2N6 Programmation 2

avec python™

Les relations entre les classes :

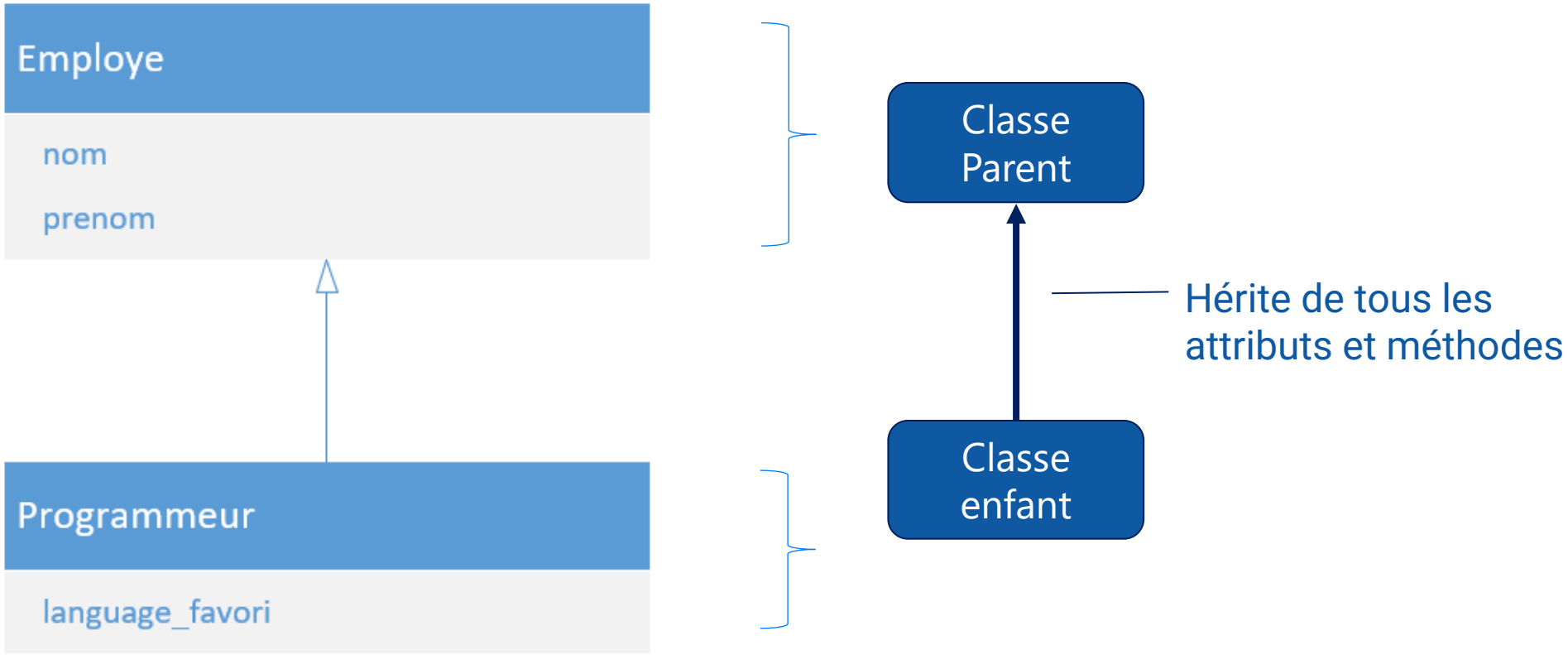
(et la nomenclature)

- Héritage
- Composition
- Agrégation
- Cardinalité

Héritage (en UML)

retour

- L'héritage est illustré ainsi dans UML



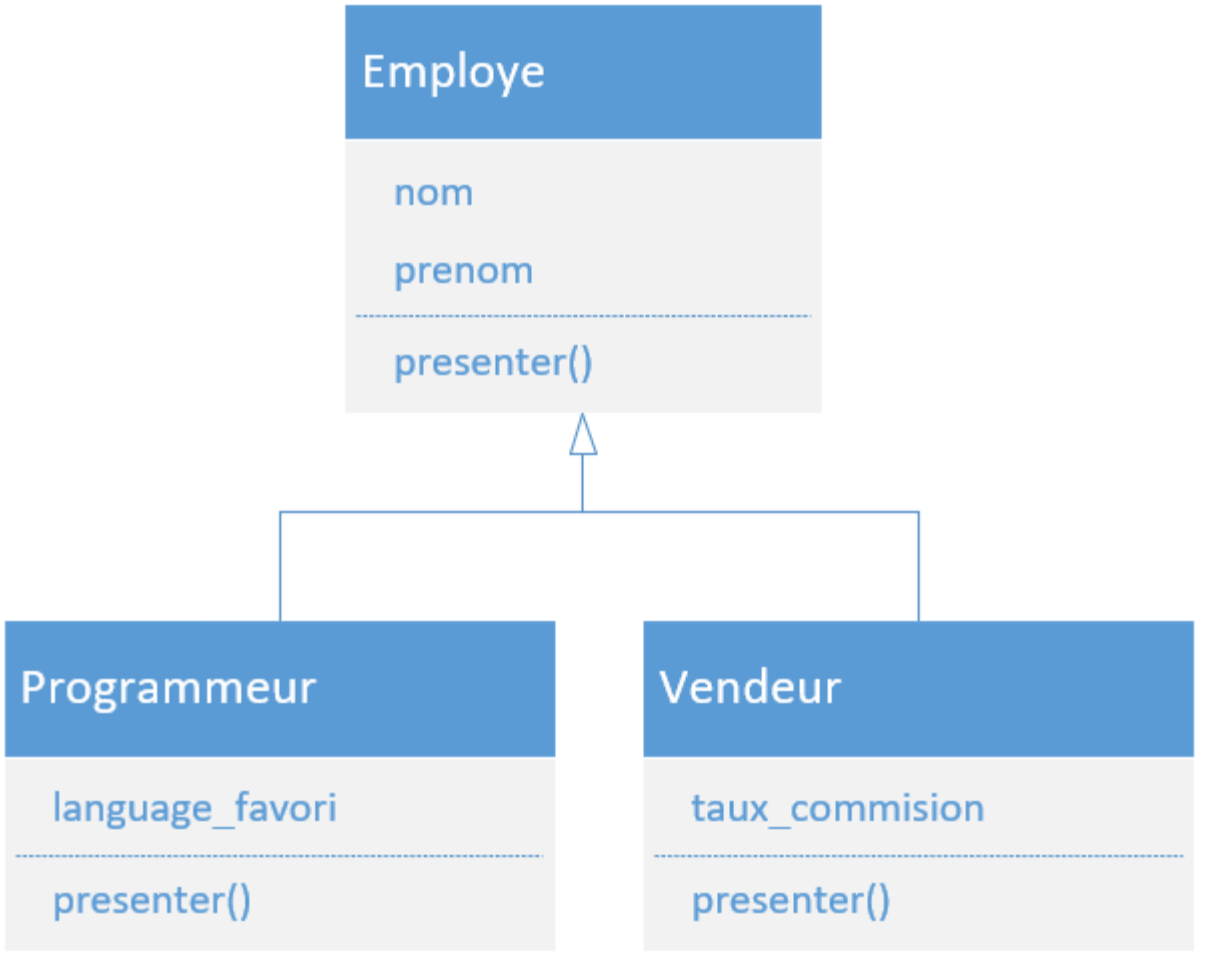
Un Programmeur EST un Employe

Héritage et polymorphisme

plusieurs formes

- Le **Polymorphisme** :
  - En programmation objet : la capacité d'une variable, d'une fonction, ou d'un objet à prendre plusieurs formes

- Programmeur et vendeur peuvent tous deux être utilisés dans toutes les situations où la classe Employe peut être utilisée mais la méthode "presenter()" est redéfinie et agira différemment.



Héritage et polymorphisme

```python
class Employe:
    def __init__(self, nom, prenom) -> None:
        self.nom = nom
        self.prenom = prenom
    def __str__(self):
        return f"Je m'appelle {self.prenom} {self.nom} et je
        fait partie de l'équipe du CEM."
```

```python
class Programmeur(Employe):
    def __init__(self, nom, prenom, langage) -> None:
        super().__init__(nom, prenom)
        self.langage_favori = langage
    def __str__(self):
        return (super().__str__() + "\n" +
                f"Je travaille en tant que programmeur en {self.
                langage_favori}" )

class Vendeur(Employe):
    def __init__(self, nom, prenom, taux) -> None:
        super().__init__(nom, prenom)
        self.taux_commission = taux
    def __str__(self):
        return (super().__str__() + "\n" +
                "Je travaille en tant que vendeur avec
                commission" )
```

Héritage et polymorphisme

```
28    emp = Employe("Gallant","Pierre")
29    print(emp)
30    emp_prog = Programmeur("Tremblay","Ana","Python")
31    print(emp_prog)
32    emp_vendeur = Vendeur("Borne","Margaut",2)
33    print(emp_vendeur)
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL .NET INTERACTIVE

```
Je m'appelle Pierre Gallant et je fait partie de l'équipe du CEM.
Je m'appelle Ana Tremblay et je fait partie de l'équipe du CEM.
Je travaille en tant que programmeur en Python
Je m'appelle Margaut Borne et je fait partie de l'équipe du CEM.
Je travaille en tant que vendeur avec commission
```

Signature

- Lorsqu'on implémente une nouvelle classe ou un nouveau groupe de classes interreliées **:**
  on commence par écrire leur signature
- La signature consiste est le « *squelette* » d'une classe.
- Elle consiste en le nom des attributs et méthodes, ainsi que les paramètres que les méthodes prennent

```python
class Employe:
    def __init__(self, nom, prenom) -> None:
        pass
    def __str__(self):
        return

class Programmeur(Employe):
    def __init__(self, nom, prenom, langage) -> None:
        pass
    def __str__(self):
        return

class Vendeur(Employe):
    def __init__(self, nom, prenom, taux) -> None:
        pass
    def __str__(self):
        return
```

Cardinalité

- La cardinalité permet d'indiquer le sens d'une relation entre des classes différentes ainsi que le nombre d'entités en relation.

1 à 0..1	Une entité à aucune ou une instance
1 à 1	Une entité à une instance exactement
1 à 0..N ou **1 à N**	Une entité à aucune ou plusieurs instances
1 à 1..N	Une entité à une instance ou plusieurs (au moins une)

La combinaison d'objets

- Dans les relations un à plusieurs, il y a deux concepts importants :
  - **Composition** : Un objet est fait de 1 ou plusieurs autres objets
  - **Agrégation** : Un objet possède ou regroupe d'autre(s) objet(s)

R19 COMPOSITION

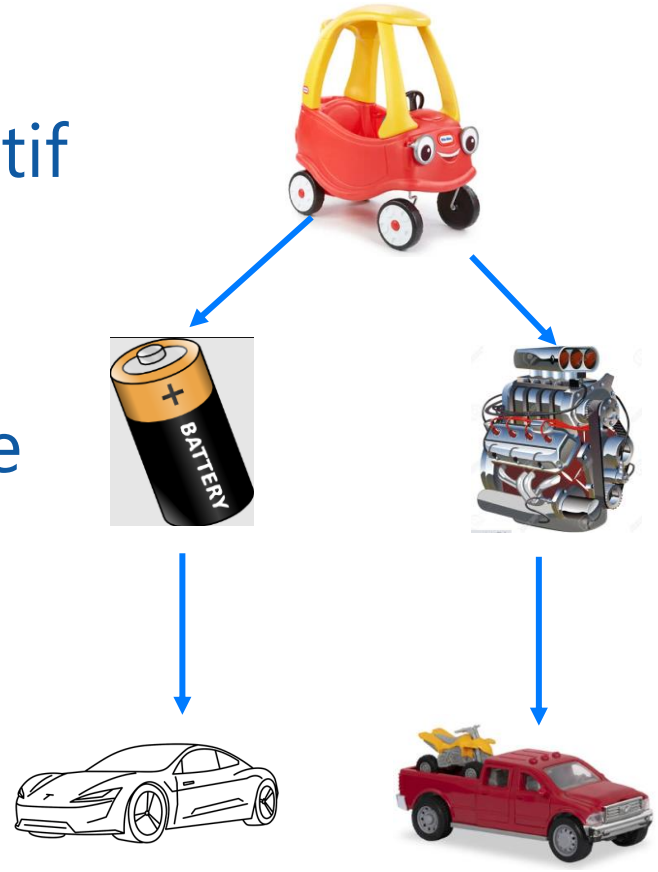
- Concept en programmation objet.
- Un objet dit "composite" est un objet qui "*possède*" d'autre objets de classes différentes.
- Lorsqu'on supprime un objet "composite", on va normalement aussi supprimer les objets que le compose.

Voir exemple

Ex : Voitures par héritage transitif

retour

- L'héritage est transitif en Python.
- Une classe hérite de tous les attributs et méthodes de tous ses ancêtres.




```python
class Voiture:
    def __init__(self,marque) -> None:
        self.marque = marque
```

```python
class Voiture_moteur(Voiture):
    def __init__(self, marque,reservoir):
        super().__init__(marque)
        self.reservoir = reservoir
```

```python
class Pickup(Voiture_moteur):
    def __init__(self, marque, reservoir,puissance):
        super().__init__(marque, reservoir)
        self.puissance = puissance
```

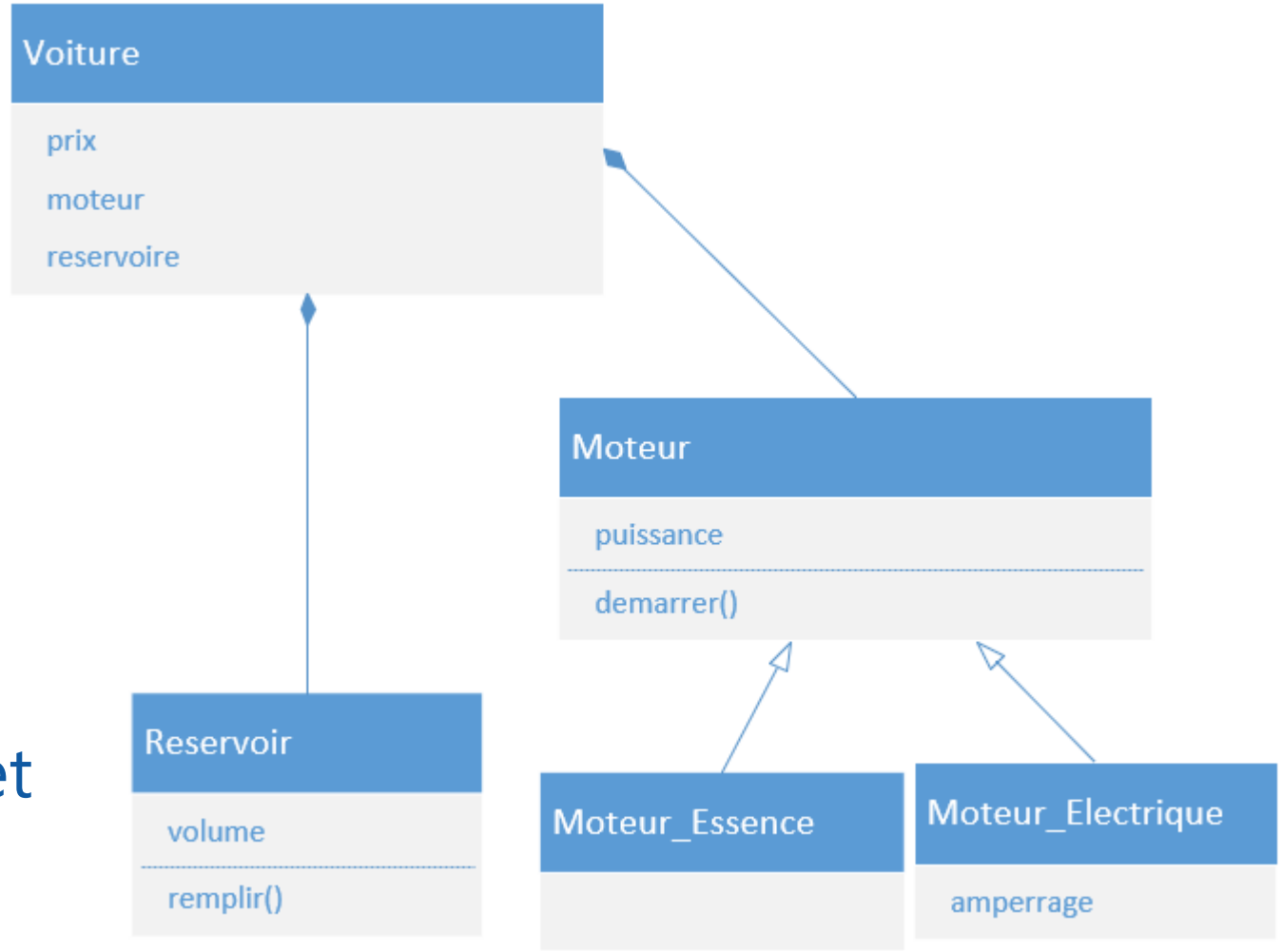
Ex : Voitures par composition

- La classe Moteur contient les propriétés des objets Moteur
- La sous-classe Moteur_Electrique hérite de la classe Moteur
- La classe Voiture contient un objet Moteur et un objet Reservoir.

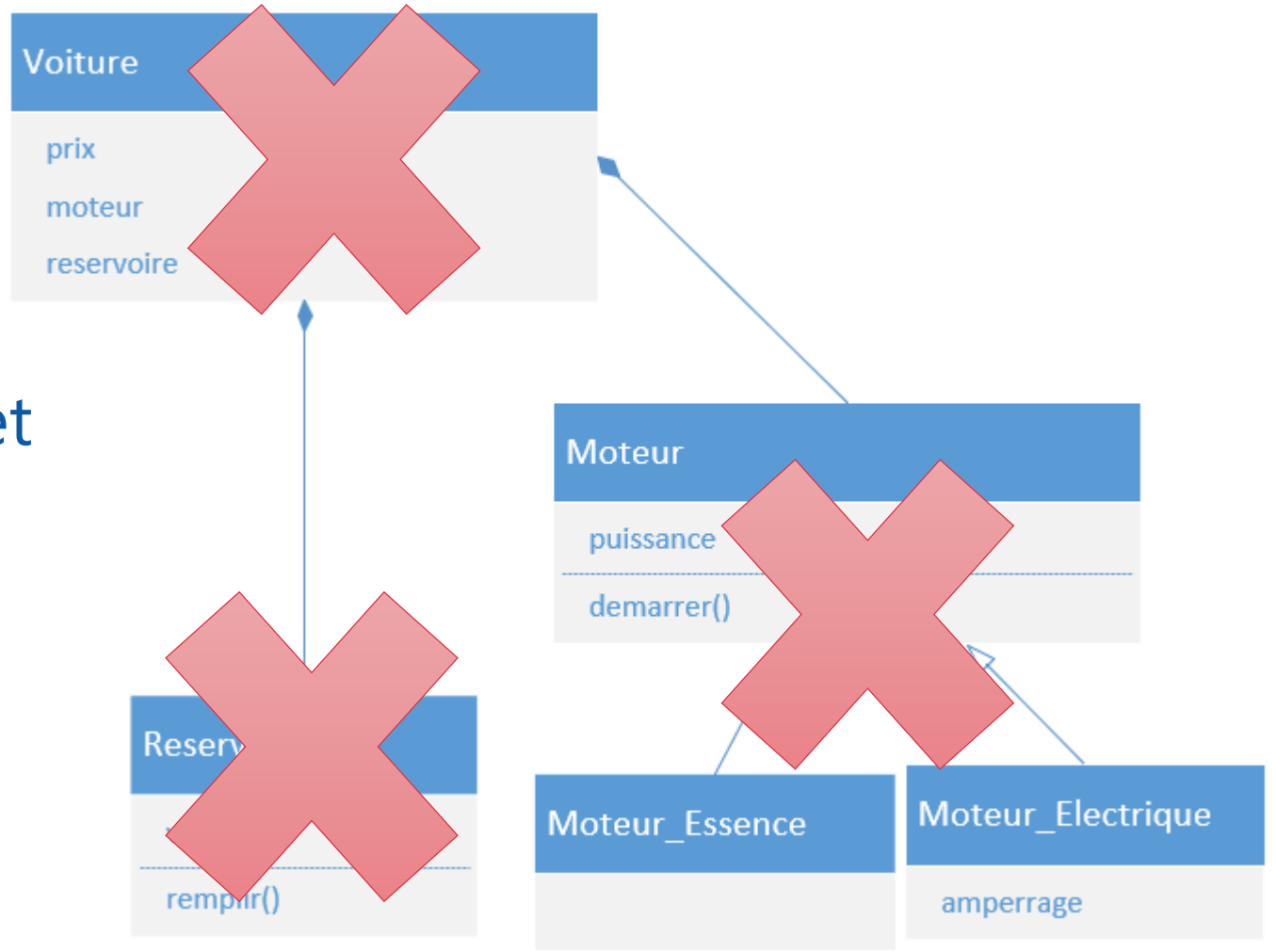
```
1  class Moteur:
2      def __init__(self) -> None:
3          pass
4
5  class Moteur_Electrique(Moteur):
6      def __init__(self) -> None:
7          super().__init__()
8
9  class Reservoir:
10     def __init__(self) -> None:
11         pass
12
13 class Voiture:
14     def __init__(self,moteur,reservoir,prix):
15         pass
16
```

Ex : Voitures par composition (UML)

- La classe Moteur contient les propriétés des objets Moteur
- La sous-classe Moteur_Electrique hérite de la classe Moteur
- La classe Voiture contient un objet Moteur et un objet Reservoir.



Ex : Voitures par composition (UML)



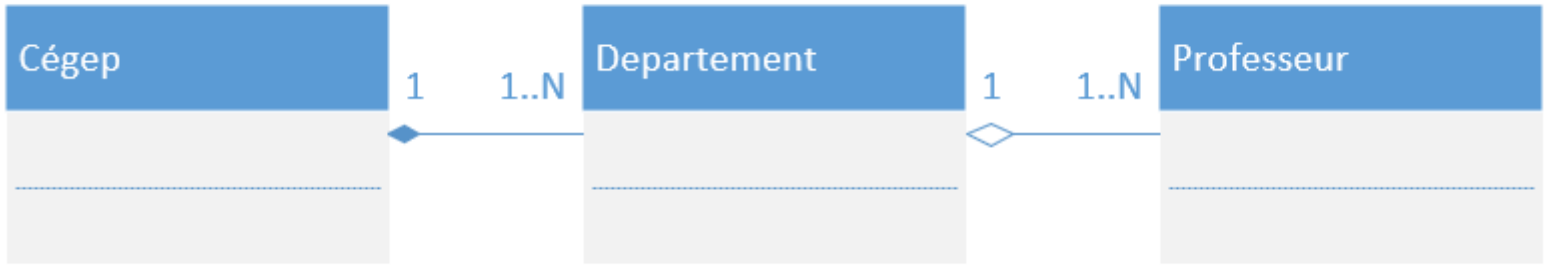
- La classe Voiture contient un objet Moteur et un objet Reservoir.
- Si une voiture est supprimé, son moteur et son réservoir le sont aussi.

Agrégation

- On parle d'agrégation lorsqu'un objet va faire référence à un ou plusieurs objets sans pour autant en être le propriétaire.
- Contrairement à la composition, lorsqu'un "agrégat" est supprimé, les objets auxquels il fait référence ne le sont pas.

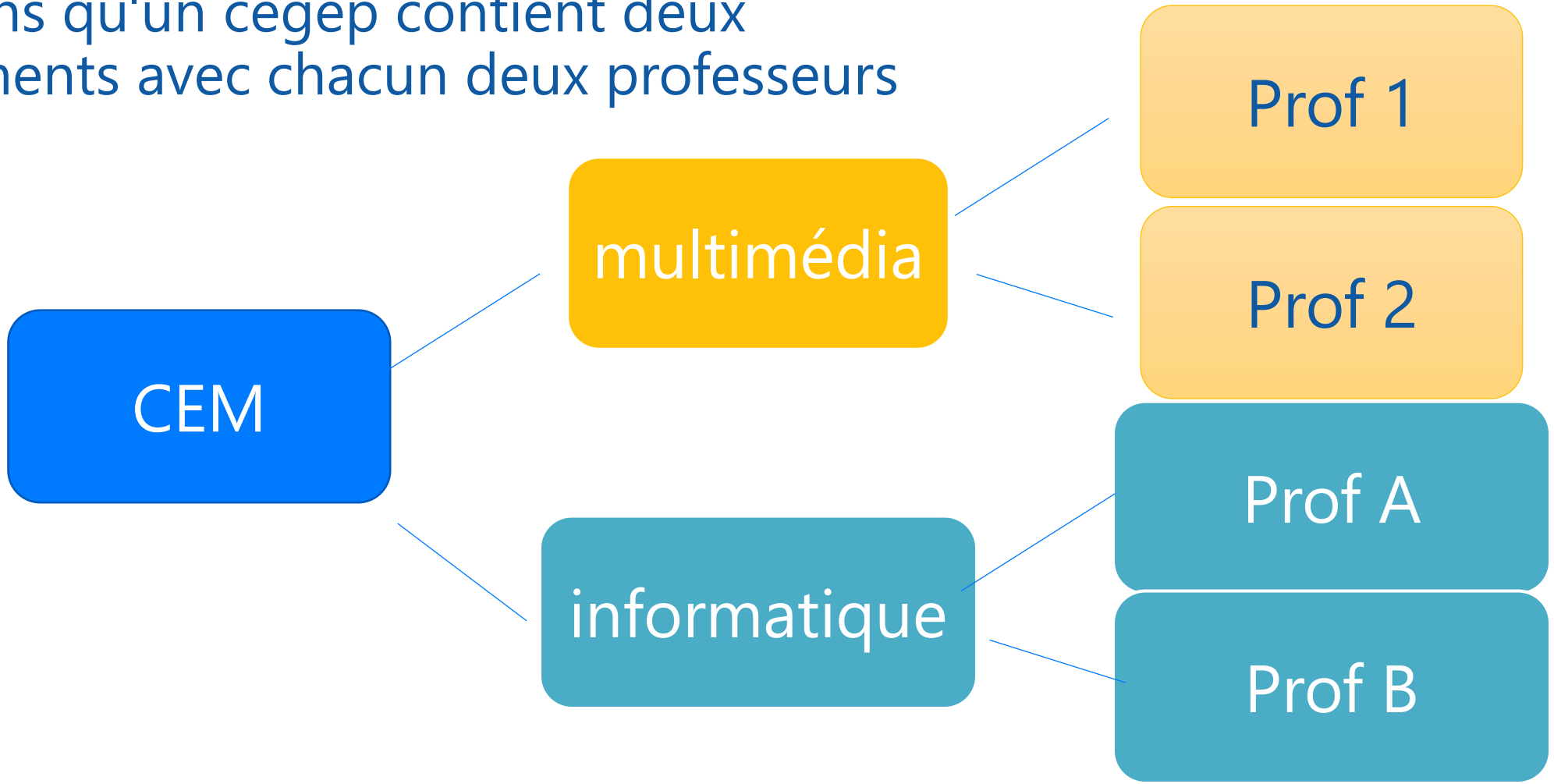
Agrégation

- Le cégep contient plusieurs département (composition) qui engagent chacun plusieurs professeurs (agrégation)



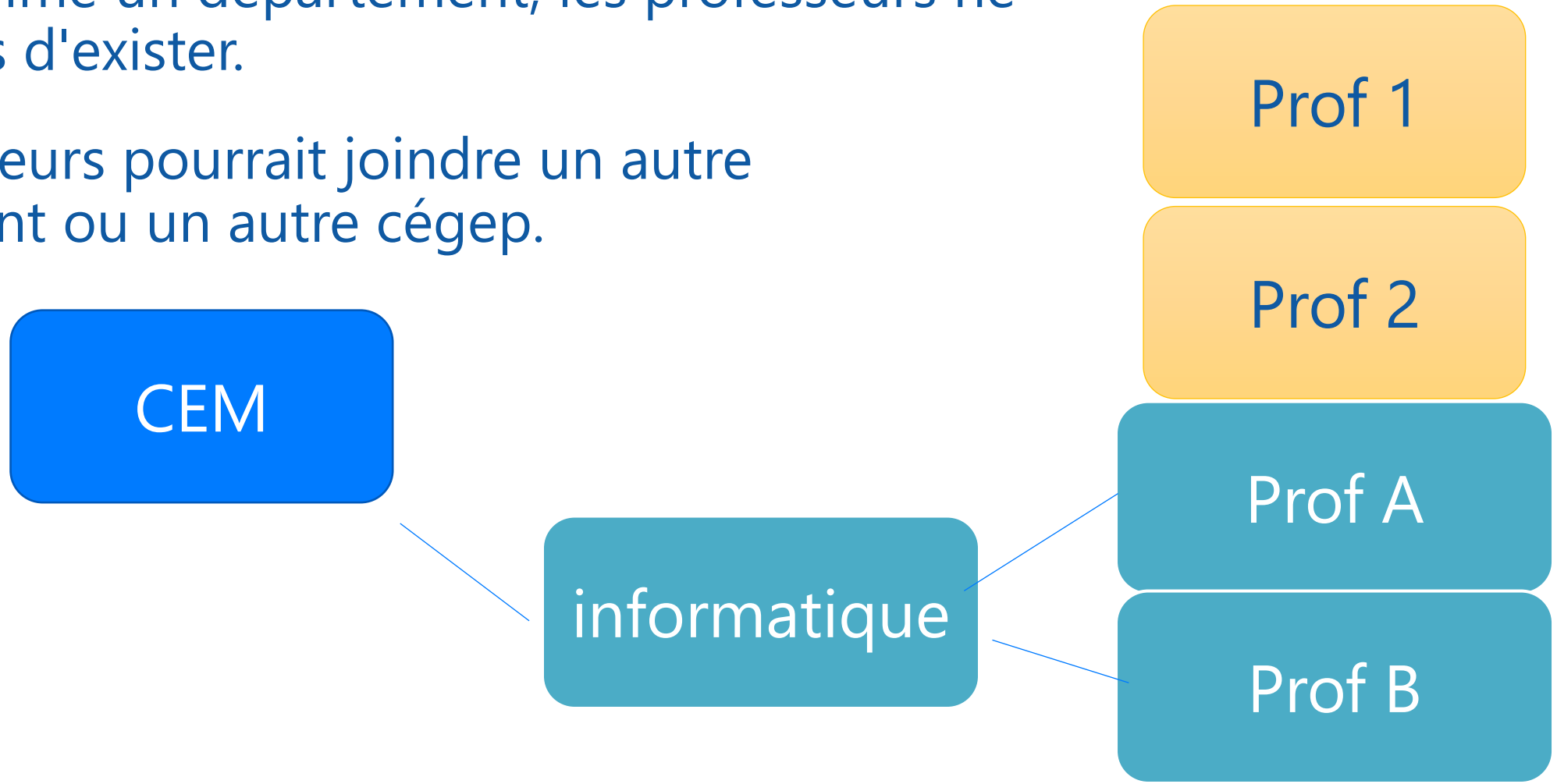
Ex : Agrégation

- Supposons qu'un cégep contient deux départements avec chacun deux professeurs



Ex : Agrégation

- Si on supprime un département, les professeurs ne cessent pas d'exister.
- Les professeurs pourrait joindre un autre département ou un autre cégep.



Ex : Agrégation vs Composition

- Par contraste, si on supprime le cégep, les départements cesseront d'exister (il s'agit alors de composition)