# Rapport TP5 – LOG1410

## 2. Patron Visiteur

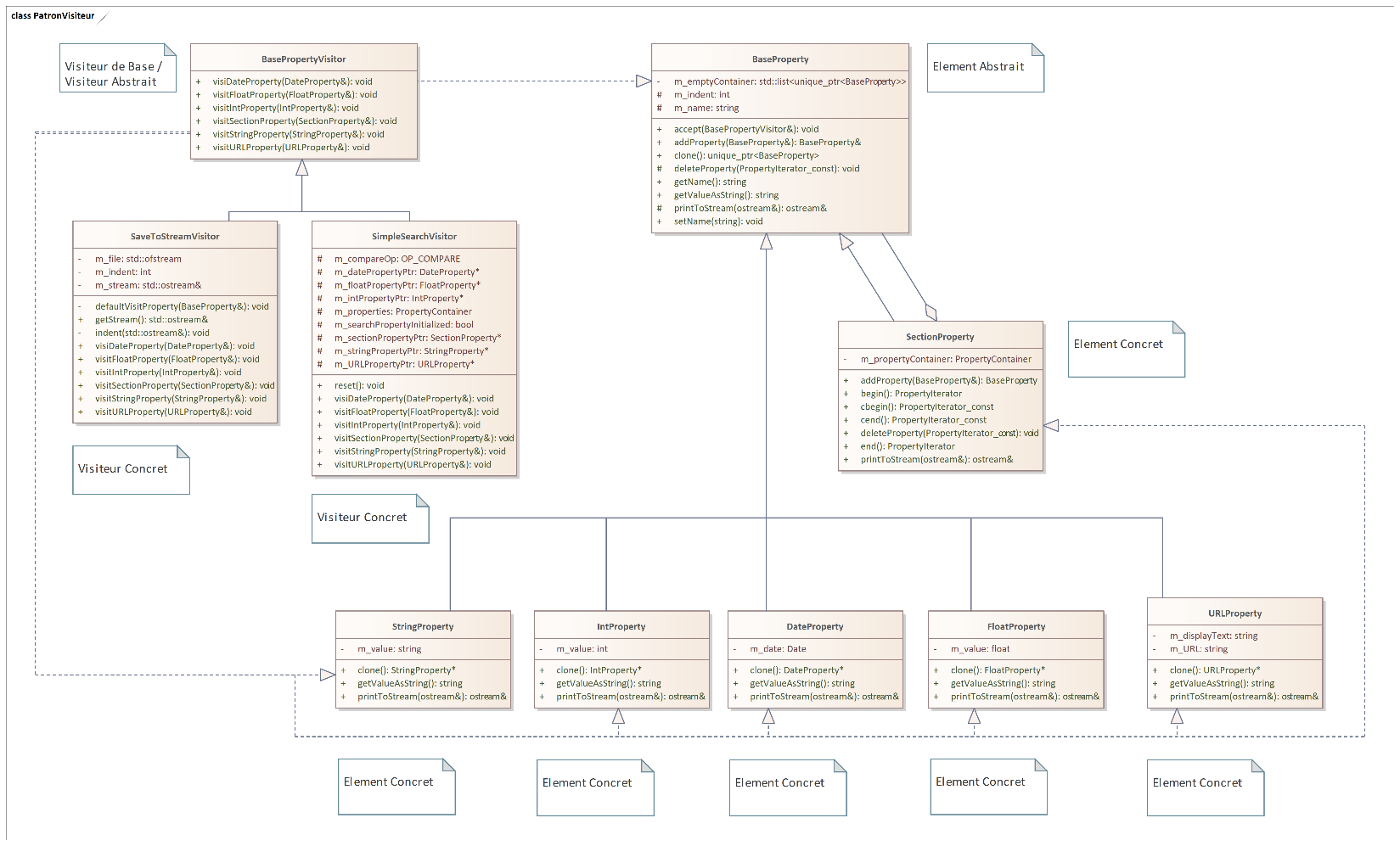
### Question 2.1

Le patron visiteur permet de séparer un ou des algorithmes du ou des objets sur lequel celui-ci opère ou a un effet. Dans le cas de PolyMordus, nous avons deux visiteurs :

* Le *SaveToStreamVisitor* qui prend en charge l’opération d’affichage sur un *ostream* d’une propriété associée à un enregistrement
* Le *SimpleSearchVisitor* qui a pour rôle comparer différentes propriétés selon un critère donné, en utilisant les différents opérateurs de comparaison du C++ (less, equal\_to, not\_equal\_to…)

### Question 2.2

Voici le diagramme UML représentant le patron visiteur dans l’application PolyMordus :



### Question 2.3

Mettre la méthode « *printToStream()*» au sein de *BaseProperty* fait qu’il y a un fort couplage entre l’affichage textuel et la structure même des propriétés ce qui peut freiner l’évolution du code. Utiliser un visiteur simplifie la classe *BaseProperty* qui ne dispose maintenant que d’une méthode « *accept()* » et n’a plus à implémenter sa propre logique d’affichage. Cela respecte les principe SOLID et facilite l’ajout de fonctionnalités à l’application

### Question 2.4

Si on souhaite ajouter une nouvelle classe *XProperty* dérivée de *BaseProperty*, il faudrait rajouter ces éléments :

* Ajouter la méthodes *visitXProperty* dans *BasePropertyVisitor,* et l’implémenter dans *SimpleSearchVisitor* et *SaveToStreamVisitor*
* Implémenter la méthode *accept* dans *XProperty*

## Patron Stratégie

### Question 3.1

Le patron stratégie permet de définir plusieurs algorithmes différents en les regroupant dans une « famille ». Ses algorithmes sont ensuite utilisés dans des classes séparées, et sont interchangeables. Dans notre cas, le patron stratégie est utilisé dans *SimpleSearchVisitor*, et les différents algorithmes possibles sont les six opérateurs de comparaisons du langage C++. En effet, sela permet de choisir de faire une recherche pour des éléments, différents, égaux, inférieurs et supérieurs à un certain critère spécifié dans la classe.

### Question 3.2

Voici le diagramme UML représentant le patron stratégie dans l’application PolyMordus :

