Programmation par Aspect

Version d'Eclipse à utiliser : `/home/cregut/tp_cregut/eclipse-jee-photon-4.8/eclipse`

1 Comprendre les principes de la programmation par aspect

AspectJ est un projet de la fondation Eclipse : `https://www.eclipse.org/aspectj/`

Exercice 1 : Comprendre les termes de la programmation par aspect
Pour comprendre la programmation par aspect, lire les trois premiers chapitres (I, II et III) de `https://skebir.developpez.com/tutoriels/java/aspectj/`. Donner la définition des termes suivants :

1. Aspect (Aspect)
2. Point de jonction (joinpoint)
3. Coupe (pointcut)
4. Greffon (advice)
5. Mécanisme d'introduction
6. Tissage (weaving)

Exercice 2 : Comprendre un exemple
Lire et comprendre l'exemple du cache ajouté sur les expressions arithmétiques : `https://skebir.developpez.com/tutoriels/java/aspectj/#LIV`.

2 La programmation par aspect avec AspectJ

Dans cette partie, nous allons travailler avec les classes des comptes bancaires : `CompteSimple`, `CompteCourant`... Même si le code source de ces classes est fourni, l'idée est de ne pas le modifier !

Exercice 3 : Tracer les appels des méthodes des comptes
Définir un aspect pour tracer tous les appels aux méthodes publiques des classes `CompteSimple` et `CompteCourant`. On suivra les étapes suivantes :

1. définir une coupe qui correspond aux méthodes publiquess de CompteSimple et CompteCourant, quelque soit leur nombre de paramètres.
2. définir un greffon (advice) pour cette coupe qui affiche d'abord simplement la signature de la méthode appelée sur la sortie standard. On utilisera : `thisJoinPoint.getSignature()`.
3. Vérifier le fonctionnement en exécutant la classe `ExempleComptes`.

4. Modifier l'affichage pour afficher le nom de la méthode et la valeur de ses paramètres séparés par des virgules. On utilisera `getArgs()` sur `thisJoinPoint`.

Exercice 4 : Signaler les débits élevés
Si une opération de débit dépasse un certain montant (par exemple 200 euros), l'opération devra être signalée. Ceci pourrait se traduire par un message électronique ou un SMS envoyé au titulaire du compte pour l'informer de l'opération et lui laisser la possibilité de la contester.

On pourrait aussi implanter un système de double authentification en envoyant un code par SMS au titulaire du compte pour qu'il confirme qu'il est à l'origine de l'opération.

Ici, nous simulerons l'envoi d'un SMS et un système de confirmation.

1. Définir un aspect DebitEleve qui affiche un message quand on débit d'un montant supérieur à une certaine somme est fait. La limite déclenchant l'affichage du message sera défini comme un attribut de classe de l'aspect que l'on pourra appeler LIMITE avec pour valeur 450. Pour simuler la confirmation, on fera un tirage aléatoire (java.util.Random.nextInt(10)). Le débit sera confirmé si l'entier est ≤ 2. Sinon, une exception `OperationNonConfirmeeException` sera levée.

Exercice 5 : Ajouter un observateur sur les comptes
Sans modifier le code des classes fournies (on pourrait n'avoir accès qu'à leur version compilée et pas à leur code source), on veut ajouter un mécanisme d'observateur sur les classes CompteSimple et CompteCourant. Nous allons nous appuyer sur la classe abstraite Observable et l'interface Observer de l'API Java 8. Il s'agit donc, via l'aspect exclusivement, de :

1. faire hériter `CompteSimple` de `Observable`.
2. ajouter sur la classe `CompteSimple` une méthode `avertir(double)` qui fait un appel à `setChanged()` et `notifyObservers(Object)`. Cette méthode prendra en paramètre le montant de l'opération, un **`double`**.
3. appeler la méthode `avertir(double)` quand le solde du compte change. On passera en paramètre de `notifyObservers` le montant ajouté (positif) ou retiré (négatif) au solde du compte.

1. La classe de test fournie CompteObservateurTest ne compile pas. Pourquoi est-ce normal ?
2. Définir un aspect qui rend la classe `CompteSimple` observable.
3. Tester l'aspect en utilisant la classe de test fournie.

3 Programmation par aspect et patron Visiteur

Exercice 6 : Opération sur une structure de données (expressions arithmétiques)
Intéressons nous aux expressions arithmétiques.

1. Définir un aspect sur les opérations arithmétique qui calcule ne nombre de traits utilisés par les différents opérateurs d'une expression. On compte un trait pour la soustraction (ou l'opposé), 2 traits pour l'addition et 3 traits pour la multiplication.
2. Comparer la programmation par aspect et le patron de conception visiteur.