

# **Conception Uml par Bouuml**

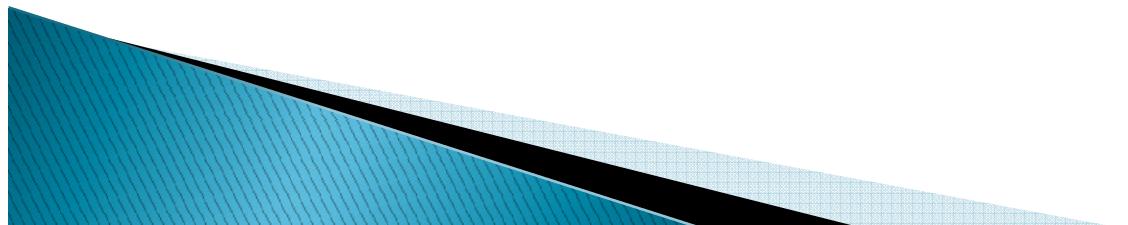
Présenté par:  
**AYADI Malika**  
**ROUISSI Mohamed**  
**RADI Said**



**Master 1 Informatique**

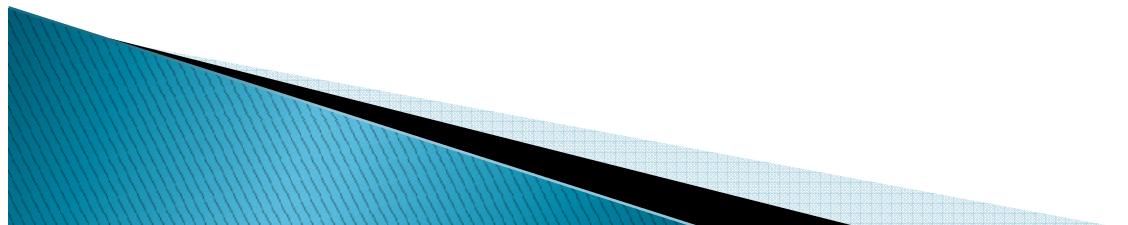
# Présentation

- C'est quoi Bouml ?
- Pourquoi Bouml ?
- Uml et Bouml
- Conception
- Plug-out
- Génération de Code
- Reverse Engineering
- Avantages/Inconvénients Bouml



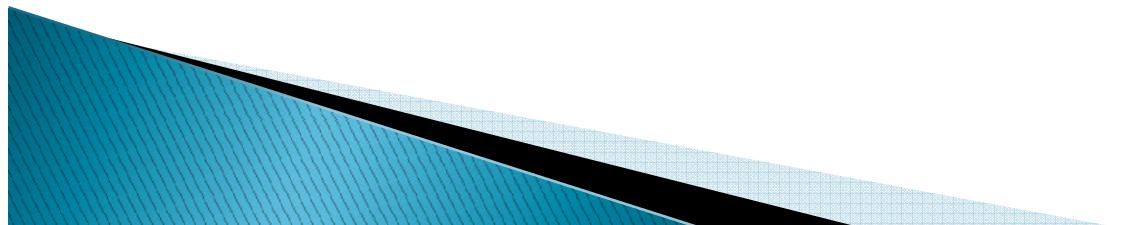
# C'est quoi Bouml ?

- BOUML est une série de logiciels comprenant un modeleur UML et plusieurs programmes externes dont des générateurs de code et reverse.
- Il est développé en C++ au dessus de Qt, ce qui permet son utilisation sous *Windows* et la constellation *Unix* (*Linux*, *Solaris*, *Mac OS X* etc.).



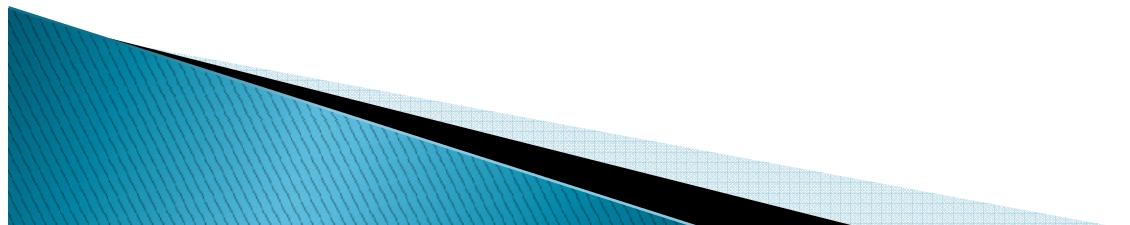
# Pourquoi Bouml ?

- Le but de BOUML est de permettre une utilisation allant des besoins à la génération de code
- Les langages pris en compte à ce jour étant *C++, Java, Php* et *IDL* .



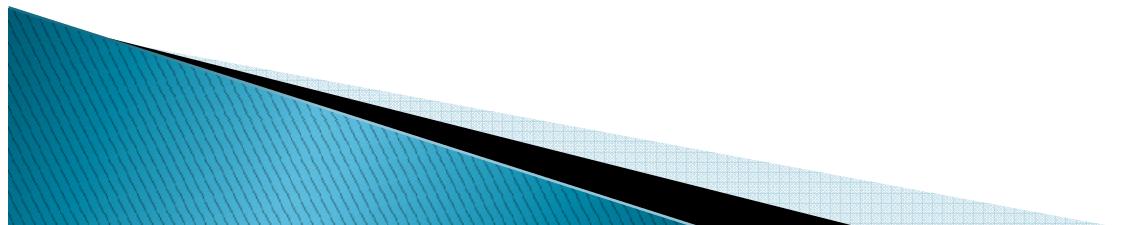
# Uml et BOUML

- Uml définit treize types de diagramme différents, BOUML permet de modéliser neuf d'entre eux :
  - Diagrammes de cas d'utilisation.
  - Diagramme d'objet.
  - Diagramme de classes.
  - Diagrammes de composants.
  - Diagrammes de déploiement.
  - Diagrammes d'activités.
  - Diagrammes de collaboration.
  - Diagrammes de séquence.
  - Diagrammes d'état– transitions.



# Conception

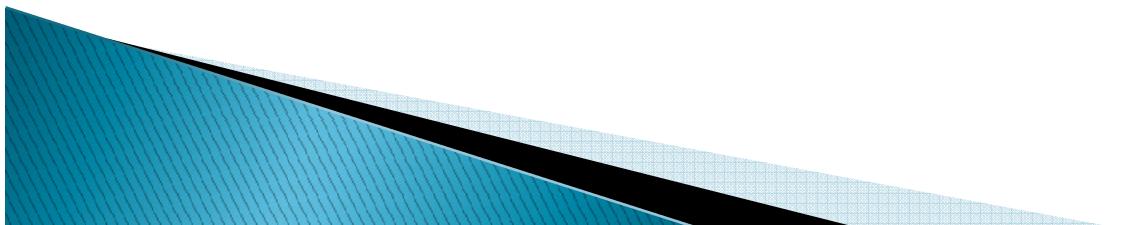
- on va présenter les diagrammes les plus utilisés avec UML:
  - Diagrammes de classes
  - Diagramme de séquences.



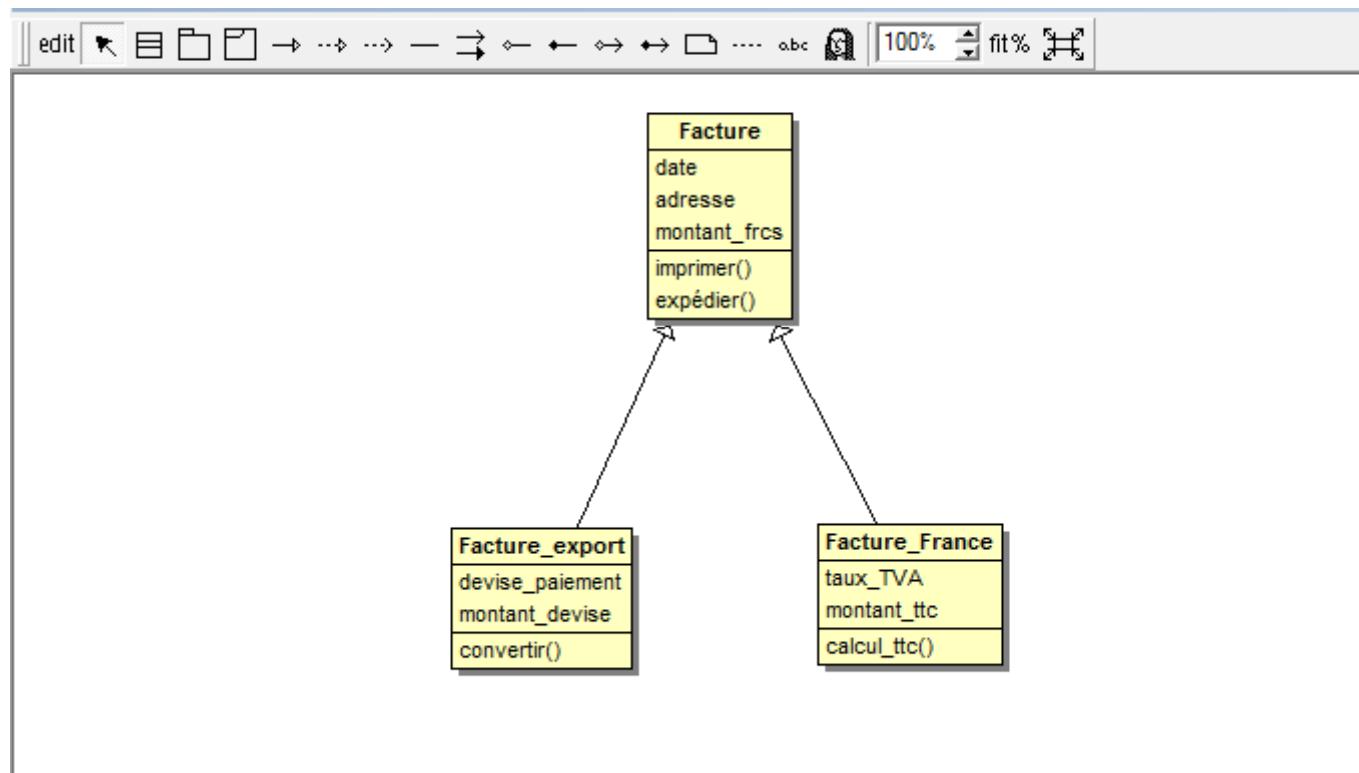
# Conception

## Diagramme de classes

- Pour créer un diagramme de classes, il faut d'abord créer une « class view ». Cette vue contiendra les classes.
- Pour cela il faut faire un clic droit sur le projet puis *new class view*.
- Enfin, lui ajouter un diagramme de classes grâce à un clic droit sur la nouvelle vue, puis *new class diagram*



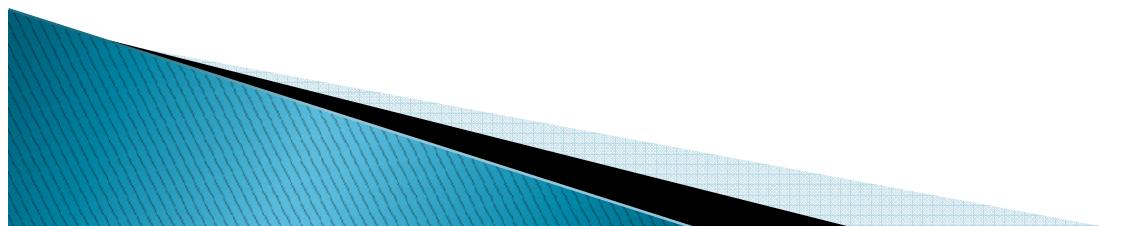
# Exemple diagramme de classe



# Conception

## Diagramme de séquences

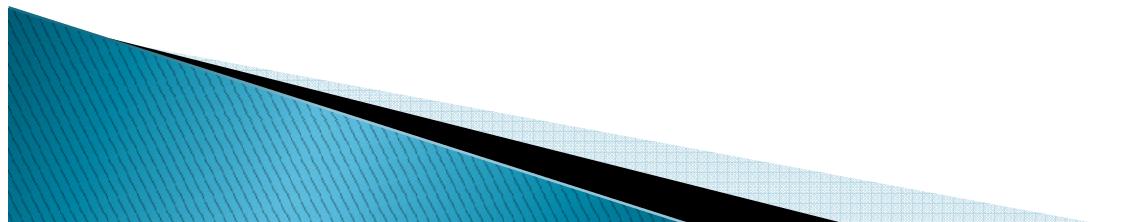
- Le diagramme de séquence doit être créé dans une « use case view ». Contrairement à la « class view », celle ci pourra contenir, en plus du diagramme de séquences et des objets associés,des acteurs.



# Plug-out

Définition:

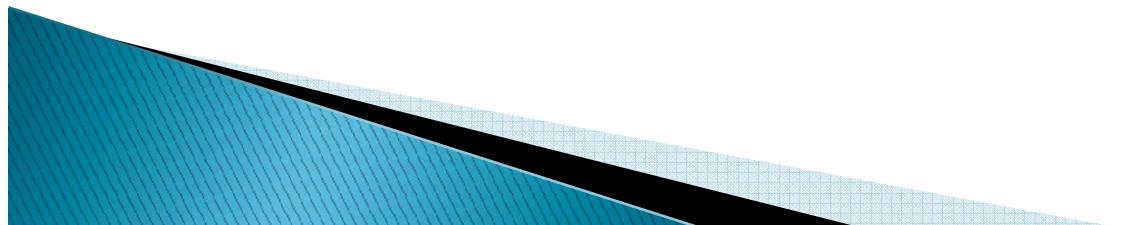
- Sont des programmes permettant d'accéder et/ou de modifier automatiquement des modèles.
- Les générateurs de code et reverse en sont des exemples.



# Plug-out

A quoi ça sert?

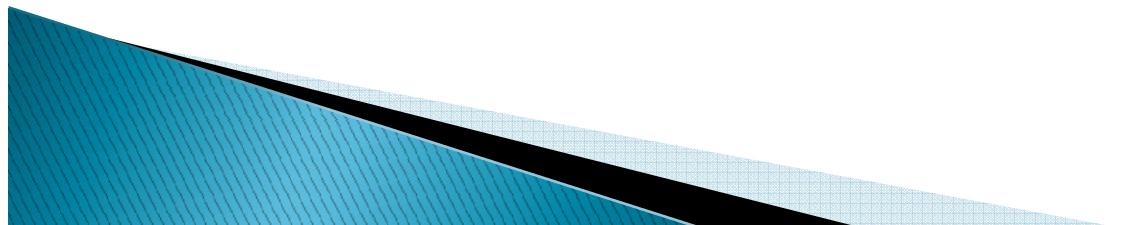
- Les plug-outs permettent de rajouter des fonctionnalités.
- Mais aussi de générer du code, de la documentation...etc



# Plug-out

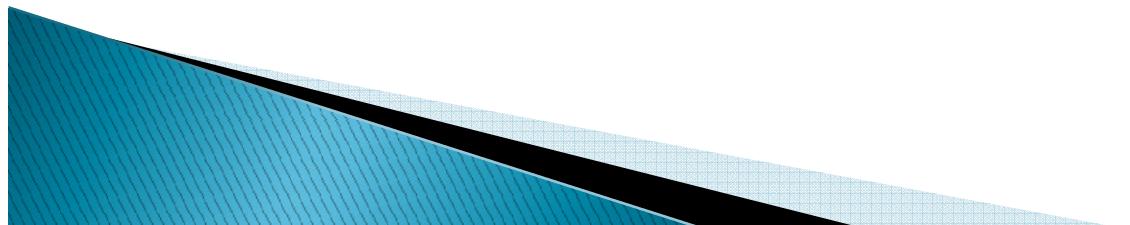
Exemple:

- Le générateurs Html est un *plug-outs* accédant au modèle pour en extraire le contenu.
- Le générateur de code et reverse C++ et java.
- On peut aussi créer nos propres plug-outs avec du C++ ou du java.



# Génération de code

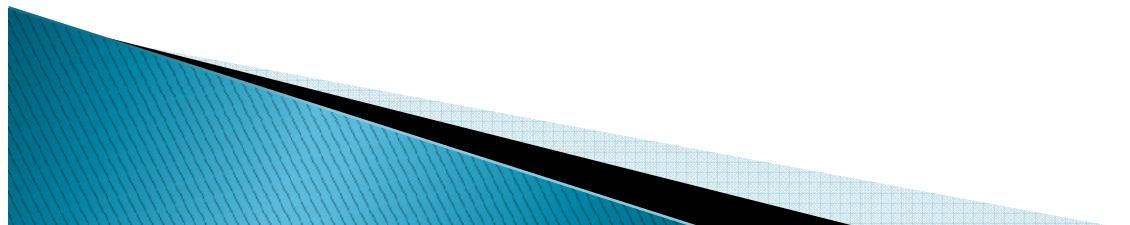
- BOUML permet de générer un squelette de code à partir des différents diagrammes dessinés
- Si cette fonctionnalité est aujourd'hui présente dans la majorité des AGLs (ateliers de génie logiciel) dite de conception, elle n'en reste pas moins très appréciable.



# Génération de code

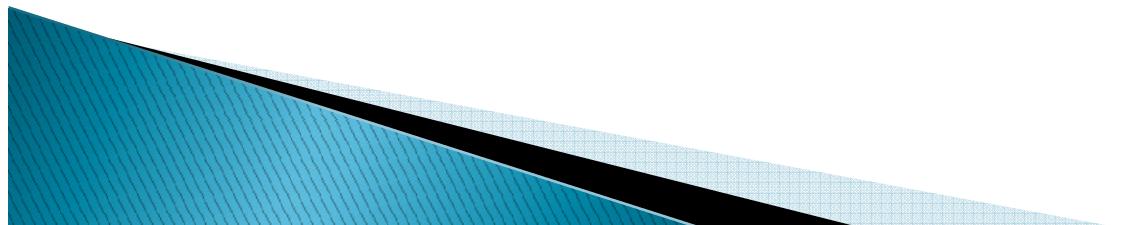
Pour générer du code:

- crée une « deployment view ».
- Associer cette vue à la « class view » où se trouvent nos classes à exporter. Double-cliquer sur la « class view » permet de l'éditer et de changer le champ « deployment view ».
- Associer à chaque classe un « artifact » avec un clic droit sur cette classe. Il s'agit du fichier qui la contiendra.



# Génération de code

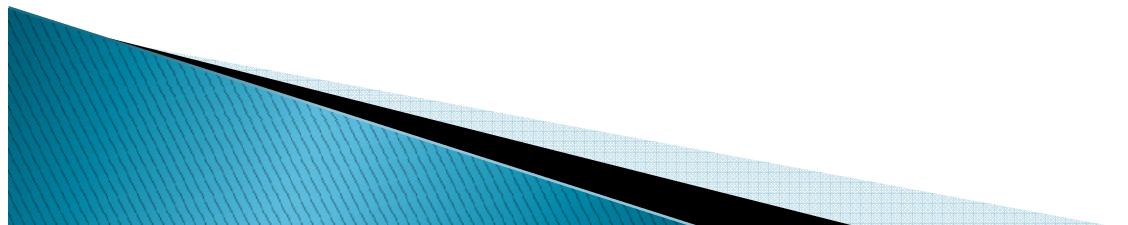
- Choisir le répertoire où seront créés les fichiers avec un clic droit sur le projet puis *Edit generating settings/Directory*.  
un clic droit sur le projet suivi de *Generate, language* générera le code.



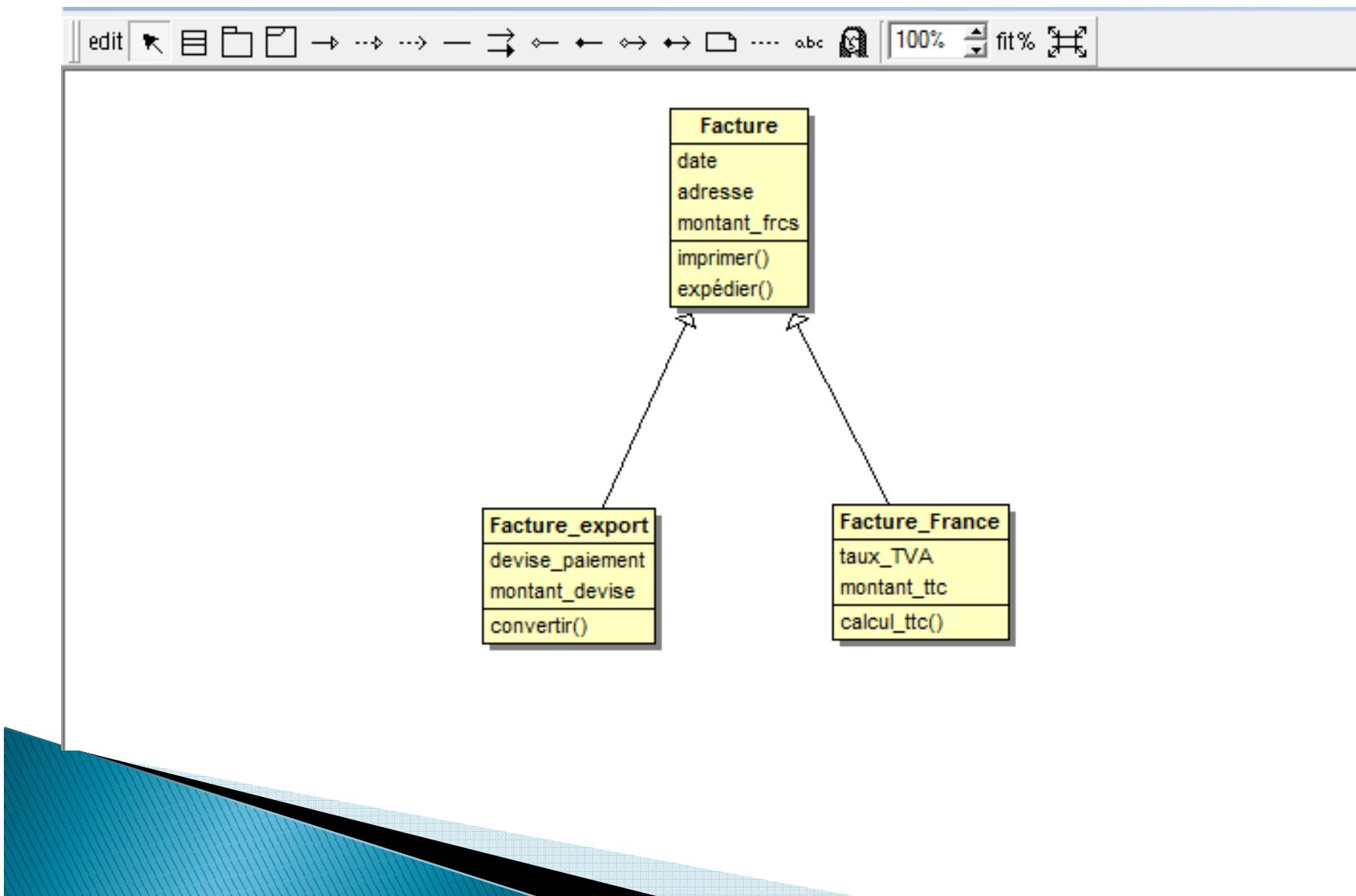
# Génération de code

## Export en java

- Avant tout il faut aller dans le menu *Languages* et choisir *java* (sinon rien ne marche)
- Pour générer du code java bien fait avec le corps des fonctions, il faut faire un clic droit sur chaque méthode et choisir <<*Edit Java Body*>>. On pourra mettre /e contenu qu'on veut.



# Génération de code



# Génération de code (exemple:Java)

```
1  
2 class Facture {  
3     private date;  
4  
5     private adresse;  
6  
7     private montant_frcs;  
8  
9     public imprimer() {  
10    }  
11  
12    public expédier() {  
13    }  
14  
15}  
16
```

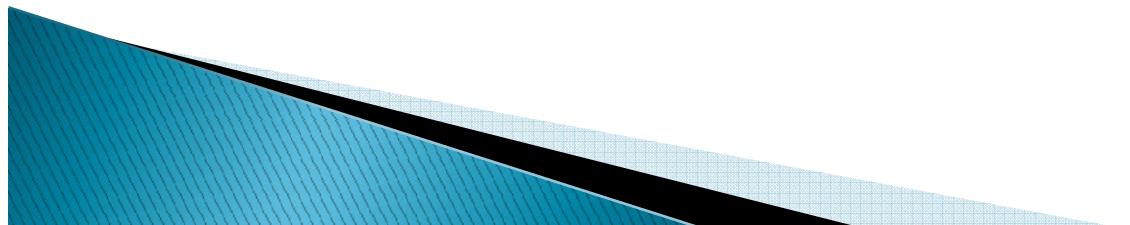
```
1  
2 class Facture_export extends Facture {  
3     private devise_paiement;  
4  
5     private montant_devise;  
6  
7     public convertir( devise) {  
8    }  
9  
10}  
11
```

```
1  
2 class Facture_France extends Facture {  
3     private taux_TVA;  
4  
5     private montant_ttc;  
6  
7}  
8
```

# Génération de code

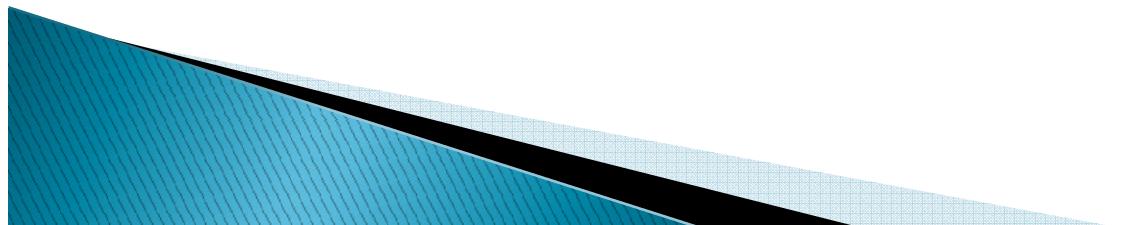
## Export en C++

- C'est la même chose qu'en java sauf que au début il faut bien cocher dans le menu *Languages le langage C++.*
- Les fichiers générés contiennent un fichier.h et un fichier.cpp



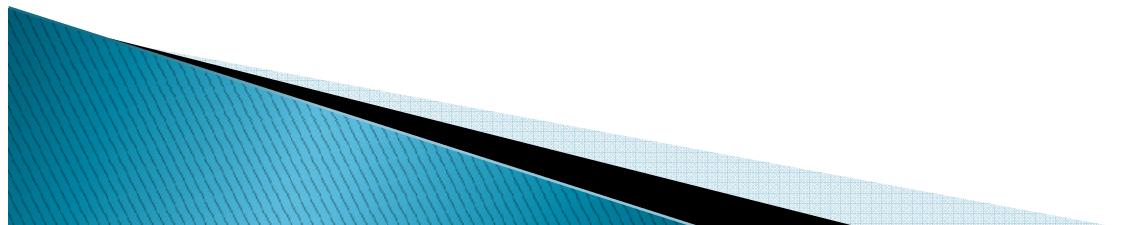
# Reverse Engineering

- Il s'agit de générer des diagrammes de classes avec des sources, Java, C++, PHP ou Idl.



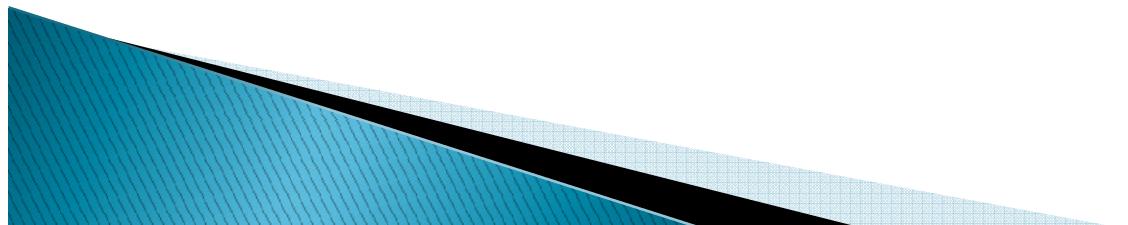
# Reverse Engineering

- Tout d'abord il faut activer le langage utilisé comme dans précédemment.
- Dans le menu *Tools* il faut choisir *reverse* suivi du langage et ensuite choisir le package que l'on veut utiliser.



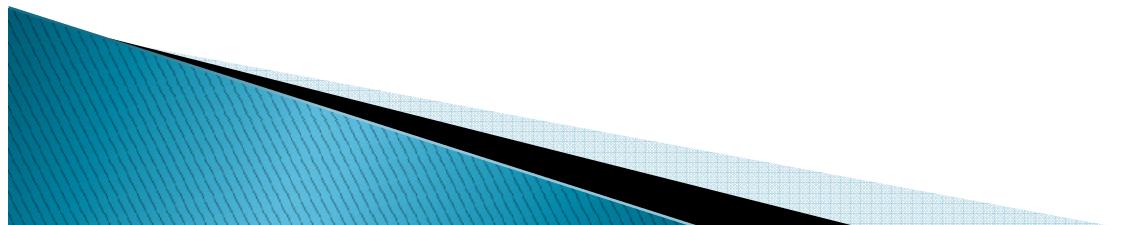
# Avantages BOUML

- BOUML est très rapide et demande peu de mémoire pour modéliser plusieurs milliers de classes.
- Définition des attributs et des opérations est assez ergonomique.
- La documentation est très bien faite et complète.
- Il est gratuit, open source, et multiplateforme



# Inconvénient Bouml

- La génération de code à partir des diagrammes n'est pas simple.
- L'entrée des paramètres et la modification partiel sur des parties de code sont de vrai handicapes.



# Liens Utilisés

- ▶ <http://bouml.free.fr/doc/>
- ▶ <http://www.framasoft.net/article3966.html>
- ▶ <http://bouml.free.fr/doc/>
- ▶ <http://www.developpez.net/forums/>