Examen Design Pattern et Programmation Orientée Aspect

Durée : 3H30 | Documents autorisés

On souhaite créer une Framework qui permet de Générer des diagrammes UML d'un modèle ainsi que le code source de l'application avec différents langages (Java, C#, etc…).

Un Modèle UML se compose de plusieurs Diagrammes. Il existe différents types de diagrammes : Digramme de classe, Diagramme Use case, Diagramme de séquences, Diagramme d'Objets, etc. Dans ce problème, nous nous intéresserons uniquement au diagramme de classe.

Un diagramme de classes est défini par :

- Un ensemble d'entités qui peuvent être des classes, des interfaces, des énumérateurs ou des annotations ou des Records. Chaque entité est définie par son nom.
    - Une classe est définie par un groupe d'attributs, un groupe de méthodes (concrètes ou abstraites) et un groupe de constructeurs. Chaque attribut est défini par son nom, son type de données, sa visibilité (privé, publique, protégé), si l'attribut est statique ou non, si l'attribut est final ou non. Pour chaque classe, on doit pouvoir aussi préciser si elle est publique, abstraite, statique ou finale. Une méthode est définie par son nom, son type de retour et une liste de paramètres. Chaque paramètre est défini par son nom et son type. Pour chaque méthode, on doit préciser sa visibilité (publique, privée ou protégée), statique, final ou abstraite.
    - Une interface est définie par un groupe de méthodes abstraites.
    - Un énumérateur est défini par une liste de valeurs
    - Une annotation est définie par une liste de propriétés
    - Un Record est une classe qui un seul constructeur avec paramètres avec des getters.
- Un ensemble de relations qui trois types : Héritage, implémentation et associations.
    - Pour une association, on distingue deux types : les associations unidirectionnelles et les relations bidirectionnelles.
    - Chaque association est définie par l'entité source, qui doit être une classe, et l'entité destination (classe, interface, énumérateur ou annotation). Pour chacune des entités source et destination, on précise un tableau de deux cardinalités. Une cardinalité minimale qui peut être soit ZERO ou UN et la cardinalité maximale qui peut être soit UN ou MANY.
    - Une cardinalité est représentée par un énumérateur.

- Une relation de type Héritage est définie par une entité source et une entité destination.

Pour cette application on souhaite la doter des fonctionnalités suivantes :

- Définir une opération qui permet de générer le code source d'un diagramme de classes en choisissant une stratégie de génération pour chaque langage de programmation. Le modèle doit rester extensible pour donner la possibilité d'ajouter de nouvelle implémentation pour d'autres langages futurs. Dans cette application, il est demandé de définir une implémentation qui permet de générer le code java du diagramme de classe qui consiste à générer la structure des classes, interfaces, enumérateurs, annotations du diagramme de classe en prenant en considération les relations en entre ses différentes entités.
- Exporter un diagramme de classes en différents formats : Sérialisation Binaire, XML, JSON, SVG etc. en laissant le modèle ouvert aux extensions futurs.
- Définir un aspect de journalisation en utilisant une annotation @Log pour journaliser toutes actions effectuées sur le diagramme de classes
- Définir un aspect pour verrouiller des méthodes en utilisant l'annotation @Lock. Une méthode verrouillée ne pourra jamais s'exécuter.
- Donner la possibilité de créer des groupes de classes qui peuvent contenir d'autres groupes
- Adapter une ancienne implémentation qui permet de générer le code en l'intégrant comme stratégie de générer de code du Framework
- Quand l'état d'un attribut d'une classe change, d'autres observateurs (à définir) de l'application sont notifiés.
- Appliquer d'autres Pattern que vous jugez important pour votre modèle

Travail demandé

Rendre un rapport et le code source du Framework, répondant aux questions suivantes :

1. Établir un diagramme de classes de ce Framework en utilisant les design patterns les plus appropriés. En plus des designs patterns exigés par l'énoncé du problème, vous pouvez proposer d'autres design patterns que vous voyez pertinents pour améliorer la qualité du Framework
2. Faire une implémentation JAVA de ce Framework
3. Créer une application de Test du Framework en choisissant un use case
4. Ajouter des améliorations à votre implémentation selon vos propositions