IUT Informatique **2ème Année**
Université d'Orléans
G. Cleuziou

Module : Conception et Programmation Objet Avancées (CPOA)

TP3 Atelier de Génie Logiciel (1h30)

L'objectif de ce TP est de mettre en œuvre le design pattern « observateur » sur l'application *Bank* en cours de développement. Pour cela vous pourrez repartir du programme obtenu en fin de TP2 (préférable) ou à défaut, reprendre le code fonctionnel proposé sous Célène.

PARTIE A. Mise en place du pattern « observateur » sous Umbrello

Conception d'un composant graphique :

1. Ajouter une classe *Widget* qui spécialise la classe JFrame (de javax.swing) et qui contient un attribut *message* de type JLabel (également présent dans Swing).

Un objet Widget correspond à une fenêtre graphique qui affichera des messages d'information sur un compte auquel il sera « abonné ».

2. Définir la classe qui correspondra aux objets « observables » et la classe qui correspondra aux objets observateurs puis mettre en place la conception du pattern « observateur ». Cette étape doit conduire à l'ajout :
    - d'une classe *Observable*,
    - d'une interface *IObservateur*,
    - d'une association entre l'observable et l'observateur,
    - d'une relation de généralisation vers l'observable
    - d'une réalisation de l'interface d'observateur

Préparation et génération de code :

3. Ajouter dans les bonnes classes les opérations :
    - *notifierObservateurs(String)* : pour notifier tous les observateurs d'un objet observable
    - *alerter(String)* : pour permettre à un observateur de réagir à la notification d'un observable

4. Générer le code Java des trois classes créées : *Widget*, *Observable*, *IObservateur*.

PARTIE B. Implémentation fonctionnelle du pattern « observateur »

Implémentation du *Widget :*

1. Importer le package *swing* puis ajouter à la classe un constructeur *Widget(String titre)* qui initialise un objet Widget ainsi :
    - affichage du message (JLabel) par défaut « Compte ok ! »
    - ajout tu titre « title » à la fenêtre (méthode *setTitle()*)

- dimensionner la fenêtre : par exemple 300x200 (méthode *setSize()*)
- définir l'opération *alerter(String info)* de sorte à ce que le message affiché devienne *info*.

Implémentation de l'*Observable* :

2. Vérifier la structure de la classe ainsi que la possibilité d'ajouter/supprimer des observateurs
3. Définir l'opération *notifierObservateurs(String info)* qui consiste à activer l'opération *alerter(String info)* sur chaque observateur
4. Ajouter la généralisation sur la classe qui doit être observable (cette classe n'ayant pas fait l'objet de génération de code)

Implémentation du fonctionnement *Observable*/*IObservateur* :

5. Modifier le menu principal de sorte à permettre l'ajout d'un widget affichant les informations d'alertes sur un compte défini.
6. Faire en sorte que le widget affiche « Compte bloqué ! » dès que le compte est bloqué (pour un compte courant) puis « Compte ok ! » dès que le compte est débloqué.

Test de fonctionnement sur le scénario suivant :

7. Ajouter un compte courant « CCP1 »
8. Ajouter un widget associé au compte « CCP1 »
9. Réaliser un crédit de 100€ sur le compte « CCP1 »
10. Réaliser un débit de 250€ sur le compte « CCP1 »

Limite actuelle de notre conception :

11. Poursuivre le scénario précédent en ajoutant un second widget sur le même compte...

PARTIE C. Utilisation de l'API Java

Le design pattern « observateur » existe par défaut en JAVA via les API de la classe *Obervable* et de l'interface *Observer*.

Rechercher et étudier ces deux API et modifier le programme précédent de sorte à les exploiter.

→ Rendez-vous sur Célène pour un QCM de restitution de connaissances (5').