I.U.T. LUMIÈRE

UNIVERSITÉ
LUMIÈRE LYON 2

POO 2/3
-
Modélisation Objet
UML / Unified Modeling Language

Pierre Parrend
IUT Lumière Lyon II, 2005-2006
pierre.parrend@univ-lyon2.fr

CERTAINS DROITS RESERVES

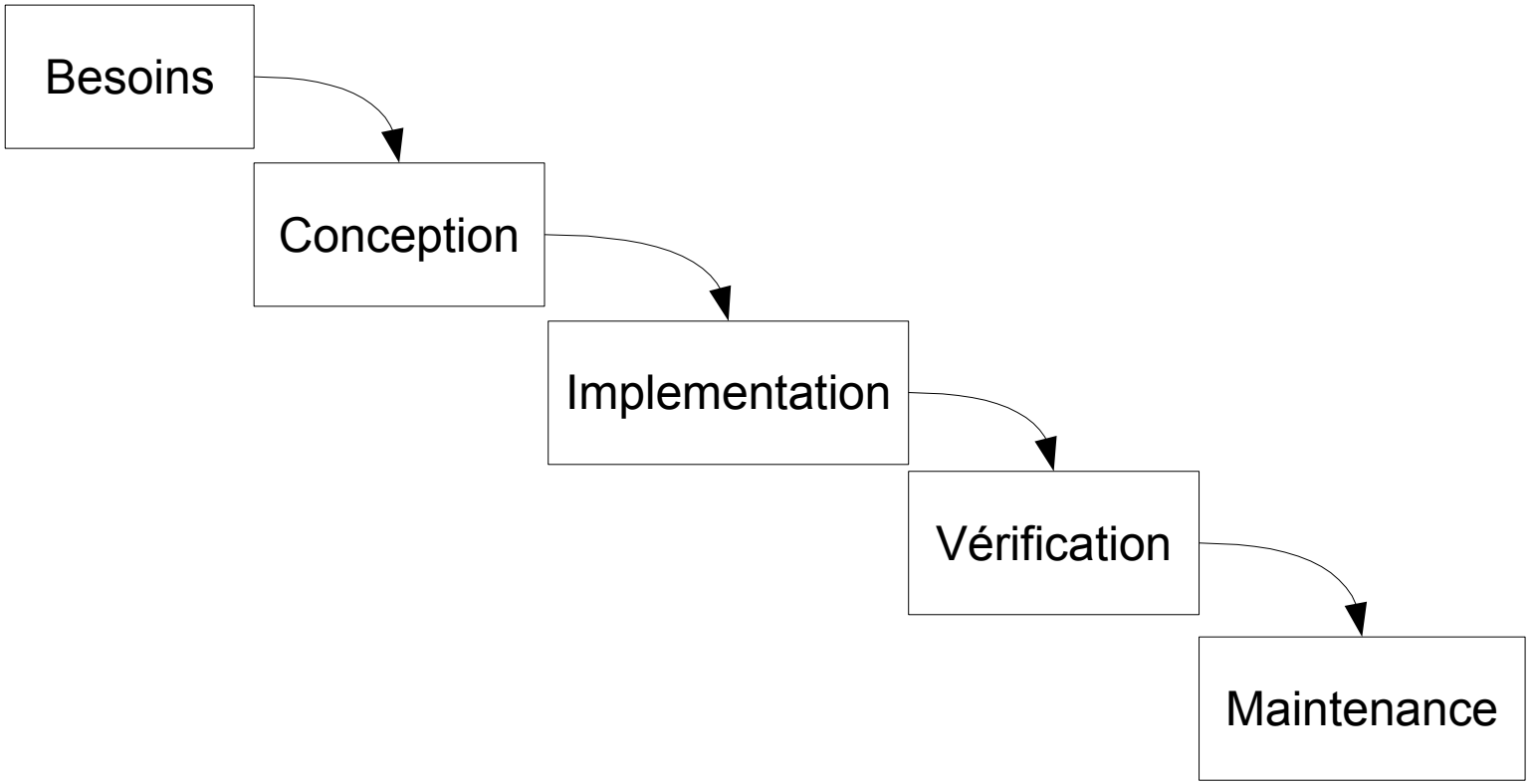
Sommaire

- Les cycles de vie du logiciel
- Le diagramme UML de Cas d'Utilisation
- Le diagramme UML de Classes
- Le diagramme UML de Séquence
- Autres diagrammes UML
- Les Design Pattern

Pour Programmer en Objet, il faut être capable de concevoir en Objet

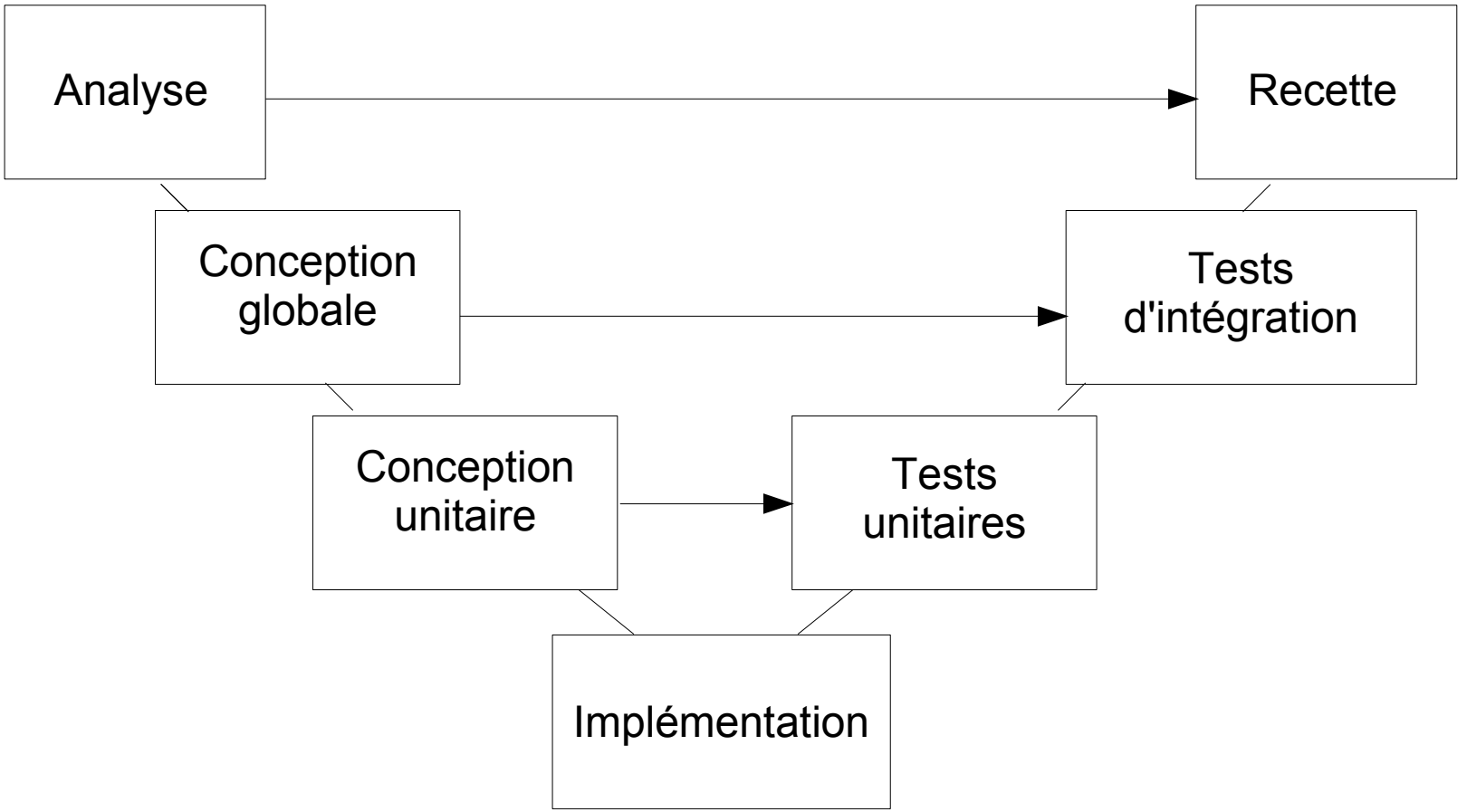
Le Cycle de Vie du Logiciel

- **Modèle en Cascade**



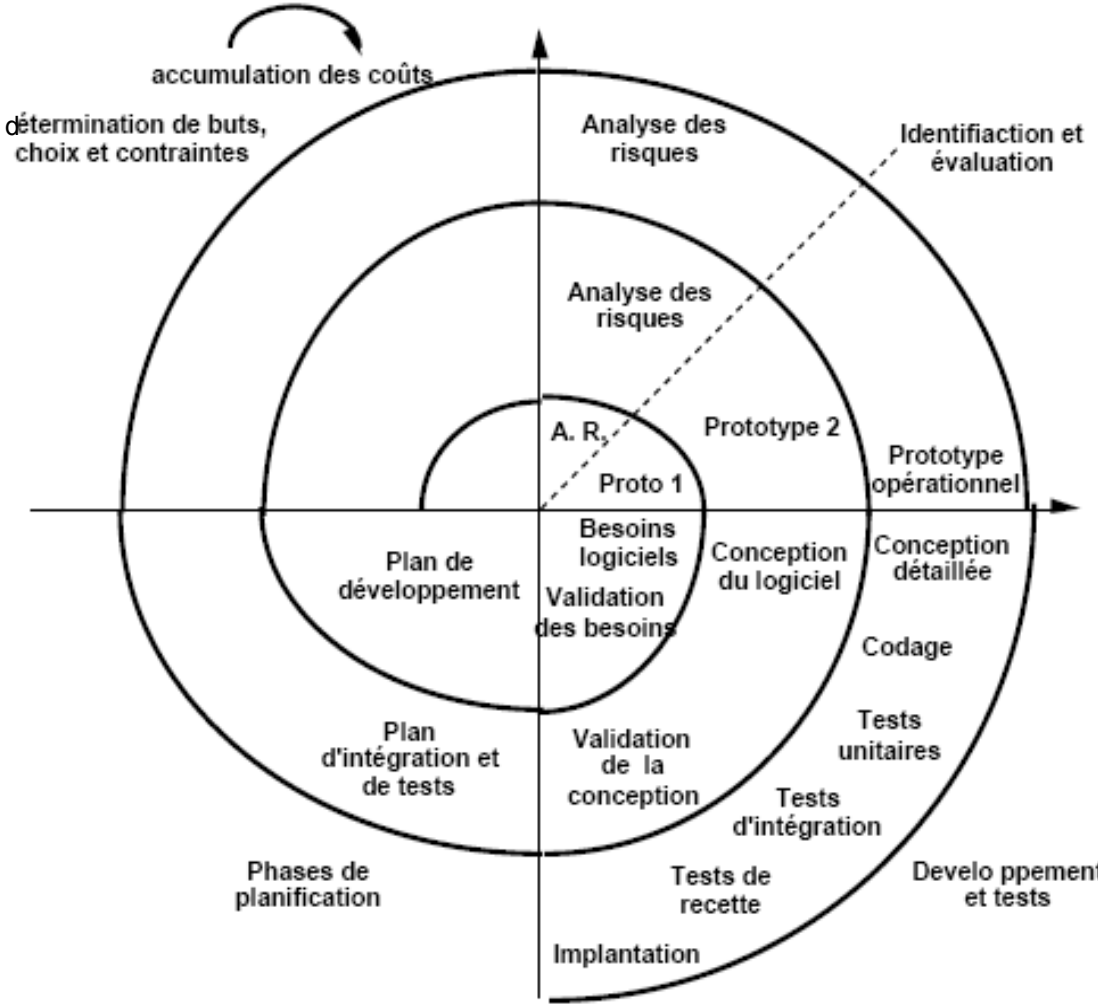
Le Cycle de Vie du Logiciel

- **Modèle en V**



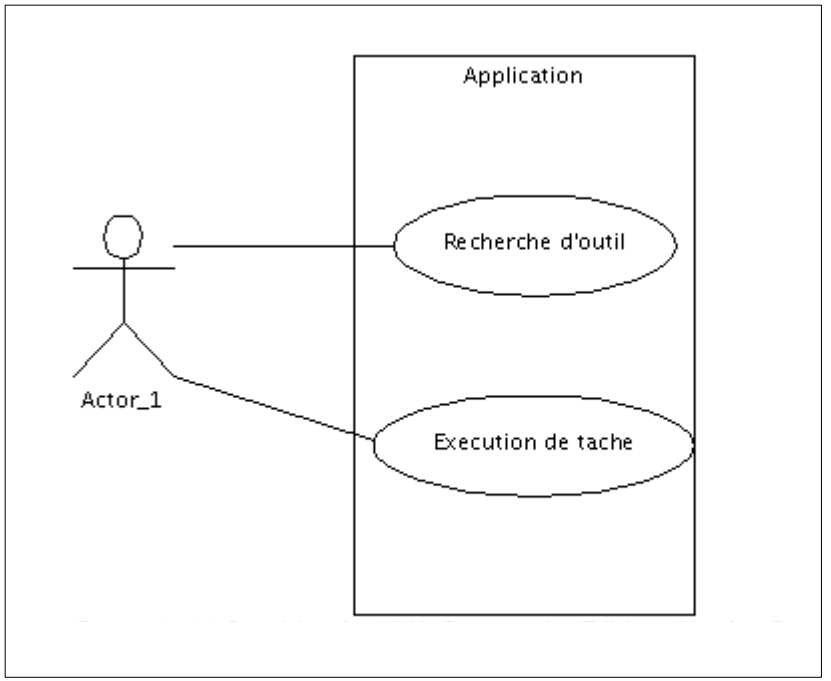
Le Cycle de Vie du Logiciel

- **Modèle en Spirale**



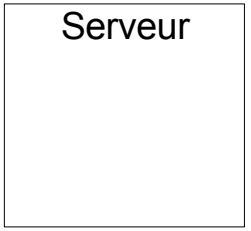
Le Diagramme de Cas d'Utilisation

- Principe
  - Représentation des interactions entre un système et les utilisateurs
- Example

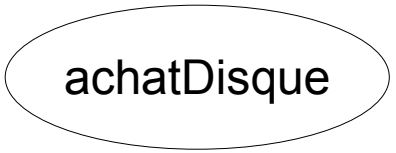


Le Diagramme de Cas d'Utilisation

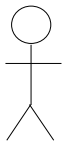
- Eléments
  - Système



  - Cas d'utilisation



  - Utilisateur

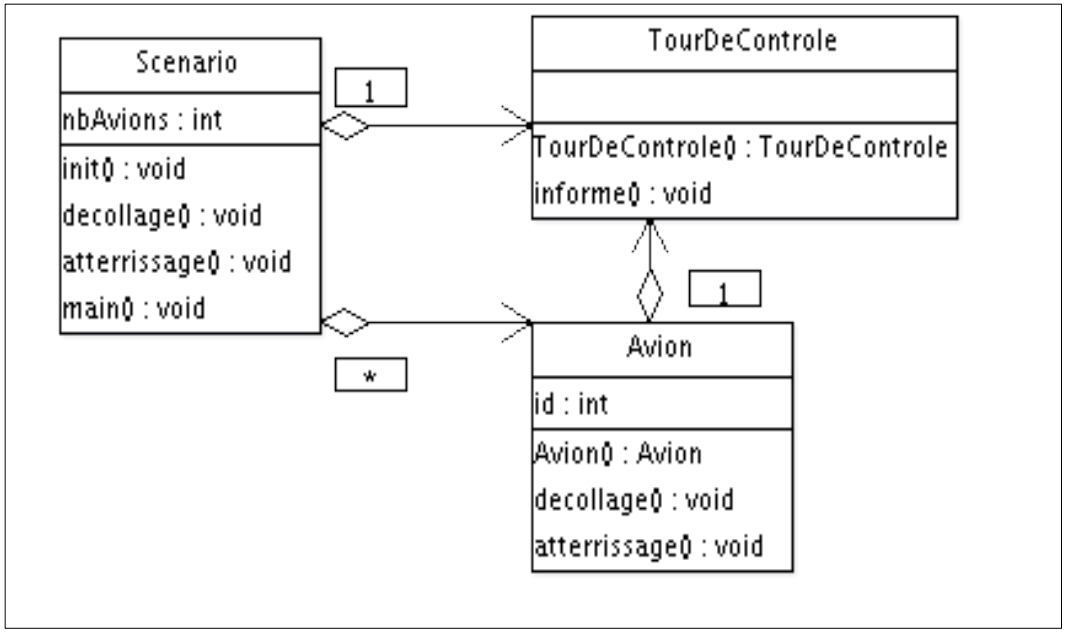


Client

Attention:
Respectez les conventions graphiques !!

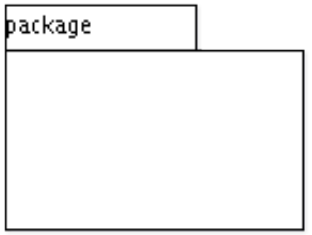
Le diagramme de Classe

- Principe
  - Représentation des Classes d'un programme
    - de leurs membres
    - De leurs interactions
- Example

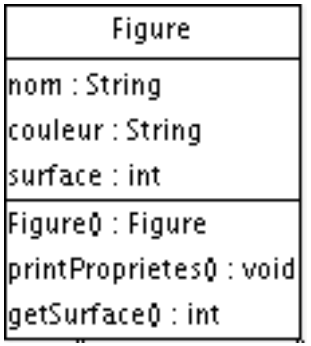


Le diagramme de Classe

- Eléments
  - Package

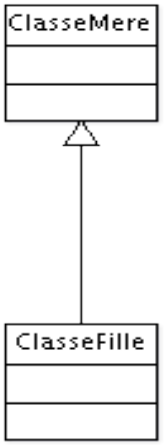


  - Classes, attributs, méthodes

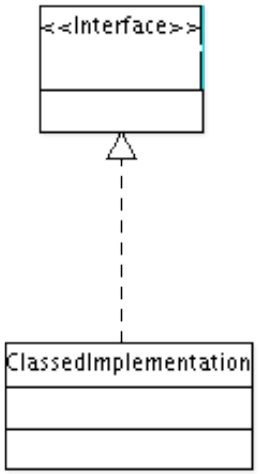


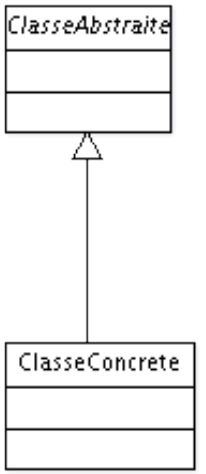
Le diagramme de Classe

- Héritage



- Interfaces et Classes Abstraites

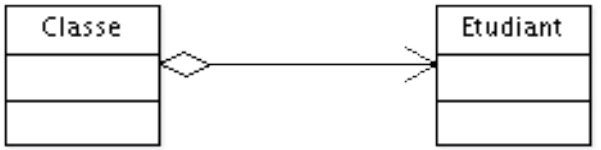




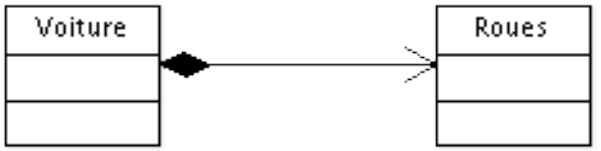
Le diagramme de Classe

- Eléments
  - Association
    - Lien simple entre deux classes

  - Aggrégation
    - Lien entre deux classes dont les cycles de vie sont indépendants

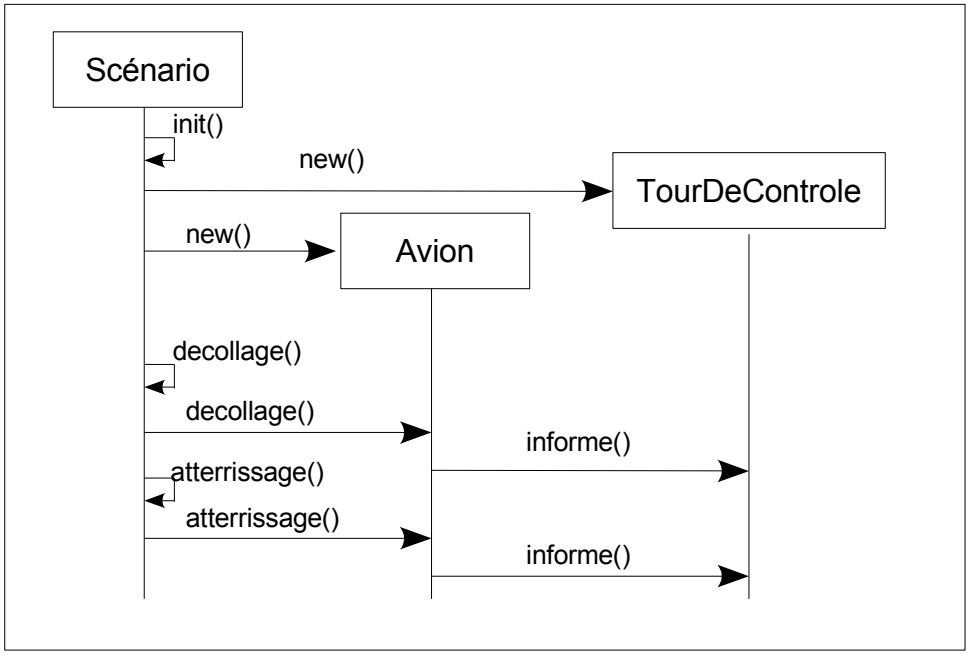


  - Composition
    - Lien entre deux classes dont les cycles de vie sont liés



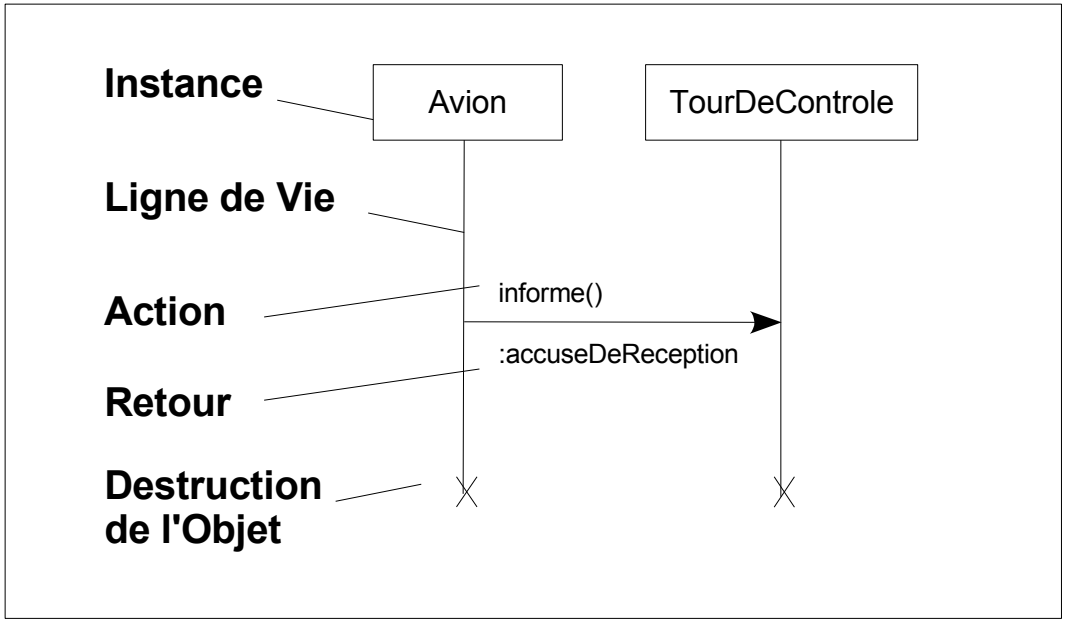
Le diagramme de Séquence

- Principe
  - Représentation des interactions entre les objets d'un programme
- Example



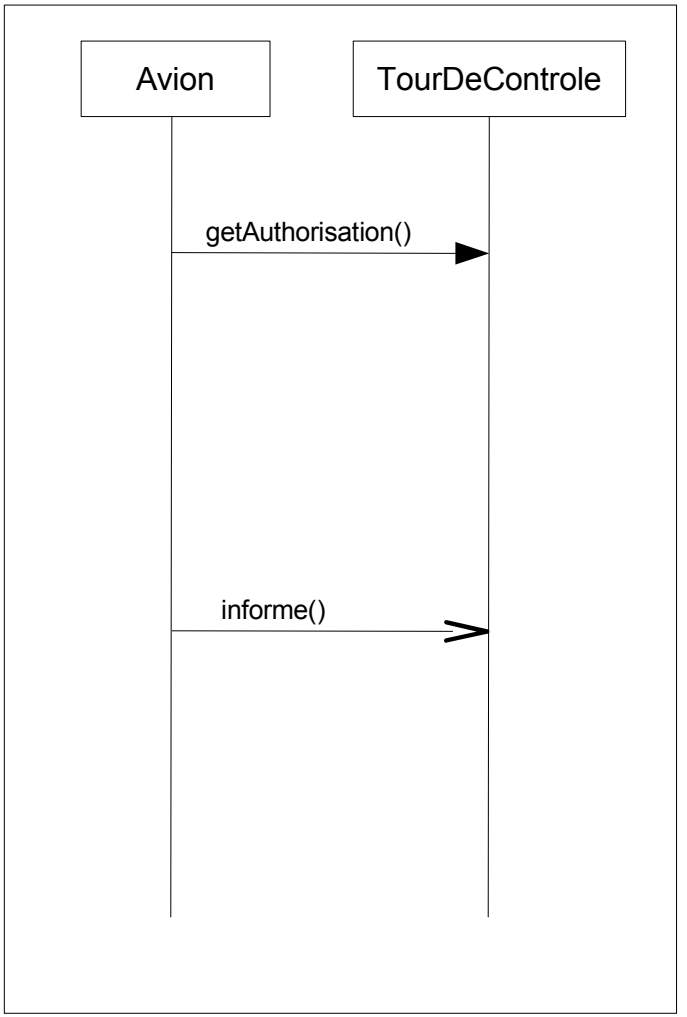
Le diagramme de Séquence

- Eléments
  - Instance, ligne de vie, action, retour



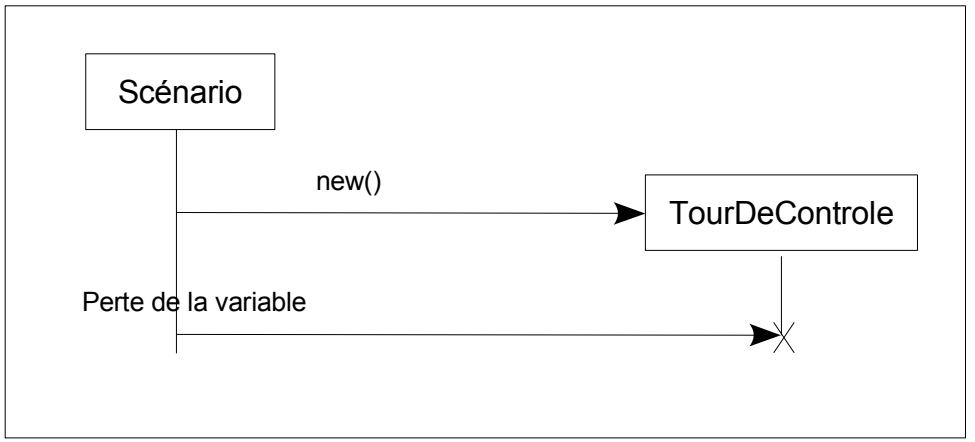
Le diagramme de Séquence

- Eléments
  - Appel synchrone
    - Avec attente de la réponse
    - Appel de méthode
  - Appel asynchrone
    - Sans attente de la réponse
    - Emission de message



Le diagramme de Séquence

- Eléments
  - creation/destruction



Autres diagrammes

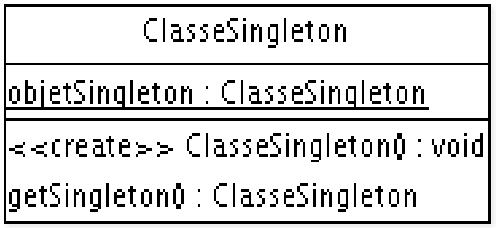
- UML 1.3
  - Diagrammes de comportement
    - Cas d'Utilisation, Etat, Activité
  - Diagrammes de Structure
    - Classes, Objets, Composants, Déploiement
  - Diagrammes d'Interactions
    - Collaboration, Séquence

Design Patterns

- Différents Design Patterns (DP)
  - = 'Patrons de Conception'
  - 'Solution à une problème de programmation récurrent'
  - Souvent représentés en UML
    - mais pas obligatoirement
  - DP de Création
    - Singleton, Usine
  - DP Structurels
    - Facade
  - DP comportementaux
    - Inversion de Contrôle

I.U.T. LUMIÈRE

Design Patterns



- Singleton

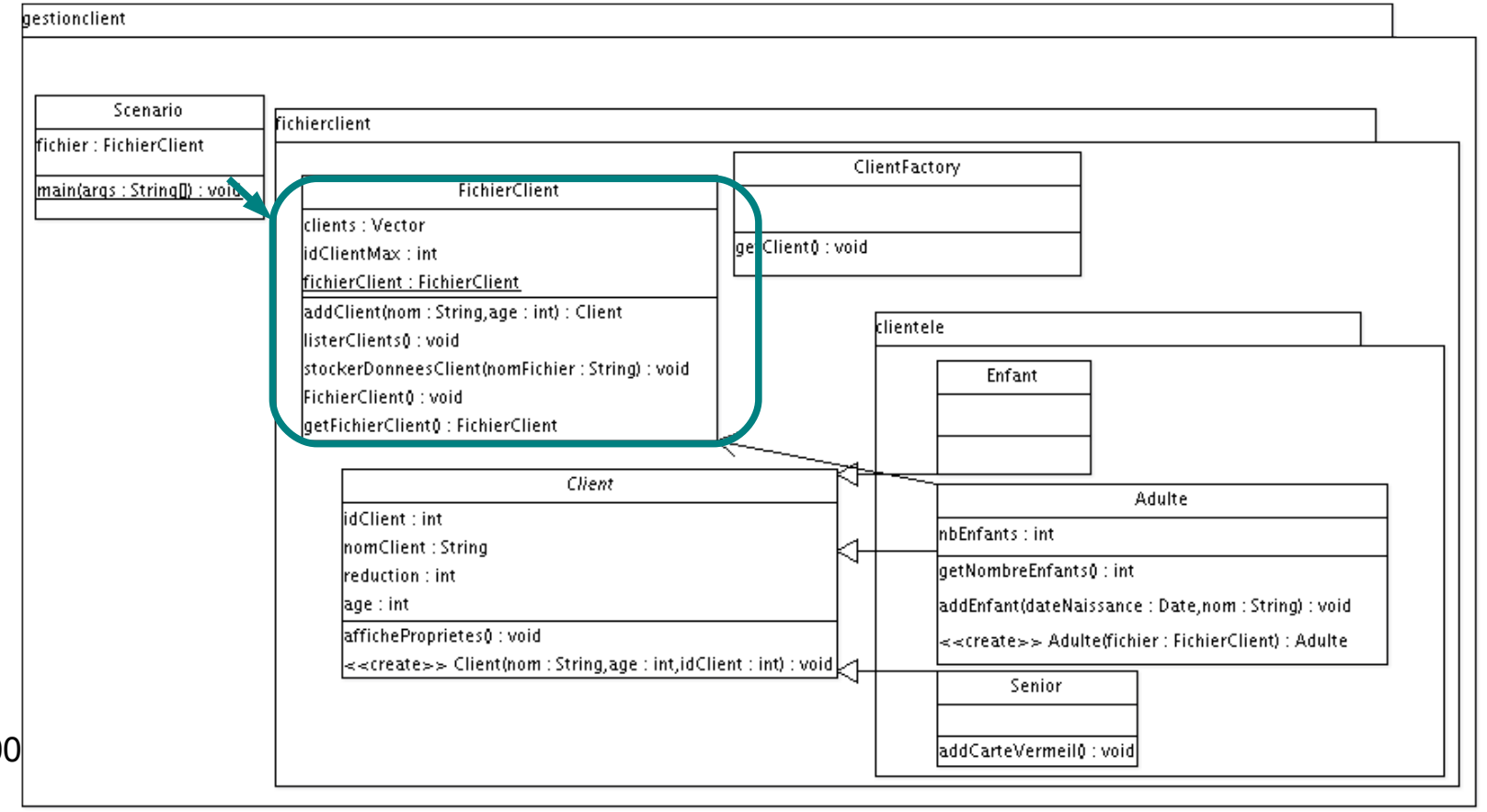
```
private static FichierClient fichierClientSingleton;

public class FichierClient
{
  private FichierClient()
  {
    System.out.println("Nouveau fichier client cree");
  }

  public static FichierClient getFichierClient()
  {
    if(fichierClientSingleton==null)
    {
       fichierClientSingleton=new FichierClient();
    }
    return fichierClientSingleton;
  }
}
```

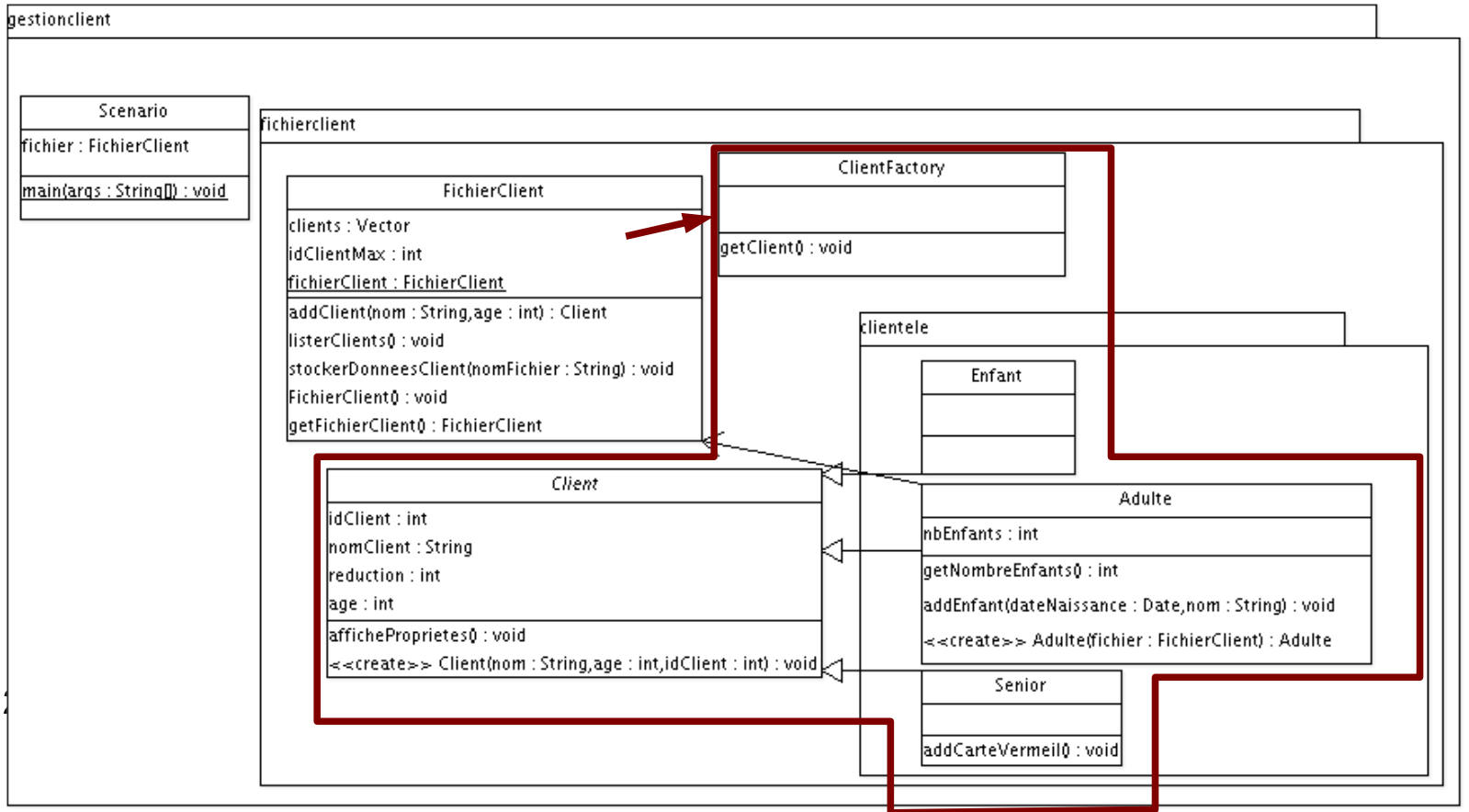
Design Patterns

- Façade
  - Le Design Pattern Façade consiste à avoir une classe comme point d'entrée unique d'une partie d'une programme.



Design Patterns

- Usine
  - Le Design Pattern Usine (Factory) consiste à utiliser une classe de génération d'objets, en fonction du contexte.



Design Patterns

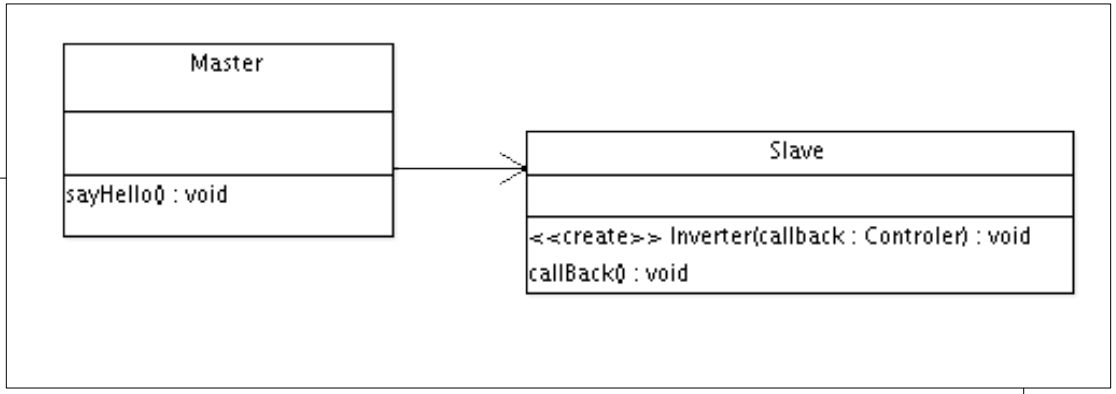
- Usine
  - Exemple

```
public class ClientFactory {

 public static Client getClient(String nom, int age, int id, FichierClient fichier){
   Client monClient;
   if(age<18)
   {
      monClient=new Enfant(nom,age,id);
   }
   else if(age<65)
   {
      monClient=new Adulte(nom,age,id,fichier);
   }
   return monClient;
  }
}
```

Design Patterns

- Inversion de Contrôle




```
package ioc;

public class Master
{
  public void sayHello(){System.out.println("Hello");}

  public static void main(String[] args){
    Slave esclave=new Slave(new Master());
    esclave.callBack();
  }
}

class Slave
{
  private Master monMaitre;
  public Slave(Master monMaitre){this.monMaitre=monMaitre;}
  public void callBack(){monMaitre.sayHello();}
}
```