UNIFIED MODELING LANGUAGE - UML

DIAGRAMME DE CLASSES

🎓 MOOC (massive open online course)
🏛 Cursus Ingénieurs CentraleSupelec - 2024/2025

Idir AIT SADOUNE
idir.aitsadoune@centralesupelec.fr

OUTLINE

Back to the begin - Back to the outline

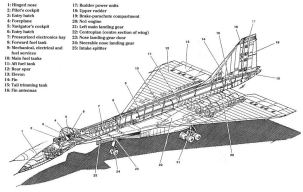
OUTLINE

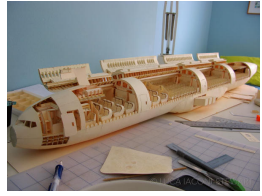
Back to the begin - Back to the outline

QU'EST CE QU'UN MODÈLE ?

- Un **modèle** est une représentation/simplification/abstraction de la réalité
  - mieux comprendre le sujet (le problème) étudié

Différents modèles d'un avion





CentraleSupélec

QU'EST CE QU'UN MODÈLE EN INFORMATIQUE ?

- Un **modèle** est une représentation simplifiée d'un système, établie dans un objectif.
- Un **modèle** doit permettre de répondre à des questions sur le système.
  - mieux comprendre les systèmes complexes.
- Un **modèle** a pour objectif de structurer :
  - les informations (données)
  - et les activités (traitements) d'un système.

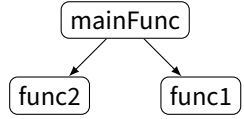
- L'**activité de modélisation** consiste à décrire de manière non ambiguë le fonctionnement futur du système afin d'en faciliter la réalisation ;
  - Spécification du système $\longrightarrow$ description des fonctionnalités du système ;
  - Conception de l'architecture $\longrightarrow$ description de la structure générale du système ;
  - Conception détaillée $\longrightarrow$ description des algorithmes et des structures de données ;

LES MÉTHODES DE CONCEPTION

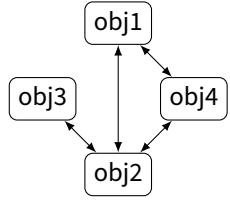
- **Une méthode de conception** est une démarche qui a pour objectif la formalisation des différentes étapes du développement d'un système/logiciel.
- **Une méthode de conception** vise à produire un système/logiciel le plus fidèle aux besoins du client.
- **Une méthode de conception** fournit une méthodologie.
- **Une méthode de conception** fournit également des notations standards permettant de concevoir des logiciels de qualité.

LES MÉTHODES DE CONCEPTION

- Il existe différentes manières pour classer les méthodes de conception. Nous citons les approches **fonctionnelle** et **orientée objet** :
  - **la conception fonctionnelle (procédurale)**
    un système est vu comme un ensemble hiérarchique de fonctions.



  - **la conception orientée objet**
    un système est un ensemble d'objets en interaction.



CONCEPTS IMPORTANTS DE L'APPROCHE OBJET

Encapsulation

Polymorphisme

Classe

Héritage

Agrégation

UML est une méthode de conception orientée objet

OUTLINE

Back to the begin - Back to the outline

POURQUOI UML ?

HISTORIQUE

- 1980 - 1994 $\longrightarrow$ apparition des langages orientés objets puis de nombreuses méthodes permettant de modéliser la conception.
- 1995 $\longrightarrow$ proposition d'une méthode commune $\longrightarrow$ Unified Method (UM).
- 1996 $\longrightarrow$ la méthode est significativement améliorée et est maintenant appelée Unified Modeling Language (UML).
- 1997 $\longrightarrow$ UML est adopté par l'Object Management Group (OMG) comme standard (UML 1.0).

QUI EST UML ?

- **UML** (Unified Modeling Language) est un langage (une notation graphique et une sémantique) de modélisation basé sur la notion d'objets.
- **UML** est considéré comme un outil pour :
  - présenter l'analyse d'un système ou les pré-requis le concernant ;
  - spécifier et concevoir des systèmes ;
  - communiquer sur des processus logiciels ou d'entreprise ;
  - documenter un système, un processus ou une organisation existants.
- **UML** se présente sous forme d'un ensemble de diagrammes qui peuvent être utilisés à différents niveaux d'un cycle de vie d'un logiciel.

LES DIAGRAMMES D'UML I

Diagrammes de structure ou diagrammes statiques

- Diagramme de classes (class diagram)
- Diagramme d'objets (object diagram)
- Diagramme de composants (component diagram)
- Diagramme de déploiement (deployment diagram)
- Diagramme des paquets (package diagram)
- Diagramme de structure composite (composite structure diagram)

Diagrammes de comportement

- Diagramme des cas d'utilisation (use-case diagram)
- Diagramme états-transitions (state machine diagram)

Diagramme d'activité (activity diagram)

LES DIAGRAMMES D'UML II

Diagrammes d'interaction ou diagrammes dynamiques

- Diagramme de séquence (sequence diagram)
- Diagramme de communication (collaboration) (communication diagram)
- Diagramme global d'interaction (interaction overview diagram)
- Diagramme de temps (timing diagram)

OUTLINE

Back to the begin - Back to the outline

C’EST QUOI UN DIAGRAMME DE CLASSES

Diagramme de classe
décrit les entités (classes et interfaces) constituant le système à modéliser et les relations statiques entre celles-ci.

DÉFINITION D'UN CLASSE UML

Classe et Objet

- Une classe regroupe des objets
- Les objets d'une même classe possédent des états et un comportement communs.
- Une classe peut représenter :
  - un concept concret (voiture, facture ...)
  - un concept abstrait (stratégie ...)
- En **UML**, une classe représente un type de classificateur.

Exemple

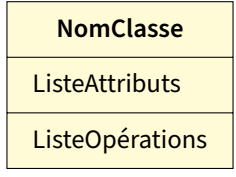
- Une classe : Personne
- Des objets : Jean, Marc, Marie, ...

DÉFINITION D'UN CLASSE UML

En **UML**, une classe est définie par un cadre rectangulaire comportant 3 zones :

1. Zone contenant le nom de la classe
2. Zone contenant les attributs de la classe
3. Zone contenant les opérations de la classe

Représentation d'une classe en UML



LES ÉLÉMENTS DE LA CLASSE EN UML

LES ATTRIBUTS D'UNE CLASSE UML

- Les caractéristiques (propriétés) sont représentées sous la forme d'attributs.
- Un attribut peut être représenté au moyen de deux notations différentes :
  1. en ligne pour des données primitives simples (entiers, réels, ...),
  2. relation pour exprimer des liens avec d'autres classes (*voir la section suivante*).

LES ÉLÉMENTS DE LA CLASSE EN UML

LES ATTRIBUTS EN LIGNE

- La forme générale d'un attribut en ligne dans une classe :

  `visibilité / nom : type multiplicité = valeur-initiale {propriétés}`

- Visibilité (par défaut public)

  +(public), -(private), #(protected), ~(package)

- Un attribut dérivé /
  c'est un attribut qui peut être calculé à partir d'autres attributs de la classe

- Multiplicité (par défaut 1)
  - `[val]` → il y a `val` fois cet attribut
  - `[min..max]` → il y a entre `min` et `max` fois cet attribut
  - `[min..*]` → il y a au moins `min` fois cet attribut
  - `[*]` → il y a un nombre indéterminé d'occurrence de cet attribut

CentraleSupélec

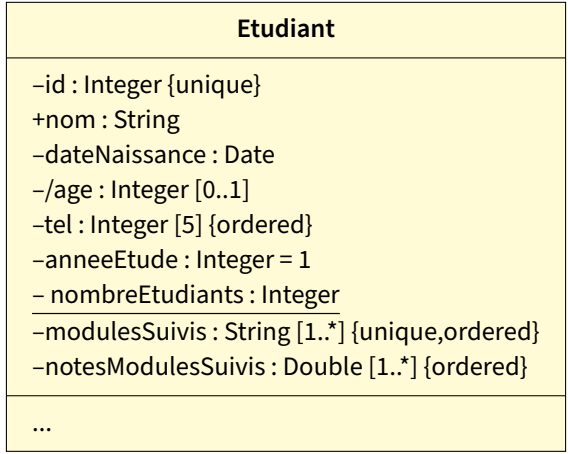
LES ÉLÉMENTS DE LA CLASSE EN UML

PROPRIÉTÉS

- `readOnly` $\longrightarrow$ valeur constante
- `static` $\longrightarrow$ attribut partagé entre tous les objets (souligné dans le diagramme **UML**)
- `unique` $\longrightarrow$ si multiplicité $>$ 1, valeurs distinctes
- `ordered` $\longrightarrow$ si multiplicité $>$ 1, valeurs ordonnées

LES ÉLÉMENTS DE LA CLASSE EN UML

EXEMPLE



CentraleSupélec

LES ÉLÉMENTS DE LA CLASSE EN UML

LES OPÉRATIONS D'UNE CLASSE UML

- Une opération permet d'invoquer une fonctionnalité offerte par une classe.

LES ÉLÉMENTS DE LA CLASSE EN UML

LES OPÉRATIONS D'UNE CLASSE UML

- Une opération permet d'invoquer une fonctionnalité offerte par une classe.
- La forme générale d'une opération dans une classe :

```
visibilité nom (paramètres) : type-retourné {propriétés}
```

LES ÉLÉMENTS DE LA CLASSE EN UML

LES OPÉRATIONS D'UNE CLASSE UML

- Une opération permet d'invoquer une fonctionnalité offerte par une classe.
- La forme générale d'une opération dans une classe :

  `visibilité nom (paramètres) : type-retourné {propriétés}`

- Un paramètre d'une opération est de la forme :

  `direction nom : type multiplicité = valeur-initiale {propriétés}`

  - la direction `in` → paramètre en entrée seule sans modification
  - la direction `out` → paramètre renvoyé en sortie seule
  - la direction `inout` → paramètre en entrée avec possibilité de modification

LES ÉLÉMENTS DE LA CLASSE EN UML

LES OPÉRATIONS D'UNE CLASSE UML

- Une opération permet d'invoquer une fonctionnalité offerte par une classe.
- La forme générale d'une opération dans une classe :

```
visibilité nom (paramètres) : type-retourné {propriétés}
```

- Un paramètre d'une opération est de la forme :

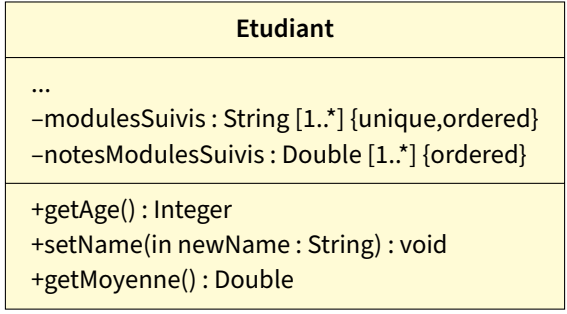
```
direction nom : type multiplicité = valeur-initiale {propriétés}
```

  - la direction `in` → paramètre en entrée seule sans modification
  - la direction `out` → paramètre renvoyé en sortie seule
  - la direction `inout` → paramètre en entrée avec possibilité de modification

La **visibilité** et les **propriétés** ont les mêmes définitions et les mêmes utilisations que dans le cas des attributs.

LES ÉLÉMENTS DE LA CLASSE EN UML

EXEMPLE



OUTLINE

Back to the begin - Back to the outline

LES ÉLÉMENTS DE LA CLASSE EN UML

LES ATTRIBUTS D'UNE CLASSE UML

- Les caractéristiques (propriétés) sont représentées sous la forme d'attributs.
- Un attribut peut être représenté au moyen de deux notations différentes :
  1. en ligne pour des données primitives simples (entiers, réels, ...),
  2. relation pour exprimer des liens avec d'autres classes.

LES RELATIONS D'ASSOCIATION

- Un système est constitué d'un ensemble d'objets qui interagissent entre eux. (des classes avec des relations entre elles).

LES RELATIONS D'ASSOCIATION

- Un système est constitué d'un ensemble d'objets qui interagissent entre eux. (des classes avec des relations entre elles).
- Une relation d'association entre deux classes peut être interprétée par :
  - "... a un ...",
  - "... est propriétaire de ...",
  - ou "... est composé de ...".

LES RELATIONS D'ASSOCIATION

- Un système est constitué d'un ensemble d'objets qui interagissent entre eux.
  (des classes avec des relations entre elles).
- Une relation d'association entre deux classes peut être interprétée par :
  - "... a un ...",
  - "... est propriétaire de ...",
  - ou "... est composé de ...".
- Une relation d'association doit être stable.
  (elle dure dans le temps et elle est non ponctuelle).

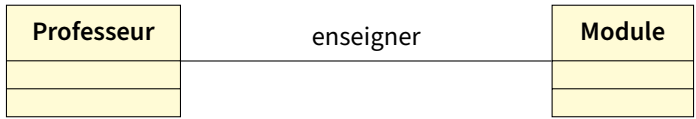
LES RELATIONS D'ASSOCIATION

- Un système est constitué d'un ensemble d'objets qui interagissent entre eux.
  (des classes avec des relations entre elles).
- Une relation d'association entre deux classes peut être interprétée par :
  - "... a un ...",
  - "... est propriétaire de ...",
  - ou "... est composé de ...".
- Une relation d'association doit être stable.
  (elle dure dans le temps et elle est non ponctuelle).
- Il est possible de nommer une relation d'association.

LES RELATIONS D'ASSOCIATION

- Un système est constitué d'un ensemble d'objets qui interagissent entre eux. (des classes avec des relations entre elles).
- Une relation d'association entre deux classes peut être interprétée par :
  - "... a un ...",
  - "... est propriétaire de ...",
  - ou "... est composé de ...".
- Une relation d'association doit être stable. (elle dure dans le temps et elle est non ponctuelle).
- Il est possible de nommer une relation d'association.

Exemple d'une relation d'association



CentraleSupélec

LES RELATIONS D'ASSOCIATION

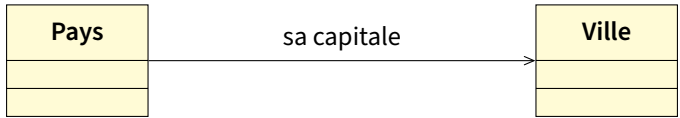
LA NAVIGABILITÉ DANS UNE ASSOCIATION

- Une association est par défaut bidirectionnelle

LES RELATIONS D'ASSOCIATION

LA NAVIGABILITÉ DANS UNE ASSOCIATION

- Une association est par défaut bidirectionnelle
- Une association peut être orientée (sens de navigation)



CentraleSupélec

LES RELATIONS D'ASSOCIATION

LES RÔLES DANS UNE ASSOCIATION

- En plus du nom, nous pouvons définir le rôle que joue chaque classe dans une association

LES RELATIONS D'ASSOCIATION

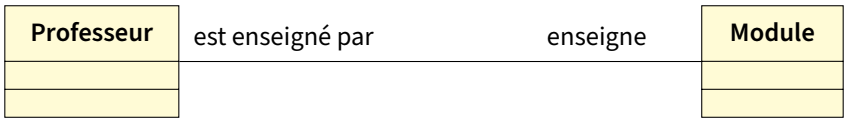
LES RÔLES DANS UNE ASSOCIATION

- En plus du nom, nous pouvons définir le rôle que joue chaque classe dans une association
  - nom de l'extrémité d'une association

LES RELATIONS D'ASSOCIATION

LES RÔLES DANS UNE ASSOCIATION

- En plus du nom, nous pouvons définir le rôle que joue chaque classe dans une association
  - nom de l'extrémité d'une association



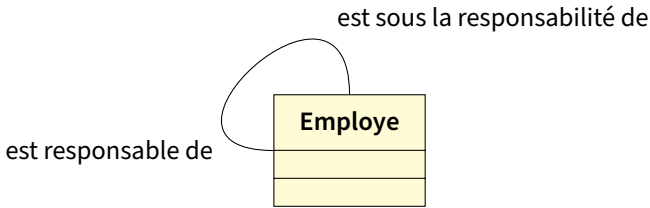
LES RELATIONS D'ASSOCIATION

D'AUTRES EXEMPLES

CentraleSupélec

LES RELATIONS D'ASSOCIATION

D'AUTRES EXEMPLES



CentraleSupélec

LES RELATIONS D'ASSOCIATION

MULTIPLICITÉ DANS UNE ASSOCIATION

- spécifie, dans une association, le nombre d'instance d'une classe qui sont liées à une instance de l'autre classe

LES RELATIONS D'ASSOCIATION

MULTIPLICITÉ DANS UNE ASSOCIATION

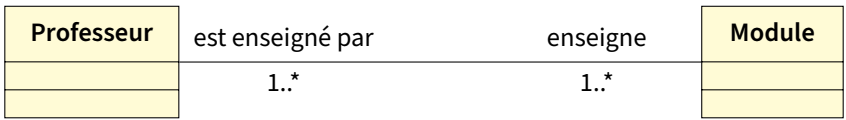
- spécifie, dans une association, le nombre d'instance d'une classe qui sont liées à une instance de l'autre classe
  - `1`, `0..1`, `M..N`, `*`, `0..*`, `1..*`

CentraleSupélec

LES RELATIONS D'ASSOCIATION

MULTIPLICITÉ DANS UNE ASSOCIATION

- spécifie, dans une association, le nombre d'instance d'une classe qui sont liées à une instance de l'autre classe
  - `1`, `0..1`, `M..N`, `*`, `0..*`, `1..*`



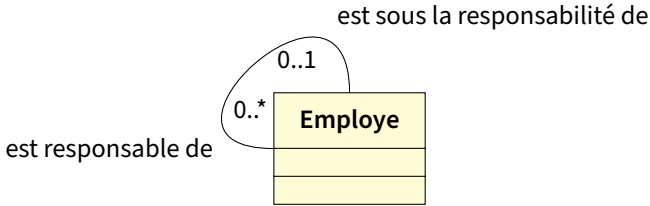
CentraleSupélec

LES RELATIONS D'ASSOCIATION

AUTRES EXEMPLES

LES RELATIONS D'ASSOCIATION

AUTRES EXEMPLES

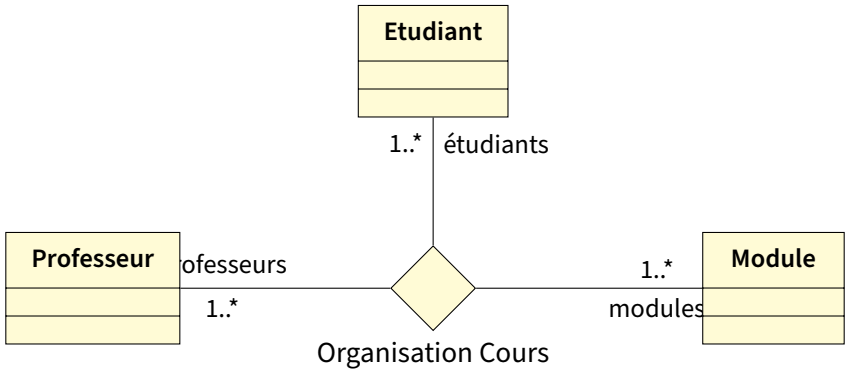


CentraleSupélec

LES RELATIONS D'ASSOCIATION

LES ASSOCIATIONS N-AIRES

- C'est une association qui relie plus de deux classes



LES RELATIONS D'ASSOCIATION

LES ASSOCIATIONS D'AGRÉGATION

- Une association d'agrégation est une relation entre classes qui est plus forte qu'une relation d'association classique.

LES RELATIONS D'ASSOCIATION

LES ASSOCIATIONS D'AGRÉGATION

- Une association d'agrégation est une relation entre classes qui est plus forte qu'une relation d'association classique.
- Une association d'agrégation peut être traduite par le verbe "posséder"

LES RELATIONS D'ASSOCIATION

LES ASSOCIATIONS D'AGRÉGATION

- Une association d'agrégation est une relation entre classes qui est plus forte qu'une relation d'association classique.
- Une association d'agrégation peut être traduite par le verbe "posséder"
- Une association d'agrégation correspond généralement à une relation tout ou parties (composé/composant)

LES RELATIONS D'ASSOCIATION

LES ASSOCIATIONS D'AGRÉGATION

- Une association d'agrégation est une relation entre classes qui est plus forte qu'une relation d'association classique.
- Une association d'agrégation peut être traduite par le verbe "posséder"
- Une association d'agrégation correspond généralement à une relation tout ou parties (composé/composant)
  - Il y a agrégation si le composé dépend de l'existence de ses composants.

LES RELATIONS D'ASSOCIATION

LES ASSOCIATIONS D'AGRÉGATION

- Une association d'agrégation est une relation entre classes qui est plus forte qu'une relation d'association classique.
- Une association d'agrégation peut être traduite par le verbe "posséder"
- Une association d'agrégation correspond généralement à une relation tout ou parties (composé/composant)
  - Il y a agrégation si le composé dépend de l'existence de ses composants.

Exemple d'une association d'agrégation

CentraleSupélec

LES RELATIONS D'ASSOCIATION

LES ASSOCIATIONS DE COMPOSITION

- L'association de composition est une association d'agrégation plus forte

LES RELATIONS D'ASSOCIATION

LES ASSOCIATIONS DE COMPOSITION

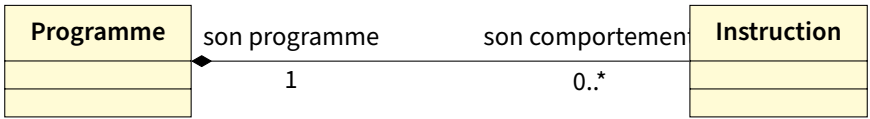
- L'association de composition est une association d'agrégation plus forte
- L'association de composition peut être traduite par le verbe "composer de"
  - Si on détruit une instance du composé, on détruit tous ses composants

LES RELATIONS D'ASSOCIATION

LES ASSOCIATIONS DE COMPOSITION

- L'association de composition est une association d'agrégation plus forte
- L'association de composition peut être traduite par le verbe "composer de"
  - Si on détruit une instance du composé, on détruit tous ses composants

Exemple d'une association de composition



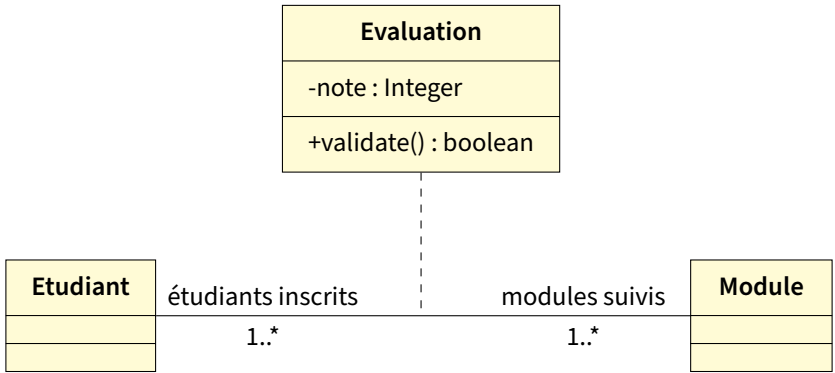
CentraleSupélec

LES CLASSES D'ASSOCIATION

- C'est une association possédant un nom, des attributs et des méthodes

LES CLASSES D'ASSOCIATION

- C'est une association possédant un nom, des attributs et des méthodes



OUTLINE

Back to the begin - Back to the outline

LA GÉNÉRALISATION/SPÉCIALISATION

- Une relation de généralisation permet d'indiquer qu'une classe constitue un cas plus général d'une autre classe.

LA GÉNÉRALISATION/SPÉCIALISATION

- Une relation de généralisation permet d'indiquer qu'une classe constitue un cas plus général d'une autre classe.
- Une relation de généralisation est utilisée pour extraire des propriétés/opérations communes à plusieurs classes afin de les regrouper dans une super-classe.

LA GÉNÉRALISATION/SPÉCIALISATION

- Une relation de généralisation permet d'indiquer qu'une classe constitue un cas plus général d'une autre classe.
- Une relation de généralisation est utilisée pour extraire des propriétés/opérations communes à plusieurs classes afin de les regrouper dans une super-classe.
- A l'inverse, une relation de spécialisation permet de décrire les spécificités d'un cas particulier d'une classe dans une sous-classe (relation "... est un ...").

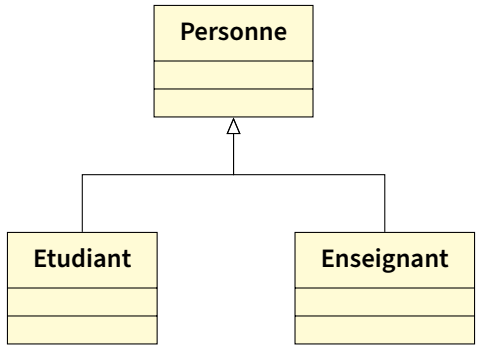
LA GÉNÉRALISATION/SPÉCIALISATION

- Une relation de généralisation permet d'indiquer qu'une classe constitue un cas plus général d'une autre classe.
- Une relation de généralisation est utilisée pour extraire des propriétés/opérations communes à plusieurs classes afin de les regrouper dans une super-classe.
- A l'inverse, une relation de spécialisation permet de décrire les spécificités d'un cas particulier d'une classe dans une sous-classe (relation "... est un ...").

Une sous classe possède, en plus de ses propres propriétés/opérations, les propriétés/opérations de ses super-classes.

LA GÉNÉRALISATION/SPÉCIALISATION

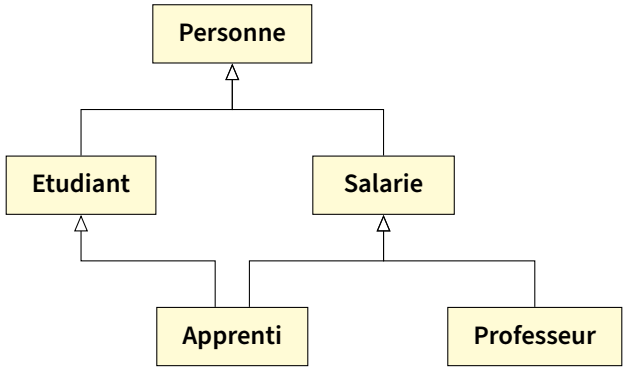
EXEMPLE



CentraleSupélec

LA GÉNÉRALISATION/SPÉCIALISATION

UN AUTRE EXEMPLE



CentraleSupélec

OUTLINE

Back to the begin - Back to the outline

LES CLASSES ABSTRAITES

- Une classe abstraite permet de regrouper des propriétés et des fonctionnalités communes à différents types d'objets.

LES CLASSES ABSTRAITES

- Une classe abstraite permet de regrouper des propriétés et des fonctionnalités communes à différents types d'objets.
- Une classe abstraite est une classe qui ne peut pas être instanciée

LES CLASSES ABSTRAITES

- Une classe abstraite permet de regrouper des propriétés et des fonctionnalités communes à différents types d'objets.
- Une classe abstraite est une classe qui ne peut pas être instanciée
- Une classe abstraite peut avoir des méthodes non implémentées.
  - ➠ Une classe possédant une méthode abstraite doit être déclarée abstraite

LES CLASSES ABSTRAITES

L'HÉRITAGE D'UNE CLASSE ABSTRAITE

- Une classe qui hérite d'une classe abstraite peut implémenter les méthodes abstraites de la classe mère.
  (sauf si la classe fille est également une classe abstraite).

LES CLASSES ABSTRAITES

L'HÉRITAGE D'UNE CLASSE ABSTRAITE

- Une classe qui hérite d'une classe abstraite peut implémenter les méthodes abstraites de la classe mère.
  (sauf si la classe fille est également une classe abstraite).

- Une classe qui hérite d'une classe abstraite peut ré-implémenter les méthodes déjà implémentées.

LES CLASSES ABSTRAITES

L'HÉRITAGE D'UNE CLASSE ABSTRAITE

- Une classe qui hérite d'une classe abstraite peut implémenter les méthodes abstraites de la classe mère.
  (sauf si la classe fille est également une classe abstraite).

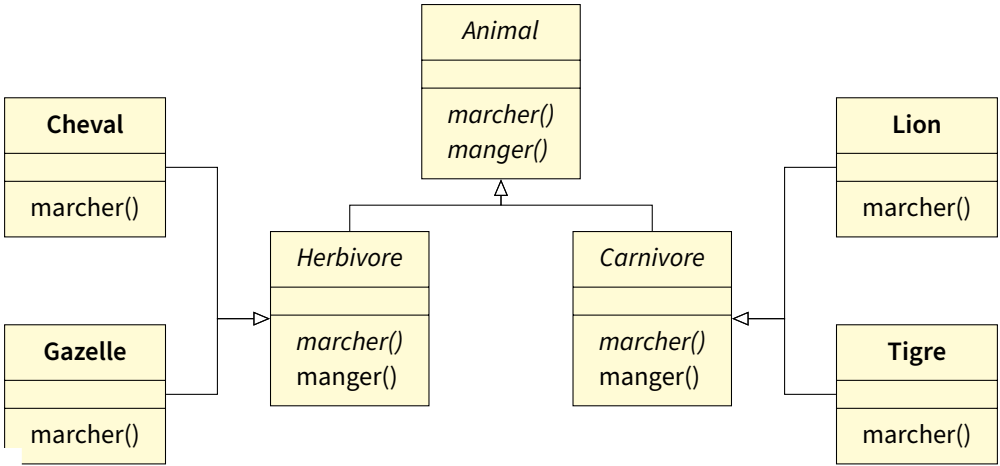
- Une classe qui hérite d'une classe abstraite peut ré-implémenter les méthodes déjà implémentées.

En **UML**, la classe/méthode abstraite est notée en *italique*

CentraleSupélec

LES CLASSES ABSTRAITES

EXEMPLE



LES INTERFACES

- Une interface est un classificateur contenant des déclarations de propriétés et de méthodes abstraites assurant un service cohérent.

LES INTERFACES

- Une interface est un classificateur contenant des déclarations de propriétés et de méthodes abstraites assurant un service cohérent.
  - Une interface est comme une classe abstraite dans laquelle aucune méthode ne serait implémentée (donc ne peut pas être instanciée).

LES INTERFACES

- Une interface est un classificateur contenant des déclarations de propriétés et de méthodes abstraites assurant un service cohérent.
  - Une interface est comme une classe abstraite dans laquelle aucune méthode ne serait implémentée (donc ne peut pas être instanciée).
  - Une interface permet de définir un ensemble de services sans se préoccuper de leurs implémentations (méthodes abstraites).

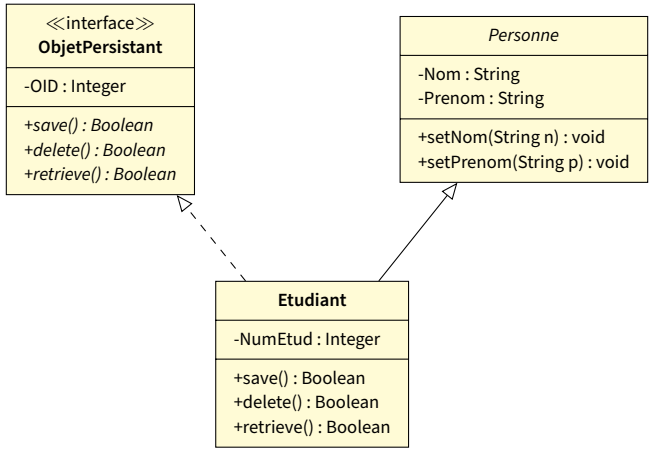
LES INTERFACES

- Une interface est un classificateur contenant des déclarations de propriétés et de méthodes abstraites assurant un service cohérent.
  - Une interface est comme une classe abstraite dans laquelle aucune méthode ne serait implémentée (donc ne peut pas être instanciée).
  - Une interface permet de définir un ensemble de services sans se préoccuper de leurs implémentations (méthodes abstraites).

- Une interface représente un contrat que doit respecter chaque classe implémentant cette interface.
  - Une classe implémentant une interface doit obligatoirement implémenter toutes les méthodes déclarées dans l'interface
    (à moins qu'elle ne soit elle-même déclarée abstraite !)

LES INTERFACES

EXEMPLE



CentraleSupélec

CONCLUSION
CLASSE ABSTRAITE vs INTERFACE

- La notion d'interface est utilisée pour représenter des propriétés transverses de classes.

CONCLUSION
CLASSE ABSTRAITE vs INTERFACE

- La notion d'interface est utilisée pour représenter des propriétés transverses de classes.
- La notion de classe abstraite doit être étendue et spécialisée.

CONCLUSION
CLASSE ABSTRAITE vs INTERFACE

- La notion d'interface est utilisée pour représenter des propriétés transverses de classes.
- La notion de classe abstraite doit être étendue et spécialisée.
- Une interface nous dit juste que telle classe possède telle propriété et assure tel service, indépendamment de ce qu'elle représente.

THANK YOU

Back to the begin - Back to the outline