POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

LOG2410 : Conception logicielle

TP 04 : **Conception à base de patrons I**

Travail fait par :

Mohamed Khairallah Gharbi (1837067)

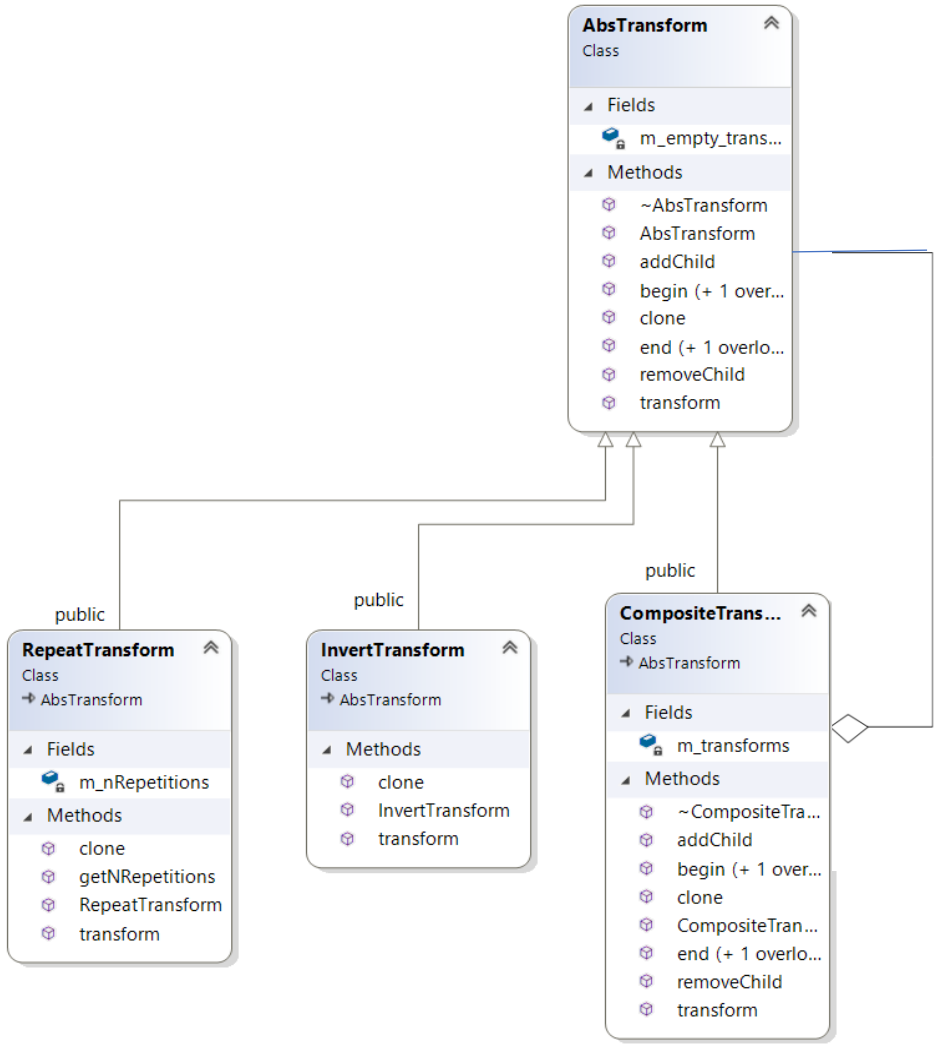
Mohamed Sameh Sellami (1917426)

Farid El Fakhry (1875036)

2— Patron Composite

2-1. a. L'intention du Patron Composite : En se référant au cours on conclut qu'un patron composite permet au client de manipuler uniformément un groupe d'objets de la même façon que s'il s'agit d'un seul objet. De ce fait il permet, son utilisation garantira un traitement composé récursivement et de façon uniforme. Son applicabilité se résume généralement sur des objets qui doivent être composés récursivement, ou dont les structures peuvent être traitées uniformément. Ce qui nous amène à notre TP. En effet, son utilisation a permis de réaliser une série de traitement des objets, et ce d'une façon « répétitive » dont le résultat est un stockage dans un objet de type composite. De ce fait on a réussi à traiter toutes les transformations uniformément grâce à la disposition sous forme d'arbres.

2-1. b. Les structures de classes réelles qui participent au patron selon l'exemple donné :



AbstractComponent est : **AbsTransform :** C'est une Superclasse (interface) qui contient l'ensemble des transformations.

Composite est : **CompositeTransform :** c'est une classe de type composite, car elle contient des RepeatTransform, qui représente une représentation récursive et répétitive d'un <<Chunk>> dans un fichier de sortie, ainsi que des InvertTransform, qui fait aussi des modifications sous forme d'inversion du <<Chunk>> également dans le fichier de sortie.

Les feuilles sont : **InvertTransform et RepeatTransform.**

2-2. Par définition du principe d'abstraction, dans la programmation orientée objet, consiste à ne fournir que les informations essentielles sur les données au monde extérieur et en masquant les détails de l'arrière-plan. De ce fait nous concluons que ces classes sont des classes abstraites :

* AbsAudioFile (contenant AudiFile et MemAudioFile).

*AbsTrabsform (contenant les transformations sur le fichier de sortie).

2-3. ***Question : Dans l'implémentation actuelle du système PolyVersion, quels objet ou classe est responsable de la création de l'arbre des composantes***

Réponse :

En parcourant les méthodes des classes on s'est attardé sur la classe CompositeTransform, qui nous semble la classe responsable de la création de l'arbre des composantes avec ses deux méthodes pertinentes qui sont :

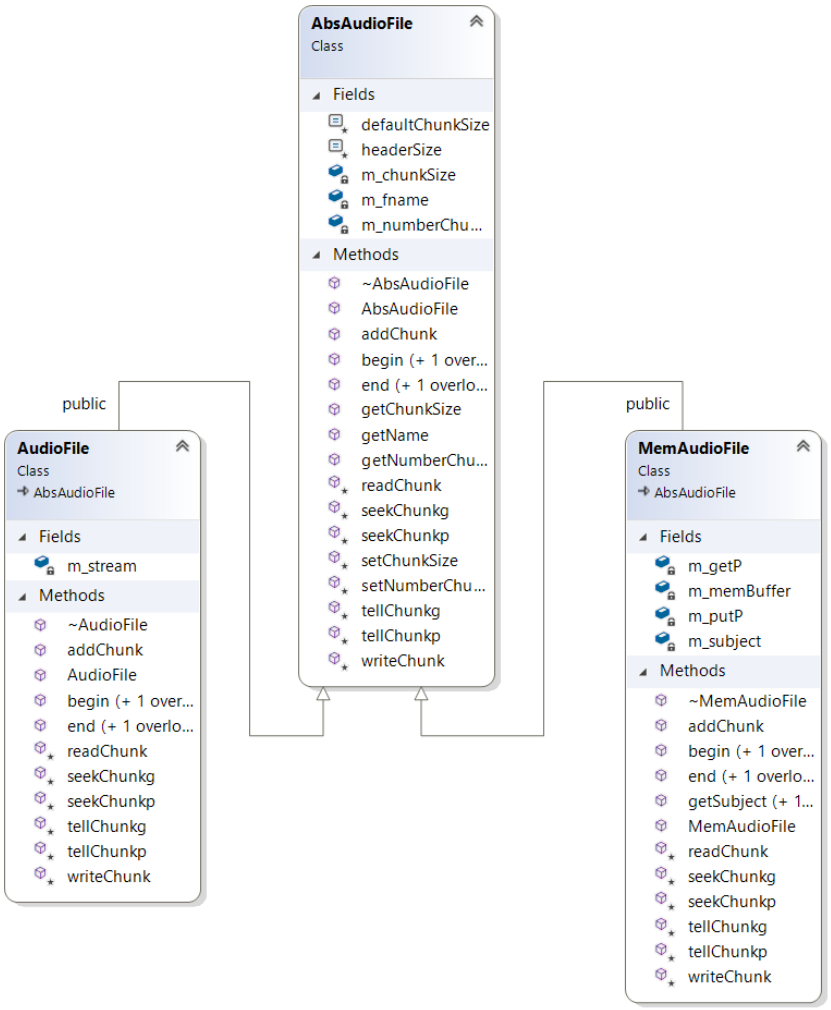
1 <<AddChild>>

2<<RemoveChild>>

3— Patron Proxy

3-1. a. L'intention du Patron Proxy : En se basant sur la définition du patron proxy (ou de conception) qui permet de fournir un substitut ou en autres ternes un espace réservé pour un autre objet avec l'intention de contrôler ce dernier ou le modifier, nous avons pu identifier la classe MemAudioFile qui représente un proxy de AudioFile.

3-2. b. Le diagramme de classe est :



AbsAudioFile : L'Interface

AudioFile : L'objet que Proxy référe.

MemAudioFile : Le remplaçant.

4— Conteneurs et Patron Iterator

4-2. a. Patron Iterator : permet l'accès à tous les membres d'une classe agrégée d'un conteneur dans notre cas sans avoir besoin de manipuler directement (sa structure interne) cette classe ou de connaître sa structure.

4-2. b. La classe iterator de la STL est utilisée.

4-3. static m_empty_tranform permet d'accéder au pointeur de début et de fin avec succès. Il permet d'indiquer la présence du chunk qui a subi une transformation.

Le mot clé static permet d'initialiser l'attribut une seule fois. Il est privé d'empêcher la modification.

4-4. Il faudrait faire des modifications à la classe TransformContainer seulement. Le principe d'encapsulation est respecté : on peut accéder à des données tout en laissant leur structure interne cache.

4-5.

***** : sert à déréférencer l'objet iterer.

⟶ : sert à retourner un pointeur vers l'élément.

Avantage : on peut choisir de retourner une référence ou bien la valeur même de l'objet contenu

Inconvénient : Peut créer des confusions (l'itérateur ressemble à un pointeur, mais ne l'est pas en effet).