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 Singleton

L’utilité principale de PolyPiano est l’analyse de la performance musicale des étudiants. Pour ce faire, parmi tout ce que propose PolyPiano, il y a un programme chargé d’exécuter cette tâche. Ce programme est unique, qui que soit l’étudiant, il va opérer de la même façon. Le fait qu’il soit en un seul exemplaire amène l’idée d’utiliser le patron Singleton. Ce patron garantit une seule instance d’une classe, et fournit un point d’accès global a celle-ci, ce qui convient parfaitement à notre situation. On peut alors créer une classe qui contiendra la méthode analysant la performance de l’étudiant.

## 3. Discussion des avantages et des inconvénients

* Patron Singleton :

Avantages :

* La classe créée est unique, et on a un point d’accès global à celle-ci, toute instance d’étudiant va faire appel au même objet pour analyser la performance.
* La classe est initialisée une seule fois tout le long, ce qui est très efficace et excellent au niveau de la performance de l’entièreté de PolyPiano.
* Le fait qu’il n’y ait qu’une instance faciliterait énormément le débogage lors de l’implémentation de PolyPiano.

Inconvénients :

* Le patron Singleton viole le principe de responsabilité unique, stipulant que chaque classe doit avoir une responsabilité distincte.
* Dans notre cas, les tests unitaires pourraient s’avérer difficiles, il est difficile d’isoler une classe dépendant d’une classe Singleton, donc certains tests unitaires pourraient s’avérer plus compliques que d’autres.
* 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 à 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 :

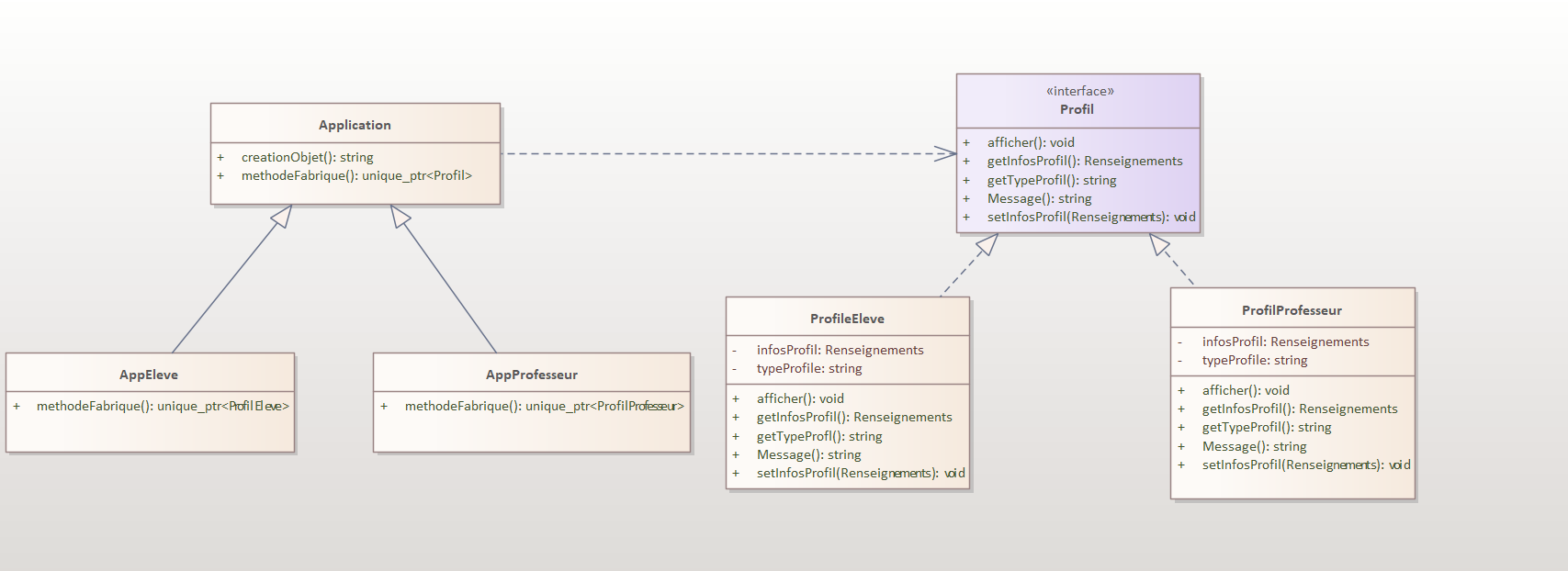


Figure 1: Diagramme de classes du patron Fabrique

* Patron Template Method :

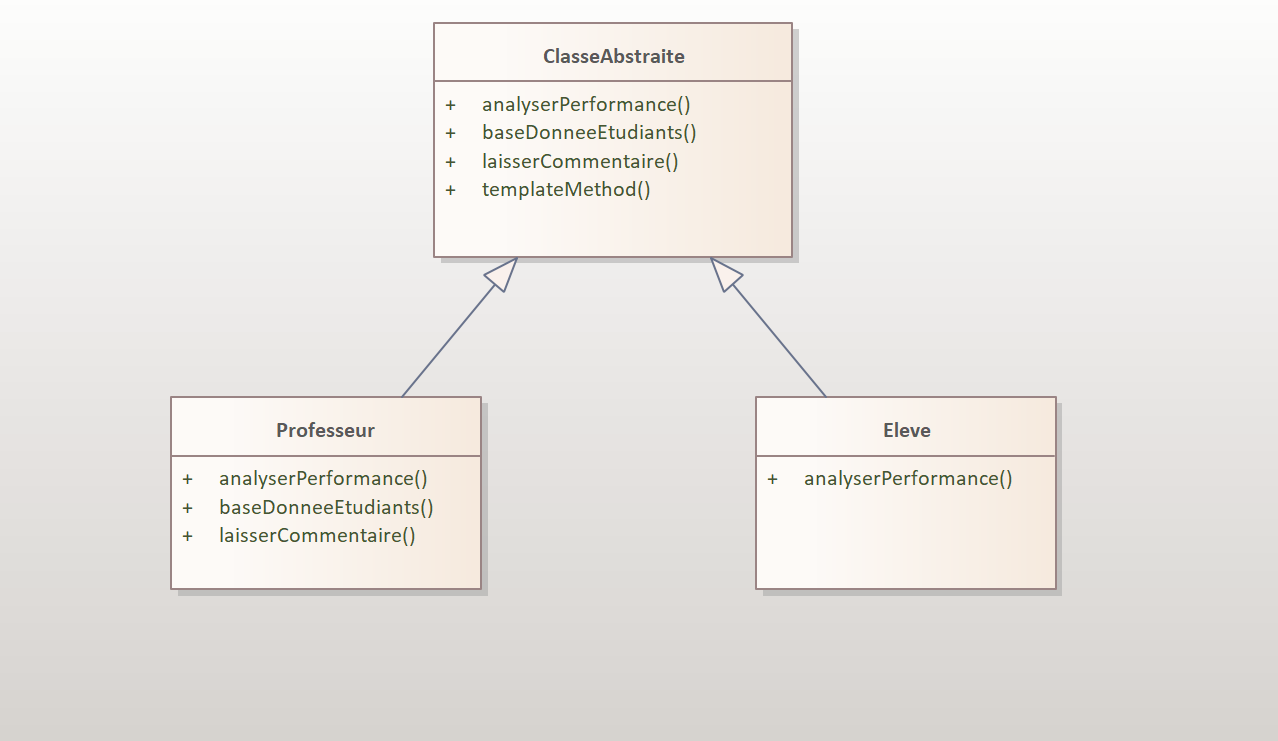


Figure 2: Diagramme de classes du patron Template Method