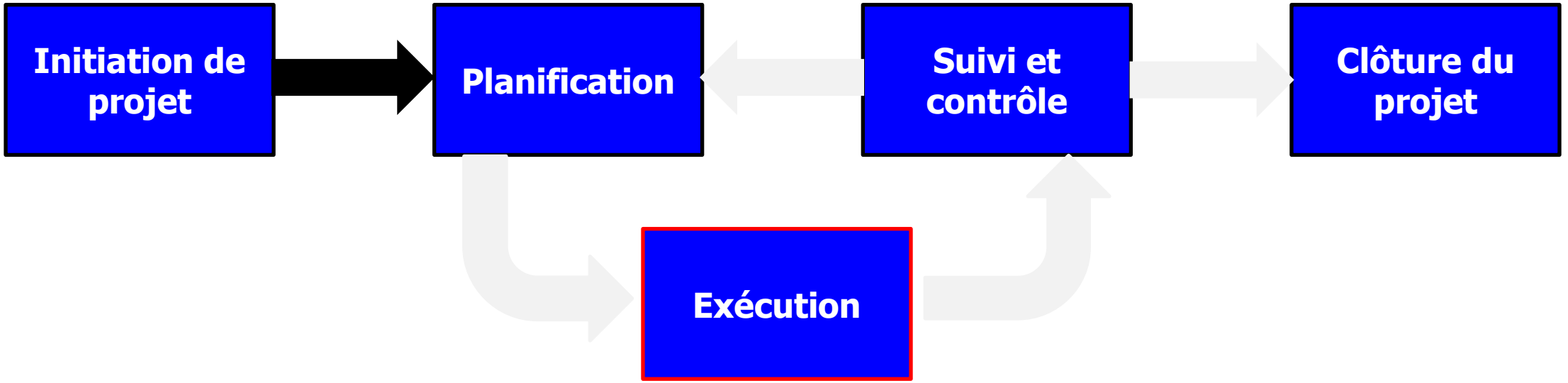
Chapitre 5

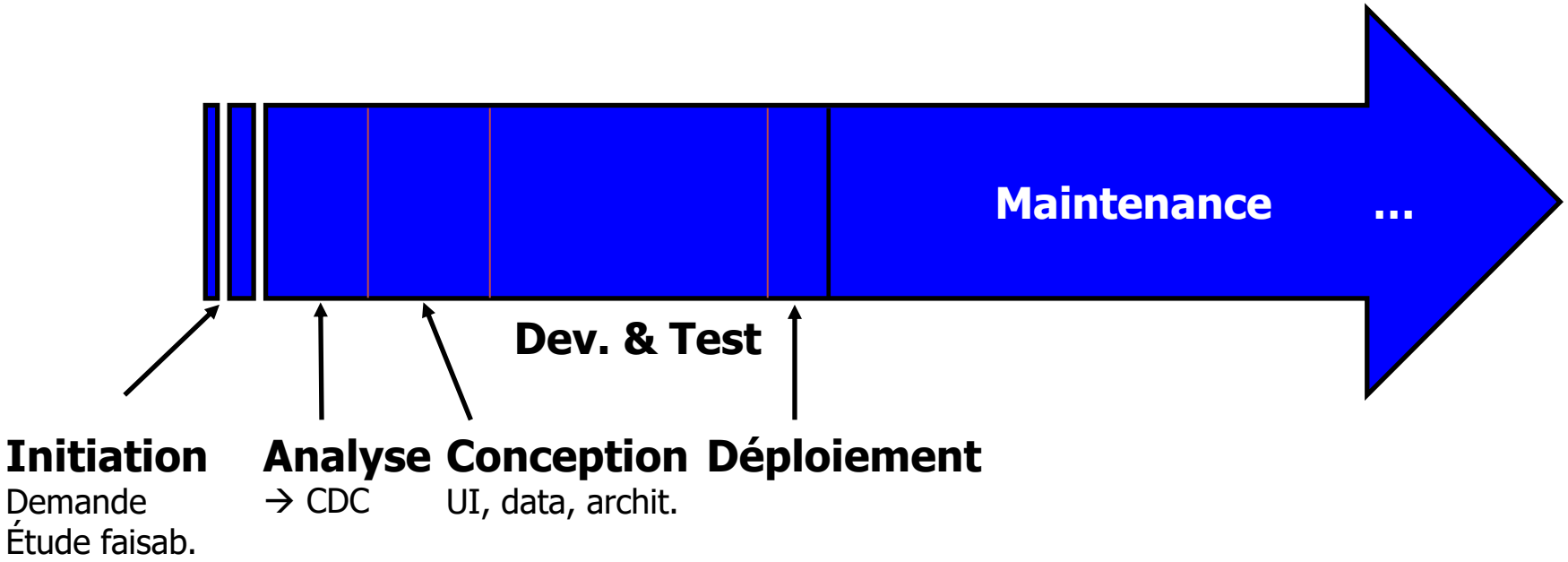
UML : Modélisation Structurelle

Vue d'ensemble



RAPPEL

Cycle de vie d'un logiciel ou système:



Objectifs

Analyse ➔ QUOI ?

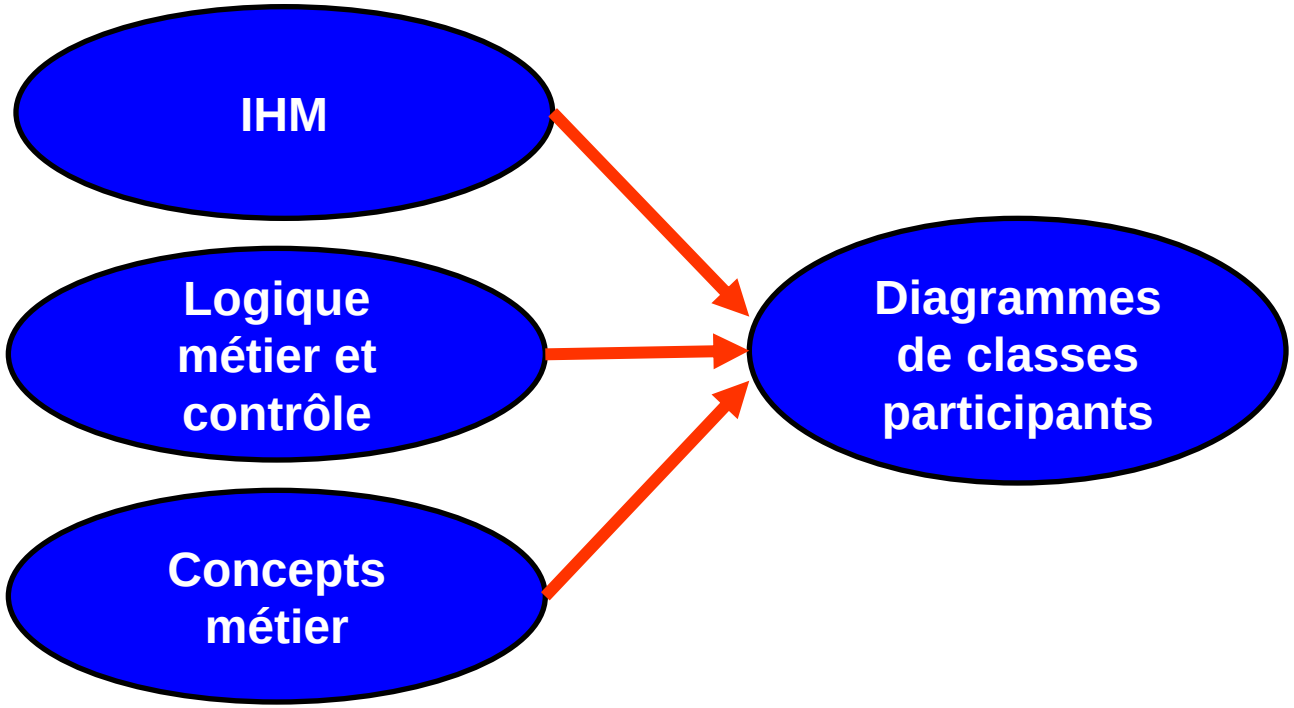
Cas d'utilisations

Conception ➔ COMMENT ?

Diagrammes de classes

Il s'agit de transformer le besoin métier exprimé, en une architecture (orientée objet).

Approche


IHM
Logique métier et contrôle
Concepts métier
Diagrammes de classes participants

Définitions (1)

La notion d'**Objet**

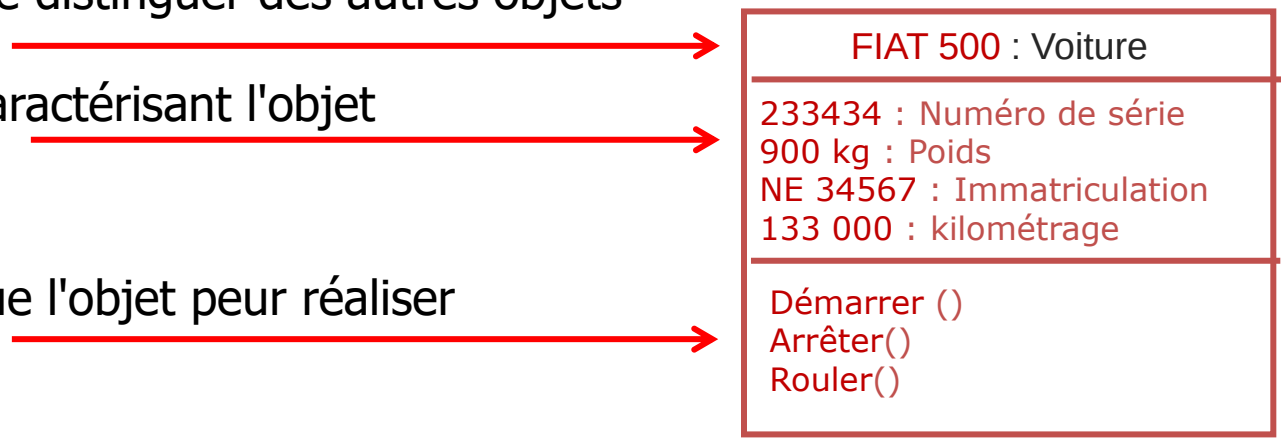
Abstraction du monde réel, sous forme de variables d'états (attributs) et d'opérations possibles (méthodes)

P.ex. une personne, un compte bancaire, une voiture

Identité : permet de le distinguer des autres objets

Attributs : données caractérisant l'objet

Méthodes : actions que l'objet peur réaliser



Définitions (2)

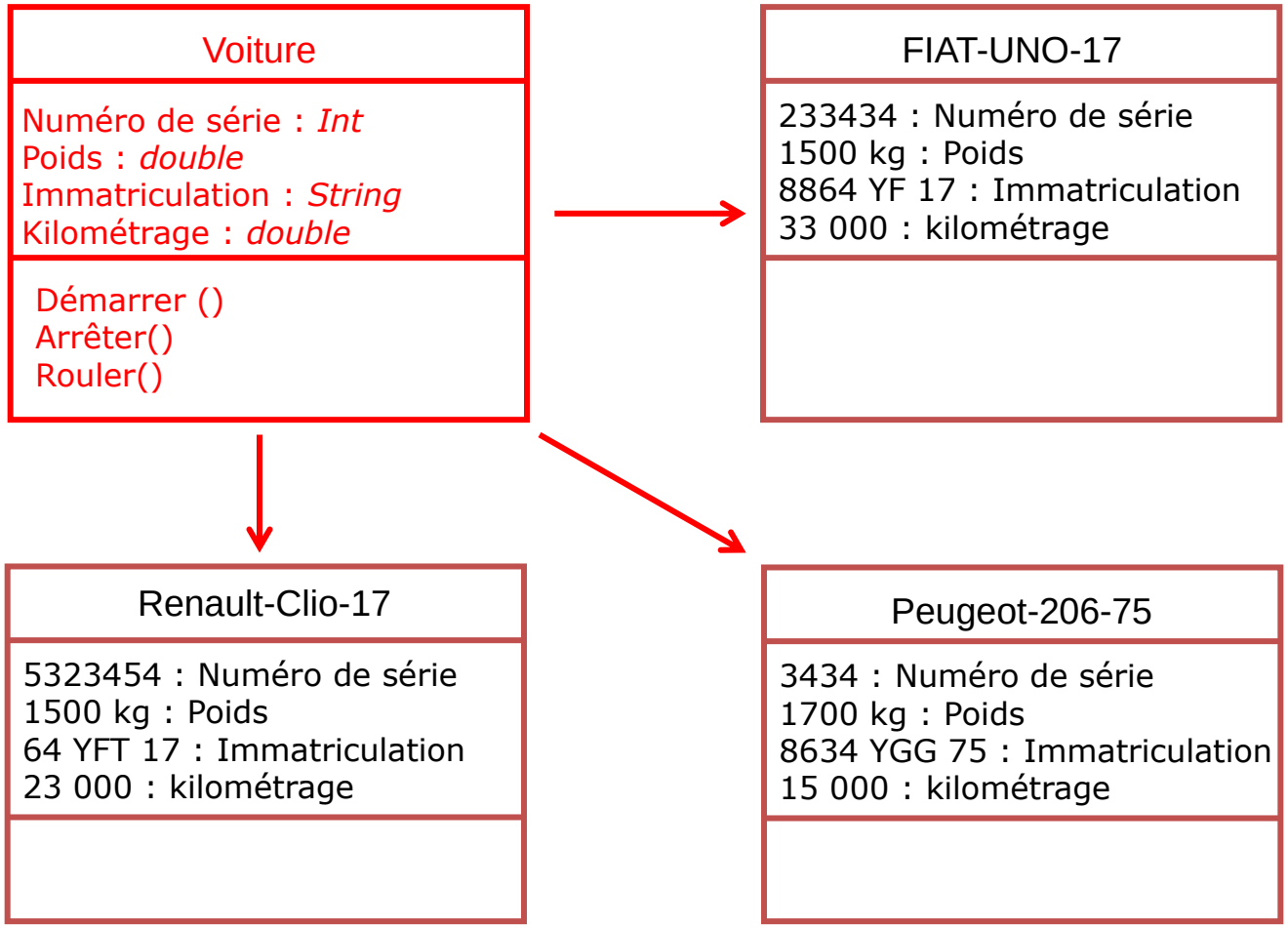
Notion de classe

Structure d'un objet = déclaration des entités qui le composent

Un objet est une instance d'une classe

Un object est issu d'une classe, c'est le produit qui sort du moule (la classe)

Classes et objets



Classe : nom

Compte

Nom

```
class Compte
{



}
```

Classe : propriétés, états

Compte

numero : string

solde : double

Attributs

```
class Compte
{

  string numero;
  double solde;




}
```

Classe : opérations, comportement

Compte

numero : string

solde : double

debiter (double) : void

crediter (double) : void

getSolde () : double

calculerNouveauNumero () : string

Méthodes

```
class Compte
{

  string numero;
  double solde;

  void debiter(double) {}
  void crediter(double) {}
  double getSolde() {}

  static void calculerNouveauNumero() {}
}
```

Classe : Visibilités

Compte

- numero : string
- solde : double

+ debiter (double) : void
+ crediter (double) : void
+ getSolde () : double
# calculerNouveauNumero () : string

+ : public
- : private
: protected

```
class Compte
{
private:
   string numero;
   double solde;
public:
   void debiter(double) {}
   void crediter(double) {}
   double getSolde() {}
protected:
   static void calculerNouveauNumero() {}
}
```

Relations

3 types

Association

Spécialisation 

Dépendance 

Relations : association

Bidirectionnelle

Permet l'échange de messages

Relations : association

```
class Person
{
  public:
    string FirstName;
    string LastName;
    List<Car> cars;
}
```

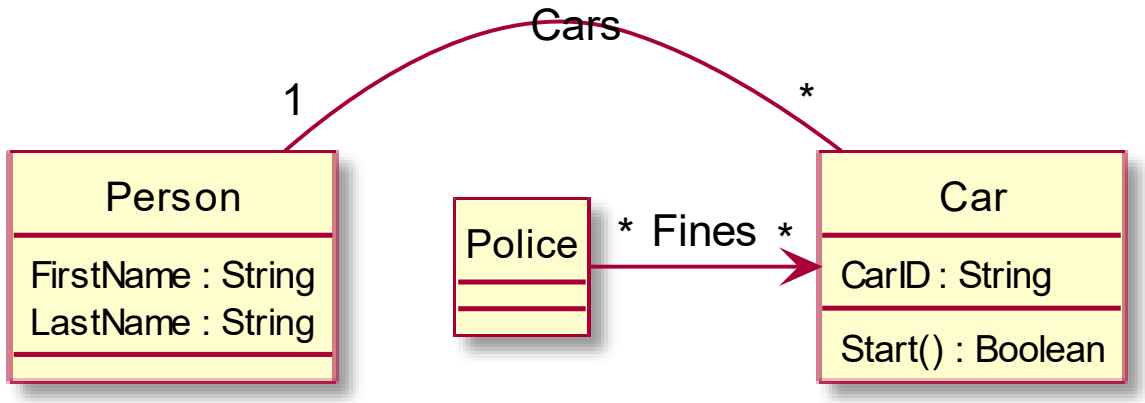
```
class Car
{
  public:
    string carID;
    Person *pOwner;
    void Start();
}
```

```
class Police
{
   List<Car> Fines;
}
```

Cardinalité

Multiplicité

Nb min et max d'instances dans la relation



0..1 : optionnel

1 : exactement 1

0..* : plusieurs

1..* : plusieurs (min. 1)

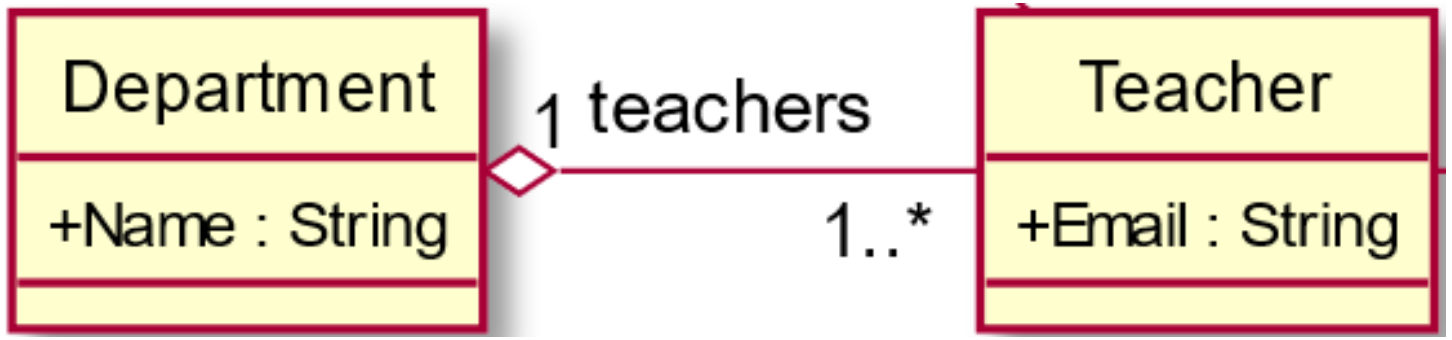
Relations : agrégation

Cas particulier d'association

Couplage plus fort

Relation d'appartenance

L'un peut exister sans l'autre



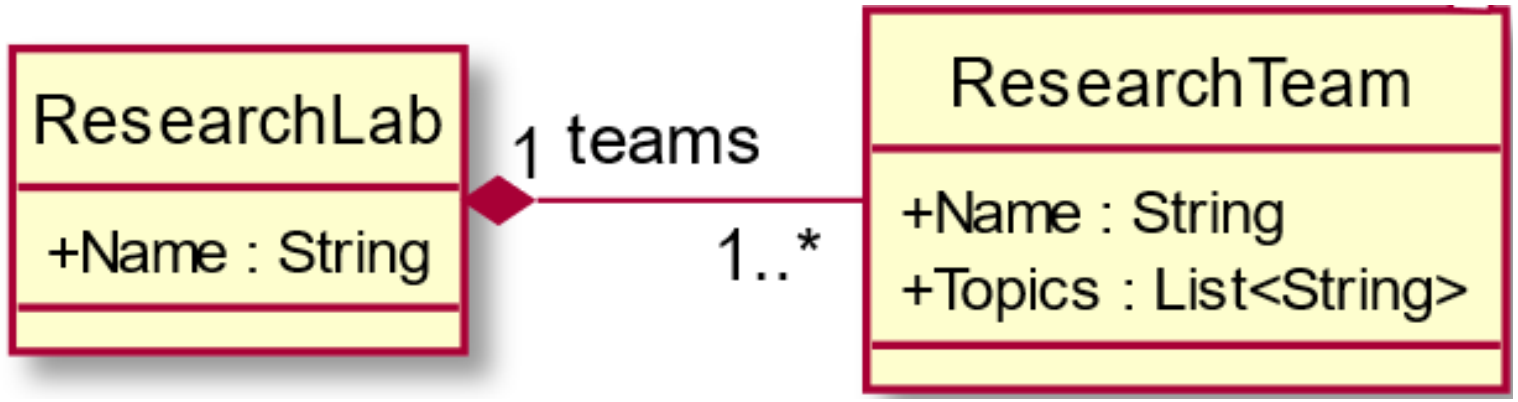
Relations : composition

Cas particulier d'association

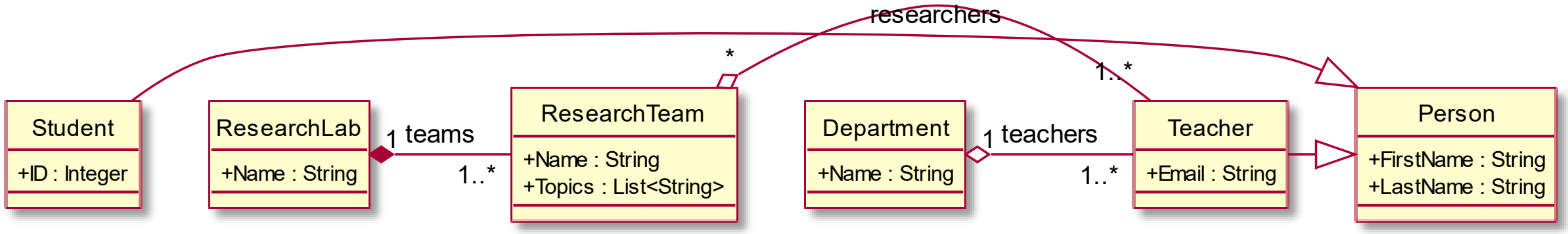
Couplage encore plus fort

Relation d'appartenance

Mais l'un **NE** peut **PAS** exister sans l'autre !



Relations : exemple

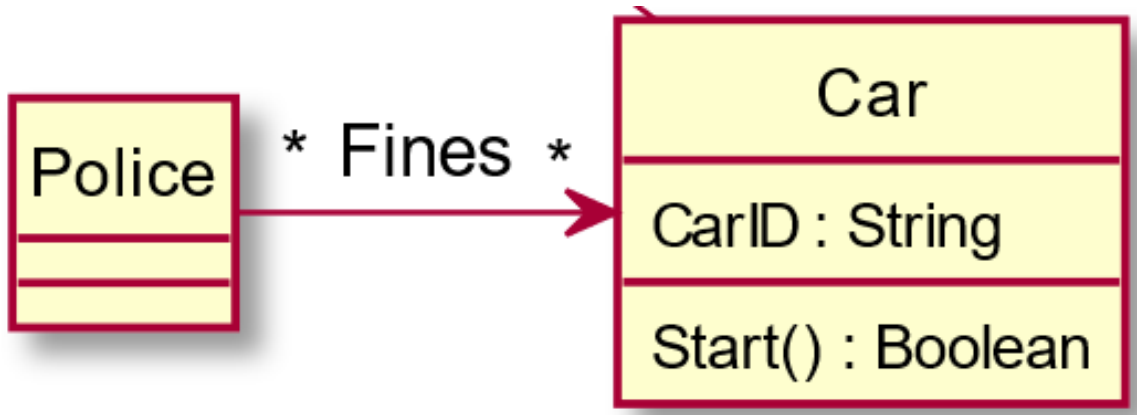


Relations : navigabilité

Une flèche qui restreint le sens de navigation

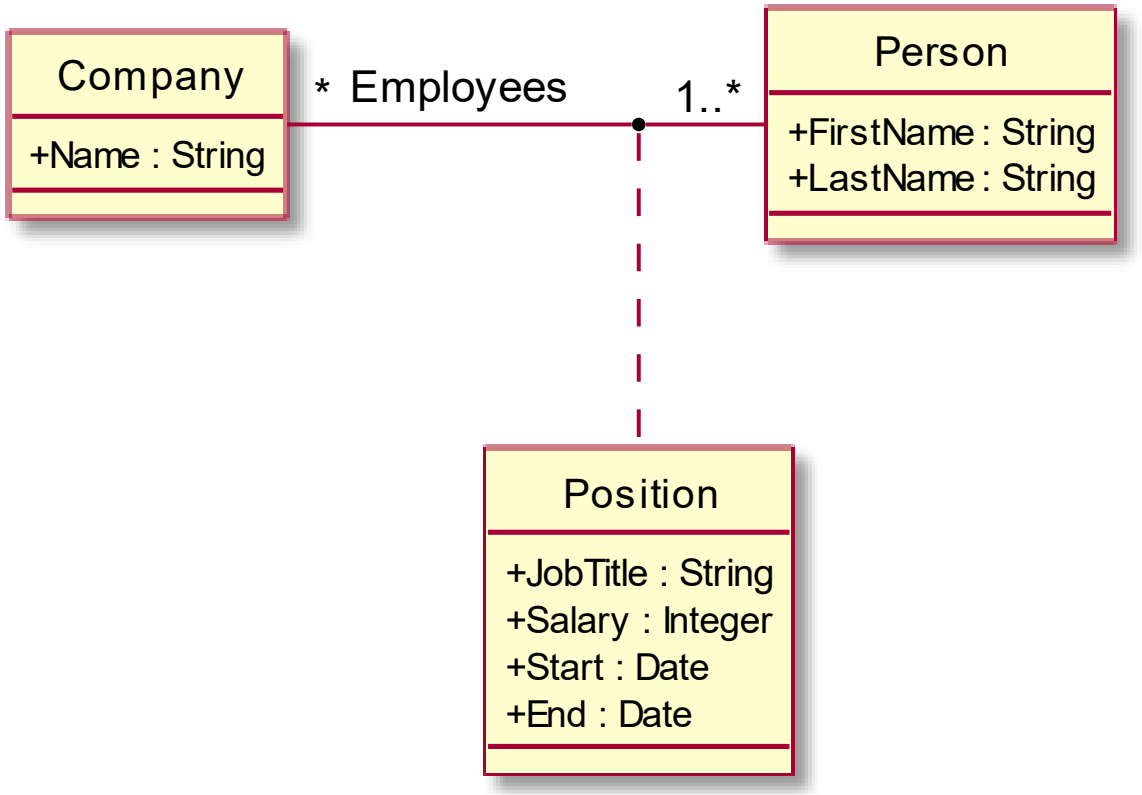
- Eviter les doubles dépendances
- Donner une précédence à une classe

les instances d'une classe ne "connaissent" pas les instances de l'autre. Par ex. la voiture ne connait pas les détails du policier



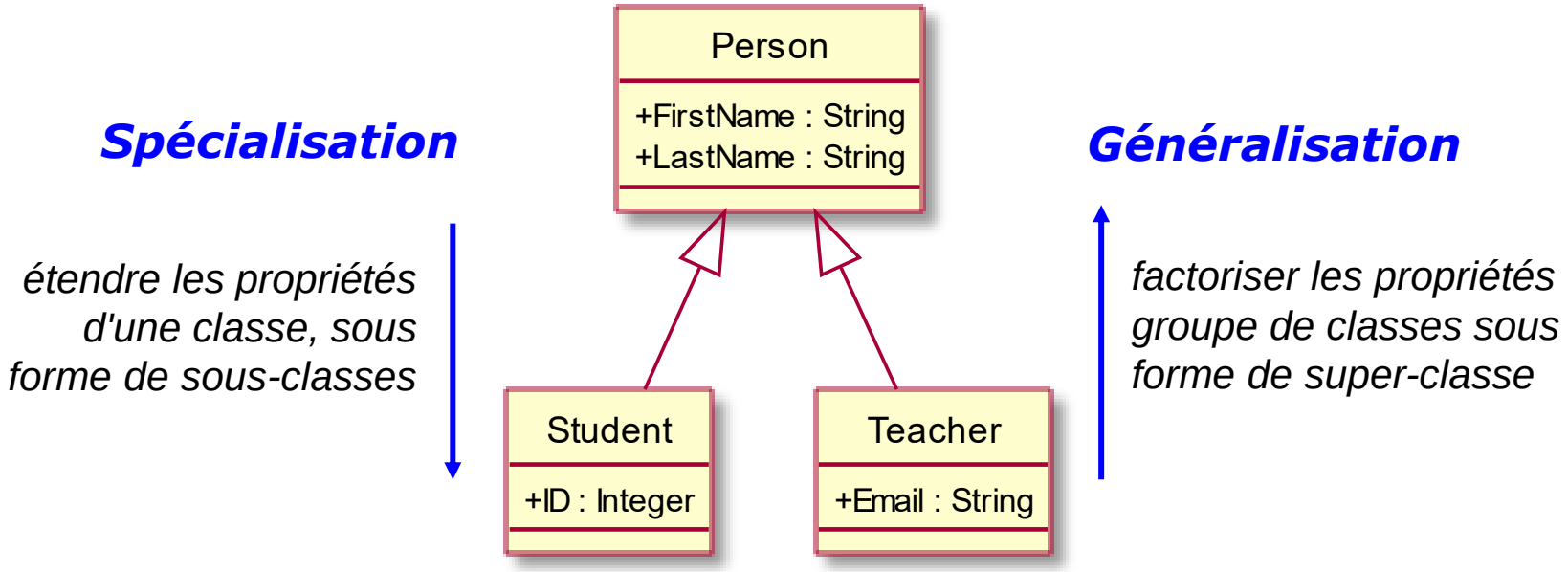
Classe-association

association porteuse d'attributs et/ou d'opérations, représentée comme une classe anonyme associée à l'association



Héritage (principe)

Permet de créer une nouvelle classe à partir d'une classe existante; la classe dérivée contient les attributs et les méthodes de sa superclasse



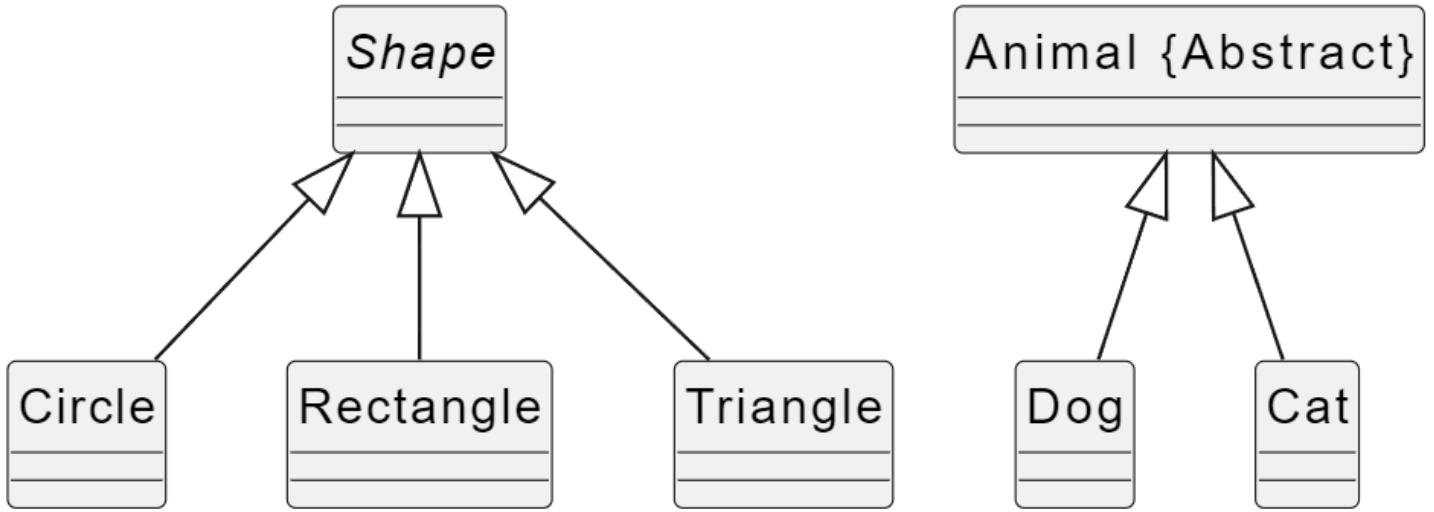
Chaque personne de l'université est identifiée par son nom, prénom
de plus, les étudiants ont un noEtudiant
de plus, les enseignants ont un numéro de téléphone interne

Classes abstraites

Une classe abstraite est simplement une classe qui ne s'instancie pas mais qui représente une pure abstraction afin de factoriser des propriétés.

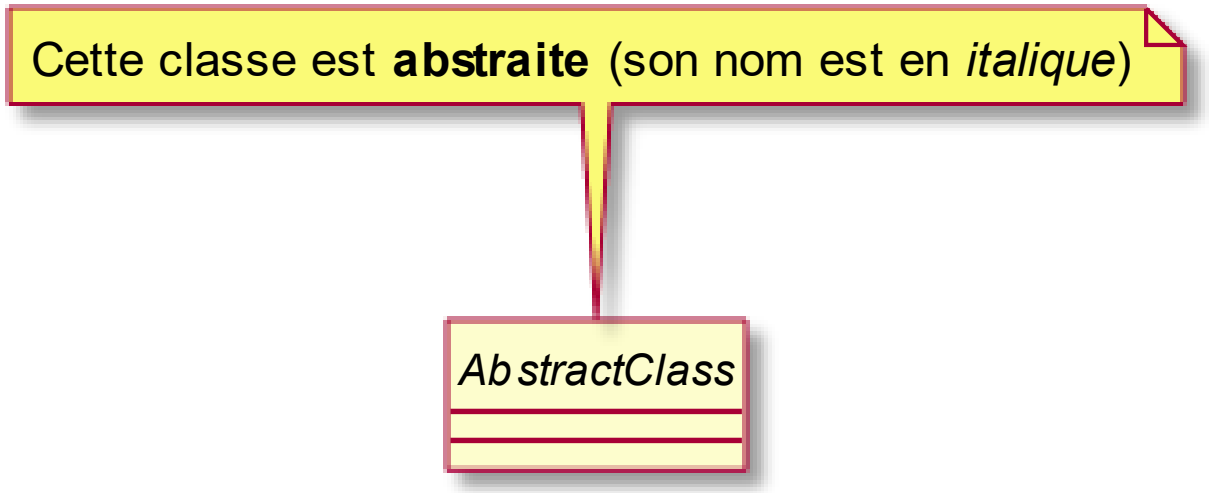
En UML, elle se note en italique ou avec **{abstract}**

(mot-clé abstract en Java, méthode pure virtual, =0, en C++, …)



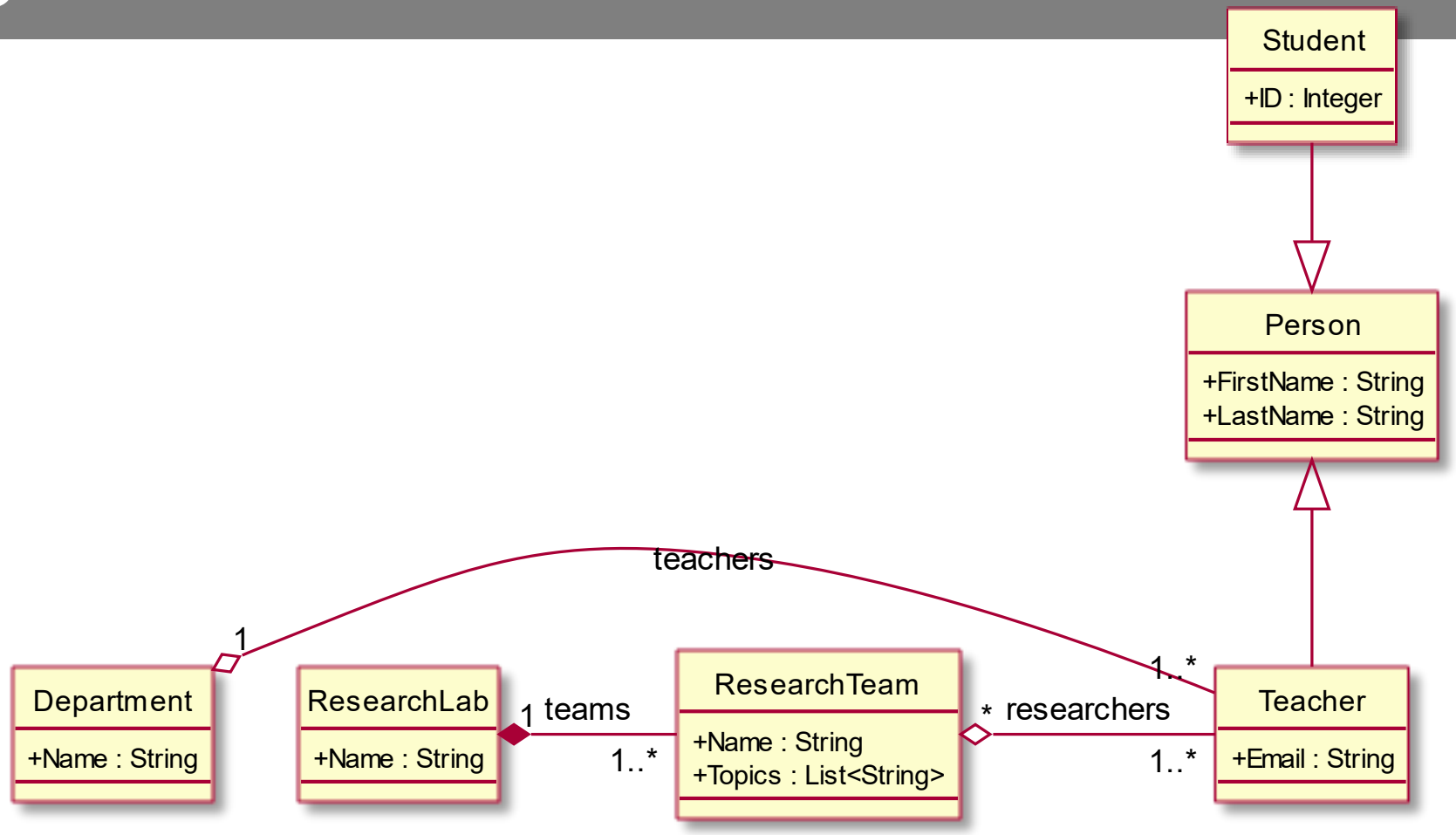
Les notes

Ajout de contraintes, de commentaires, etc…



Texte
URL
Lien à des documents
….

Diagramme de classes



Démarche de conception

Analyse du domaine / vocabulaire métier

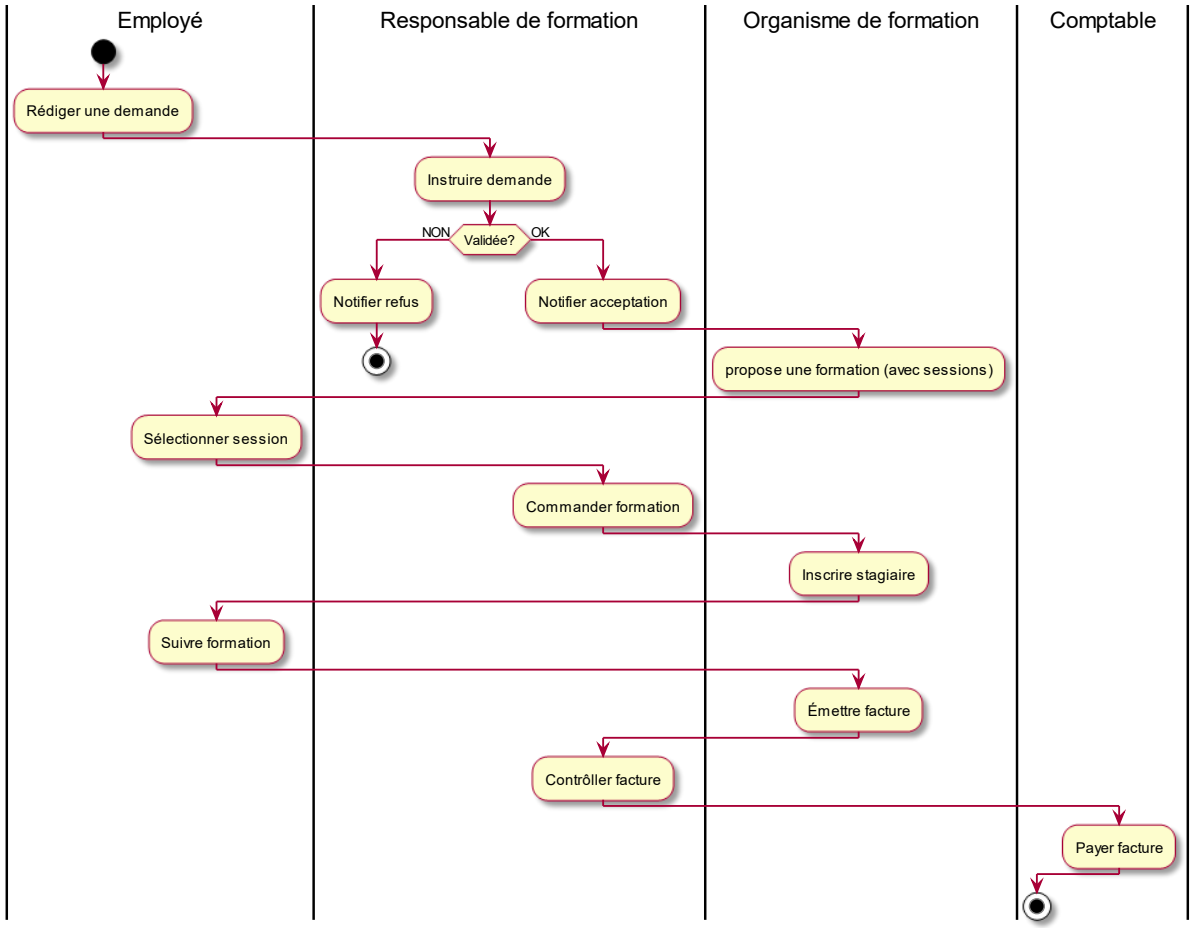
Identifier et analyser les «noms» dans les phrases

→ **classes et attributs**

Identifier et analyser les «verbes» dans les phrases

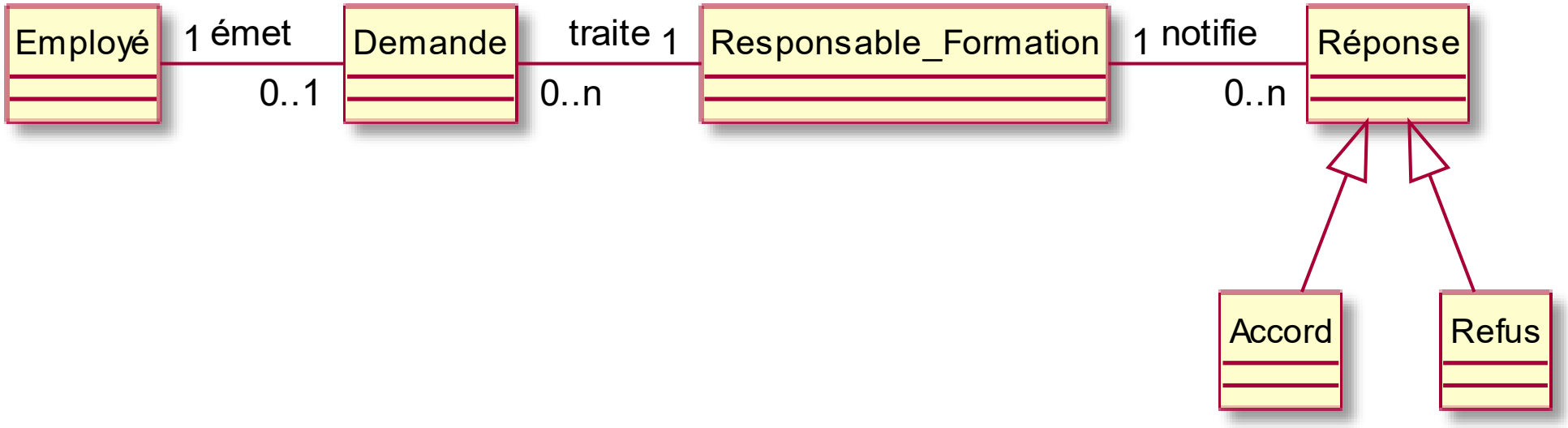
→ **opérations**

Diagramme d'activité



Etapes 1 & 2

1. 1 employé peut émettre une demande de formation.
2. 1 responsable de formation traite les demandes.
3. 1 responsable de formation notifie la réponse (accord ou refus)



Etapes 3 & 4

3. 1 organisme de formation recherche une formation (dans un catalogue) qui correspond.
4. Il notifie l'employé et lui propose une liste des prochaines sessions.
5. l'employé choisit puis participe à la session.

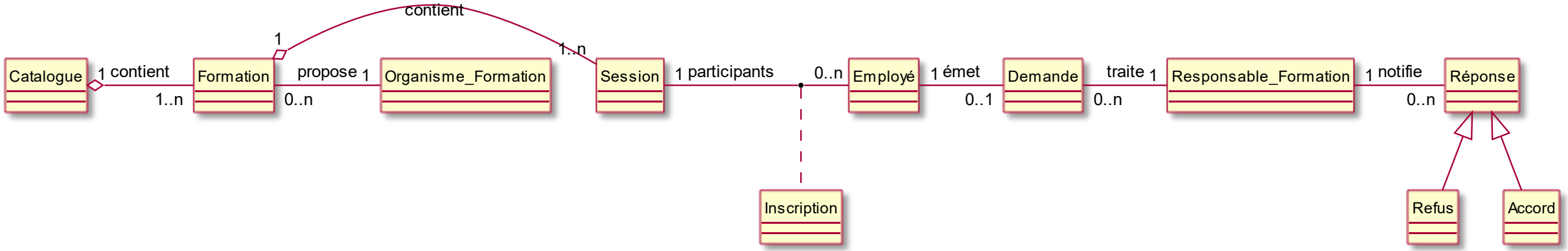
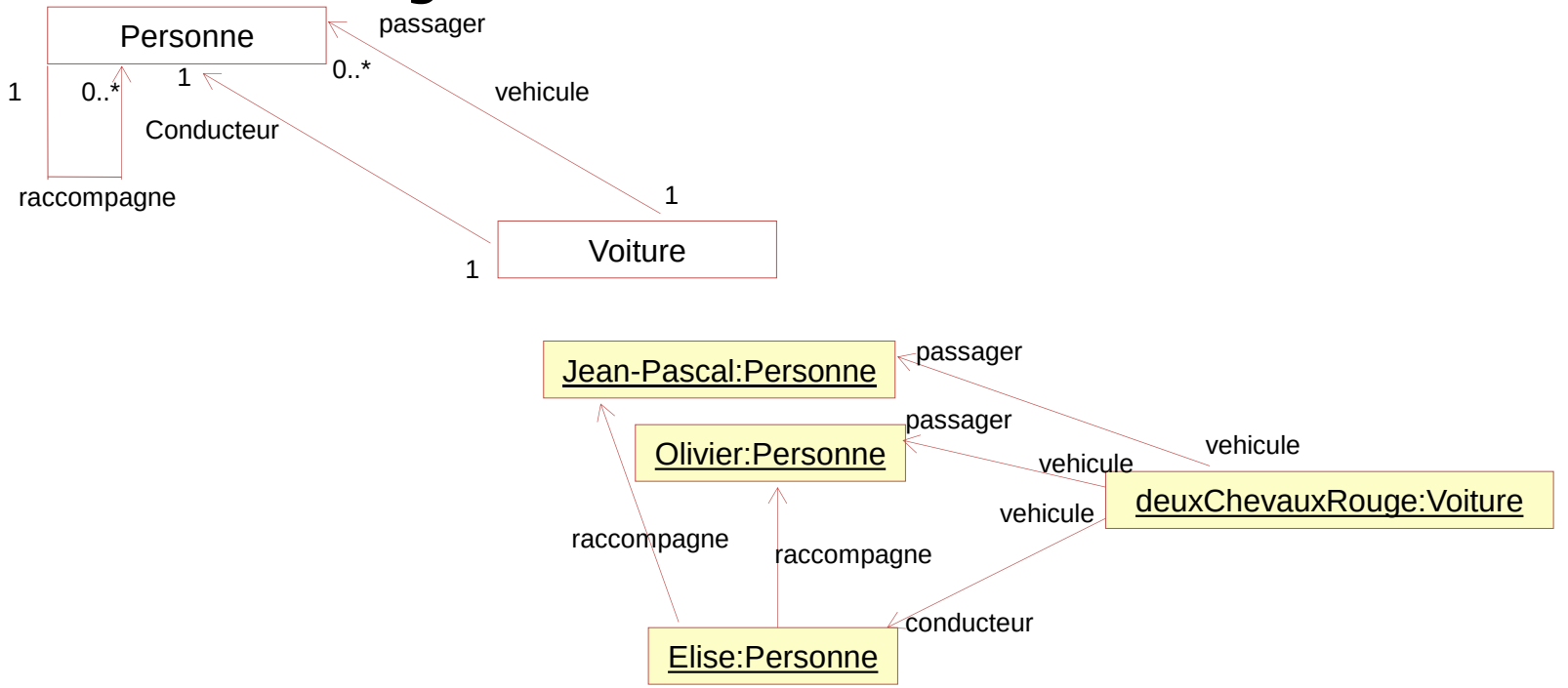


Diagramme d'objets

Montre les relations entre instances de classes. Instantané, photo d'un sous-ensemble des classes (plus simple)

permet de vérifier un diagramme de classe selon différents cas



Exercice

Faire le diagramme de classes de votre P2.

Identifier et ajouter (ou inventer si besoin) des **relations de composition et d'agrégation**