Technologie Objet

—

Relations entre classes

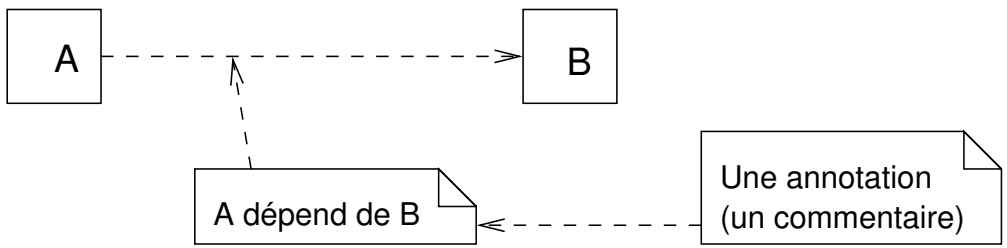
Xavier Crégut
<Prénom.Nom@enseeiht.fr>

ENSEEIHT
Sciences du Numérique

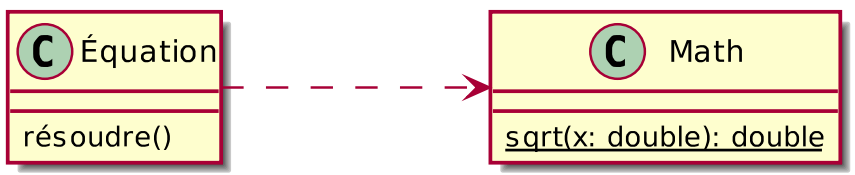
Relation de dépendance

On dit qu'il y a relation de dépendance entre une classe A et une classe B si la classe A fait référence à la classe B dans son texte. On dit que A dépend de B.

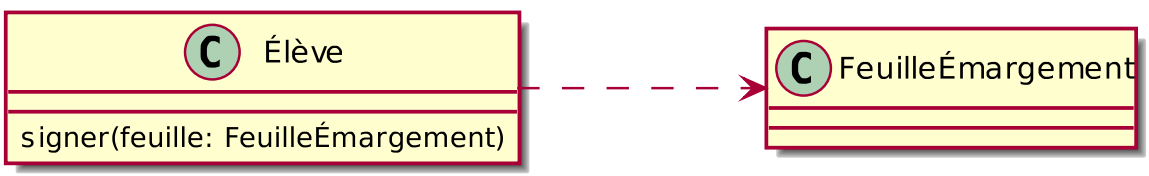
La relation de dépendance est représentée par une flèche en traits interrompus dont l'origine est la classe A et la cible la classe B.



Exemple : La classe Equation dépend de la classe Math puisque sa méthode « résoudre » utilise la méthode « sqrt » de Math.



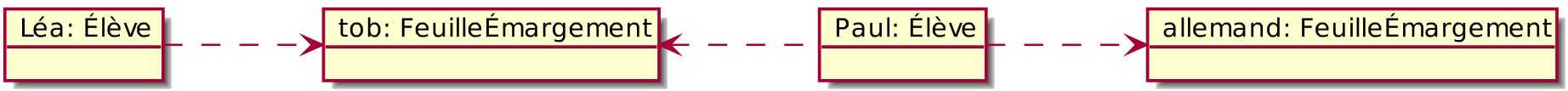
Exemple : Un élève doit signer la feuille d'émargement.



Lien de dépendance

En général, si une classe A dépend d'une classe B, il y a un lien de dépendance entre un objet de la classe A et un objet de la classe B.

Exemple : Léa, Paul... doivent chacun signer la feuille d'émargement de TOB. Plus tard, Paul signe la feuille d'émargement d'Allemand.
Il y a un lien entre l'objet Léa et l'objet feuille TOB, un lien entre l'objet Paul et le même objet feuille TOB, etc. Il y a un lien entre l'objet Paul et la feuille Allemand.



Remarque : Un lien est instance d'une relation (comme un objet est instance d'une classe).

Le lien (par extension la relation de dépendance) est momentané si B apparaît comme variable locale ou paramètre d'une méthode m de A. Le lien n'existe que pendant l'exécution de m.
Les relations de dépendance sont rarement représentées en UML (souvent peu pertinentes).

Exemple : Il n'y a un lien entre Léa et la feuille d'émargement de TOB que le temps de signer la feuille (exécution de la méthode signer).

Relation structurelle

Une relation de dépendance est dite relation structurelle si elle dure dans le temps.
C'est par exemple le cas quand B est le type d'un attribut de la classe A.

Exemple : Une personne travaille dans une entreprise.

- Un objet personne est lié à un objet entreprise
- Ce lien dure dans le temps
- La classe Personne pourrait avoir un attribut dont le type est Entreprise (et inversement).

UML propose trois relations de plus en plus précises :

- association (la plus générale)
- agrégation
- composition (la plus précise)

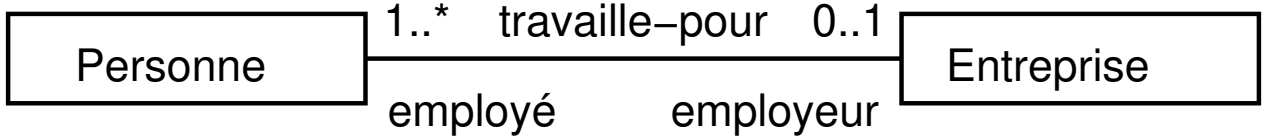
Ces relations permettent de décrire l'architecture de l'application.

Relation d'association

Définition : Une relation d'association est une relation entre deux classes qui traduit un couplage faible : chaque classe pourrait être considérée indépendamment de l'autre.

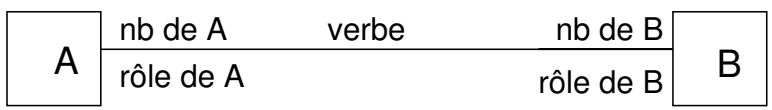
Propriété : Les objets des deux classes sont relativement indépendants et leurs durées de vie ne sont pas liées.

Exemple :



- Classes (et objets) aux extrémités de la relation indépendantes :
  - On pourrait parler de Personne sans parler d'Entreprise
  - On pourrait parler d'Entreprise sans parler de Personne
- Durées de vie des objets non liées :
  - Si une entreprise disparaît, les personnes continuent à exister.
  - Si une personne disparaît, l'entreprise continue à exister.

Forme générale :

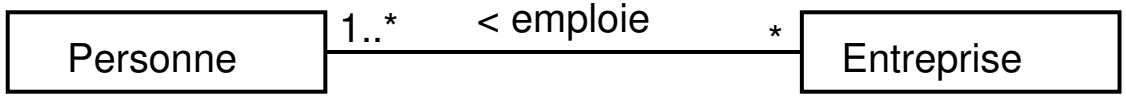


Expliquer une relation : nom et rôles

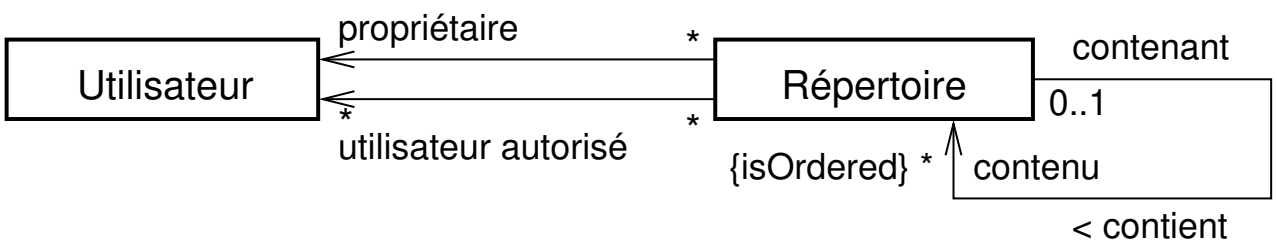
Il est essentiel de savoir ce que représente une relation et donc de préciser sa signification.

On peut nommer la relation :

- C'est un verbe (travaille-pour, emploie, etc.)
- Le nom est placé au milieu de la relation
- La lecture se fait de gauche à droite ou de haut en bas.
  Une direction peut indiquer le sens de lecture quand il n'est pas naturel.



On peut donner le rôle joué par les objets dans la relation (extrémités de la relation)

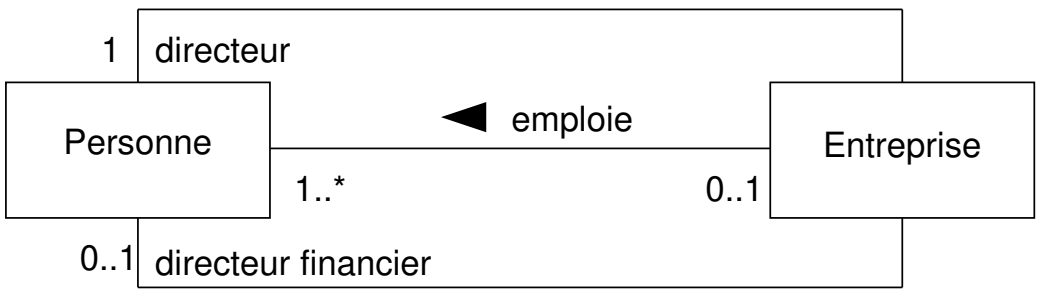


- Les rôles peuvent avoir des droits d'accès (comme les attributs).
- Si le rôle est omis, c'est le nom de la classe qui est utilisé.

Multiplicité (ou cardinalité)

Objectif : indiquer combien d'objets peuvent/doivent prendre part à la relation :

- forme générale : min..max
  - exemple : 1..4
  - le nombre d'objet est compris entre min et max inclus
  - la valeur de max peut être « * » : nombre quelconque (« illimité »)
- un entier (ex : 4) : le nombre exact d'objets (simplification d'écriture de 4..4)
- `0..1` : optionnel
- $*$ ou `0..*` : 0 ou plusieurs (un nombre quelconque y compris 0)
- `1..*` : au moins 1 (un nombre quelconque mais au moins 1)
- une multiplicité omise crée une ambigüité (souvent considérée comme 1..1)



Navigation

Définition : La navigation est le fait de traverser une relation (en fait un lien) :
à partir d'une objet d'une extrémité, on peut atteindre les objets de l'autre extrémité.

Exemple : Depuis une personne, Xavier, on obtient l'entreprise où Xavier travaille (N7).

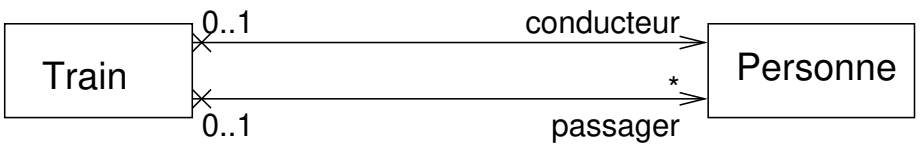
Propriété : La relation d'association est bidirectionnelle : navigable dans les deux sens.

Exemple : Partant de Xavier, on trouve son employeur, N7, et inversement, de N7 on trouve Xavier parmi les employés.

Principe : Rôles (et noms) d'association sont utilisés pour la navigation (termes en bleu)

Sens de navigation : Expliciter dans quel sens une relation peut être traversée.

- Flèche à l'extrémité d'une relation : la relation peut être traversée dans le sens de la flèche
- Croix à une extrémité : interdit le sens de navigation qui mène à la croix

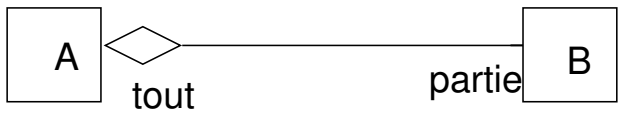


- d'un objet train on peut obtenir le conducteur ou les passagers
- d'une personne on ne peut pas retrouver le train (Train n'apparaît pas dans Personne)

Relation d'agrégation

Définition : La relation d'agrégation est un cas particulier d'association où une classe est prépondérante par rapport à l'autre : ses objets (le tout) agrégent d'autres objets (les parties).

Notation : On utilise un losange vide à l'extrémité de la relation, côté tout.



Exemple : Un département de formation s'appuie sur des gestionnaires et des enseignants.

Propriété : La relation d'agrégation permet le *partage* : le même objet peut appartenir à plusieurs liens d'agrégation.

Exemple : Le même enseignant peut intervenir dans plusieurs départements.

Remarque : Le terme utilisé en UML est *aggregation* ou *shared aggregation*.

Relation d'agrégation (2)

Voici quelques indices d'une relation d'agrégation :

1. un objet correspond au tout, les autres aux partie
   - La promotion 1SN et les étudiants.
   - Toulouse INP et les trois écoles historiques : N7, A7 et NSAT.
2. les opérations du tout se propagent sur les parties
   - L'INP et ses écoles : les statistiques demandées à l'INP sont d'abord demandées par l'INP aux écoles puis consolidées au niveau INP.
   - Les opérations de Segment utilisent les opérations de Point.
3. il est difficile de parler du tout sans parler de ses parties
   - Segment et ses Points extrémités
   - Itinéraire et les Villes étapes

Attention : La relation d'agrégation est subjective. Elle apporte une nuance au modèle, explicite un point de vue.

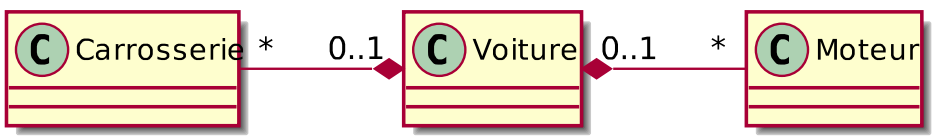
Exemple : Entre Groupe et Étudiant qui est le tout ?
Le groupe qui regroupe plusieurs étudiants ?
L'étudiant qui est inscrit à plusieurs groupes (1SN, langue, sport, BDE...) ?
La réponse dépend du point de vue (gestionnaire du groupe ou étudiant) !

Relation de composition

Définition : La relation de composition est un cas particulier de la relation d'agrégation. Elle dénote un couplage plus fort qui lie la durée de vie des objets : quand le tout est détruit, les parties sont aussi détruites.

Notation : On utilise un losange plein à l'extrémité de la relation, côté tout.

Exemple : Une voiture « est composée » d'une carrosserie, un moteur, etc.



Vocabulaire : On parle d'aggrégation forte.
L'objet du côté du losange (A, Voiture) est appelé l'agrégat ou composite.

Propriété : La composition interdit le partage. Un même objet ne peut appartenir qu'à un seul lien de composition (peu importe la relation de composition dont le lien est instance).

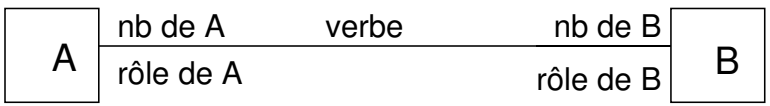
Exemple : Un même moteur ne peut pas être lié à une voiture et un camion !

Remarque : Un épaviste pourrait récupérer des éléments de la voiture avant sa destruction !
Remarque : La relation de composition est particulièrement importante pour les langages sans ramasse-miette car elle permet de savoir qui doit libérer la mémoire occupée par les objets.
En réalité, elle est importante dans tout système.

Relation entre classes : synthèse

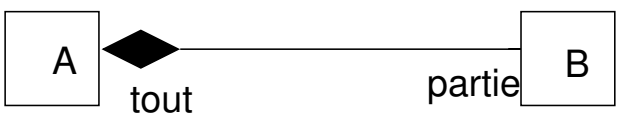
Une application est constituée de plusieurs classes dont le couplage est caractérisé par des relations dites de délégation :

- **association :** couplage faible correspondant à une relation symétrique entre objets relativement indépendants (durées de vie non liées) ;



- **agrégation :** association non symétrique avec couplage plus fort
  - relation tout et parties
  - relation de subordination : propagation des opérations de A vers B
  - on ne peut pas parler de A sans parler de B

- **composition :** agrégation forte (par valeur) :
  - La durée de vie des objets « partie » est liée à celle du « tout »
  - Pas de partage possible.



Relations entre classes : exercices

Exercice 1 Dessiner un diagramme de classes faisant apparaître un site web, des pages HTML et un « webmaster ».

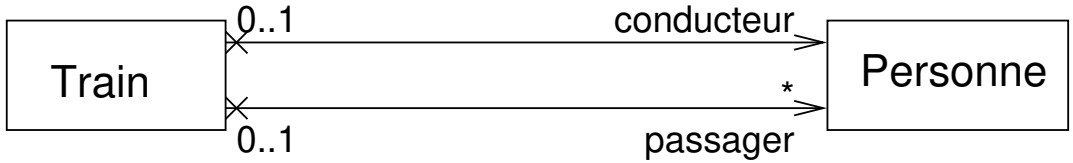
Exercice 2 Indiquer quelles relations il serait possible de définir entre :

- Livre et Page
- Une UE et les étudiants inscrits à l'UE
- Itinéraire et Gare
- Segment et Point

Exercice 3 Que penser d'attributs de types primitifs ?
Et plus généralement de la rubrique « attribut » d'UML.

Règle : Dans une classe, je jamais mettre un attribut dont le type est une classe !
Il faut le représenter par une relation entre les deux classes, le nom de l'attribut étant utilisé comme rôle.
Exceptions : On peut utiliser des attributs de type classe pour les classes considérées comme des types de base (String, Date, etc.) et quand il s'agit d'une relation de composition.

Traduction en Java



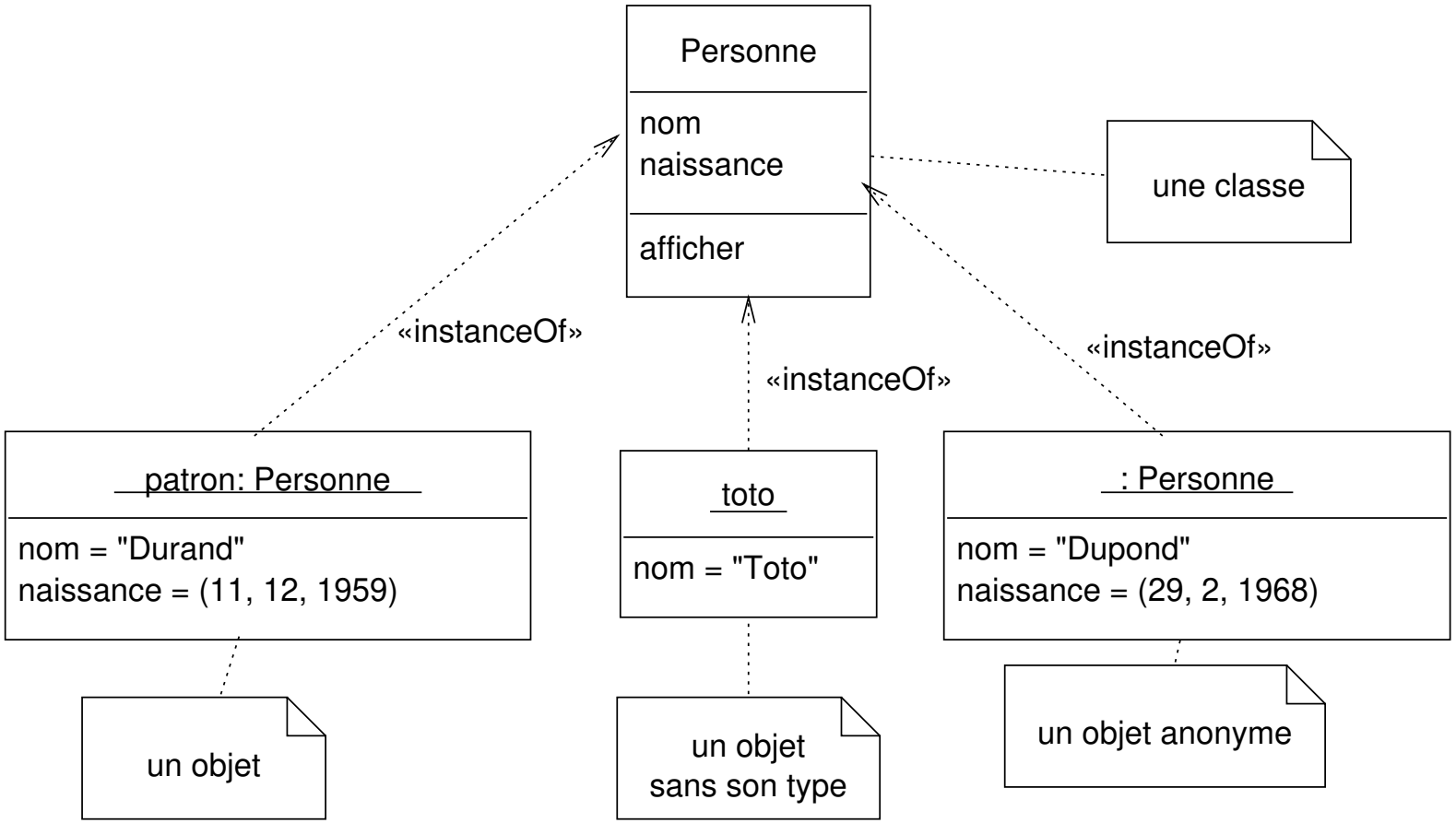
Principe : la relation « conducteur » entre Train et Personne se traduit par :

- un attribut appelé « conducteur » (nom du rôle) dans la classe Train
- le type de l'attribut est la classe au bout de la relation : Personne
- si la multiplicité avait été supérieure à 1, on aurait utilisé une structure de données

```
1   public class Train {
2       private Personne conducteur;
3           // Pour une multiplicité de 1 ou 0..1 (null ?)
4
5       private Personne[] passagers;
6           // Au lieu d'un tableau, il est préférable
7           // d'utiliser une structure de données adaptée.
8       ...
```

- En UML, on ne met pas toujours la croix quand la flèche est à l'autre extrémité.
- En Java, pas de distinction entre association, agrégation et composition :
  - c'est toujours une poignée
  - C'est au programmeur de le gérer !
  - Plusieurs stratégies possibles pour la composition... (cf TD)

Représentation des objets



Un objet est représenté sous la forme « nom : Type » souligné. Les deux sont optionnels.
On peut aussi faire apparaître la valeur des attributs de l'objet (*slot* en UML)
Remarque : le souligné se perd...

Diagramme d'objet

Diagramme d'objet : Une configuration possible respectant un diagramme de classe.

Exemple de diagramme de classe :

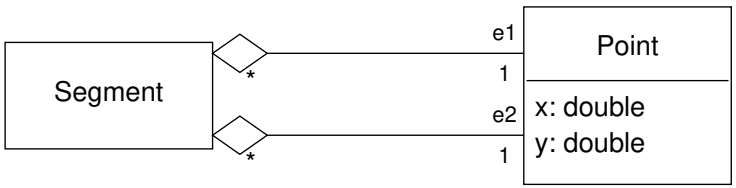
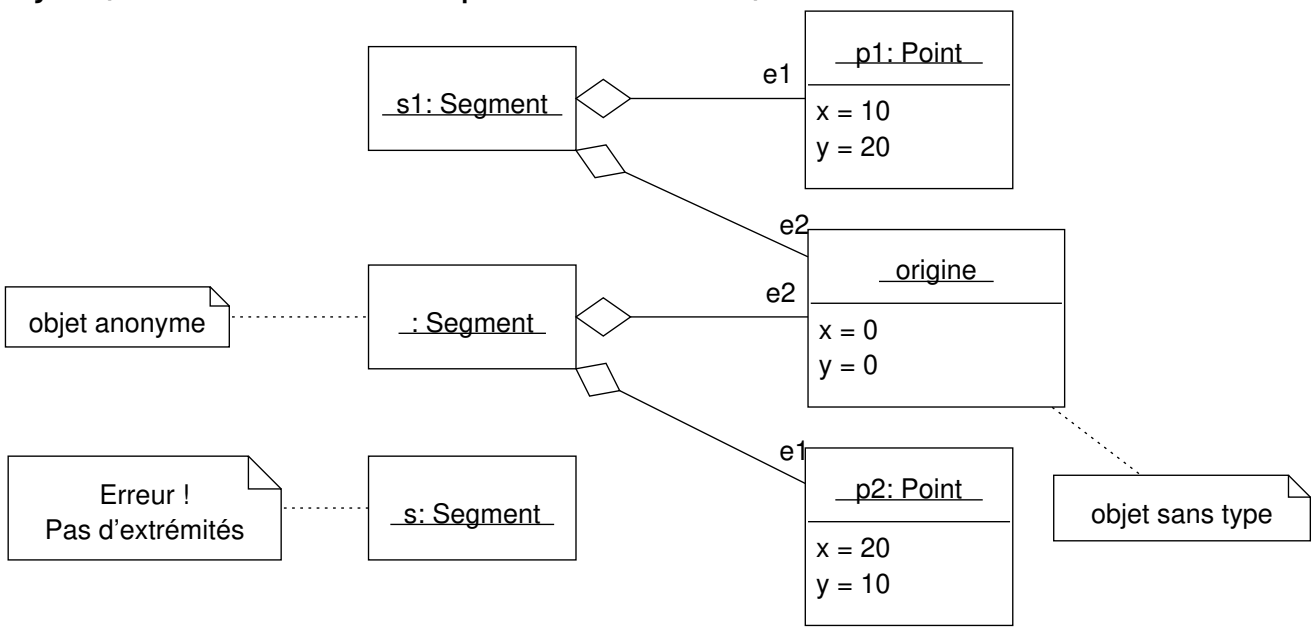


Diagramme d'objet (attention, s n'est pas conforme !) :



Conformité

Un diagramme d'objet est conforme à un diagramme de classe ssi :

- tout objet du diagramme d'objet est instance d'une classe du diagramme de classe
  - chaque attribut de l'objet correspond à un attribut de la classe
  - la valeur des attributs de l'objet respecte le type de l'attribut dans la classe
- tout lien du diagramme d'objet est instance d'une relation du diagramme de classe
- le nombre de liens respecte les multiplicités exprimées sur le diagramme de classe
- la relation de composition est respectée :
  - un objet ne peut apparaître extrémité que d'un seul lien de composition !

Quelques remarques sur le diagramme d'objet :

- Généralement, on ne fait pas apparaître les losanges (agrégation, composition)
- Les rôles sont souvent omis (si pas d'ambiguïté)