Paul Clas 1846912 & Mazigh Ouanes 1921035 LOG2410 - Hiver 2019

**Conception à base de patrons 1**

1. **Objectif**
2. **Patrons Composite**
    1. **Identifez les points suivants**
        a. **L'intention du patron Composite**

L'intention du patron Composite est de créer une structure de classes en arbre avec des structures composites. Cela nous permet de traiter récursivement et uniformément les objets.

**b. La Structure des classes réelles qui participent au patron ainsi que leurs rôles. Faite un diagramme de classes avec Enterprise Architect pour l'instance du patron composite. Ajouter des notes en UML pour indiquer les rôles, et exportez le tout en pdf).**

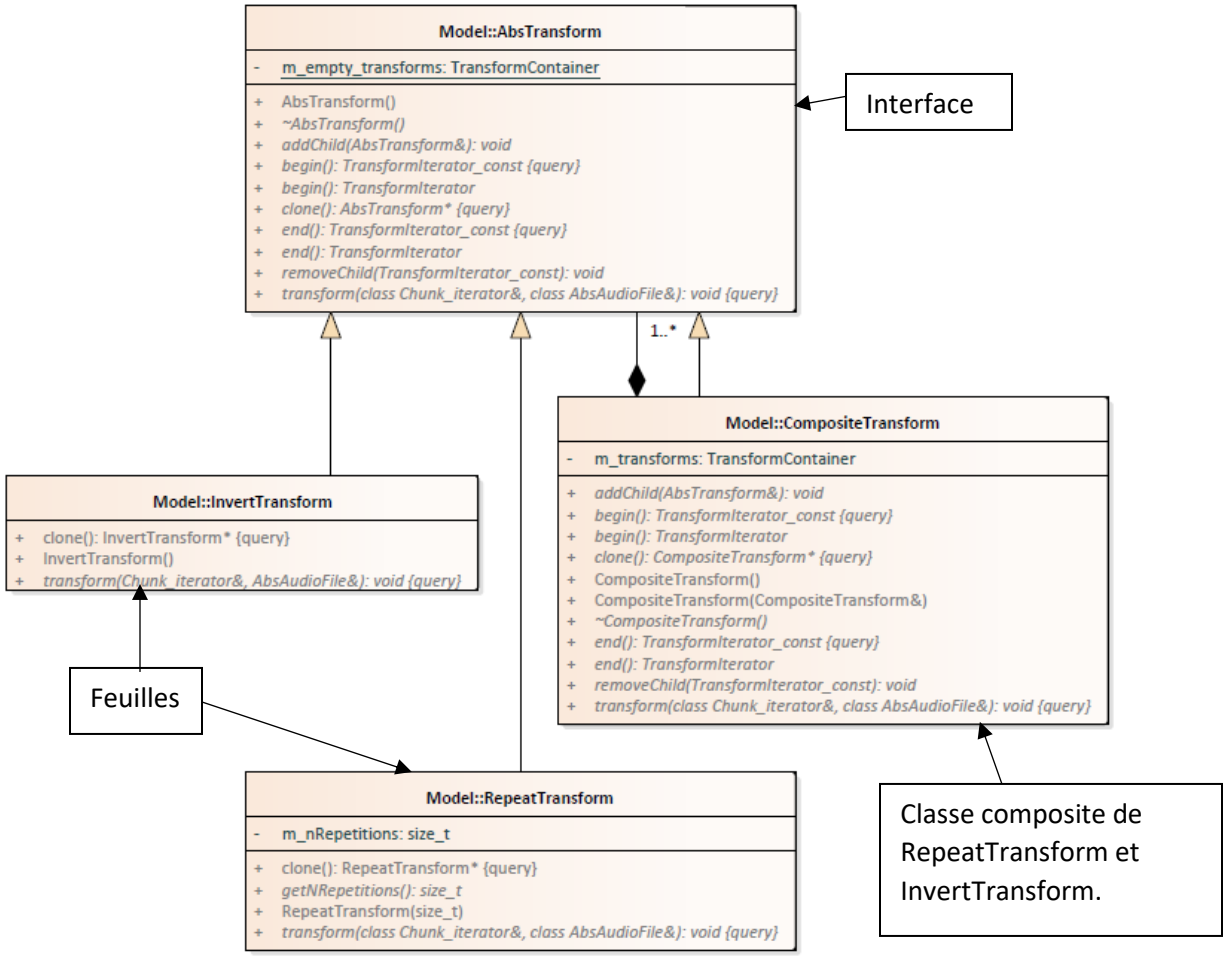


Diagramme de classe du patron Composite (version .pdf en Annexe 1)

2. **Identifiez toutes les abstractions présentent dans la conception du TP4, et pour chacune, identifiez les responsabilités spécifiques qui lui ont été assignées.**

AbsTransform : Classe interface des trois transformations RepeatTrasnform, InvertTrasnform, CompositeTransform.

AbsAudioFile : Cette classe fait l'interface de AudioFile et MemAudioFile.

3. **Dans l'implémentation actuelle du système PolyVersion, quel objet ou classe est responsable de la création de l'arbre des composantes.**

La classe CompositeTransform est responsable de la création de l'arbre des composantes avec addChild() et removeChild().

3. **Patron Proxy**
   1. **Identifiez les points suivants :**
      a. **L'intention du patron Proxy**

L'intention du patron Proxy permet de substituer une classe/objet à une autre tout en permettant d'utiliser les méthodes de la classe/objet originale sur la classe/objet proxy. La classse MemAudioFile est un proxy pour la classe AudioFile. Un proxy ajoute une indirection à l'utilisation de la classe à substituer (Gamma,1994).

**b. La structure des classes réelles qui participent au patron ainsi que leurs rôles. Faite un diagramme de classes avec Enterprise Architect pour l'instance du patron proxy. Ajouter des notes en UML pour indiquer les rôles, et exportez le tout en pdf.**

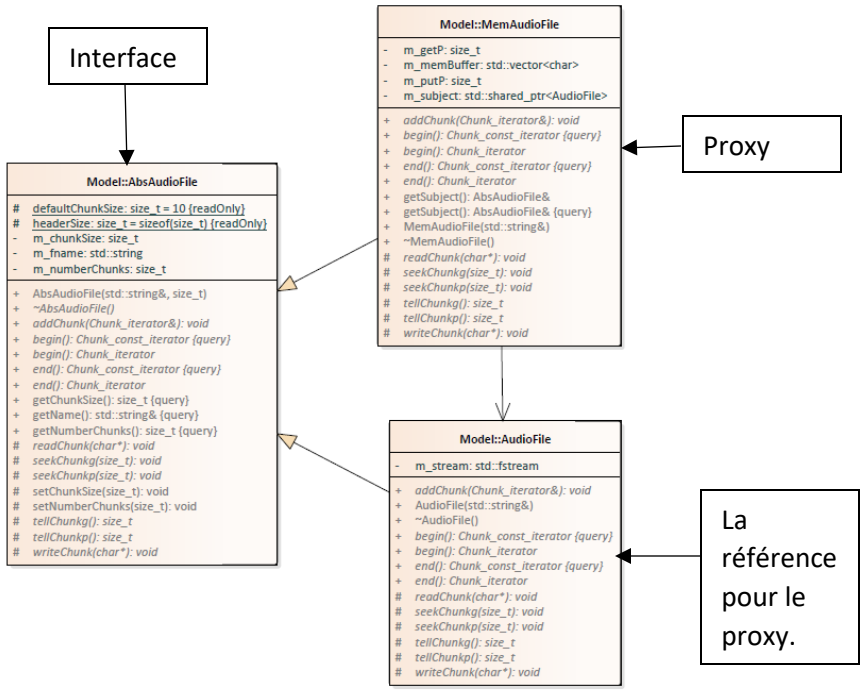


Diagramme de classe du patron proxy (version .pdf en Annexe 2)

4. **Conteneurs et Patron Iterator**
    1. **Identifiez les points suivants :**
        a. **L'intention du patron Iterator**

Ce patron permet d'accéder séquentiellement aux éléments d'un ensemble sans connaitre les détails techniques du fonctionnement de l'ensemble (Gamma, 1994). L'Iterator permet de parcourir un conteneur en accédant séquentiellement à tous ses éléments pour faire une transformation, tout en isolant la structure interne du conteneur. Dans le cadre du TP4, nous utilisons TransformIterator pour pouvoir utiliser les méthodes d'un objet/classes agrégats (comme un conteneur vector) sans exposer sa structure interne.

**b. La classe de conteneur de la STL utilisée pour stocker les enfants dans la classe Composite et les classes des Iterators utilisés dans la conception qui vous a été fournie.**

La classe de conteneur de la STL utilisée pour stocker les enfants dans la classe Composite est celle de Vector. De plus, les classes des Iterators TransformBaseIterator et TrasnformBaseIterator_const (qui représentent respectivement les classe TransformIterator et TransformIterator_const) nous ont été fournies.

2. **Expliquez le rôle de l'attribut statique m_empty_transforms défini dans la classe AbsTransform. Expliquez pourquoi, selon vous, cet attribut est déclaré comme un attribut statique et privé.**

Le rôle de l'attribut statique m_empty_transforms défini dans la classe AbsTransform est de retourner un élément valide(chunk) vide lorsque nous utilisons la classe abstraite AbsTransform. Cet attribut est utilisé pour faire retourner un Iterator en échouant silencieusement.

Cet attribut est déclaré privé pour respecter encapsulation et parce que c'est seulement AbsTransform qui va l'utiliser bien qu'on le déclare static.

Cet attribut est statique car il est indépendant lors de chaque différente utilisation.

3. **Quelles seraient les conséquences sur l'ensemble du code si vous décidiez de changer la classe de conteneur utilisée pour stocker les enfants dans la classe Composite? On vous demande de faire ce changement et d'indiquer toutes les modifications qui doivent être faites à l'ensemble du code suite au changement. Reliez la liste des changements à effectuer à la notion d'encapsulation mise de l'avant par la programmation orientée-objet. À votre avis, la conception proposée dans le TP4 respecte-t-elle le principe d'encapsulation ?**

Les conséquences seraient une perte de l'uniformité des manipulations de la structure de l'arbre.

Il va falloir modifier toutes les méthodes spécifiques à chacun des conteneurs pour les manipuler. Il faut changer les classes de conteneur vector en classe list. Il faudra aussi changer l'utilisation de notre patron iterator. La modification du code à faire serait :

using TransformContainer = std::list<TransformPtr>; .

La conception proposée dans le TP4 respecte le principe d'encapsulation car tous les éléments de notre conteneur restent isolés de l'utilisateur. La différence de conteneur ne changera pas la visibilité du contenu.

4. **Les classes dérivées TransformIterator et TransformIterator_const surchargent les opérateur « * » et « -> ». Cette décision de conception a des avantages et des inconvénients. Identifiez un avantage et un inconvénient de cette décision.**

**Avantages :** Accès facile aux méthodes des transformations.

**Inconvénients :** Difficile de déterminer sur quels objets ils pointent.

## 5. Bibliographie

Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides, Design Patterns : Elements of Reusable Object-Oriented Software, Addison-Wesley, 1994, 395 p. (ISBN 0-201-63361-2, lire en ligne [archive]), p. 233-245

## 6. Annexe

Dans l'ordre les annexes 1 et 2 pour le diagramme de classe du patron composite et proxy.