TP1+2: UML et Génération de code

Le but de ce TP est d'aborder le principe de la génération de code à partir de diagrammes de classe UML. Les objectifs sont les suivants :

- comprendre le lien entre les diagrammes de classe/diagrammes d'objets UML et les langages orientés objets comme Java
- manipuler concrètement ces correspondances en utilisant des outils de modélisation pour la transformation UML (genmymodel et MagicDraw) et l'environnement Eclipse (pour la manipulation du code Java)
- revisiter les exemples vus en TD
- comprendre le lien entre diagrammes de classe/diagrammes d'objets UML et JPA/les bases de données relationnelles comme SQL
- comprendre les stratégies de génération de code ainsi que leurs limites ou leurs différences

1 Un diagramme de classe UML simple

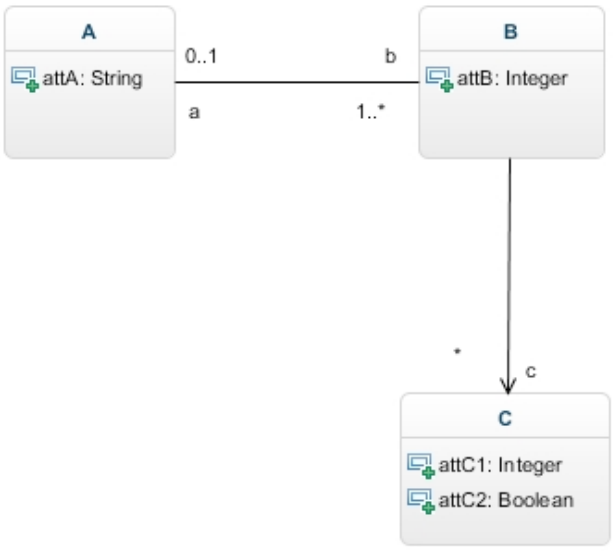


Figure 1 : diagramme de classe

Question #1: Elaborer 3 diagrammes d'objet conformes à ce diagramme de classe

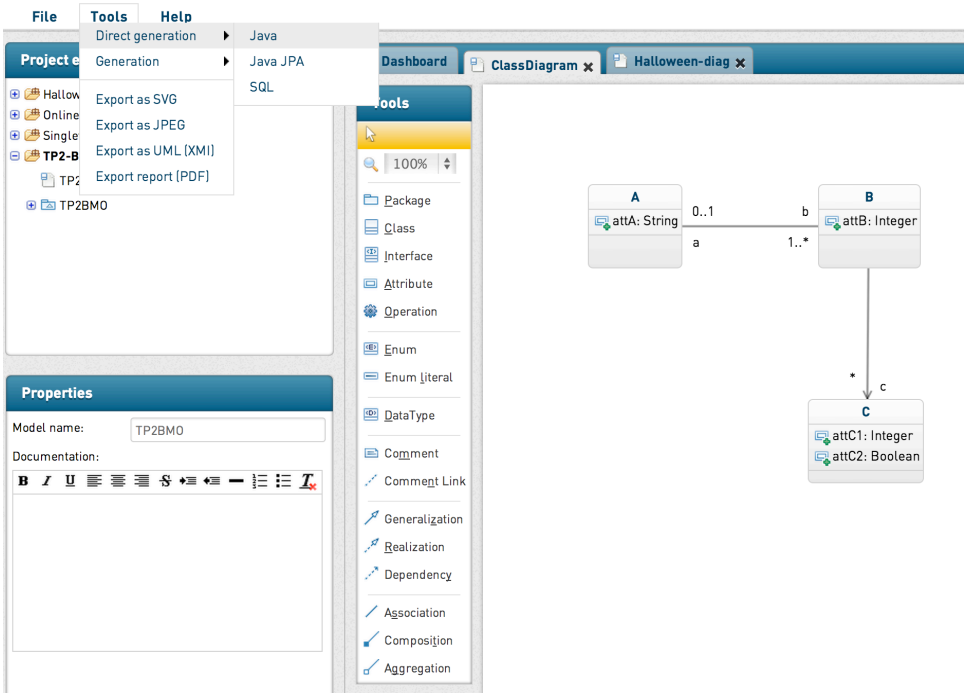
Nul besoin d'éditeur à ce stade : élaborer ces diagrammes sur une feuille.

Question #2 : Donner le code Java correspondant à ces 3 diagrammes d'objet

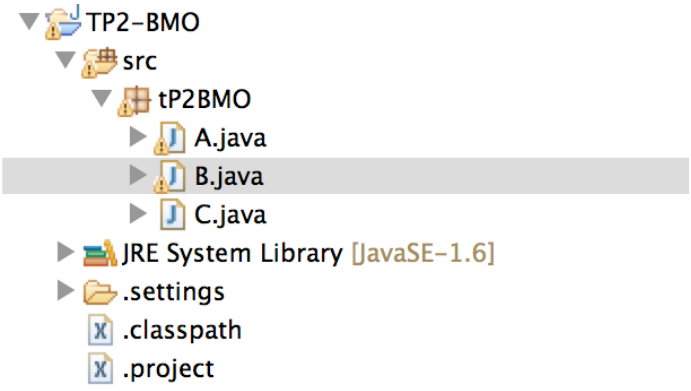
Cette fois ci, nous avons besoin d'un éditeur de modélisation.
Dans un premier temps nous allons utiliser genmymodel : http://www.genmymodel.com un outil Web, collaboratif développé en France.
L'inscription est gratuite.
Récemment une limite d'utilisation (e.g., pas plus de 20 éléments par projet) a été ajoutée, mais cela ne devrait pas être un frein majeur au déroulement du TP[1].

Inscrivez-vous en ligne puis créez un nouveau projet

Dans ce projet, créer un nouveau diagramme de classe et éditer ce diagramme pour obtenir celui de la Figure 1. Ensuite, générer le code Java correspondant.
Vous devez obtenir une archive « zip » de ce code.



Nous allons maintenant étudier et exploiter le code Java généré à l'aide d'Eclipse.
Créer un nouveau projet Java (File -> new -> Java project…)
Importer (via File -> Import -> File system) dans votre projet le contenu de votre archive « zip ».



[1] Il sera certainement nécessaire de supprimer des modèles au fur et à mesure ou de créer plusieurs projets

Vous pouvez maintenant vérifier votre réponse à la **Question #2** avec une implémentation

2 Un diagramme de classe UML (bis)

Question #3: Réitérer l'exercice pour le diagramme de classe suivant (i.e., les deux questions précédentes s'appliquent au diagramme de classe de la Figure 2)

Question #4 : Elaborer un diagramme d'objet dans lequel au moins 3 objets B ont été créés et au moins deux objets Z. Ecrire le code Java correspondant à ce diagramme d'objet. Ecrire également le code Java qui implémente la logique de suppression d'un objet Z.

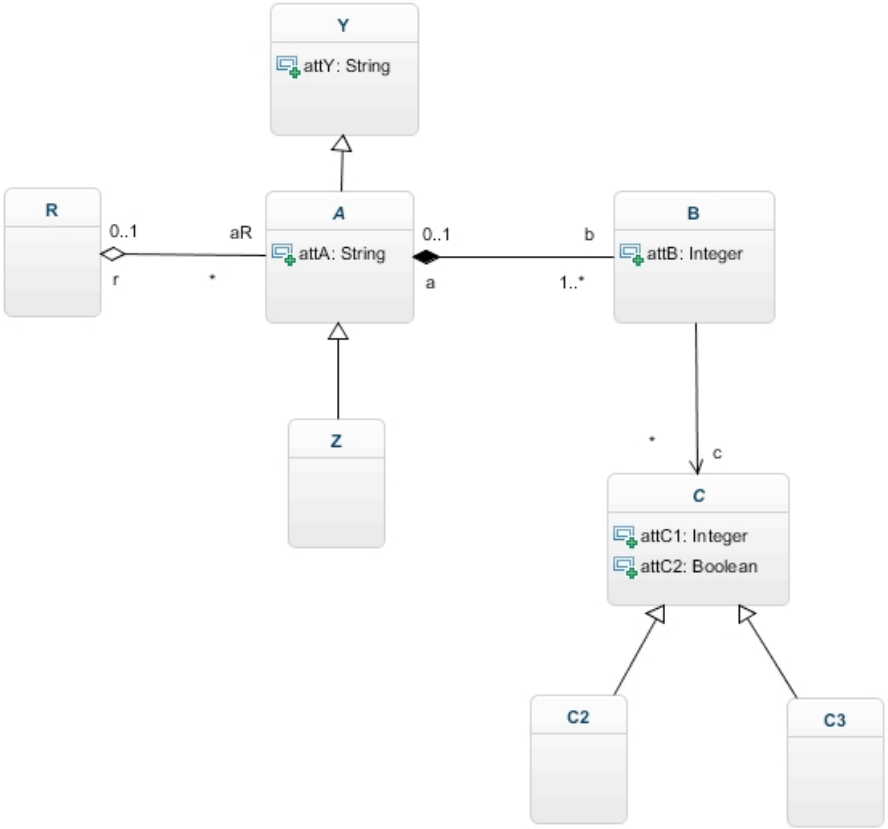


Figure 2 : un diagramme de classe plus complexe (note : les classes ***A*** et ***C*** sont abstraites)

3 Des objets aux classes

Question #5: Elaborer le diagramme d'objet équivalent au code Java ci-dessous, ainsi que le diagramme de classe associé. Est-il possible d'automatiser la rétro-ingénierie de code Java vers diagramme de classe UML ?

```
1  B b = new B()
2  A a = new A2()
3  a.b = true
4  a.c = b
5  a.d = 76
6  A a2 = new A3()
7  a2.d = 87
8  a2.c = new B2()
```

4 Transformation UML-Java

Question #6: Etudier le code généré dans les questions précédentes puis décrire les règles de transformation qui ont abouti à la génération de ce code à partir du diagramme UML.

5 Transformation UML-SQL, JPA

Cette partie aborde quelques notions avancées.

Question #7 : Générer le code SQL correspondant au diagramme de classe de la Figure 3. Pourquoi n'y a-t-il que deux tables?

Question #8 : JPA (pour Java Persistence Architecture) est une API pour faciliter le « mapping » entre les objets Java et les bases de données relationnelles (comme SQL). Générer le code JPA. Etudier le mécanisme pour supprimer des objets instances de G.

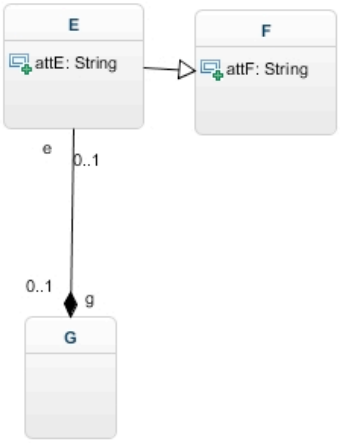
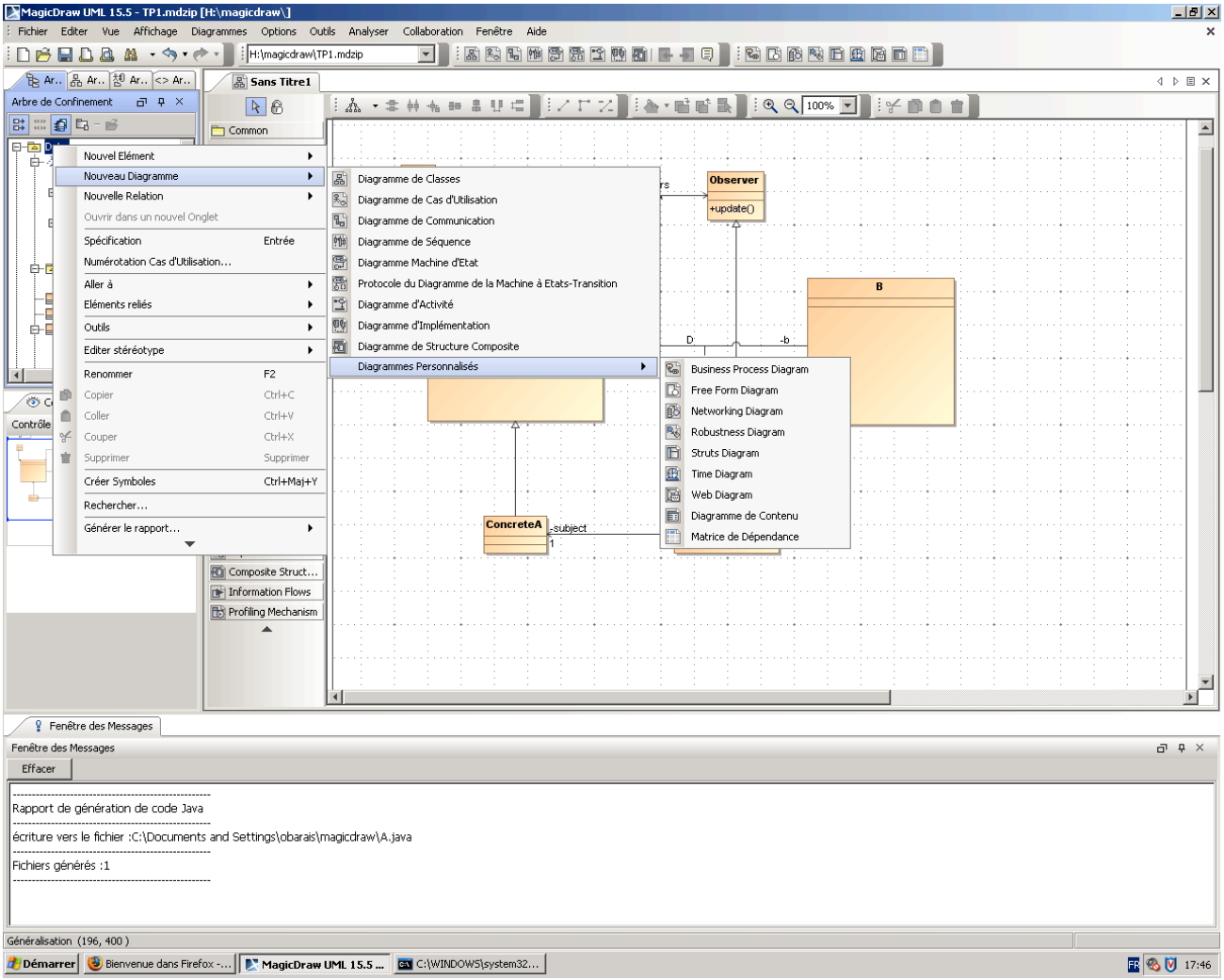


Figure 3 : un autre diagramme de classe

6 Avec Magic Draw UML

Plutôt que d'utiliser Genmymodel, nous allons utiliser un autre outil de modélisation et un autre générateur de code, Magicdraw.

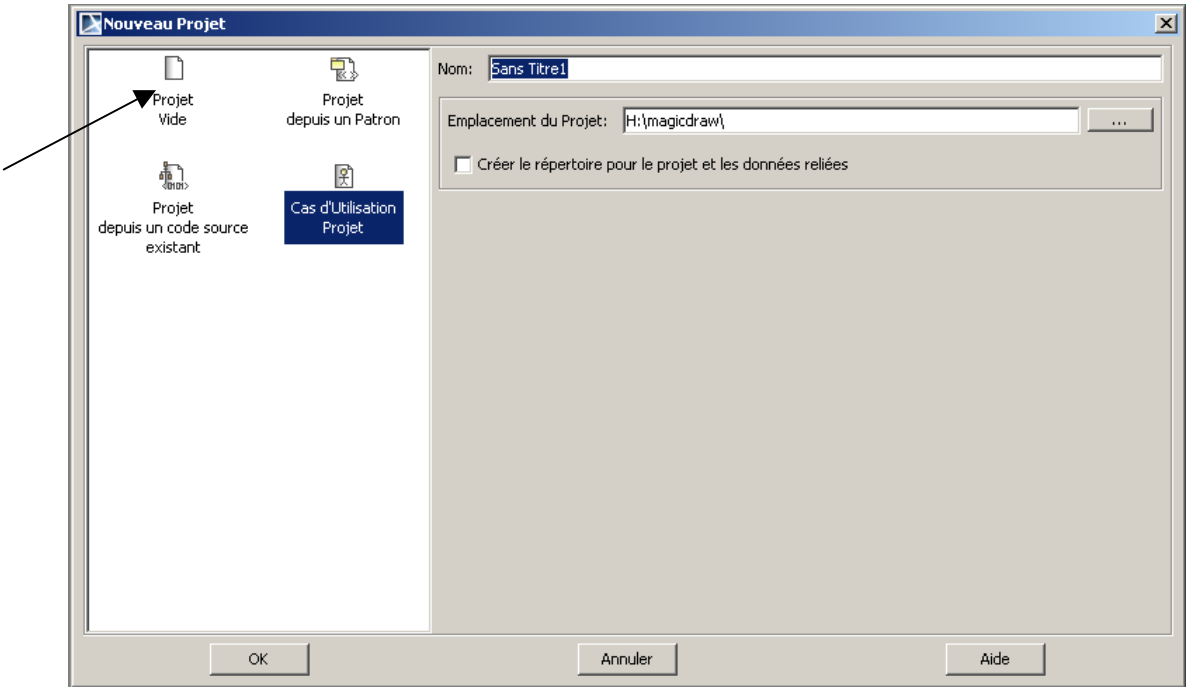
MagicDrawUML est un outil de style WYSIWYG modélisant divers diagrammes UML. Cet outil répond aujourd'hui à la problématique d'avoir un atelier de génie logiciel UML professionnel. Une vingtaine de types de diagrammes sont disponibles (classe, séquence, use case, objet, déploiement, …). Pour une description plus détaillée, l'on pourra se référer à http://www.www.magicdraw.com/



Les différents types de diagrammes proposés par MagicDraw.

Création d'un projet

Menu : *Fichier* -> *Nouveau Projet*



Création d'un projet MagicDraw.

Création d'un package

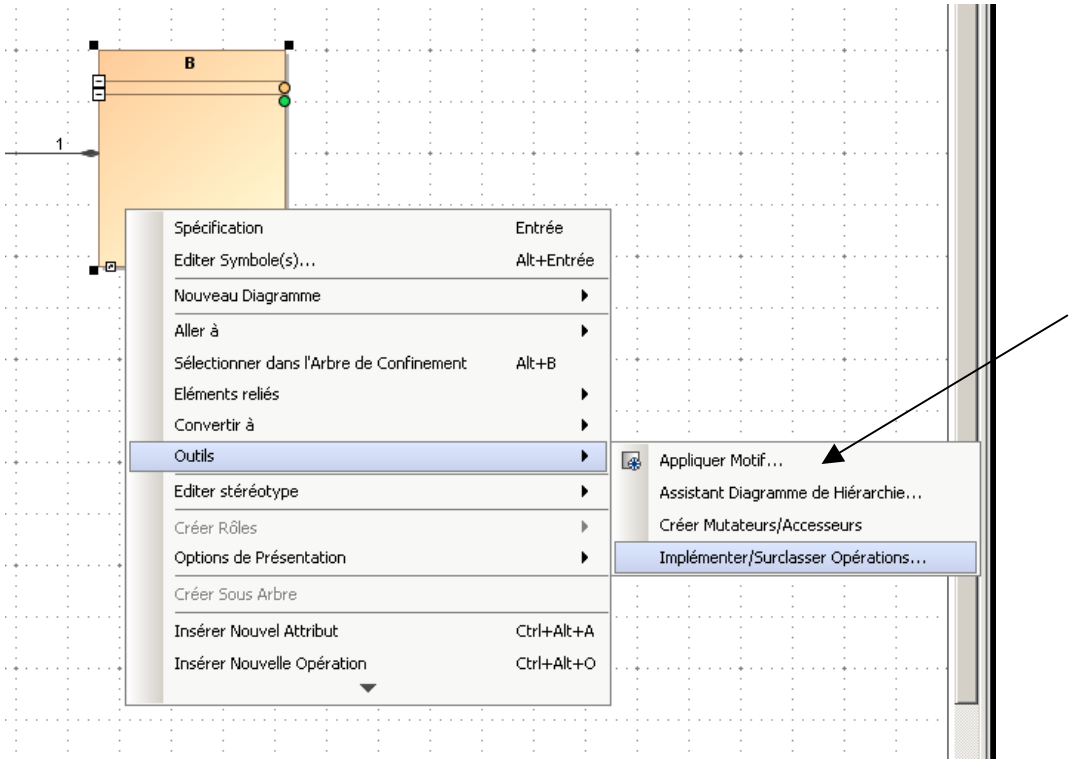
Sur le package Data : Clic droit -> nouvel élément -> Package

Création d'un diagramme de classes

Sur le package Data : Clic droit -> nouveau Diagramme -> Diagramme de classes

Restructuration et application de motif (« design pattern »)

Clic droit sur une classe -> Outils -> Appliquer Motif

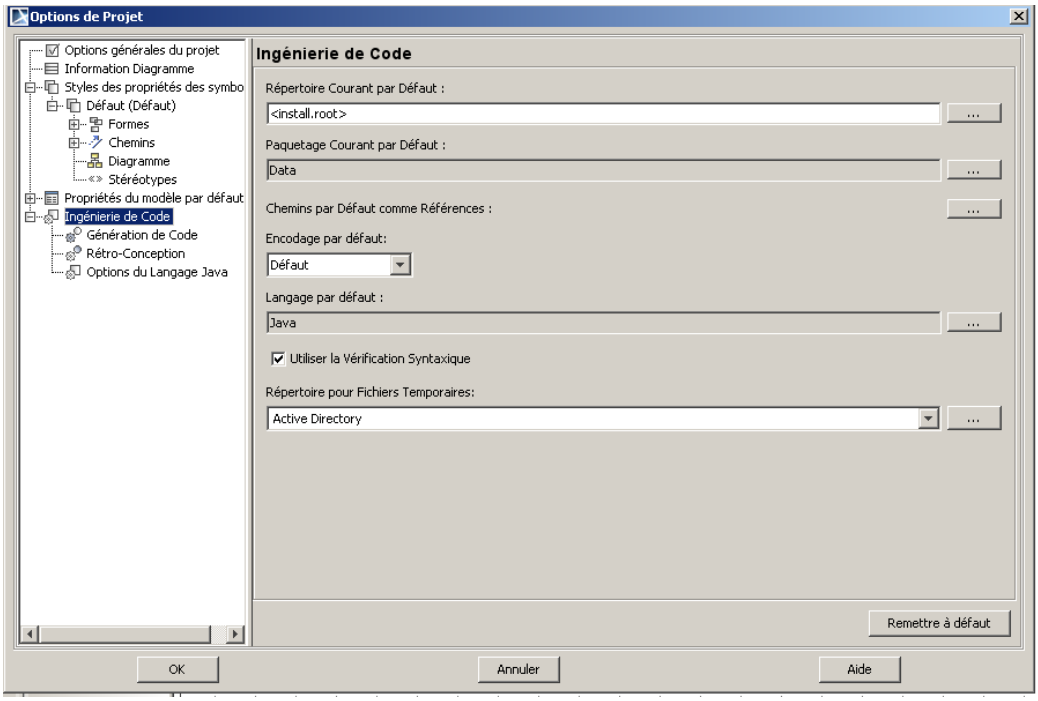


Application d'un patron de conception par MagicDraw.

Génération de code

Option projet -> Ingénierie de code -> répertoire courant par défaut.

Choisissez le répertoire source à la racine de votre projet Java de votre workspace Eclipse.



Génération de code dans MagicDraw.

Puis sélectionnez les classes à générer et faites Ctrl + G.

Travail à effectuer

- Faites les diagrammes UML vu sur le TD1-2 :
  - arbre généalogique ;
  - compilateur
- Revisiter la partie 4 (Transformation UML-Java) avec Magicdraw. Décrire les règles de transformation qui ont abouti à la génération de ce code à partir du diagramme UML. Quelles sont les différences avec Genmymodel ? Discuter les différentes stratégies de génération de code.
- Ecrire le programme Java correspondant au programme de l'exercice du compilateur, i.e., utilisez les classes générées par Magicdraw à partir de votre diagramme de classes UML de l'exercice du compilateur