INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Glossaire du jour :

- Cast *(rappel)*
- **Interface**
- **Formalisme UML**

Pour aller plus loin en Programmation Orientée Objet

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Rappel des termes importants déjà vus (1A + 2A)

Orienté objet : éléments déjà acquis 1A

- Classe
- Types non primitifs
- Instance
- Surcharge
- toString
- equals
- Encapsulation
- Accesseurs
- **this**
- **public**
- **private**
- **new**

Orienté objet : éléments nouvellement acquis 2A

- Héritage
- Polymorphisme
- Formalisme UML
- Classe mère / fille
- Type statique
- Type dynamique

- **protected**
- **super**
- **extends**
- **abstract**

Définition / Important

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Classes abstraites et **Interfaces** en Java

membre de UNIVERSITÉ DE LYON

Rappel sur les classes et méthodes abstraites

- Une classe ancêtre peut être déclarée "abstract" et comprendre des méthodes "abstract".
  - On ne peut pas créer d'objets à partir d'une classe abstraite.
  - Une méthode abstraite n'a pas d'implémentation
  - L'implémentation d'une méthode abstraite est réalisée dans les classes dérivées

- Finalement, une classe abstraite permet de définir dans une classe de base des fonctionnalités communes à toutes ses descendantes, tout en leur imposant de redéfinir certaines méthodes.

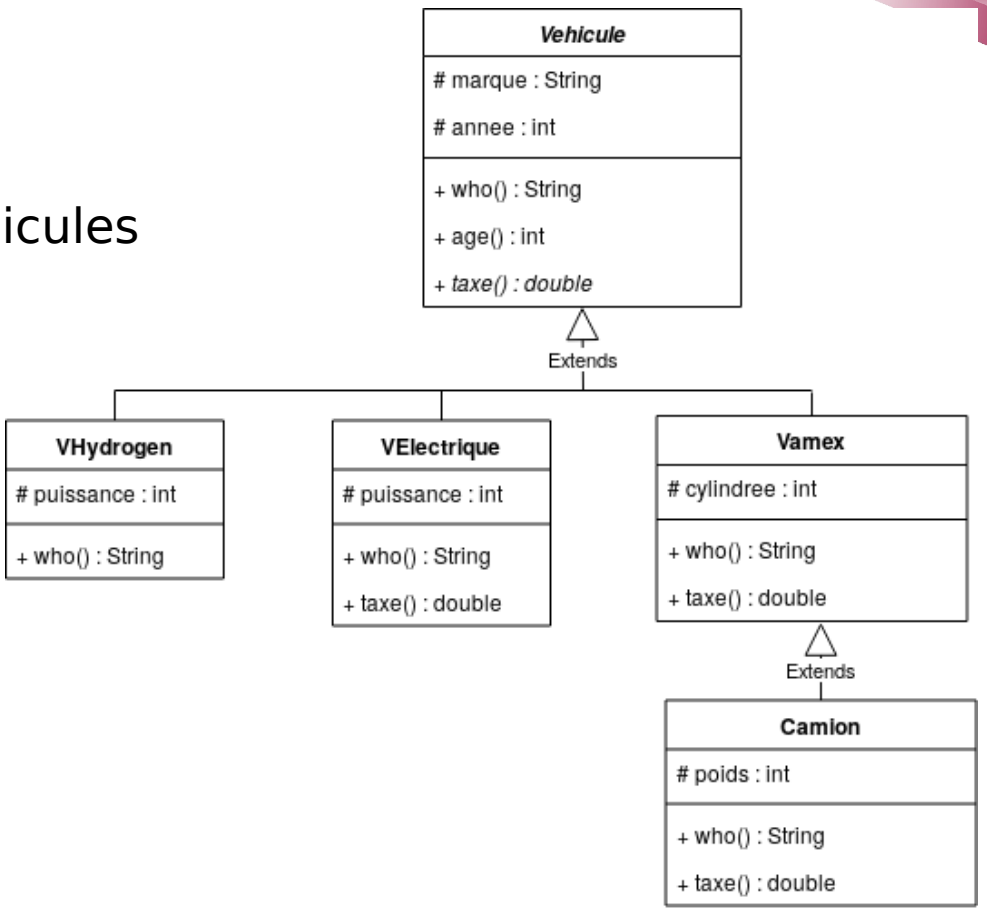
- Si l'on considère une classe abstraite n'ayant que des méthodes abstraites et n'ayant aucun attribut, on aboutit à la notion d'**interface**.

Interfaces
Commençons par le bilan

- Une interface :
  - c'est un type *(Comme les classes)*
  - c'est un « contrat » réutilisable
- Différence classe / interface :
  - Interface ne contient que des signatures *(ni code, ni attributs)*
  - Mot clef : **implements**
  - On ne **peut pas instancier** une interface
- Objectif :
  - déclarer une caractéristique transversale à plusieurs classes

Prenons un exemple

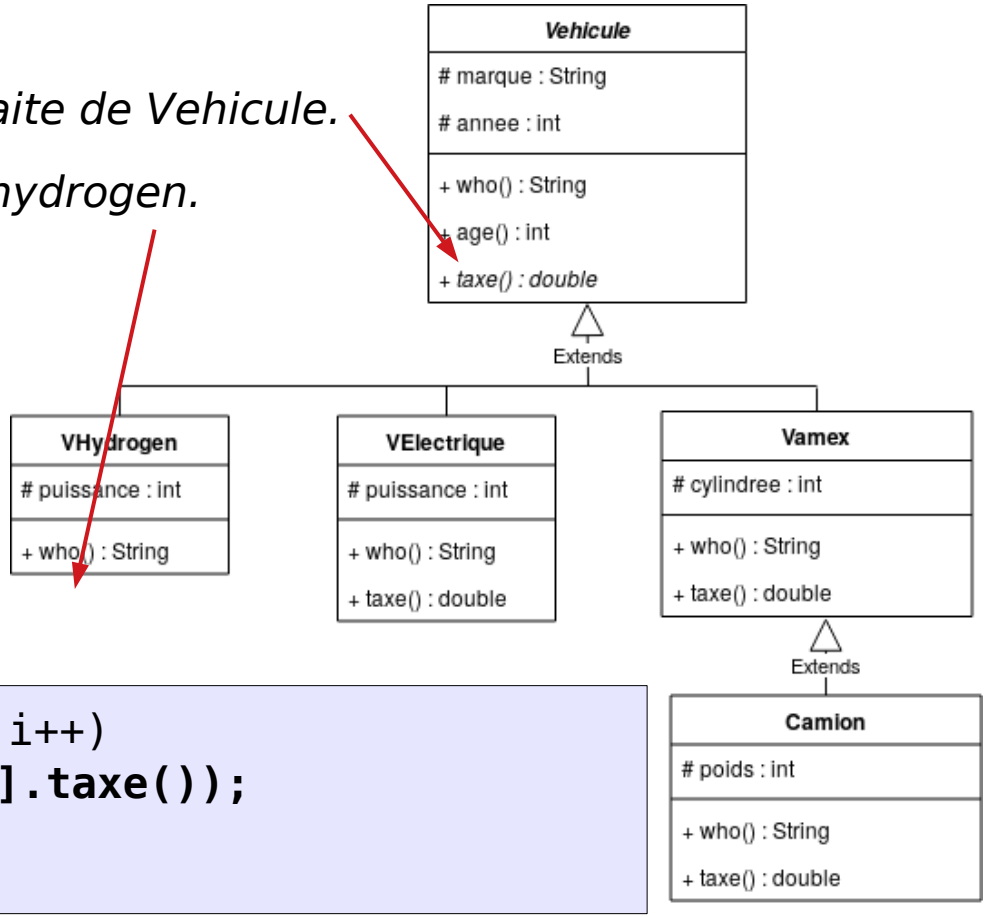
- On veut afficher la taxe des véhicules seulement lorsqu'il y en a une.
- Problème :
  VHydrogen n'a pas de taxe



Comment faire ?

Version courte mais incorrecte méthodologiquement

- *On rajoute taxe() en méthode abstraite de Vehicule.*
- *On doit donc redéfinir taxe() pour Vhydrogen.*
- *On rajoute une méthode taxe() qui renvoie 0.*

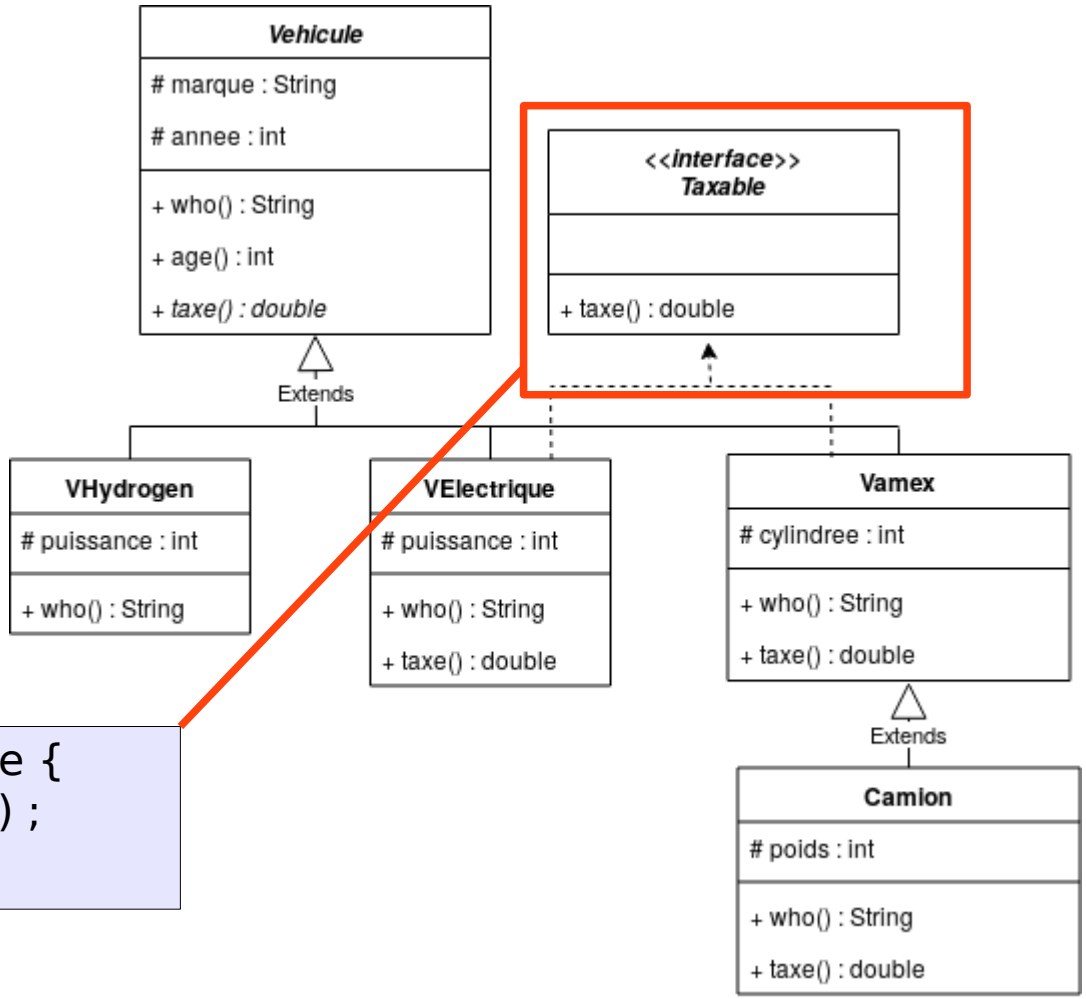



```
for(int i=0; i<myTab.length; i++)
  System.out.println(myTab[i].taxe());
```

- Problèmes :
  - Beurk !
  - 0 va être affiché pour les véhicules qui n'ont pas de taxe.

Une solution :
définir un type factorisant la caractéristique voulue

- Déclarer un « contrat » spécifique :
  - Un nom
  - Une liste de **signatures** de méthodes




```
public interface Taxable {
    public double taxe() ;
}
```

Mise en œuvre (1/3)

- Implémenter Taxable:
  - Nécessite de fournir toutes les méthodes annoncées
  - Puis on change la signature de la classe Vamex

```
public interface Taxable {
      public double taxe() ;
}
```

```
public class VAMEX extends Vehicule implements Taxable  {
      protected int cylindree;
      public VAMEX( int uneAnnee, String uneMarque,int uneCylindree){
            super(uneAnnee,uneMarque) ;
            cylindree= uneCylindree;
      }
      public double taxe(){
            return (cylindree*0.1+50);
      }
}
```

Mise en œuvre (2/3)

- Idem pour VElectrique

```
public interface Taxable {
      public double taxe() ;
}
```

```
public class VElectrique extends Vehicule implements Taxable  {
      protected double puissance;
      public VElectrique( int uneAnnee, String uneMarque,int unePuiss){
            super(uneAnnee,uneMarque) ;
            puissance= unePuiss;
      }
      public double taxe(){
            return (puissance*0.43+12);
      }
}
```

La classe Camion n'a pas besoin d'implémenter l'interface puisque sa classe mère le fait.

Mise en œuvre (3/3)

- Polymorphisme sur une interface :
  - Une interface est un type
    - Implémenter une interface c'est acquérir ce type
  - On peut tester si une variable a pour type une interface

```
myTab[0] = new VAMEX(  2005, "Ford",  1500 );
myTab[1] = new Camion( 2009, "Scania", 4200, 2.5 );
myTab[2] = new Camion(  2013, "Varta", 66, 3);
myTab[3] = new VAMEX(  2012, "Toyota", 70 );

for(int i=0; i<myTab.length; i++){
      if (myTab[i] instanceof Taxable) {
            Taxable s = (Taxable)myTab[i];
            System.out.println(s.taxe());
      }
}
```

instanceof permet de tester le type d'un objet

Interfaces
Le bilan

- Une interface :
  - c'est un type *(Comme les classes)*
  - c'est un « contrat » réutilisable
- Différence classe / interface :
  - Interface ne contient que des signatures *(ni code, ni attributs)*
  - Héritage multiple d'interfaces possibles
  - Mot clef : **implements**
  - On ne **peut pas** instancier une interface
- Objectif : déclarer une caractéristique transversale à plusieurs classes.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Application au TP Courbe

membre de UNIVERSITÉ DE LYON

Application au TP Courbe

- On a une nouvelle courbe :
  - c'est une ligne polygonale supposé ouverte (sinon c’est un polygone)
  - Elle a un barycentre, une longueur mais ...
  - .... on ne peut pas calculer son aire()
- Besoin d’une interface AireCalculable()
  - Cercle et Polygone l’implémente (PolygoneRegulier descendant de Polygone l’implémente automatiquement)
  - aire() n’est plus une méthode abstraite de Courbe

Application au TP Courbe

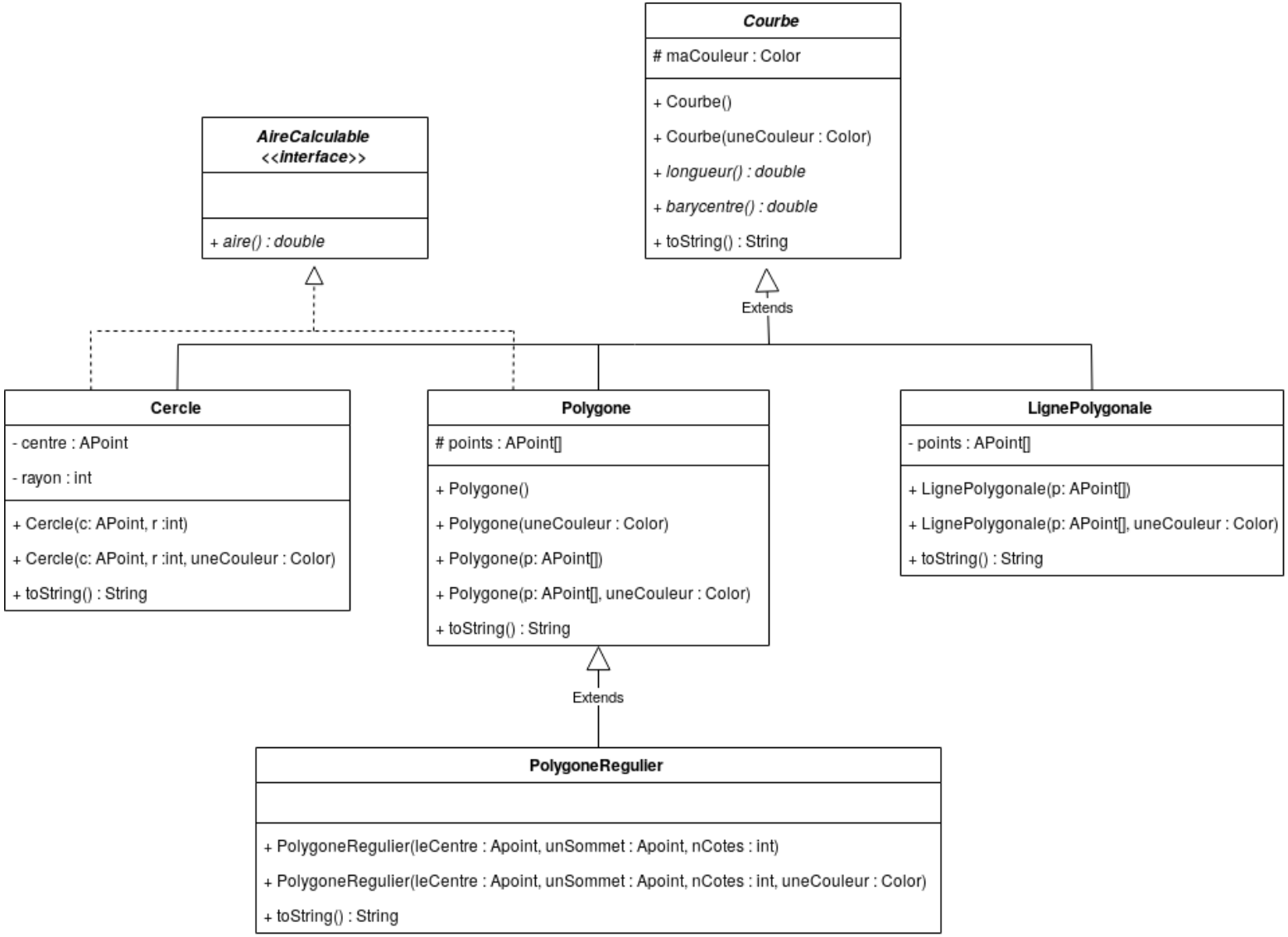
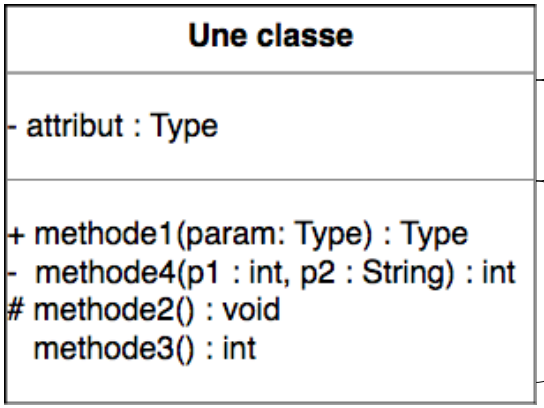


Diagramme de classes

Représentation d'une classe

Rappels de première année



Zone 1 : Nom de la classe

Zone 2 : Liste des attributs.

Notation :

<visibilité> <nom> : <Type>

Zone 3 : Liste des méthodes.

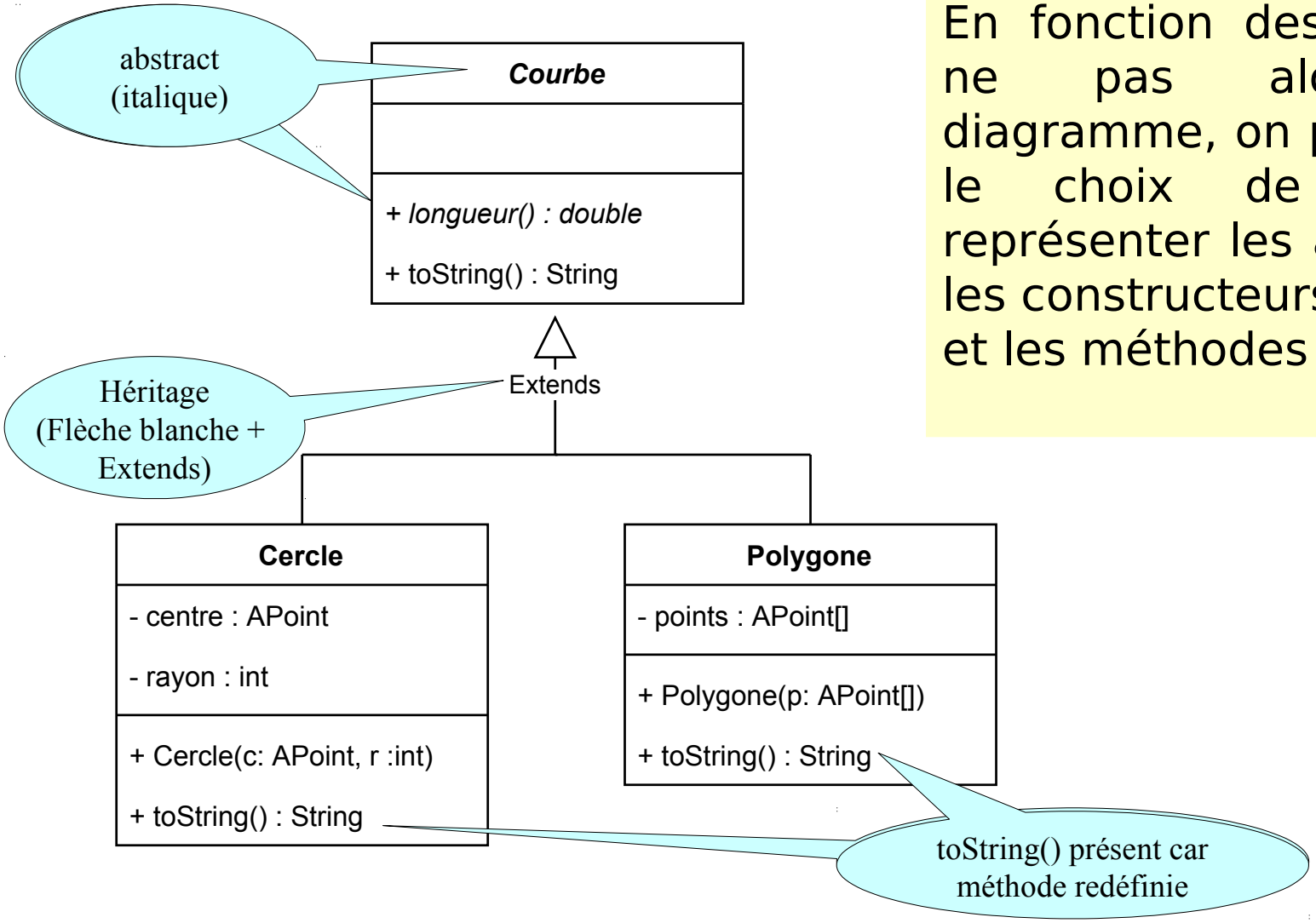
Notation :

<visibilité> <nom> (<paramètres>) : <TypeRetour>

4 visibilités possibles *(3 seulement au programme)* :

Visibilité	Notation	Remarques
public	+	
package		*Parfois noté* ~. Hors programme
protected	#	
private	-	

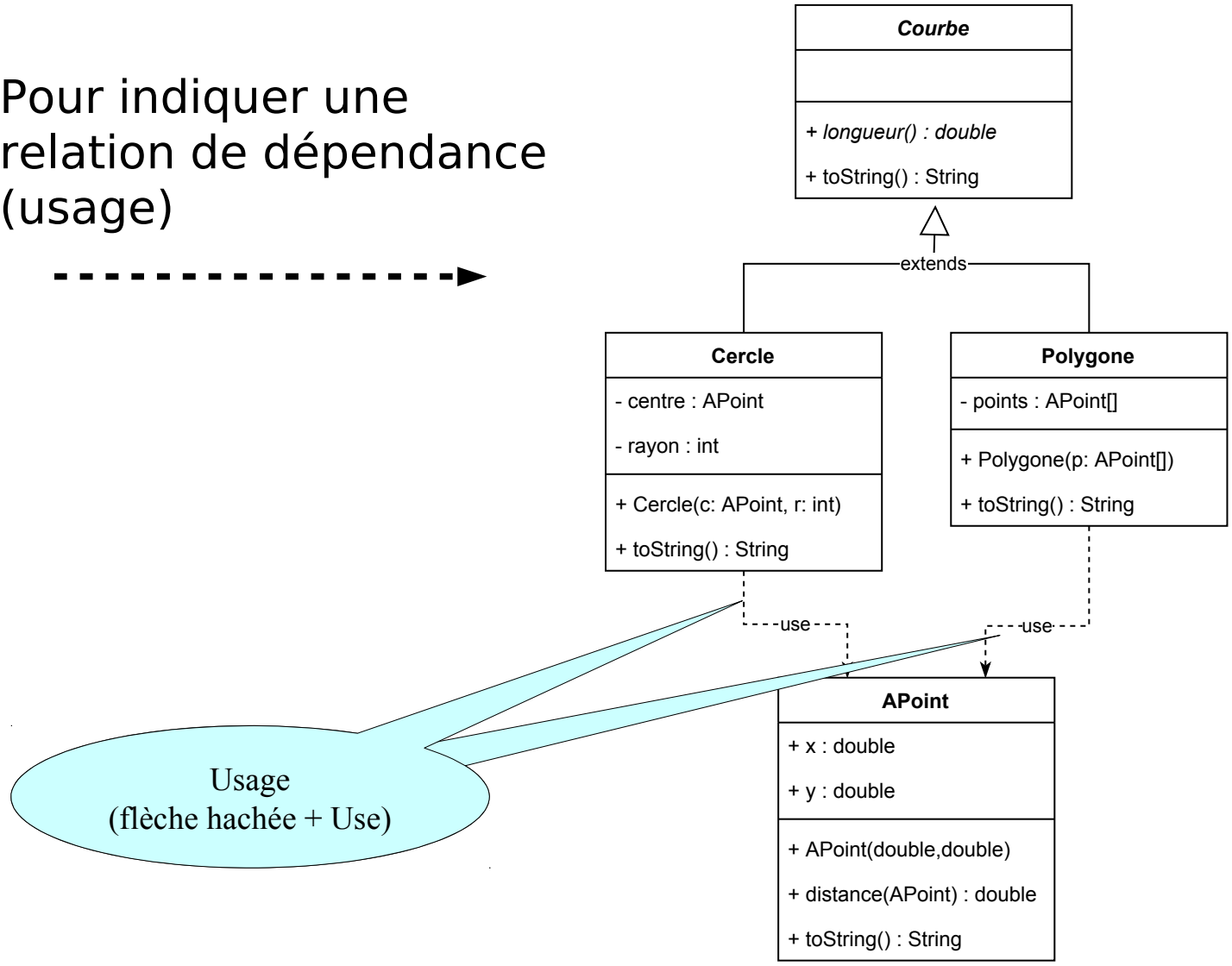
Exemple de diagramme de classes



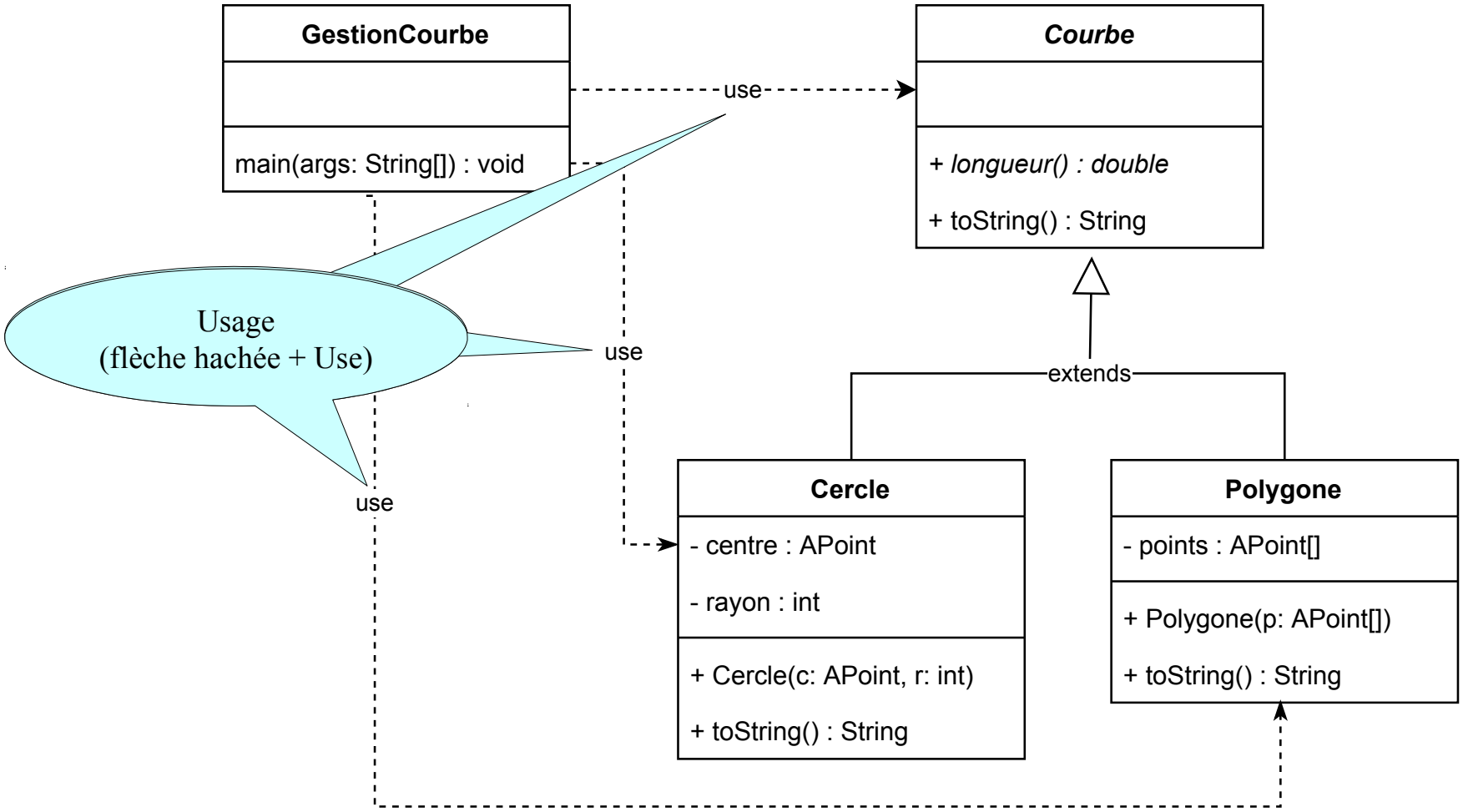
En fonction des cas, pour ne pas alourdir le diagramme, on pourra faire le choix de ne pas représenter les accesseurs, les constructeurs, toString() et les méthodes abstraites

Exemple de diagramme de classes

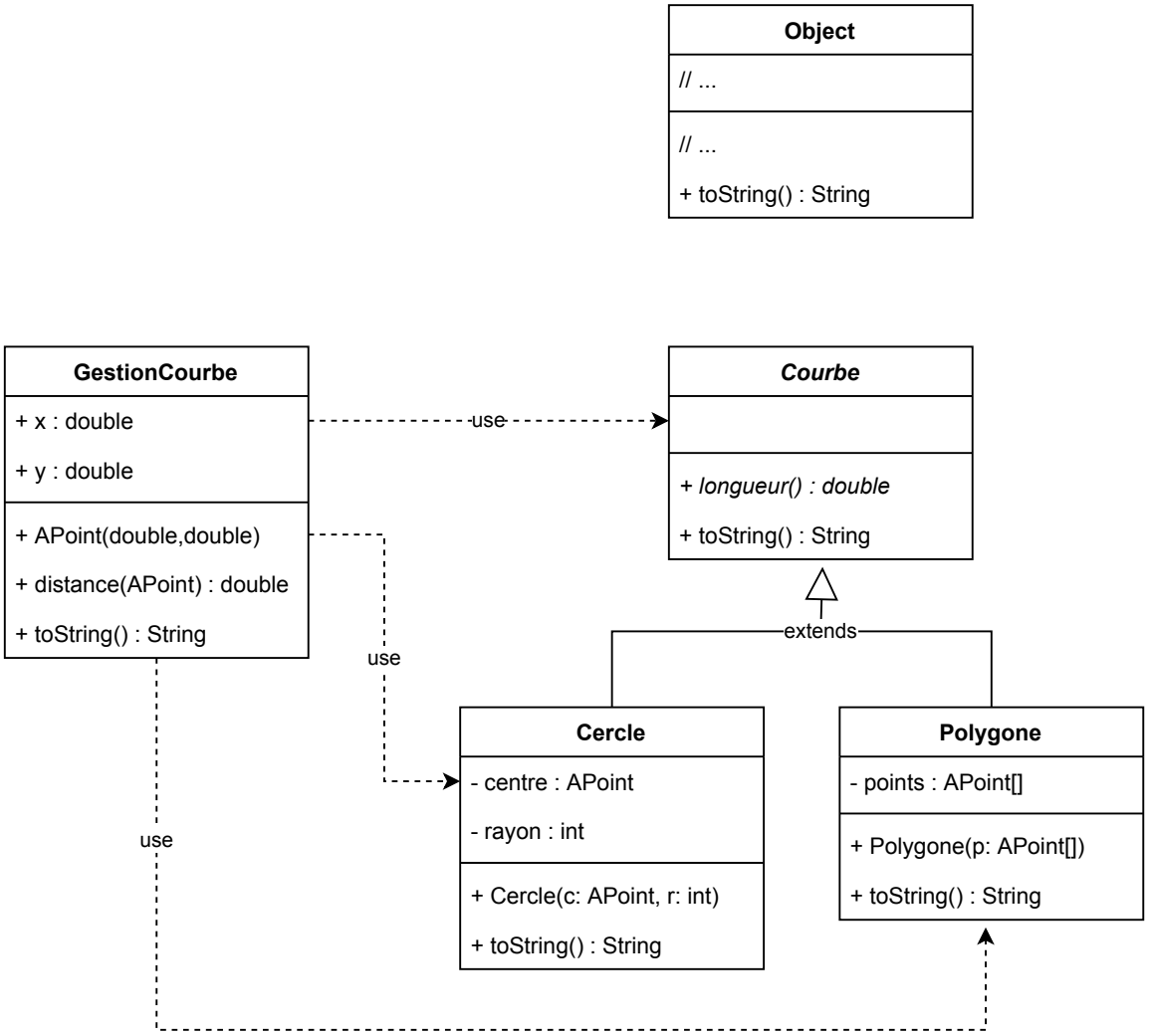
Pour indiquer une relation de dépendance (usage)



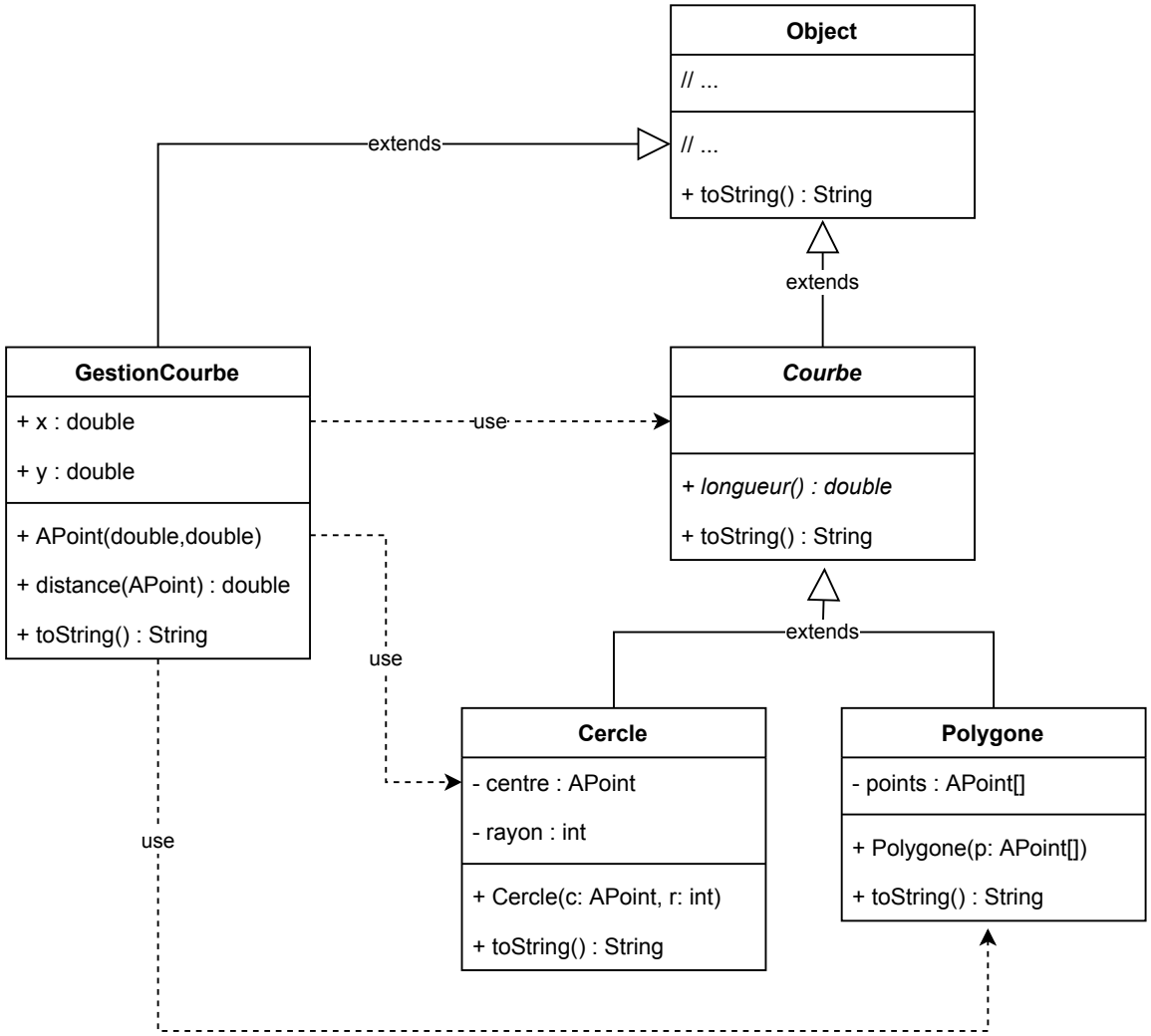
Exemple de diagramme de classes



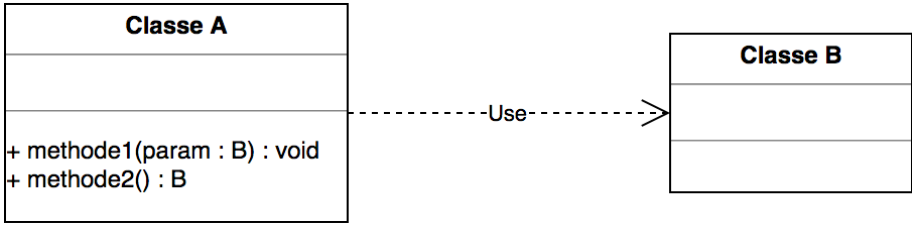
Et avec la classe Object ?



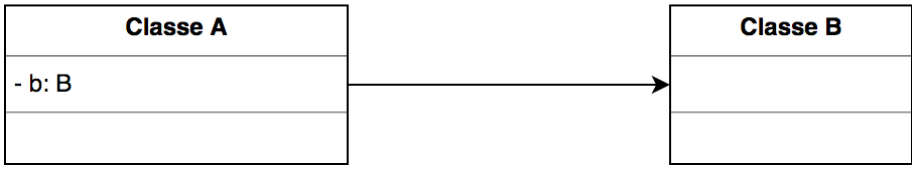
Et avec la classe Object ?



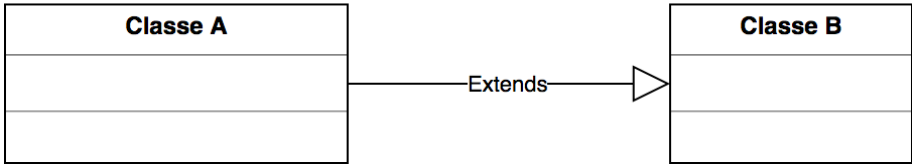
Synthèse des liaisons possibles entre classes
(Première et deuxième année)



Relation de dépendance (Usage) :
La classe A manipule des variables de type B.



Relation structurelle (association) :
la classe A possède au moins un attribut de type B.



Relation de spécialisation (Héritage) :
La classe A hérite de la classe B.

Outils pour modéliser

- Outil en ligne recommandé :

  https://www.draw.io/

- Solutions alternatives :
  - Applications à installer :
    - ArgoUML : distribution java, gratuite, multi-plateforme
    - StarUML : open source
    - yEd : gratuit, multi-plateforme
    - Bouml
    - Rational rose (IBM) : version essai gratuite
    - Open Modelsphere : gratuit
  - Outils gratuits à utiliser en ligne :
    - Lucidchart
    - GenMymodel
    - yuml.me : génération automatique du graphe à partir d'une expression