous allons développer pas à pas une petite application suivant l'architecture MVC. Les notions illustrées dans ce TP sont expliquées dans la section 3.1 du site INGInious.

1 Découverte du code

Vous trouverez sur le Campus Virtuel un fichier zip contenant une application (incomplète) permettant de gérer une petite bibliothèque, avec une interface graphique et une interface console. Importer ce code dans Eclipse, et prenez le temps d'explorer sa structure : Quelles sont les différentes classes ? A quoi servent-elles ?

Vous remarquerez que la structure de ce programme est très similaire à l'exemple donné sur le site Inginious. N'hésitez pas à commencer par observer et tester ce dernier avant de vous lancer dans le TP.

2 Modèle

Le modèle est composé de deux classes : `Livre` et `Bibliothèque`. Elles sont complètement implémentée, sauf pour ce qui est du rôle de *modèle*. C'est la classe `Bibliothèque` qui jouera ce rôle.

Modifiez `Bibliothèque` pour qu'elle remplisse le rôle de modèle. Vous devez ajoutez deux types de modification :

1. Tout d'abord, transformer la classe en objet `Observable`
2. Ensuite, notifier les observateurs en cas de modification de l'état de la bibliothèque.

Observez ensuite la classe `test.BibliothequeMVC`. Il s'agit de la classe qui démarrera tous les acteurs de l'application. Créez une instance de `Bibliothèque` dans le constructeur de `BibliothèqueMVC`.

3 Contrôleur

Cette application ne possède qu'une seule classe faisant office de contrôleur. Elle sera donc instanciée deux fois : Une fois pour contrôler la GUI, une seconde fois pour contrôler l'interface console. Le code de cette classe doit donc servir dans les deux cas de figures !

— Implémentez la méthode `empruntеLivre()` : Si l'emprunt a pu avoir lieu au niveau du modèle, alors le contrôleur affiche un message de succès sur la vue. Sinon, il informe l'utilisateur de l'échec de l'emprunt (conseil : observez ce qu'offre la classe `BibliothequeVue`)

— Implémentez la méthode `rendreLivre()` : elle signale au modèle que le livre est rendu et affiche un message sur la GUI.

Ensuite, dans le constructeur de la classe `test.BibliothequeMVC`, créez un contrôleur pour l'interface console et un autre pour l'interface GUI.

4 Vues

Deux vues vous sont proposées : une GUI et une interface console. Ne soyez pas surpris si ces classes vous paraissent illisibles : Elles utilisent les notions d'interface graphique et de thread que nous n'avons pas encore vus. Essayez malgré tout de comprendre le fonctionnement général.

Notez que ces deux vues héritent de la classe abstraite `BibliothequeVue`, qui regroupe leur comportement commun, à savoir l'enregistrement en tant qu'observateur.

Complétez la classe `BibliothequeVue` pour que l'enregistrememt en tant qu'observateur auprès du modèle s'effectue.

4.1 GUI

— Observez la classe qui implémente la GUI. Pouvez-vous identifier le code qui est exécuté lorsque le modèle est modifié ? Justifiez.

— Quand et comment la vue interagit-elle avec son contrôleur ?

— Créez une vue GUI dans le constructeur de la classe `test.BibliothequeMVC`, et associez-la au contrôleur ad-hoc.

4.2 Console

Même questions que pour la GUI.

5 Fonctionnement

Vérifiez que vous avez bien remplacé tous les `TODO` dans l'application. Vous devez avoir complété quatre classes au total. Un fois ces vérifications effectuées, lancez l'application et vérifiez que tout fonctionne correctement.

Observez notamment si les changements effectués dans une interface sont bien reflétés dans l'autre interface ! Soyez également sûr de bien expliquer les interactions entre les trois parties de l'architecture.