Stéphane TOYO DEMANOU 04/04/2021

Amir CHERKAOUI

LOG2410 TP4 : Conception à base de patrons

2. Identifier des patrons de conception utiles pour votre cas d'étude

- Patron Fabrique

Parmi les fonctionnalités demandées, notre application doit être capable de créer des profils pour les utilisateurs. De plus, pour cela, nous allons utiliser et gérer des ressources systèmes importantes telles des structures de fichiers et des ressources réseau (serveur infonuagique). En prenant tous ces besoins en compte, le patron Fabrique nous semble le plus approprié à la tâche. En effet, il nous permet de pouvoir créer des objets sans spécifier à l'avance leurs types dans la classe de base et ainsi (grâce à l'héritage) nous offre de la flexibilité dans la suite de l'implémentation. Il nous permet également de réutiliser des objets déjà crée ce qui nous économisera des ressources systèmes car il nous faudra une méthode qui soit à la fois capable de créer des objets mais aussi de les retenir en mémoire pour les utiliser plus tard. Toutes ces caractéristiques correspondent fortement au patron Fabrique.

- Patron Template Method

PolyPiano a un processus d'authentification, chaque utilisateur a besoin de se connecter. De plus, il y a deux types de comptes, les comptes élèves, et les comptes professeurs. Chacun ayant des accès différents aux ressources de l'application. En effet, le professeur a par exemple accès a la base de données de tous ses étudiants, ainsi qu'a la progression de ceux-ci, lui permettant de laisser des commentaires s'il en ressent le besoin. Le patron Template Method va permettre dans notre cas de s'assurer que l'utilisateur est bien un professeur avant de lui donner accès a ses élèves et à leurs performances.

3. Discussion des avantages et des inconvénients

- Patron Fabrique :

Avantages :

- La Fabrique fournit un point d'extension à ses sous-classes pour qu'elles puissent créer des objets (Produits Concrets) spécialisés.
- La Fabrique offre plus de flexibilité qu'une simple création d'objet avec *new*.
- Réduit le coulage en désolidarisant le Créateur des objets produits. Les sous-classes spécifiques d'objets ne sont plus liées directement au code Client ce qui augmente la portabilité du code
- On respecte le principe de responsabilité unique.
- Respect du principe ouvert/fermé, on peut ajouter de nouveaux objets produits a créer sans endommager les anciens.

Inconvénients :

- Le code peu vite devenir très complexe
- Obligation d'introduire une nouvelle sous-classe pour chaque nouveau produit concret que l'on veut ajouter.
- L'héritage multiple est assez difficile à gérer dans des codes très complexes.

- Patron Template Method

Avantages :

- Ce patron favorise la réutilisation de code, on ne trouvera donc pas de duplication car dans notre cas toutes les classes vont provenir de la même classe abstraite ou de base.
- Il permet d'imposer des règles de surcharges, c'est-à-dire, laisser les clients surcharger quelques parties de l'algorithme.

Inconvénients :

- Les clients sont limites dans la modification du code a cause du squelette de l'algorithme, qu'il ne faut donc pas modifier sous peine de casser le code.
- Plus il y a d'étapes, plus le patron est complexe, cela devient difficile à maintenir.
- Le débogage peut vite devenir compliqué vu la similitude entre les classes. Il faut donc être rigoureux et documenter le code le plus possible pour éviter de faire des erreurs.

4. Les diagrammes de classes détaillés

- Patron Factory Method :

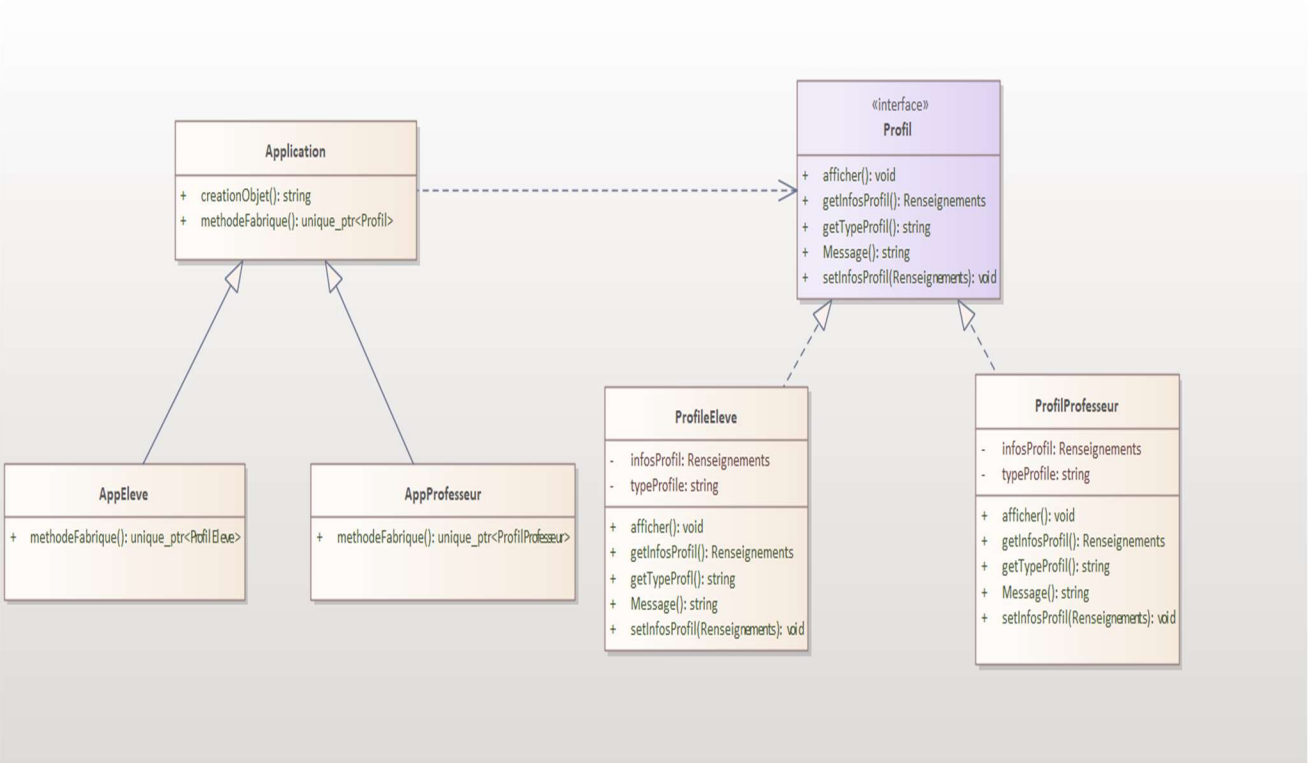

Figure 1: Diagramme de classes du patron Fabrique

- Patron Template Method :

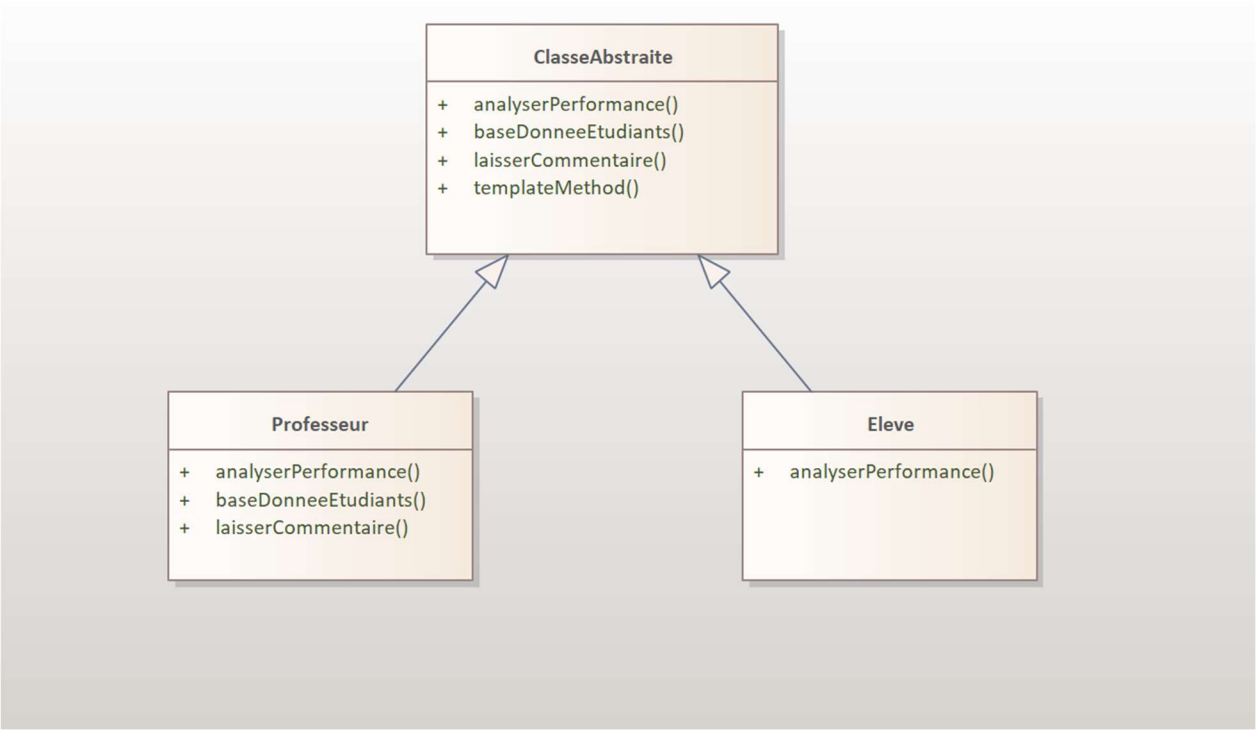

Figure 2: Diagramme de classes du patron Template Method