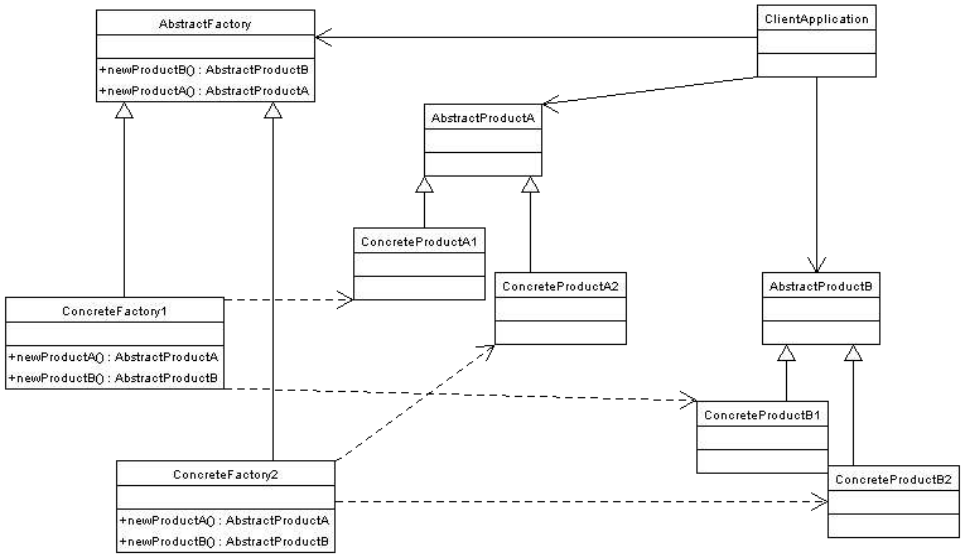
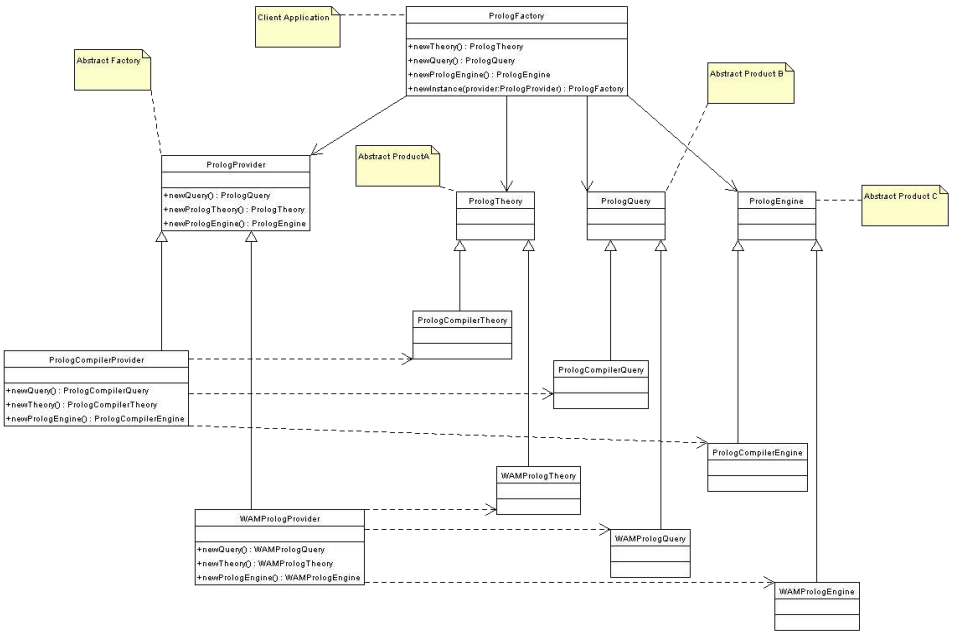
Étude de l'interopérabilité de deux langages de programmation basée sur la machine virtuelle de Java.

**J2PI :**

**Architecture** : Abstract Factory Pattern :




L'interface de Java vers Prolog est une instanciation du pattern Abstract Factory :




Une classe PrologFactory est chargée de fournir au programme Java les différentes classes et interfaces représentant le moteur Prolog en Java. Ces classes sont :

- ***PrologEngine*** : une abstraction d'un moteur Prolog générique. L'idée est que tous les programmes Java voient le même moteur Prolog quelle que soit l'implémentation sous-jacente;
- ***Theory*** : une abstraction d'une théorie Prolog qui sera utilisée par le PrologEngine;
- ***Query*** : une abstraction d'une requête Prolog qui sera évaluée par PrologEngine;

Outre ces classes, l'interface utilise aussi les classes et interfaces utilitaires suivantes :

- ***Result*** et ***ResultSet*** : représentant le résultat de l'évaluation d'une requête par le PrologEngine;
- ***Variable*** : représentant une variable Prolog;

**Ce qu'il reste à faire** pour l'interface Java vers Prolog :

1. Ajouter des méthodes permettant l'affichage des résultats au sein de Result et ResultSet;
2. Remplacer la génération des exception NoResultException et NoMoreResultException par la génération d'un Result ou d'un ResultSet vide et ajouter des champs et des méthodes pour pouvoir vérifier si un Result(Set) contient ou non un Résultat;
3. Gérer les requêtes multiples via l'introduction de findNext(Query q) dans PrologEngine. Dans l'implémentation avec le compilateur Prolog de Engel, la gestion des requêtes multiples pourra être réalisée via l'emploi d'une Hashtable et d'une classe XxxQuerySolver reprenant l'implémentation du PrologEngine actuel. La Hashtable contiendra alors le lien entre une Requête, utilisée comme clé, et le PrologCompilerQuerySolver correspondant + gestion des exceptions et overrider equals(Object o) et hashcode();
4. Se pencher sur l'utilité de fonctions avancées telles que assert(Term) ou revoke(Term) sachant que :
    1. Elles ne sont pas nécessaire et serait juste la par mesure de convénience pour le programmeur;
    2. Elles ne sont pas implémentées par toutes les interfaces avec un moteur Prolog;
5. Implémenter le Provider pour tuProlog;
6. Réaliser une démonstration (Tours de Hanoï ou 8 Reines, par exemple);

**P2JI :**

**Architecture** : précompilateur ou un truc équivalent et Abstract Factory pattern pour les méthodes factory permettant le passage d'un format d'extension propre à l'interface au format propre à l'implémentation de Prolog sous-jacente.

**Idée** : toute la mécanique des extensions sera cachée au programmeur Prolog. Pour lui, tout ce qu'il devra utiliser est le IDL permettant d'indiquer qu'un appel à Java se produit. Toute la partie gestion des extensions et précompilation sera du ressort de l'implémentation de l'interface.
Toutefois, le schéma suivit sera le même que pour l'interface J2PI : utilisation d'un **pattern Abstract Factory**. Une classe ou interface représentant une ***extension abstraite*** passée en argument à une ***méthode factory*** qui fournira l'extension pour le moteur Prolog et se chargera de la traduction du IDL en instructions pour ce moteur. Ici aussi l'utilisation d'un ***provider*** permettra la construction d'un mécanisme générique.

Il faut donc :

1. Définir le format d'extension interne à l'interface;
2. Définir les directives ou le IDL permettant de déclarer l'utilisation d'une extension côté Prolog;
3. Définir les classes de l'interface;
4. Implémenter pour Engel et tuProlog;
5. Démonstration : côté Engel, la démonstration pourrait consister en l'ajout d'extension arithmétiques ou d'entrées/sorties au compilateur. Pour tuProlog... je ne sais pas encore, on verra bien.

**Rédaction :**

**Plan :**

I. Intro : en gros elle est terminée, il manque juste une petite discussion sur la granularité des requêtes au moteur Prolog;
II. Spécification de l'interface : ici plusieurs parties :
    1. Une petite introduction pour présenter la spécification et expliquer qu'on va avoir deux interfaces
    2. J2PI : interface Java vers Prolog, là tout est quasiment terminé, il faut juste remettre à jour les spécification des méthodes et ajouter une section sur l'architecture et l'intuition qui lui est liée (moteur Prolog générique, a priori on ne parle pas du pattern ici);
    3. P2JI : ici tout reste à faire, suivre un plan comparable a J2PI
III. Implémentation de l'interface :
    1. J2PI : décrire le pattern et son implémentation, dire comment on l'instancie, parler de wrapper/factory et justifier et donner les guides d'implémentation pour le programmeur;
IV. Exemples
    1. Exemples pour J2PI;
    2. Exemples pour P2JI;

V. Annexes
    1. Implémentation avec Oolong Prolog : problèmes et solutions;
    2. Implémentation avec tuProlog : problèmes et solutions;