

# Notion d'association

# Objectifs

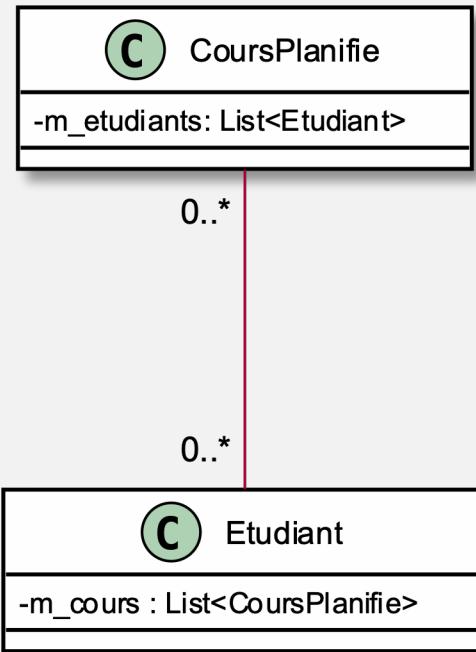
- Comprendre la notion d'association
- Représenter les associations et leurs cardinalités en UML
- Traduire les associations en code C#
- Représenter les interactions entre les objets

# Association

- On parle d'association quand une classe fait référence à une autre classe ou elle-même dans ses données membres
- On trouve généralement les associations avec le lien « **a un** » :
  - Une personne **a un** nom (string), un prénom (string), etc.
  - Une note **a un** étudiant et un cours
  - Une facture **a des** lignes de facture
  - Etc.

# Notation UML – Association plusieurs à plusieurs

- Association

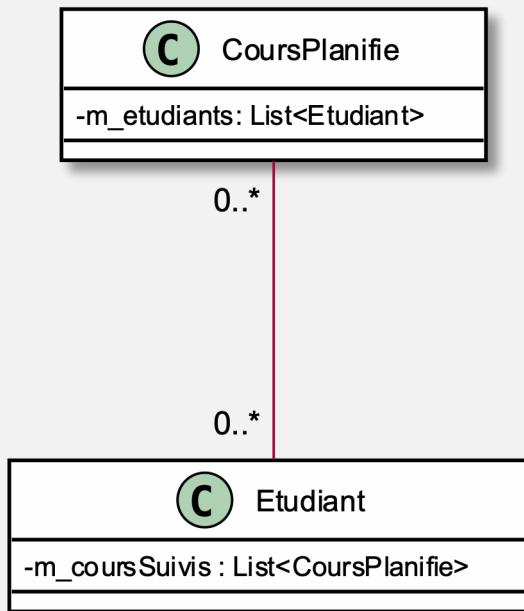


```
public class CoursPlanifie
{
    private List<Etudiant> m_etudiants;
    // ...
}

public class Etudiant
{
    private List<CoursPlanifie> m_coursSuivis;
    // ...
}
```

# Notation UML – Association plusieurs à plusieurs

- Association dirigée (navigabilité)

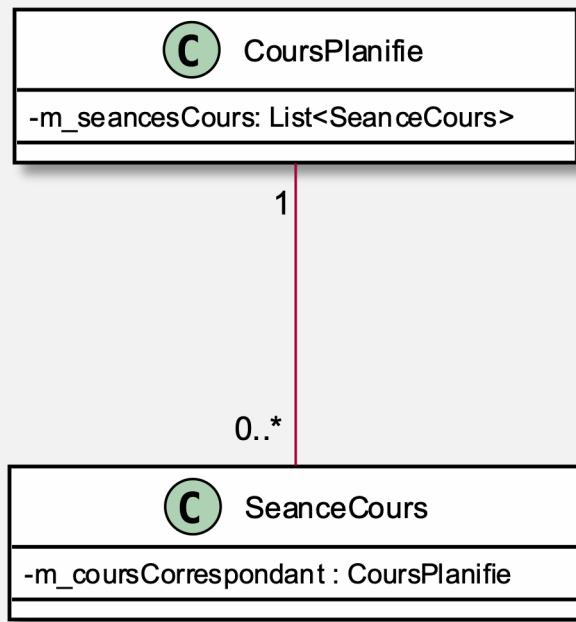


```
public class CoursPlanifie
{
    private List<Etudiant> m_etudiants;
    // ...
}

public class Etudiant
{
    // ...
}
```

# Notation UML – Association un à plusieurs

- Association

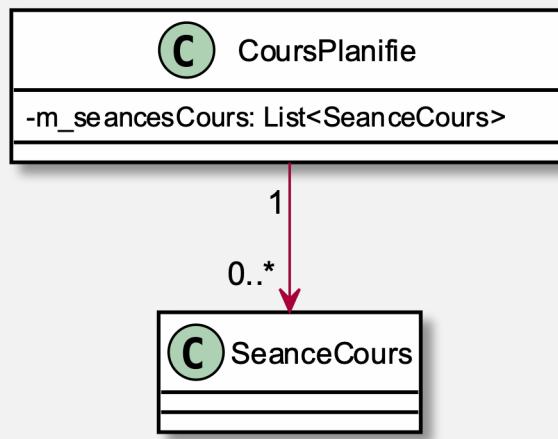


```
public class CoursPlanifie
{
    private List<SeanceCours> m_seancesCours;
    // ...
}

public class SeanceCours
{
    private CoursPlanifie m_coursCorrespondant;
    // ...
}
```

# Notation UML – Association un à plusieurs

- Association dirigée (navigabilité)

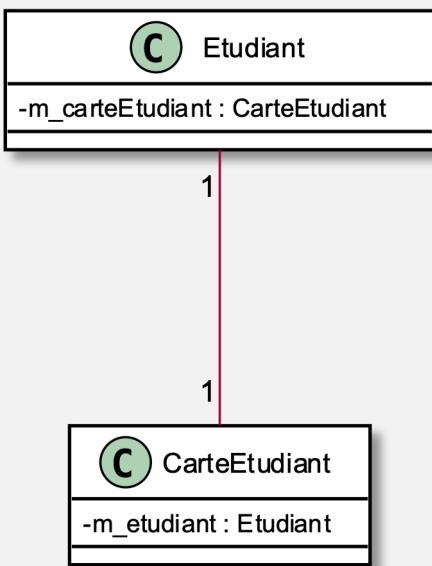


```
public class CoursPlanifie
{
    private List<SeanceCours> m_seancesCours;
    // ...
}

public class SeanceCours
{
    // ...
}
```

# Notation UML – Association un à un

- Association

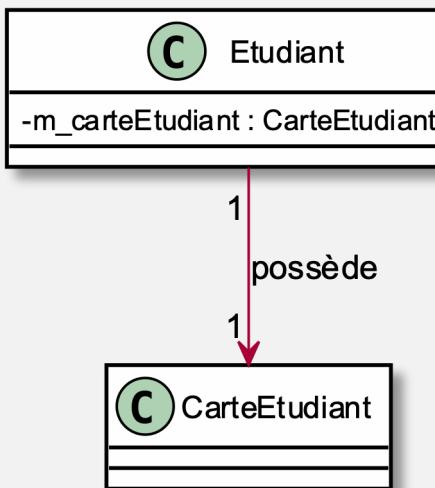


```
public class Etudiant
{
    private CarteEtudiant m_carteEtudiant;
    // ...
}

public class CarteEtudiant
{
    private Etudiant m_etudiant;
    // ...
}
```

# Notation UML – Association un à un

- Association dirigée (navigabilité)



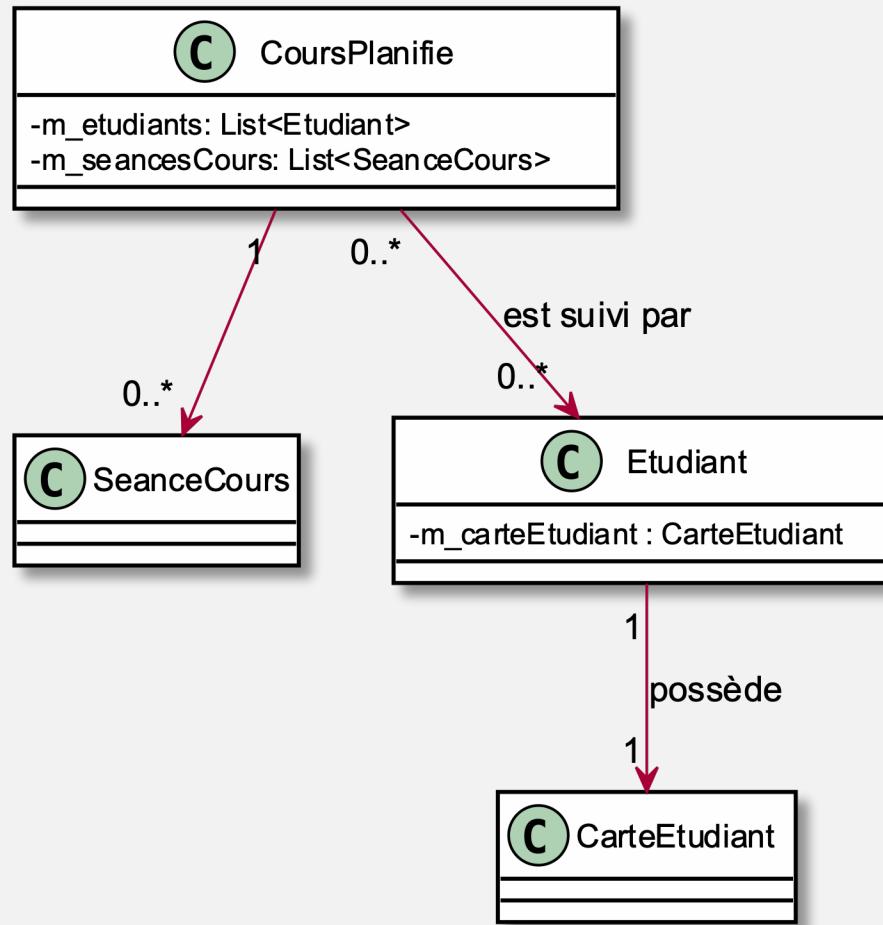
```
public class Etudiant
{
    private CarteEtudiant m_carteEtudiant;
    // ...
}

public class CarteEtudiant
{
    // ...
}
```

# Notation UML – Cardinalités

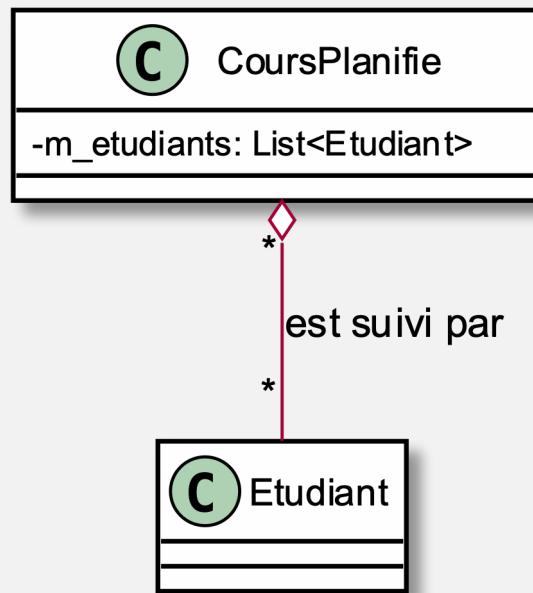
- 0..1 : est associé à zéro ou un élément
- 1 : est associé à un élément
- 0..\* ou n ou \* : est associé à zéro ou n éléments
- 1..\* ou + : est associé à un ou n éléments (contient donc au moins un élément)
- 1..8 : est associé à un ou deux ou ... ou huit éléments
- 8 : est associé à huit éléments

# Exemple partiel (Tout ce qui précède)



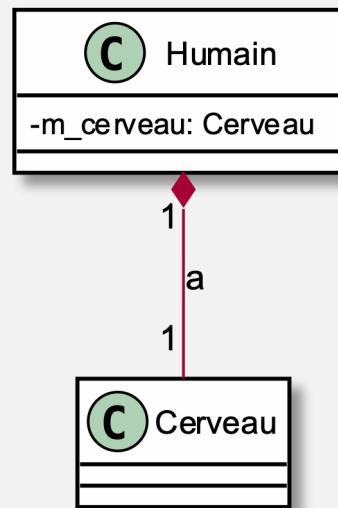
# Notation UML – Composition / agrégation

- Composition vs agrégation (Pas utilisées dans ce cours)
  - Agrégation : les éléments existent de manière indépendante.
    - Exemple : l'étudiant existe même si le cours n'existe pas



# Notation UML – Composition / agrégation

- Composition vs agrégation (Pas utilisées dans ce cours)
  - Composition : l'élément « enfant » n'existe pas s'il n'a pas de parent
    - Exemple : le cerveau n'existe pas seul, si on « efface » l'humain, on « efface » aussi le cerveau



# Interactions entre les objets ?

- En UML, il existe plusieurs types de diagrammes représentant les interactions
- Dans notre cas, nous allons utiliser les diagrammes de séquence
- Diagramme de séquence :
  - Représente les interactions entre acteurs selon l'ordre chronologique
    - Représente la séquence des événements
  - Cet représentant représente un scénario et non tous les cas possibles
  - Le temps est représenté verticalement
  - Les acteurs, généralement des objets, communiquent par envois de messages
  - Dans notre cas nous n'aurons que des messages synchrones avec ou sans retour

# Diagramme de séquence

