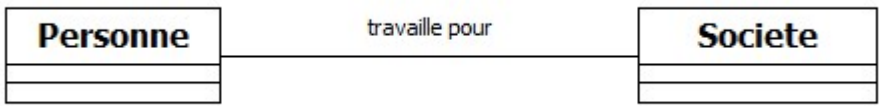
Relation d'association entre classes

Diagramme de classe

- Une application objet est constituée d'objets qui collaborent entre eux pour exécuter des traitements.
- Ces objets ne sont donc pas indépendants, ils ont des relations entre eux.
- Comme les objets ne sont que des instances de classes, il en résulte que les classes elles-mêmes sont reliées.
- Les relations peuvent être de 3 types :
  - Association
  - héritage,
  - dépendance.
- Ces relations peuvent être définies en UML à l'aide d'un diagramme de classe.

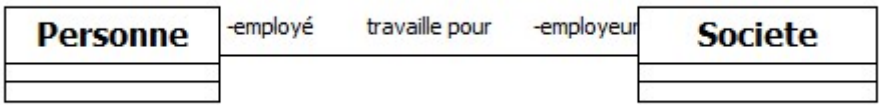
Association

- Deux classes entretiennent une relation d'association lorsque certains de leurs objets ont besoin s'envoyer des messages pour réaliser un traitement.
- Une association exprime une connexion sémantique bidirectionnelle entre deux classes.
- Elle peut avoir un nom qui décrit la nature de la relation.



Le rôle

- On peut être amené à préciser le rôle que joue chaque classe dans l'association, notamment lorsqu'un couple de classe est relié par deux associations sémantiquement distinctes:
  - Le rôle spécifie la fonction d'une classe pour une association donnée.
  - Les rôles sont optionnels, mais conseillés pour la lisibilité du modèle et du code qui sera généré.



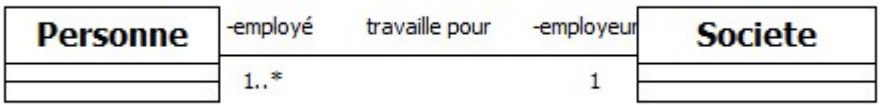
- La personne joue le rôle d'employé pour une société.
- La société joue le rôle d'employeur pour une personne

La cardinalité

- Elle précise le nombre d'instances qui participent à une relation :

Cardinalité	Rôle
1..1	Un et un seul
1	Un et un seul
0..1	Zéro ou un
m..n	De m à n
*	De zéro à plusieurs
0..*	De zéro à plusieurs
1..*	De un à plusieurs

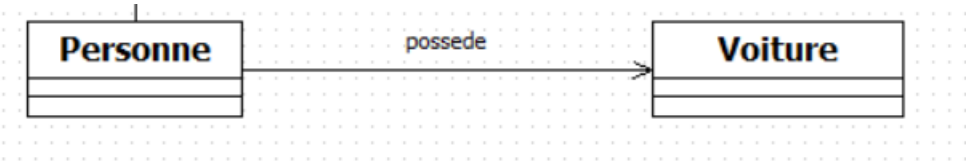
Exemple



- Une personne travaille pour une et une seule société.
- Une société à un ou plusieurs employés.

Navigabilité

- Par défaut, une association est « navigable » dans les deux sens.
- La navigabilité permet de restreindre cette navigation pour indiquer que les instances d'une classe ne connaissent pas les instances d'une autre.

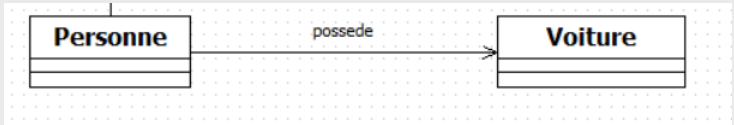


- Dans notre cas on veut exprimer que seule la personne a une visibilité sur une voiture
- La voiture n'a donc pas connaissance de son propriètaire.

Mapping d'une association

- La réalisation d'une association en Java s'effectue en spécifiant dans la classe un attribut de la classe avec laquelle on est en relation.

```
public class Personne
{
        // attributs ( private )
        private Voiture voiture ;
        …
}
```



- On utilisera le nom du rôle comme nom de l'attribut.

Agrégation

- L'agrégation est une association non symétrique, qui exprime un couplage fort et une relation de subordination.
  Elle représente une relation de type "ensemble / élément".

- Une agrégation peut notamment (mais pas nécessairement) exprimer :
  - qu'une classe (un "élément") fait partie d'une autre ("l'agrégat"),
  - qu'un changement d'état d'une classe, entraîne un changement d'état d'une autre,
  - qu'une action sur une classe, entraîne une action sur une autre.

- UML ne définit pas ce qu'est une relation de type "ensemble / élément", mais il permet cependant d'exprimer cette vue subjective de manière explicite.

- A un même moment, une instance d'élément agrégé peut être liée à plusieurs instances d'autres classes (l'élément agrégé peut être partagé).

- Une instance d'élément agrégé peut exister sans agrégat (et inversement) : les cycles de vies de l'agrégat et de ses éléments agrégés peuvent être indépendants.

Composition

- La composition est une agrégation forte (agrégation par valeur).
- Les cycles de vies des éléments (les "composants") et de l'agrégat sont liés :
  - si l'agrégat est détruit (ou copié), ses composants le sont aussi.
- A un même moment, une instance de composant ne peut être liée qu'à un seul agrégat.
- Les "objets composites" sont des instances de classes composées.