# Patron Visiteur

## Question 1 : Intention et avantages

L’intention du patron visiteur permet d’ajouter, de supprimer et ainsi de gérer aisément de nouvelles opérations sur une classe sans la modifier. Les opérations deviennent indépendantes de cette classe.

## Question 2 : Diagrammes de classes

Les diagrammes se décomposent en deux parties. D’une première part, on retrouve les visiteurs, et de l’autre on retrouve les éléments sur lesquels les visiteurs agissent. Les visiteurs possèdent une classe abstraite commune qui est *AbsFileVisitor*. S’en suit des deux sous-classes *FileStringReplace* et *FileChecksumCalculator*. Ces deux classes permettent la visitent des éléments, qui eux accepteront la visite. Les éléments possèdent une classe abstraite commune qui est *AbsAudioFile*. S’en suit des deux sous-classes *AudioFile* et *MemAudioFile*.

Voir les diagrammes *DiagrammeDeClasse\_FileChecksumCalculator.pdf* et *DiagrammeDeClasse\_FileStringFindReplace.pdf* .

## Question 3 : Ajout d’une nouvelle sous-classe

Il sera très simple d’ajouter de nouvelles sous-classes dérivées de *AbsAudioFile*, sans pour autant devoir effectuer de nombreux changements aux classes qui y sont reliées. En effet, c’est l’avantage que permet le patron Visiteur. Puisque que chaque sous-classe de *AbsAudioFile* représente un élément qui peut accepter des visiteurs, l’ajout d’une sous-classe ne demandera qu’à ajouter une fonction de visite dans chacun des visiteurs.

## Question 4 : Transformations implémentées comme visiteur

Oui il serait possible d’effectuer les transformations selon un patron visiteur. Les transformations pourraient fonctionner de la même façon que les sous-classes *FileStringReplace* et *FileChecksumCalculator*. Puisque les transformations agissent sur les fichiers audios, on conserve les éléments, et on met visiteur les transformations, et on obtient un diagramme semblable à celui de la question 2.

# Patron Commande

1. L’intention du patron visiteur permet d’ajouter, de supprimer et ainsi de gérer aisément de nouvelles opérations sur une classe sans la modifier. Les opérations deviennent indépendantes de cette classe.
2. Voir