ENSIAS

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

M.2.5.1. Paradigme Objet

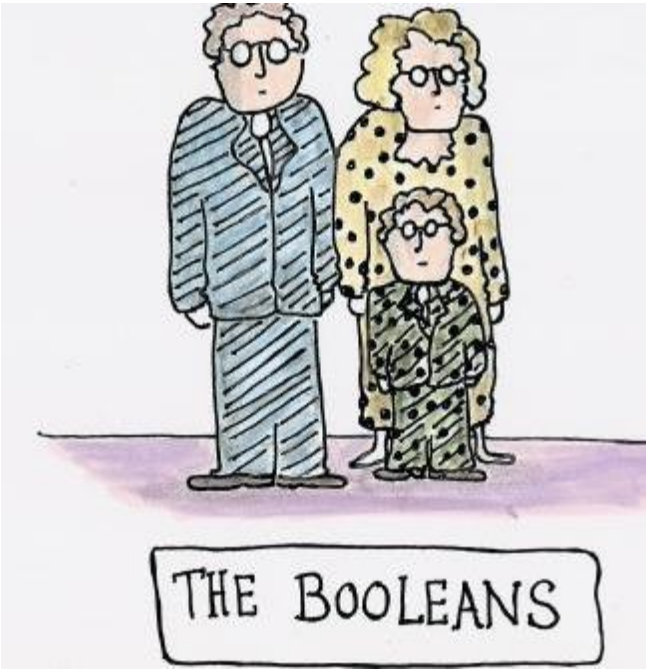
ENSIAS

Plan

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

- Heritage
- Polymorphisme
- Liaison dynamique
- Relations entre classes

Héritage



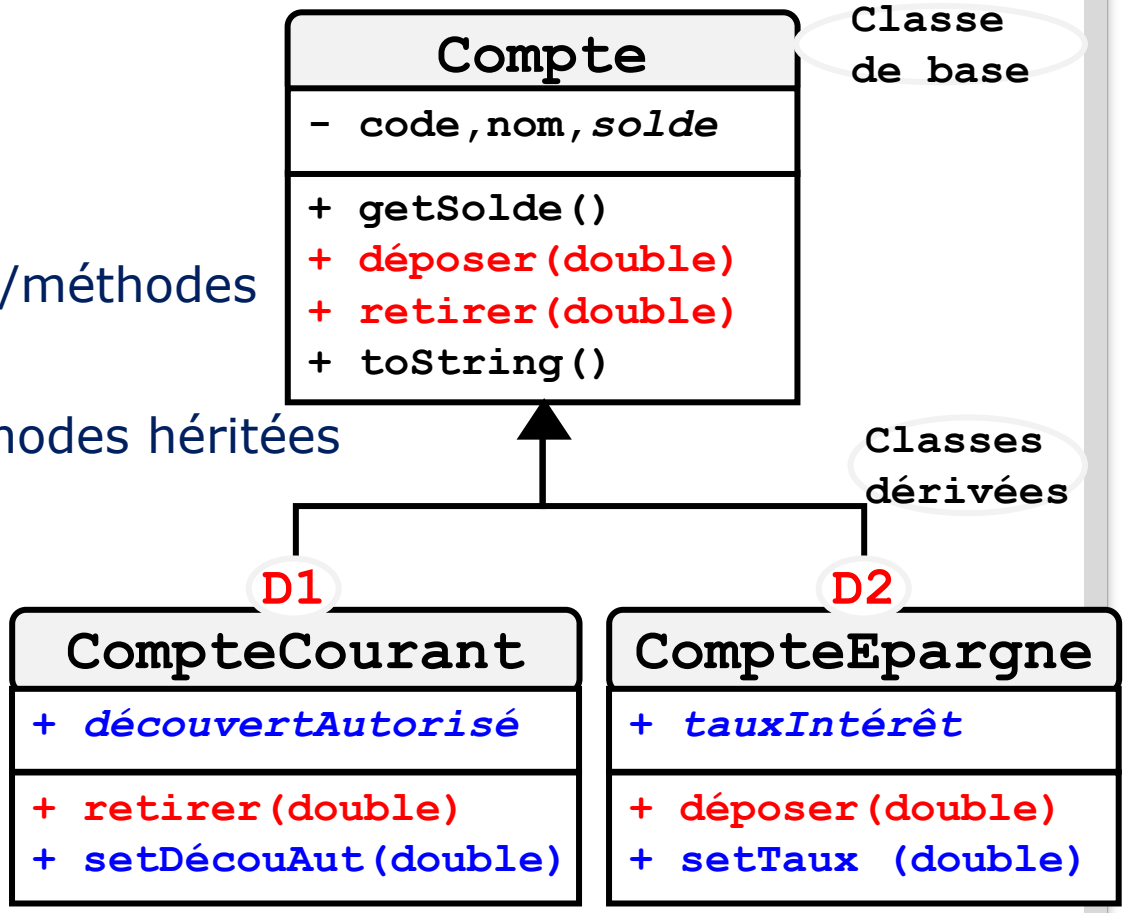
Héritage

➔ Spécialisation

1. **Enrichissement :**
   - Nouveaux attributs/méthodes
2. **Substitution :**
   - Nouvelles déf. méthodes héritées

➔ Intérêt

- Réutilisation
  - Du code
  - De l'expérience
- Eviter la redondance
  - Codes de `D1` et `D2` se répètent
  - Correspondent à 100% aux besoins



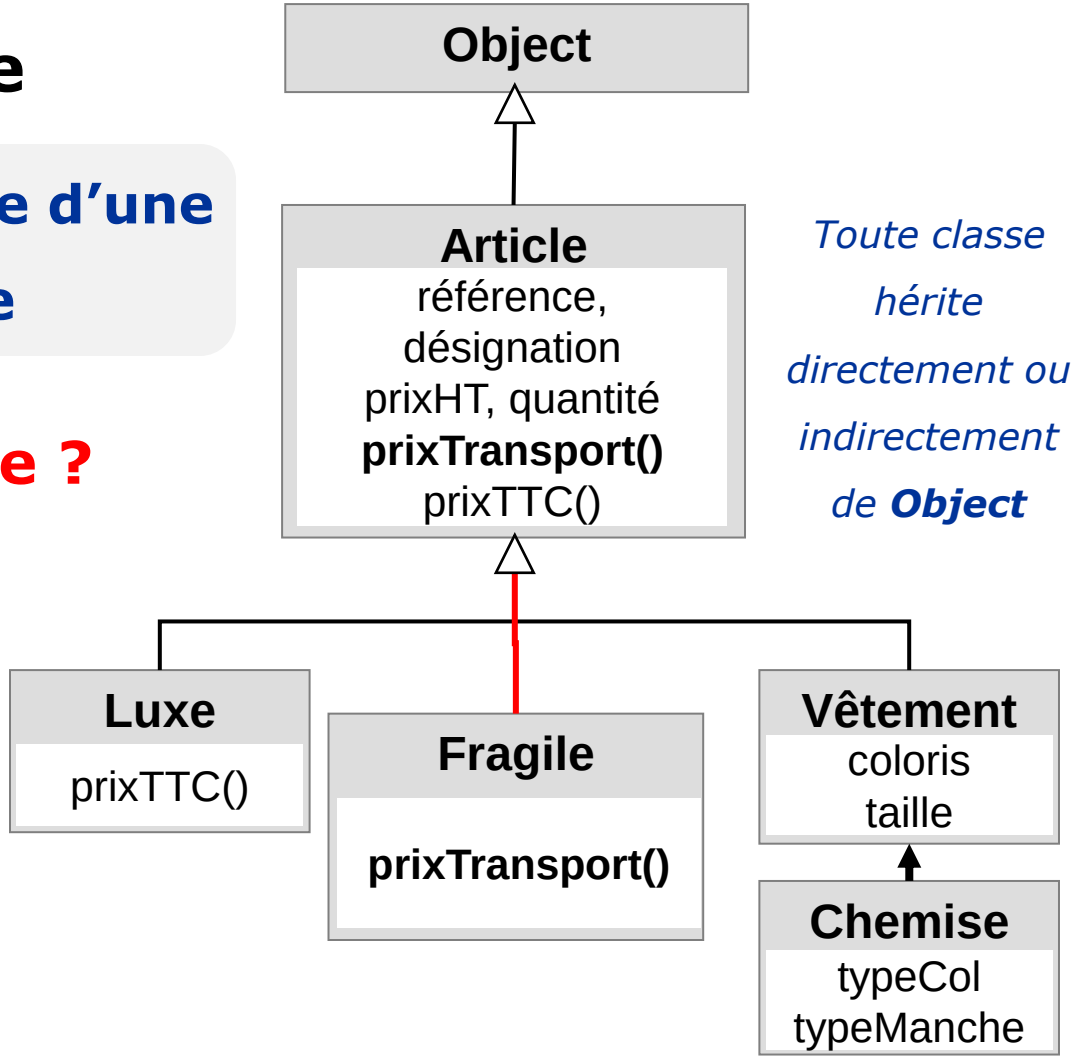
Héritage simple

➔ Arbre d'héritage

Toute classe hérite d'une seule super-classe

Choix de méthode ?

prixTransport() ?

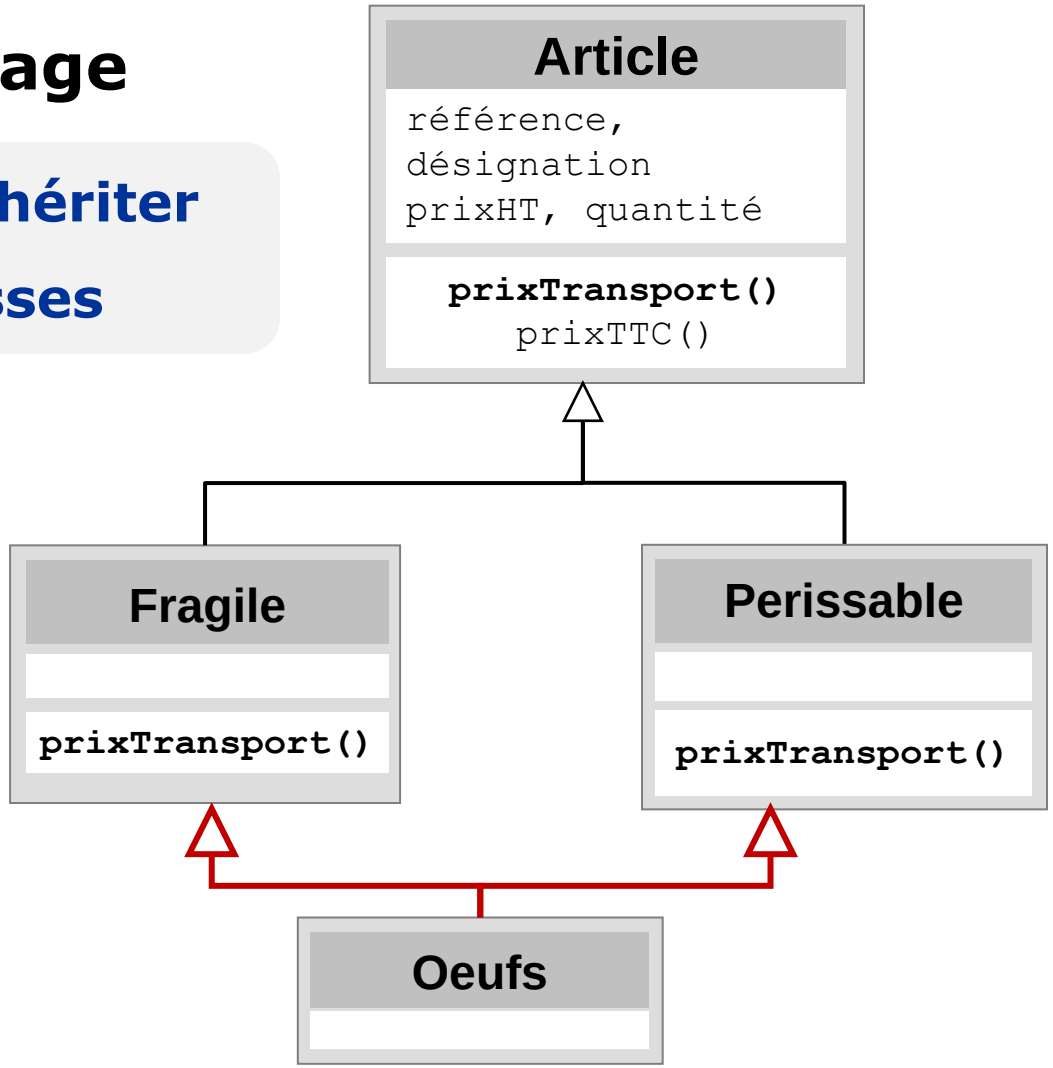


*Toute classe hérite directement ou indirectement de **Object***

Héritage multiple

➔ Graphe d'héritage

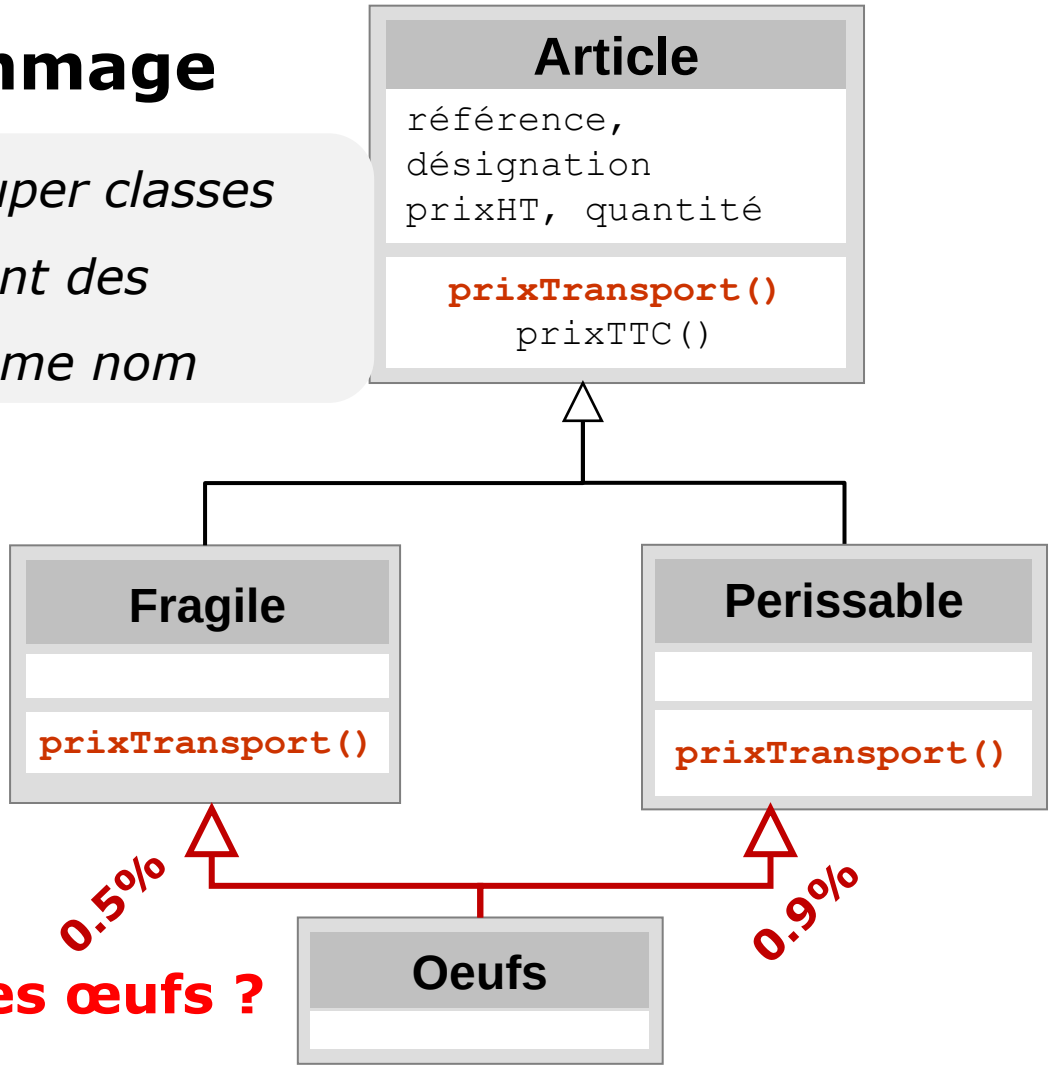
Une classe peut hériter de plusieurs classes



Héritage multiple

1. Conflit de nommage

Au moins deux des super classes d'une classe définissent des propriétés avec le même nom



Prix de transport des œufs ?

Héritage multiple

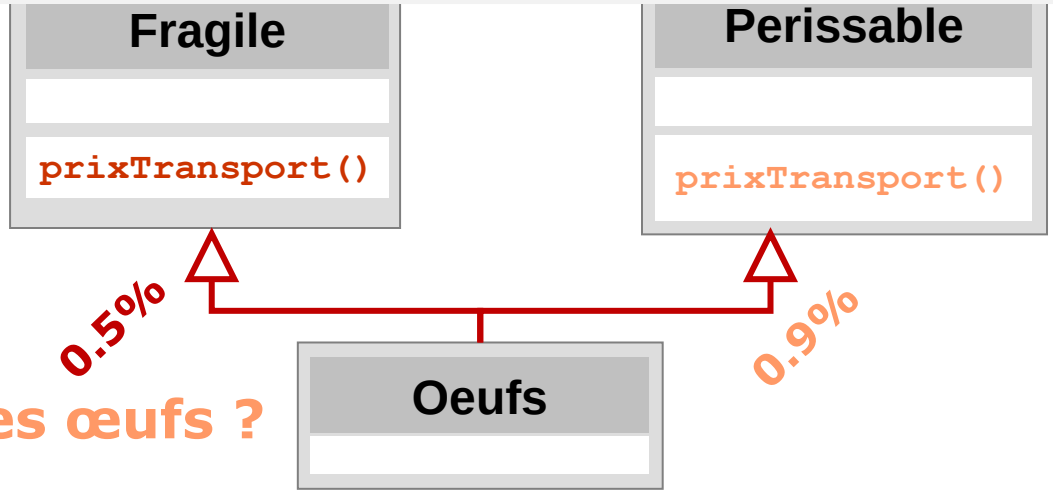
1. Conflit de nommage

➔ **Solutions**

a. Fournir un ordre des super-classes.
Définir quelle propriété sera accessible. Les autres «cachées»

Pas très pratique du fait qu'elle introduit des conséquences implicites selon l'ordre dans lequel les classes héritent les unes des autres

```
class Oeufs :
   public Fragile,
   public Perissable
```



Prix de transport des œufs ?

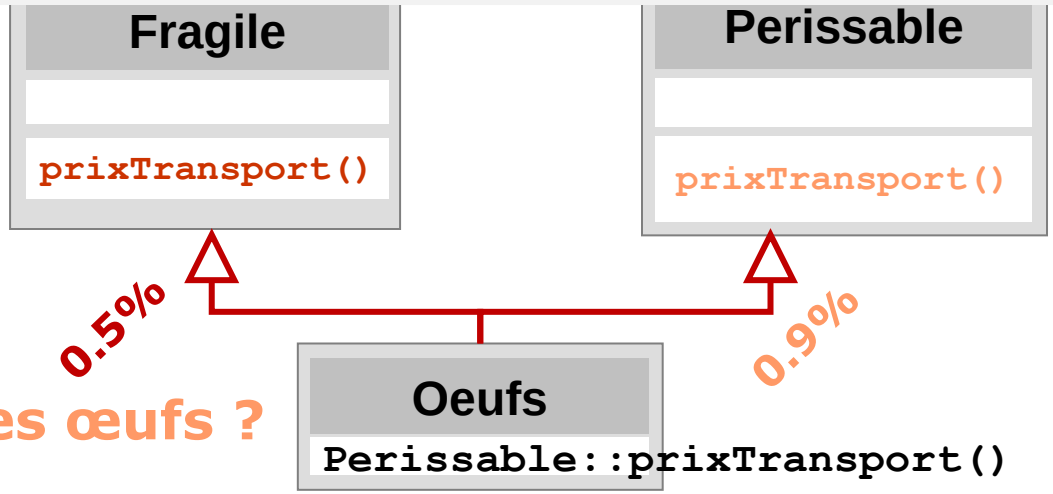
Héritage multiple

1. Conflit de nommage

➔ **Solutions**

b. La sous-classe fournit une propriété avec le nom et définit comment utiliser celles de ses super-classes

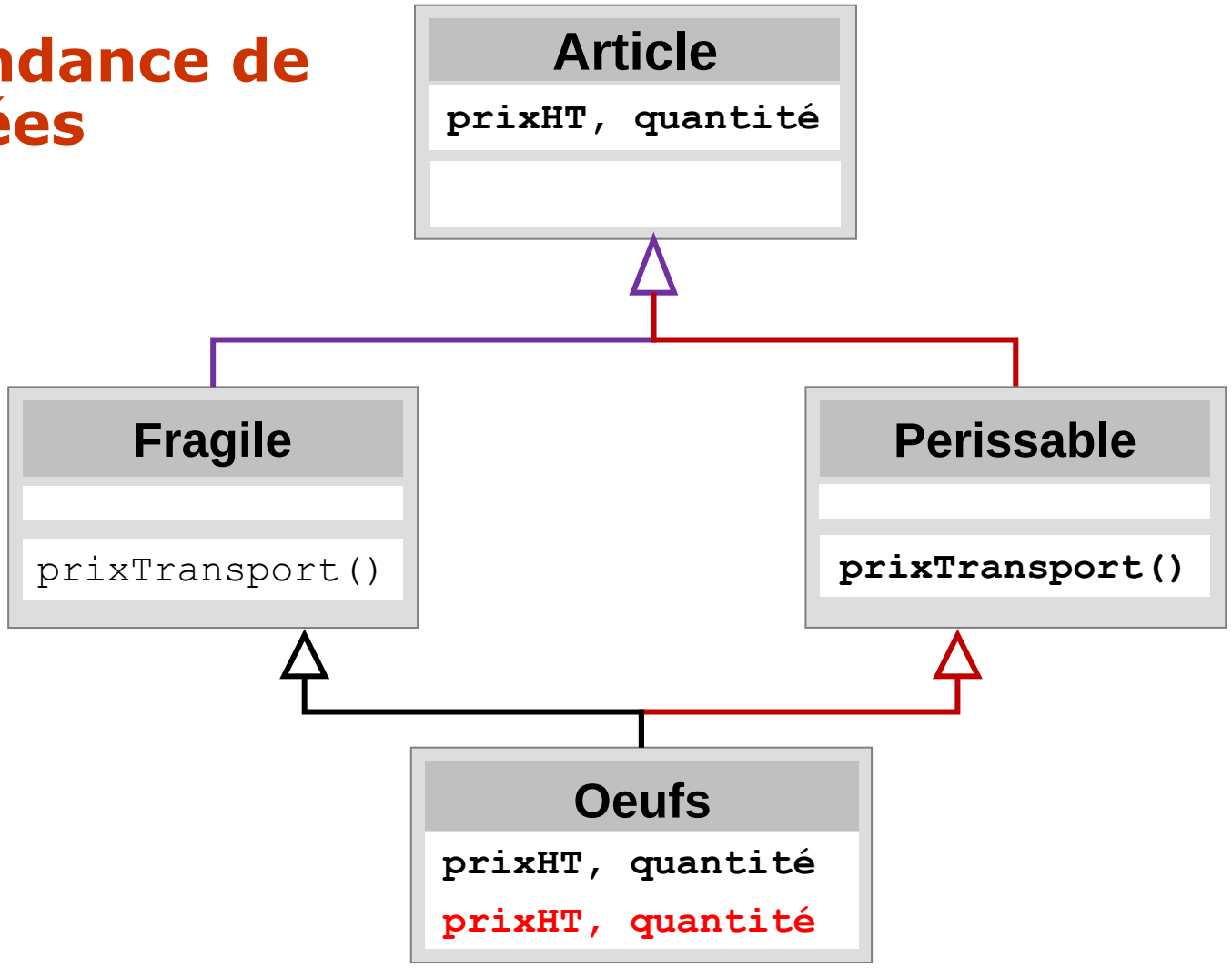
*Les sous-classes doivent **explicitement redéfinir les propriétés** qui sont impliqués dans un conflit de noms*



Prix de transport des œufs ?

Héritage multiple

2. Redondance de données



Polymorphisme

➔ Capacité pour une entité de prendre plusieurs formes

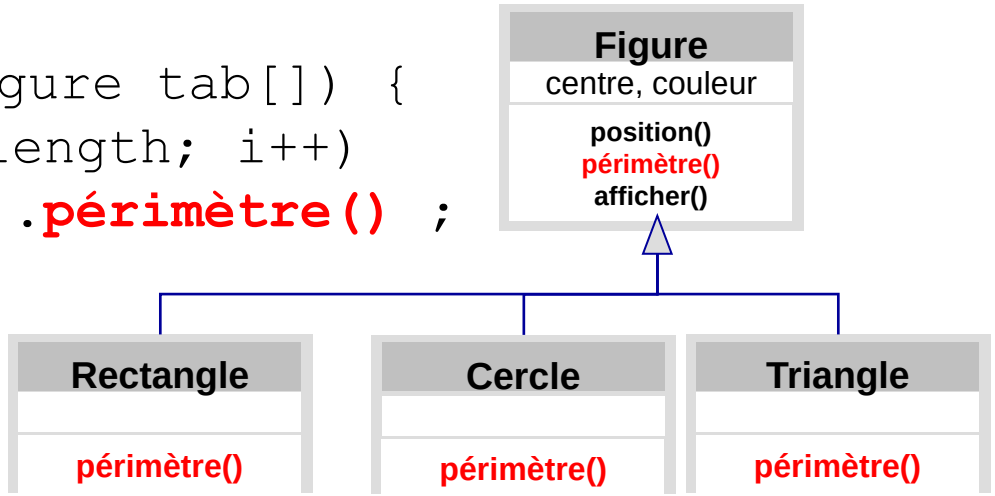
Fonction polymorphe

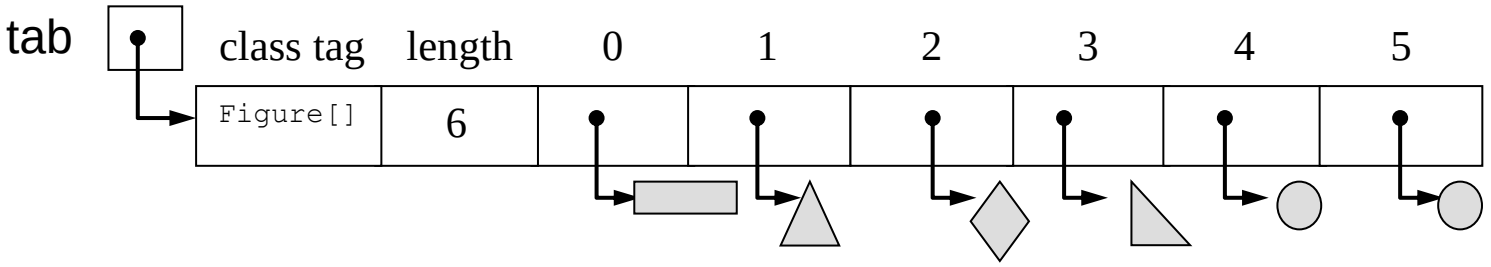
➔ Même nom de méthode mais codes différents

Polymorphisme

➔ Exemple

```
class Périmètre {
   float périmètre( Figure tab[]) {
       for(i=0; i<tab.length; i++)
          s = s+ tab[i].périmètre() ;
       return s;
   }
}
```



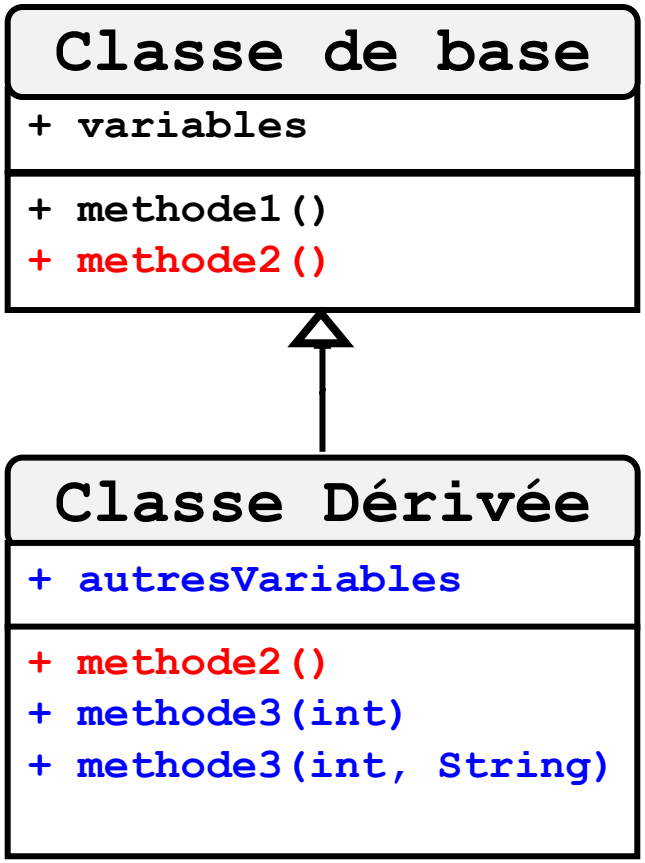


Polymorphisme

- méthode1()
  - réutilisée, héritée
- méthode2()
  - adaptée, redéfinie

  <Polymorphisme d'héritage>
- méthode3()
  - adaptée, surchargée

  <Polymorphisme paramétrique>



Polymorphisme

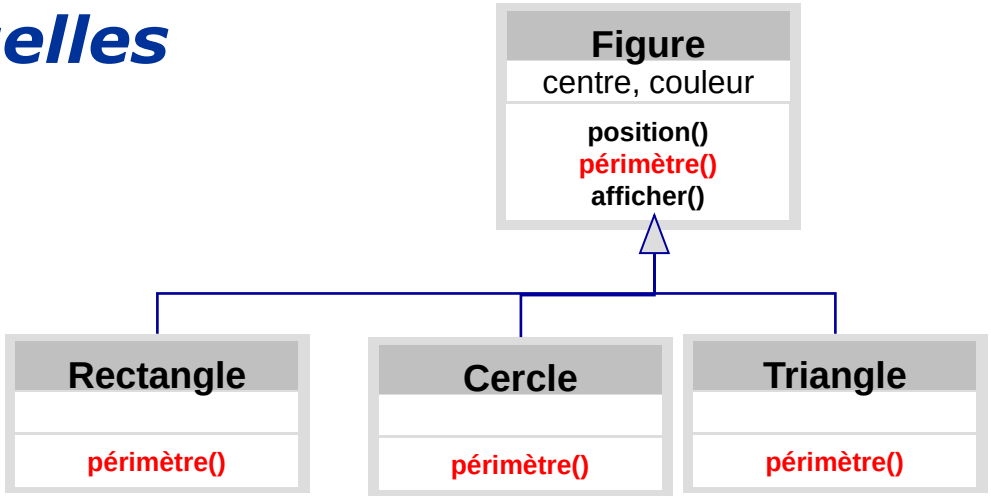
Liaison dynamique

➔ ***Mise en œuvre du polymorphisme***

Code lié à la méthode à l'exécution

➔ ***Méthodes virtuelles***

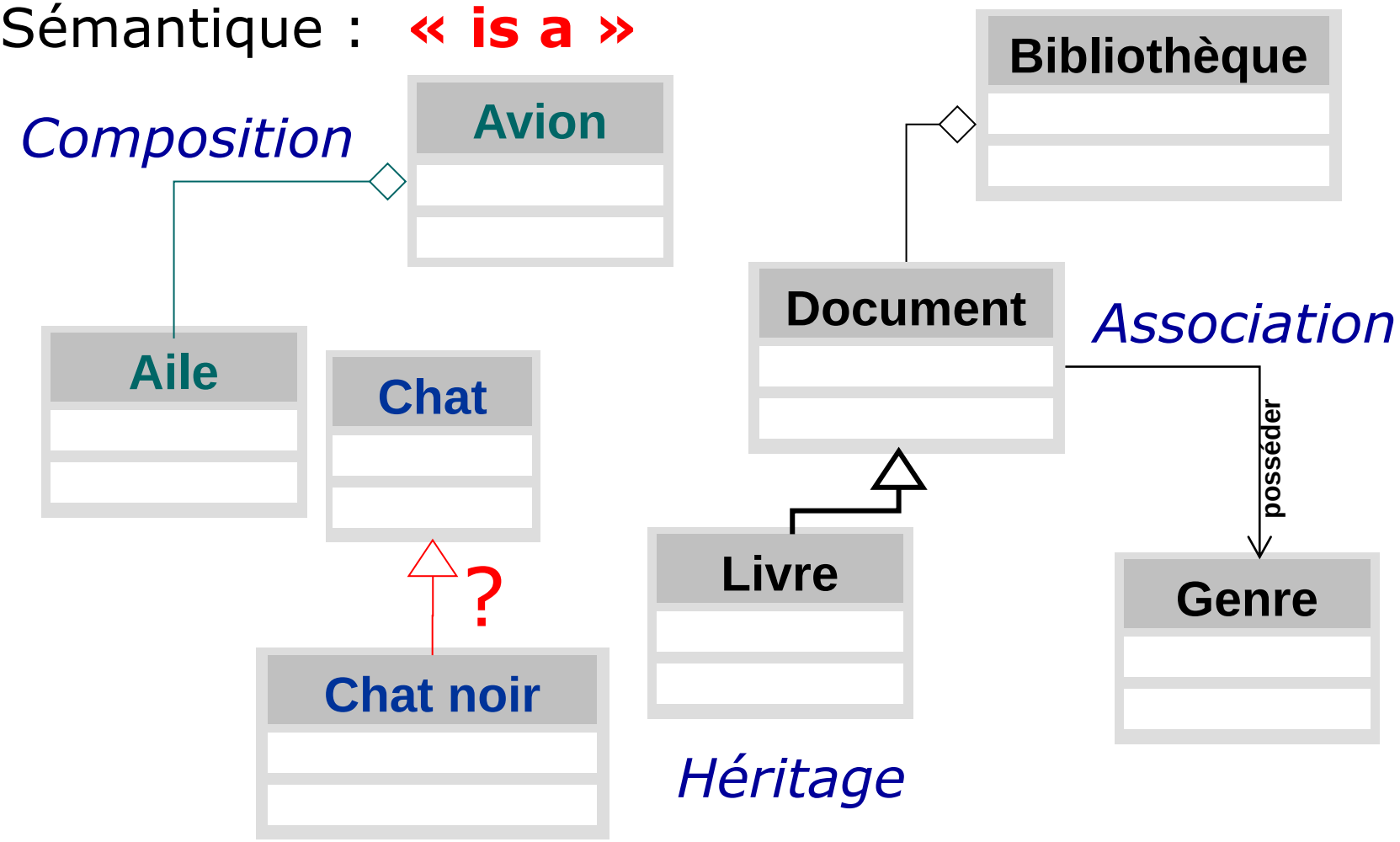
C++, Eiffel, Java



Relations entre classes

Relation d'héritage

Sémantique : « is a »



Relation d'amitié

Héritage : Hiérarchie familiale - *Partager du code*

Paquetage : Amitié - *Partager des connaissances*

```
package ma.ensias.banque;
public class Compte {
  ...
}
```

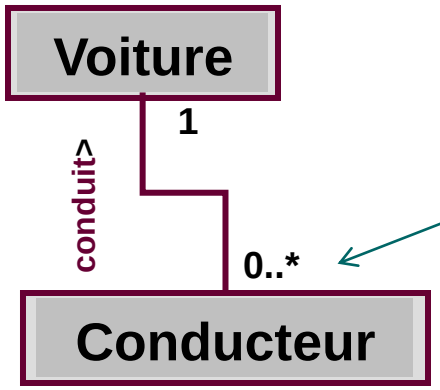
→ **Regroupement logique des classes et <interfaces>**

Swing	Java 2D

→ **Contrôle d'accès**

Relation d'association

«uses > » ➔ **Les objets sont sémantiquement liés**



Association

Cardinalité

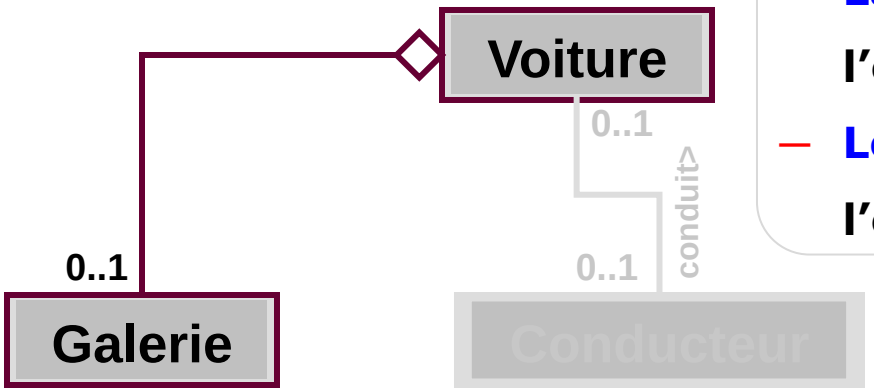
Pour une Voiture

- **elle peut avoir zéro ou plusieurs Conducteurs (concurrents) associés**

Relation d'agrégation

«is part of» ➔ **Les cycles de vie sont indépendants**

➔ **But : Meilleure encapsulation**



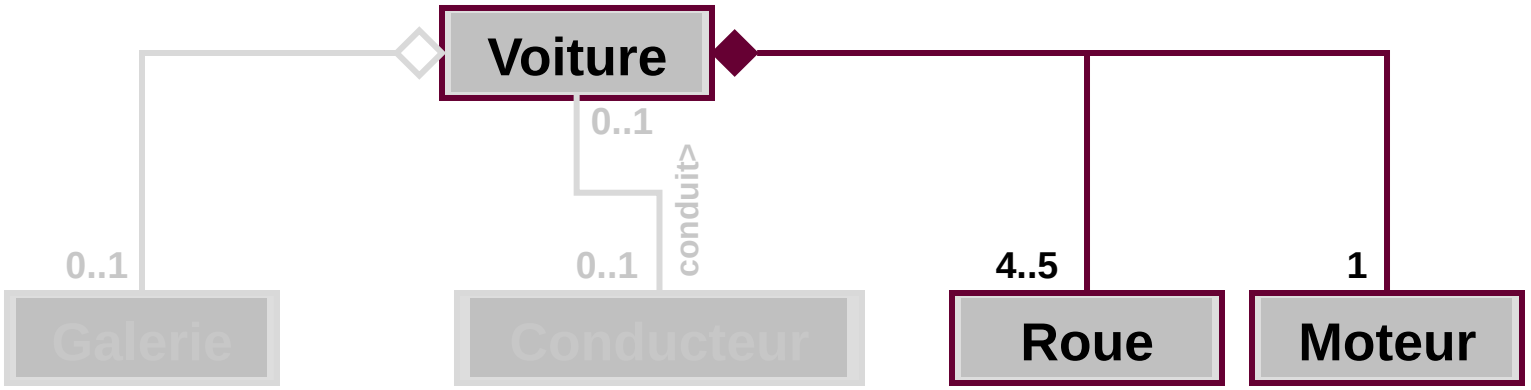
Agrégation *Association*

- La partie peut exister sans l'ensemble
- Le créateur de la partie n'est pas l'ensemble

Relation de composition

«is part of»

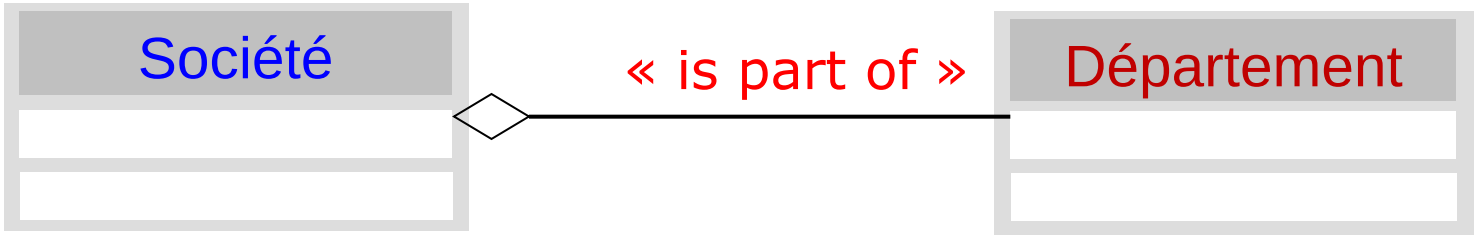
➔ **Agrégation mais avec relation plus forte**



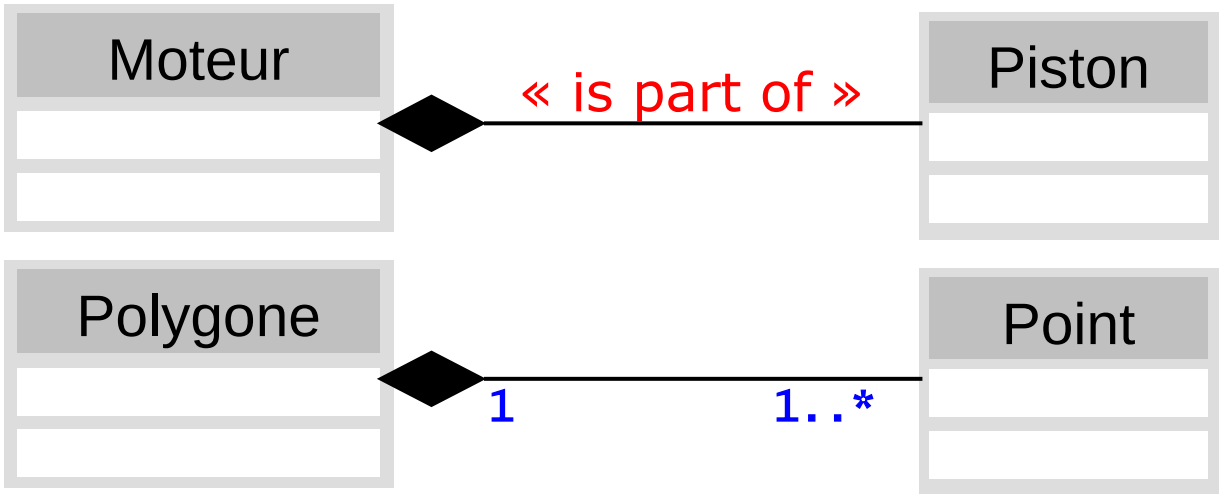
Le `Moteur` dure la vie de la `Voiture`

Exemple

➔ Agrégation



➔ Composition



Exercices d'examen

Le **Parc** informatique d'une entreprise est constitué de **Matériel** (c'est-à-dire **Imprimantes**, **PCs**, …). Chaque matériel est affecté à un ou plusieurs **Employés**. Préciser la relation entre les classes.

Matériel et **PC**		**A**	Association	**B**	Composition	**C** Héritage
Parc et **Matériel**		**D**	Association	**E**	Composition	**F** Héritage
Employé et **Matériel**		**G**	Association	**H**	Composition	**I** Héritage
Parc et **Employé**		**J**	Association	**K**	Composition	**L** Héritage

ENSIAS

Langage Java

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

Java … un aperçu

1995 : L'essor de Java

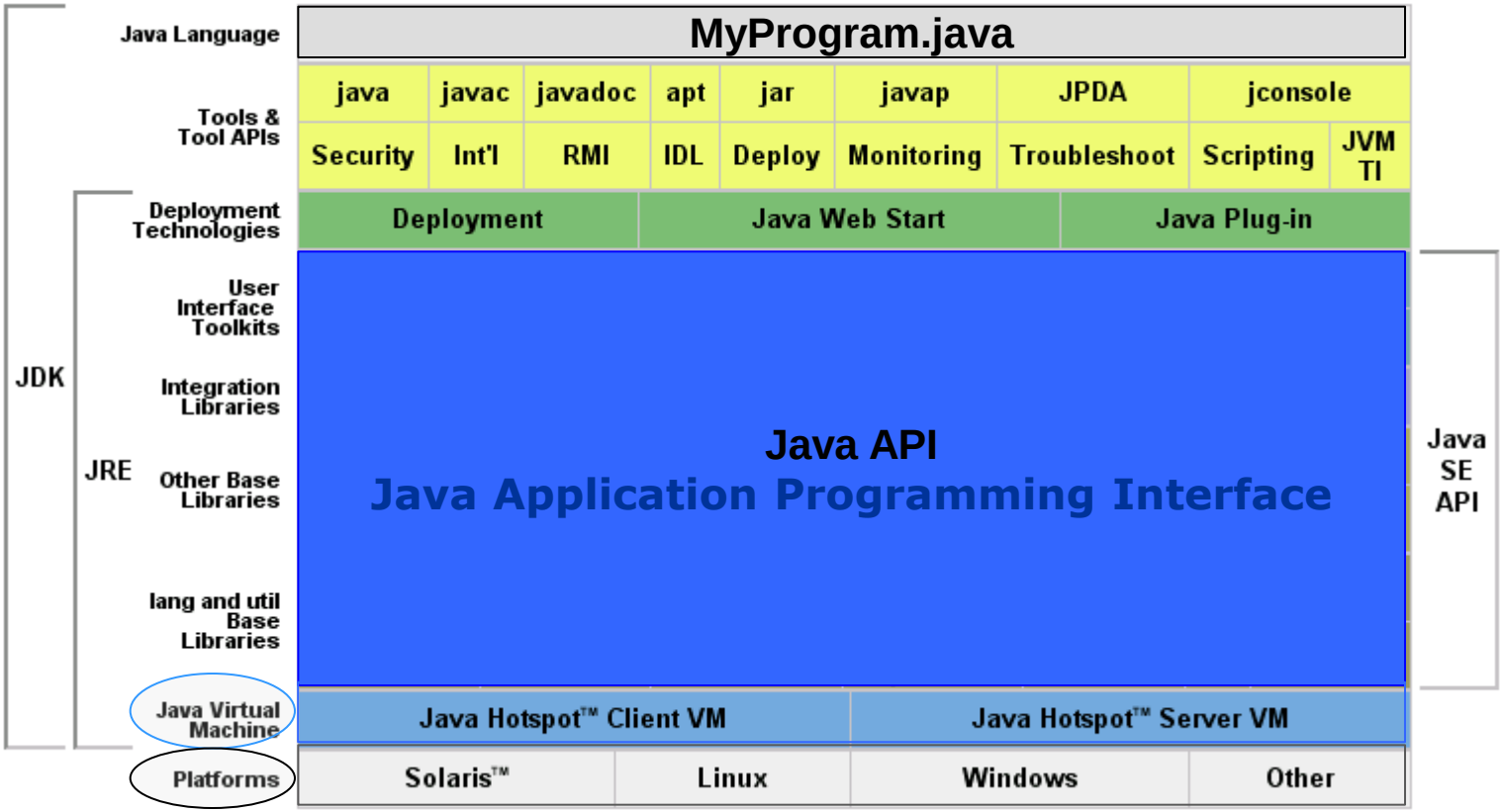
2009 ORACLE® Corporation *achète* Sun microsystems

2010 **Java : Premier langage enseigné**

Richesse fonctionnelle
+ Rigueur

Langage Java

Développer une application consiste à bien choisir les classes/packages à utiliser



ENSIAS

Langage Java

Université Mohammed V de Rabat

```
/**
 * Fichier MonProgramme.java
 **/
class MonProgramme{
    public static void main(String[] args) {
        System.out.println("Bienvenue à vous");
  }
}
```

→ **Création du code source**

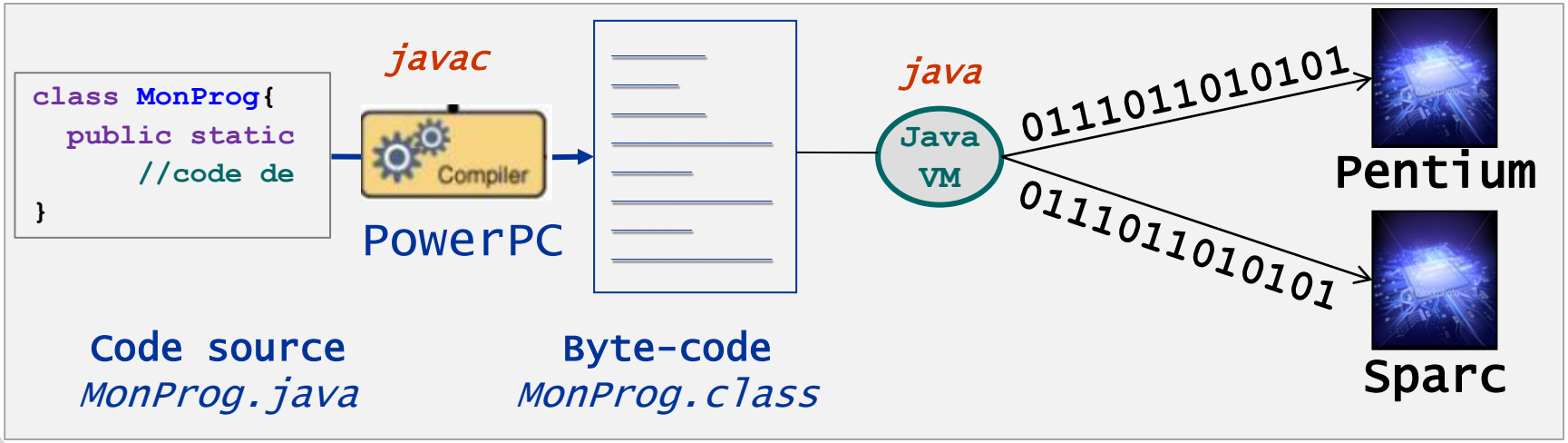
- éditeur de texte, IDE
- A partir des spécifications (UML, …)

Langage Java

→ **Compilation en Byte-Code**
- du code source

→ **Diffusion sur l'architecture cible**
- Transfert du Byte-Code seul
- Réseau, disque, …

→ **Exécution sur la machine cible**
- du Byte-Code
- Machine Virtuelle Java



Autour de main()

- **Itérations**

— **Boucle** « for(initialisation; condition; incrémentation) {… }»

```
for(int i=0; i<5; i++) System.out.println("i = " + i);

double [] tab={2.5,-5.8,7.4};
for(int i=0; i<tab.length; i++) System.out.println(tab[i]);
```

— **Boucle for each** « for(Type : var Collection) {… } »

```
int tab[]={5,8,4};
for(int v : tab)  System.out.println(v);

String []data={"Rabat", "Casa"};
for(String s : data)  System.out.println(s);
```

Différence Java / C

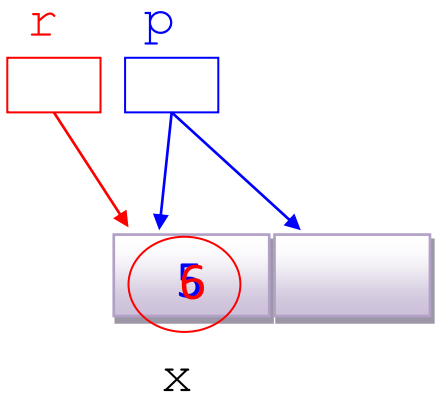
- `Java` n'a pas de préprocesseur (pas de macros)
- Pas de `const` mais `final`
- Aucune fonction ne peut exister sans être liée à une classe
- `struct`, `union`, `typedef` remplacés par `class`
- Pas de `goto`

Différence Java / C

- Pas de pointeur … références !!!

```
int x=5;
int *p; //pointeur C++
p=&x;
int &r ; //référence C++
r=&x;
```

```
p++;
r++;
```



ENSIAS

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

M.2.5.1. Paradigme Objet