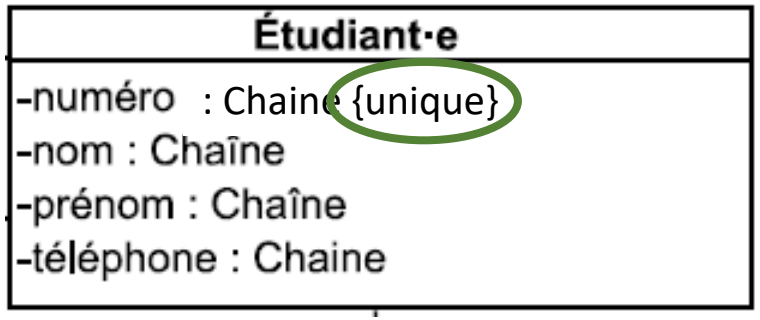
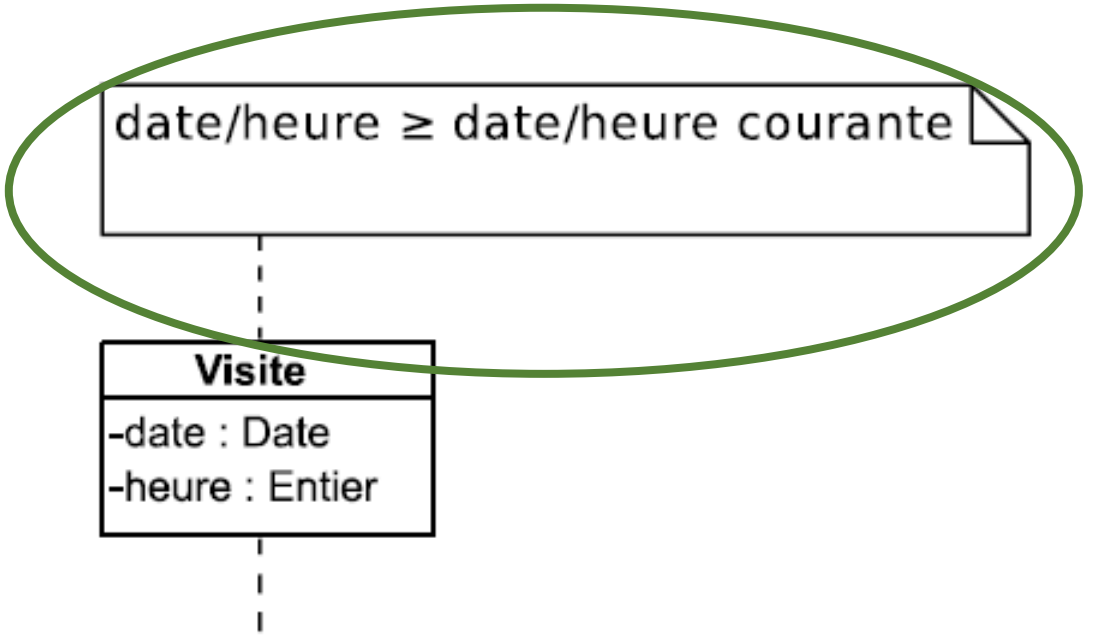
Rappels UML utiles
Diagramme de classes

1. Représenter TOUTES les contraintes

« Lorsqu'un étudiant est intéressé par une proposition, il prend un rendez-vous pour le visiter et on note son numéro d'étudiant, son nom, son prénom, son numéro de téléphone, … »



Pour les contraintes qui ne peuvent pas être exprimées graphiquement → notes textuelles ou OCL

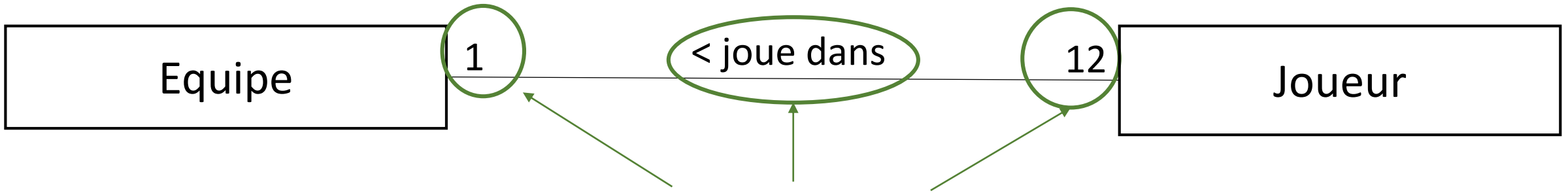


« Dans tous les cas, on ne conserve pas d'historique des visites. À la fin de la visite, le rendez-vous de la visite est supprimé »

2. Représenter une association en UML

De chaque côté navigable:
un nom de rôle et une multiplicité

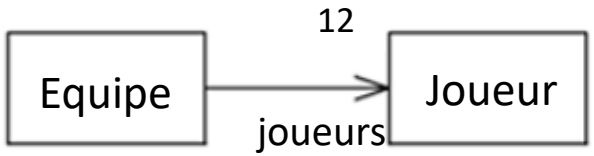
OU ALORS



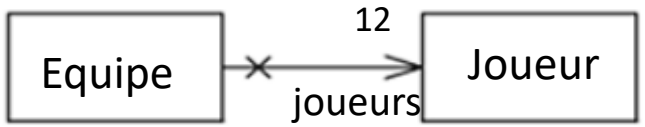
Un **nom d'association** (verbe, au milieu de l'association)
+ de chaque côté navigable: une multiplicité

3. Bien choisir: association unidirectionnelle ou bidirectionnelle ?

Unidirectionnelle



ou

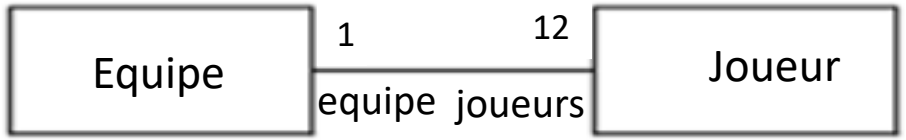


Une équipe a 12 joueurs.
A partir de l'équipe, on a accès aux joueurs qui la composent.

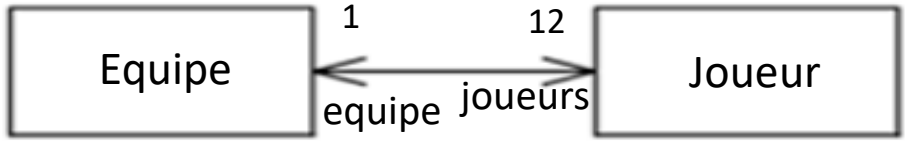
→ public class Equipe {
    private ArrayList<Joueurs> joueurs;

Par contre à partir d'un joueur, on n'a pas accès à son équipe ! (= pas de j.getEquipe() possible)

Bidirectionnelle



ou



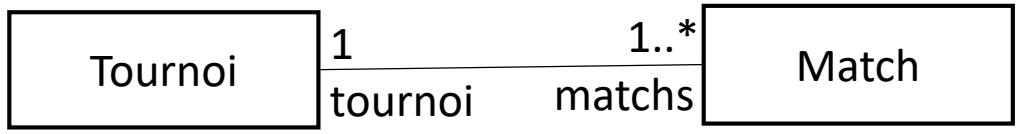
Une équipe a 12 joueurs. Un joueur a une seule équipe.
A partir de l'équipe, on a accès aux joueurs.

→ public class Equipe {
    private ArrayList<Joueurs> joueurs;

A partir de joueur on a accès à son équipe

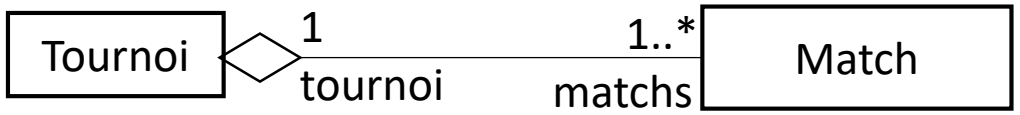
→ public class Joueur {
    private Equipe equipe;

4. Bien choisir : simple association, agrégation ou composition ?



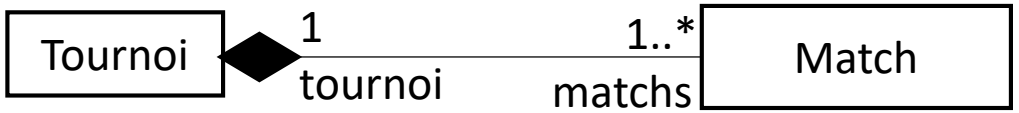
Un tournoi est associé à plusieurs matchs.
Un match est associé à un seul tournoi

Simple association



Un tournoi se compose d'un ensemble de matchs.
Un match appartient à un seul tournoi.
Si le tournoi est annulé, le match est quand même conservé.

Agrégation

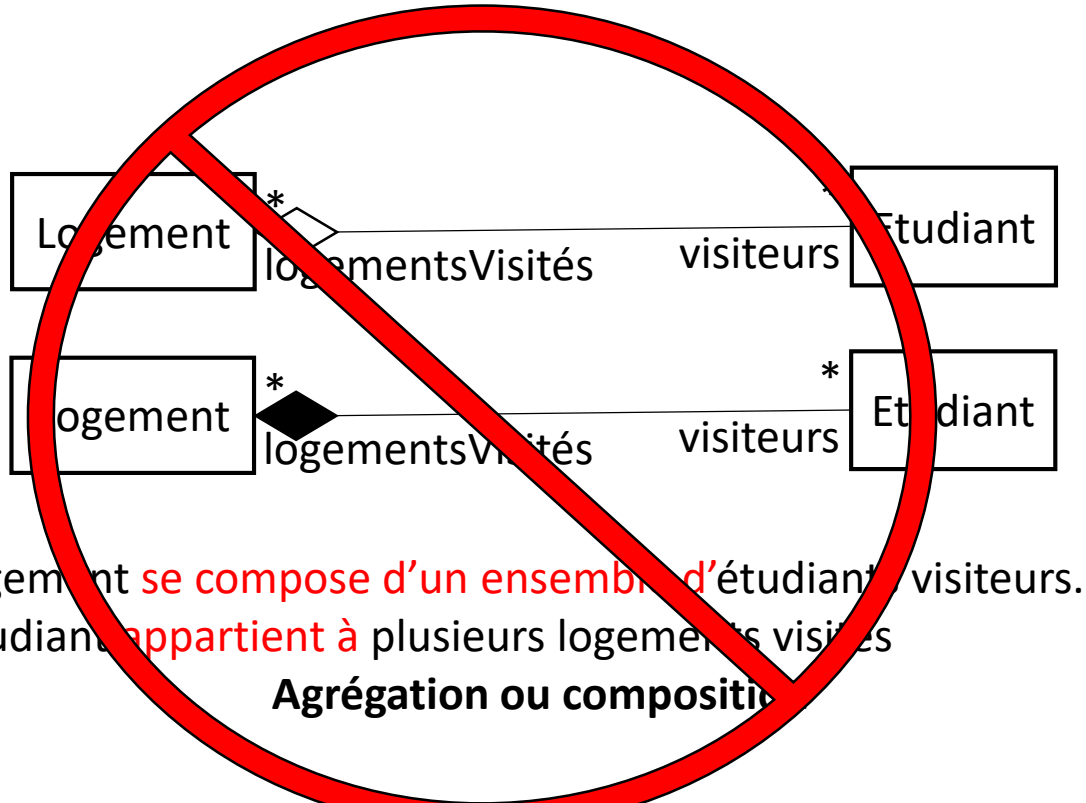


Un tournoi se compose d'un ensemble de matchs.
Un match appartient à un seul tournoi.
Si le tournoi est annulé, le match est aussi annulé.

Composition

Un logement est associé à plusieurs étudiants visiteurs.
Un etudiant est associé à plusieurs logement visités.

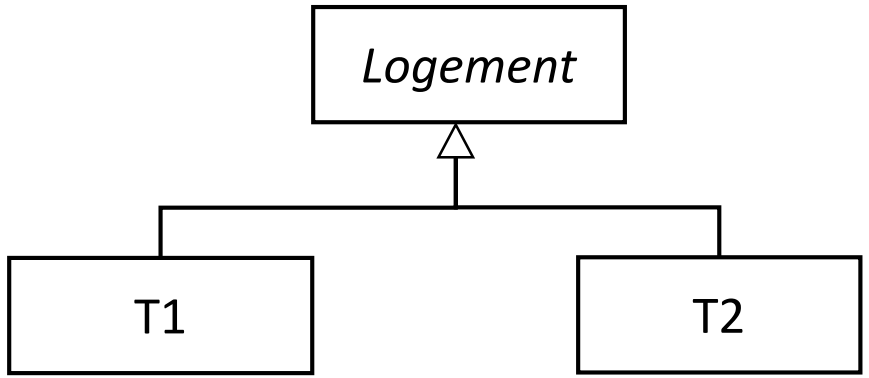
Simple association



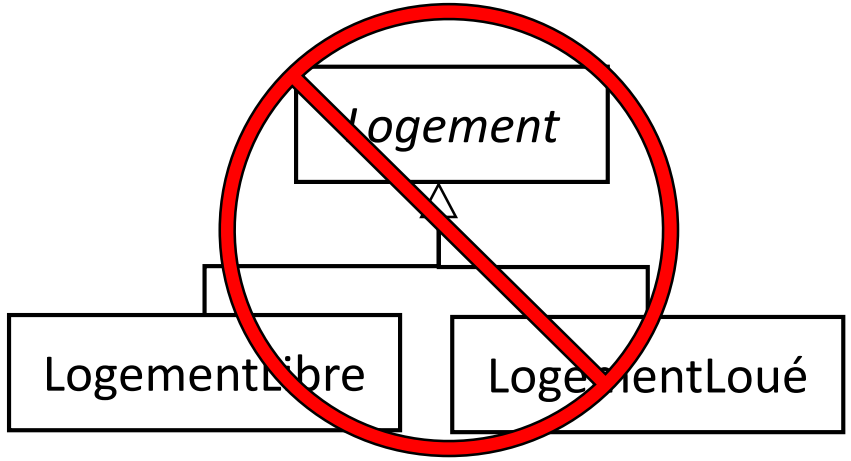
Un logement se compose d'un ensemble d'étudiants visiteurs.
Un etudiant appartient à plusieurs logements visités

Agrégation ou composition

5. Ajouter une classe ou juste un attribut ?

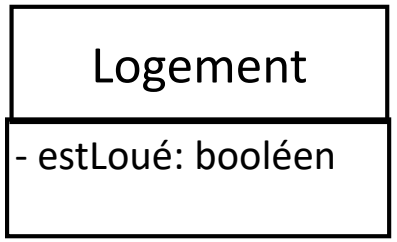


Un logement est un T1 ou un T2. La taille du logement n'est pas susceptible de changer souvent (caractéristique intrinsèque du logement)
→ T1 et T2 peuvent être des sous-classes de Logement



Problème : quand un logement change de statut (par ex. passe de libre à loué), il faut détruire l'objet Logement concerné et créer un nouvel objet d'une autre classe, alors qu'au final on parle du même logement …
→ Ici libre ou loué est susceptible de changer rapidement

Solution 1 : utiliser un attribut booléen **estLoué**



Solution 2 (mieux): utiliser une association entre Logement et Etudiant

```
public class Logement {
        private Etudiant locataire; // null si logement libre
```

6. Un attribut dans une classe est TOUJOURS de type primitif (ou classe enveloppe), JAMAIS de type classe