Génie Logiciel

4 - Diagramme de classes UML

Idir AIT SADOUNE

CentraleSupélec - Département Informatique
3, rue Joliot-Curie, 91192 Gif-sur-Yvette cedex
<http://idir.aitsadoune.free.fr>
<idir.aitsadoune@centralesupelec.fr>

Année scolaire 2015-2016, Séquence 4

CentraleSupélec

La vue statique

- La vue **statique** d'UML permet de décrire la structure du système
  - classes avec attributs (état), opérations (comportement)
  - relations entre classes
  - objets et liens (un état particulier)

La vue statique

- La vue **statique** d'UML permet de décrire la structure du système
  - classes avec attributs (état), opérations (comportement)
  - relations entre classes
  - objets et liens (un état particulier)
- Les moyens disponibles dans UML pour la vue **statique**

La vue statique

- La vue **statique** d'UML permet de décrire la structure du système
  - classes avec attributs (état), opérations (comportement)
  - relations entre classes
  - objets et liens (un état particulier)
- Les moyens disponibles dans UML pour la vue **statique**
  - ***Diagramme de classe*** : décrit une abstraction du système à l'aide de classes et des relations (description conceptuelle)

La vue statique

- La vue **statique** d'UML permet de décrire la structure du système
  - classes avec attributs (état), opérations (comportement)
  - relations entre classes
  - objets et liens (un état particulier)
- Les moyens disponibles dans UML pour la vue **statique**
  - ***Diagramme de classe*** : décrit une abstraction du système à l'aide de classes et des relations (description conceptuelle)
  - ***Diagramme d'objet*** : décrit l'état du système à un moment donné avec des objets et des liens (une instance du diagramme de classe)

La vue statique

- La vue **statique** d'UML permet de décrire la structure du système
  - classes avec attributs (état), opérations (comportement)
  - relations entre classes
  - objets et liens (un état particulier)
- Les moyens disponibles dans UML pour la vue **statique**
  - ***Diagramme de classe*** : décrit une abstraction du système à l'aide de classes et des relations (description conceptuelle)
  - ***Diagramme d'objet*** : décrit l'état du système à un moment donné avec des objets et des liens (une instance du diagramme de classe)
  - et les diagrammes de ***structure composite***, de ***paquetage***, de ***composant*** et de ***déploiement***...

Plan

Plan

Classe en UML

Classe et Objet : définition

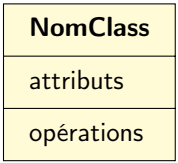
- La ***classe*** est le moule (type) permettant de créer des ***objets*** (instances) appartenant à cette classe.

Classe en UML

Classe et Objet : définition

- La ***classe*** est le moule (type) permettant de créer des ***objets*** (instances) appartenant à cette classe.

Classe en UML



Attributs

- La forme générale d'un attribut d'une classe :
  [visibilité] nom [: type] [multiplicité] [= valeur-initiale] [propriétés]

Attributs

- La forme générale d'un attribut d'une classe :
  [visibilité] nom [: type] [multiplicité] [= valeur-initiale] [propriétés]
- visibilité (par défaut public) :
  - +(public), -(private), #(protected), ~(package)

Attributs

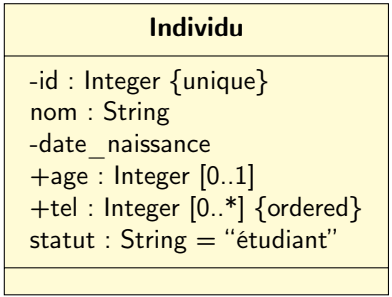
- La forme générale d'un attribut d'une classe :
  [visibilité] nom [: type] [multiplicité] [= valeur-initiale] [propriétés]
- visibilité (par défaut public) :
  - +(public), -(private), #(protected), ~(package)
- multiplicité (par défaut 1) :
  - [val] : il y a *val* fois cet attribut
  - [min..max] : il y a entre *min* et *max* fois cet attribut
  - [min..*] : il y a au moins *min* fois cet attribut

Attributs

- La forme générale d'un attribut d'une classe :
  [visibilité] nom [: type] [multiplicité] [= valeur-initiale] [propriétés]
- visibilité (par défaut public) :
  - +(public), -(private), #(protected), ~(package)
- 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
- propriétés
  - readOnly : valeur constante pour l'objet
  - static : attribut commun à tous les objets
  - unique : si multiplicité > 1, valeurs distinctes
  - ordered : si multiplicité > 1, valeurs ordonnées

Attributs

Exemple



Opérations

- La forme générale d'une opération dans une classe :
  [visibilité] nom [(liste-paramètres)] [: type-retour] [propriétés]

Opérations

- La forme générale d'une opération dans une classe :
  [visibilité] nom [(liste-paramètres)] [: type-retour] [propriétés]
- visibilité (par défaut public) :
  - +(public), -(private), #(protected), ~(package)

Opérations

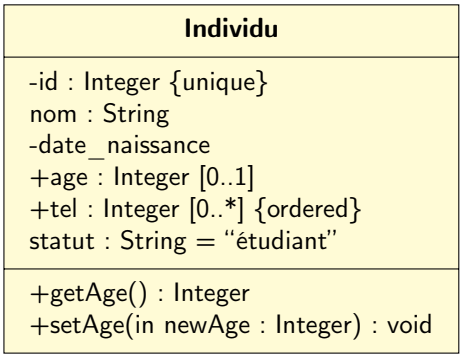
- La forme générale d'une opération dans une classe :
  [visibilité] nom [(liste-paramètres)] [: type-retour] [propriétés]
- visibilité (par défaut public) :
  - +(public), -(private), #(protected), ~(package)
- La liste des paramètres est de la forme :
  [direction] nom : type [= valeur-par-défaut]
  - Les directions sont **in**, **out** et **inout** (entrée seule sans modification, sortie seule, entrée et sortie).

Opérations

- La forme générale d'une opération dans une classe :
  [visibilité] nom [(liste-paramètres)] [: type-retour] [propriétés]
- visibilité (par défaut public) :
  - +(public), -(private), #(protected), ~(package)
- La liste des paramètres est de la forme :
  [direction] nom : type [= valeur-par-défaut]
  - Les directions sont **in**, **out** et **inout** (entrée seule sans modification, sortie seule, entrée et sortie).
- propriétés
  - readOnly : valeur constante pour l'objet
  - static : attribut commun à tous les objets
  - unique : si multiplicité > 1, valeurs distinctes
  - ordered : si multiplicité > 1, valeurs ordonnées

Opérations

Exemple



Plan

Les relations d'association

- Un système est constitué d'un ensemble de classes qui interagissent entre elles (en relation).

Les relations d'association

- Un système est constitué d'un ensemble de classes qui interagissent entre elles (en relation).
- Une relation d'**association** entre deux classes doit être stable (elle dure dans le temps et elle est non ponctuelle).

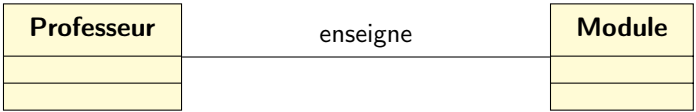
Les relations d'association

- Un système est constitué d'un ensemble de classes qui interagissent entre elles (en relation).
- Une relation d'**association** entre deux classes 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 de classes qui interagissent entre elles (en relation).
- Une relation d'**association** entre deux classes doit être stable (elle dure dans le temps et elle est non ponctuelle).
- Il est possible de nommer une relation d'**association**.

Exemple



Les relations d'association

- Une association est par défaut bidirectionnelle

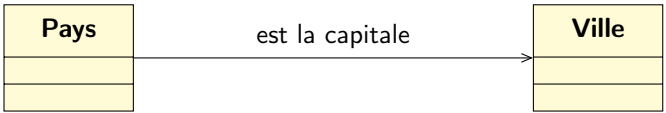
Les relations d'association

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

Les relations d'association

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

Exemple



Le rôle dans une association

- En plus du nom, nous pouvons définir le **rôle** que joue chaque classe dans l'**association**
  - nom de l'extrémité d'une association
  - indispensable pour les associations réflexives

Le rôle dans une association

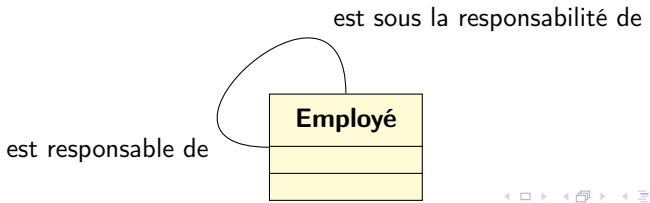
- En plus du nom, nous pouvons définir le **rôle** que joue chaque classe dans l'**association**
  - nom de l'extrémité d'une association
  - indispensable pour les associations réflexives

Exemple

Le rôle dans une association

- En plus du nom, nous pouvons définir le **rôle** que joue chaque classe dans l'**association**
  - nom de l'extrémité d'une association
  - indispensable pour les associations réflexives

Exemple



Multiplicité

- spécifie combien d'instance d'une classe sont liées à une instance d'une autre classe
  - 1, 0..1, M..N, *, 0..*, 1..*

Multiplicité

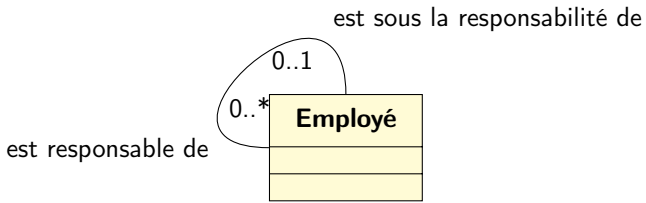
- spécifie combien d'instance d'une classe sont liées à une instance d'une autre classe
  - 1, 0..1, M..N, *, 0..*, 1..*

Exemple

Multiplicité

- spécifie combien d'instance d'une classe sont liées à une instance d'une autre classe
  - 1, 0..1, M..N, *, 0..*, 1..*

Exemple



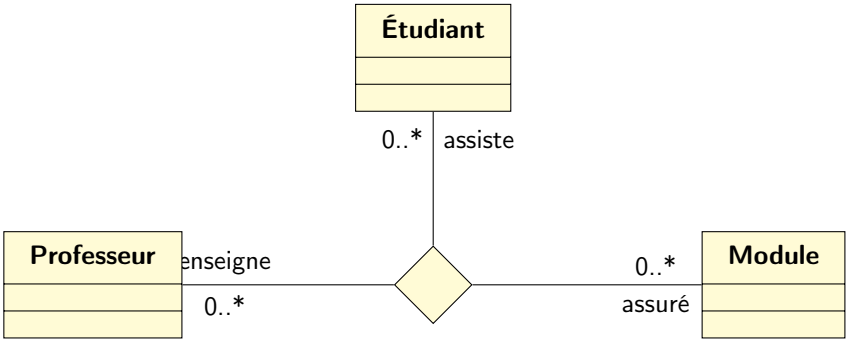
Associations n-aires

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

Associations n-aires

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

Exemple



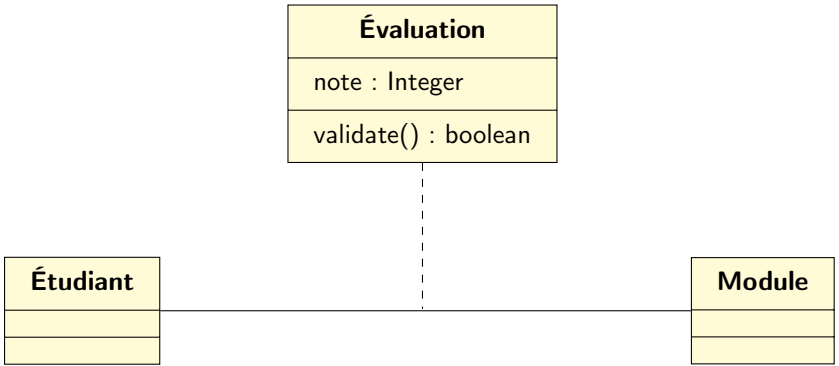
Classe d'Association

- Un attribut (une méthode) d'une relation est un attribut (une méthode) qui caractérise la relation

Classe d'Association

- Un attribut (une méthode) d'une relation est un attribut (une méthode) qui caractérise la relation

Exemple



Agrégation

- Une relation d'agrégation est un cas particulier de relation d'association.

Agrégation

- Une relation d'agrégation est un cas particulier de relation d'association.
- Elle représente un lien de subordination correspondant au verbe "**possède**"

Agrégation

- Une relation d'agrégation est un cas particulier de relation d'association.
- Elle représente un lien de subordination correspondant au verbe "**possède**"
- Elle correspond généralement à une relation tout ou parties (composé/composant)
  - Il y a agrégation si le **tout** dépend de l'existence de ses **parties** pour avoir un sens.

Agrégation

- Une relation d'agrégation est un cas particulier de relation d'association.
- Elle représente un lien de subordination correspondant au verbe "**possède**"
- Elle correspond généralement à une relation tout ou parties (composé/composant)
  - Il y a agrégation si le **tout** dépend de l'existence de ses **parties** pour avoir un sens.

Exemple

Composition

- La composition est une agrégation avec un lien plus fort
  - Si on détruit une instance d'une classe, on détruit tout ses composants

Composition

- La composition est une agrégation avec un lien plus fort
  - Si on détruit une instance d'une classe, on détruit tout ses composants
- Elle représente un lien correspondant au verbe "**est composé de**"

Composition

- La composition est une agrégation avec un lien plus fort
  - Si on détruit une instance d'une classe, on détruit tout ses composants
- Elle représente un lien correspondant au verbe "**est composé de**"

Exemple

Plan

Définitions

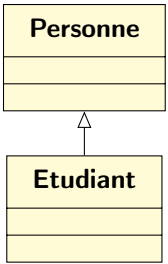
- Généralisation : factoriser les propriétés/opérations de plusieurs classes dans une super-classe.

Définitions

- Généralisation : factoriser les propriétés/opérations de plusieurs classes dans une super-classe.
- Spécialisation : capturer les particularités des instances d'une sous-classe.

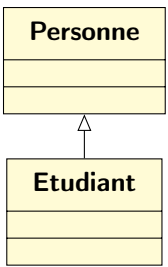
Définitions

- Généralisation : factoriser les propriétés/opérations de plusieurs classes dans une super-classe.
- Spécialisation : capturer les particularités des instances d'une sous-classe.



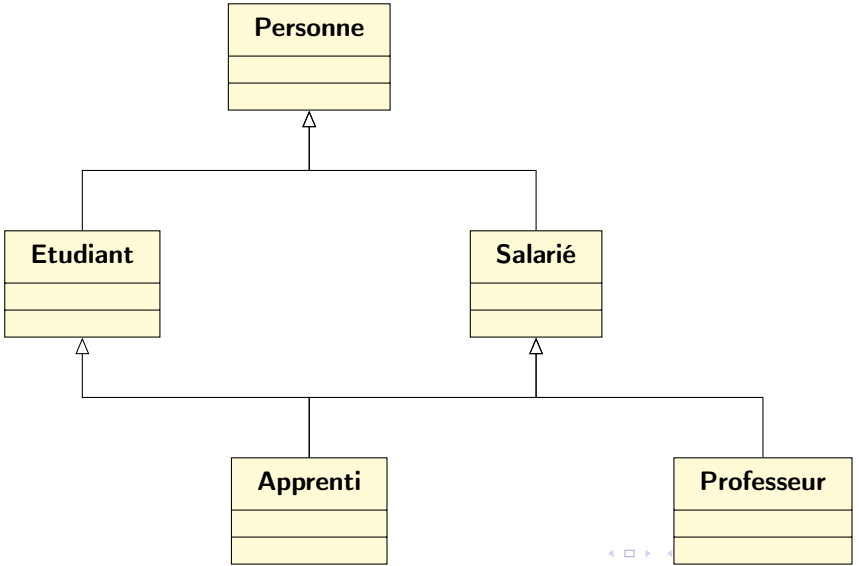
Définitions

- Généralisation : factoriser les propriétés/opérations de plusieurs classes dans une super-classe.
- Spécialisation : capturer les particularités des instances d'une sous-classe.



- Une sous classe possède les propriétés/opérations de ses super-classes.

Exemple



Contraintes

- Ce sont des propriétés concernant les liens de généralisations

Contraintes

- Ce sont des propriétés concernant les liens de généralisations
- Représentées par la contrainte entre accolades : {**contrainte**}
  - disjoint : les ensembles formés par les instances des classes filles sont disjoints

Contraintes

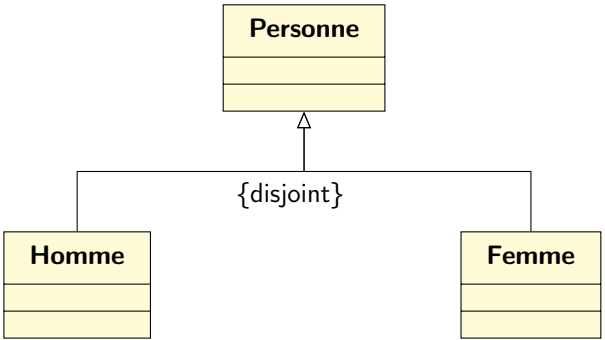
- Ce sont des propriétés concernant les liens de généralisations
- Représentées par la contrainte entre accolades : {**contrainte**}
  - disjoint : les ensembles formés par les instances des classes filles sont disjoints
  - overlapping : les ensembles formés par les instances des classes filles se recouvrent

Contraintes

- Ce sont des propriétés concernant les liens de généralisations
- Représentées par la contrainte entre accolades : {**contrainte**}
  - disjoint : les ensembles formés par les instances des classes filles sont disjoints
  - overlapping : les ensembles formés par les instances des classes filles se recouvrent
  - complete : les ensembles formés par les instances des classes filles forment l'ensemble des instances de la classe mère

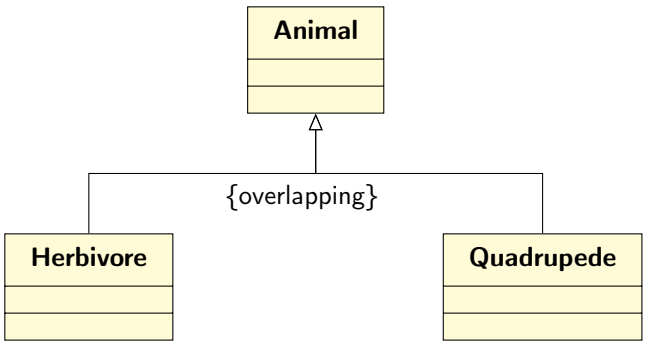
Contraintes

Exemple



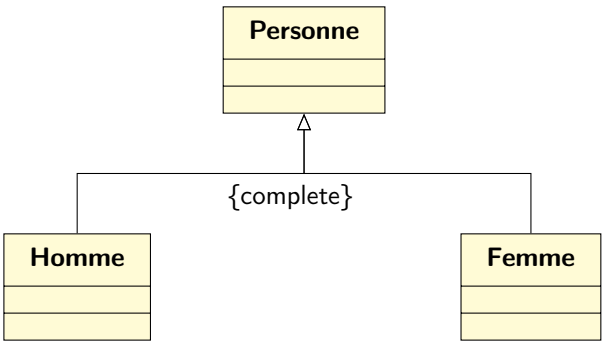
Contraintes

Exemple



Contraintes

Exemple



Plan

Définitions

- Une classe **abstraite** est une classe qui ne peut pas être instanciée

Définitions

- Une classe **abstraite** est une classe qui ne peut pas être instanciée
- Toute classe qui possède une **méthode** abstraite doit être abstraite

Définitions

- Une classe **abstraite** est une classe qui ne peut pas être instanciée
- Toute classe qui possède une **méthode** abstraite doit être abstraite
- Une classe abstraite n'a d'intérêt que parce qu'elle possède des classes filles

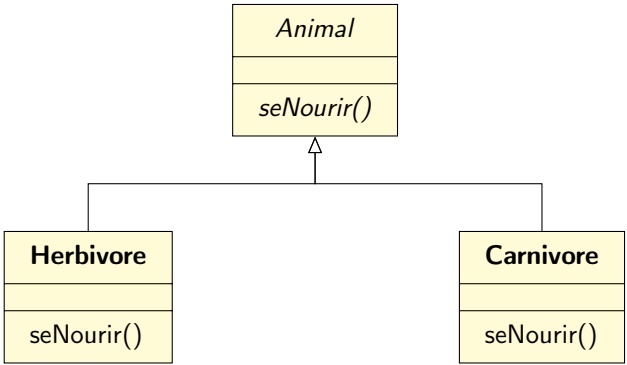
Définitions

- Une classe **abstraite** est une classe qui ne peut pas être instanciée
- Toute classe qui possède une **méthode** abstraite doit être abstraite
- Une classe abstraite n'a d'intérêt que parce qu'elle possède des classes filles

Remarque

- La classe/méthode abstraite est notée en italique

Exmple



Plan

Transcription des Classes

- A chaque classe UML correspond une classe Java

Transcription des Classes

- A chaque classe UML correspond une classe Java
- Chaque attribut de la classe UML sera un attribut de la classe Java
  - cardinalité ≤ 1 : une variable
  - cardinalité > 1 : un ensemble de variables

Transcription des Classes

- A chaque classe UML correspond une classe Java
- Chaque attribut de la classe UML sera un attribut de la classe Java
  - cardinalité ≤ 1 : une variable
  - cardinalité > 1 : un ensemble de variables
- la visibilité de la propriété est la visibilité de l'attribut

Transcription des Classes

- A chaque classe UML correspond une classe Java
- Chaque attribut de la classe UML sera un attribut de la classe Java
  - cardinalité ≤ 1 : une variable
  - cardinalité > 1 : un ensemble de variables
- la visibilité de la propriété est la visibilité de l'attribut
- Chaque opération de la classe UML sera une méthode de la classe Java
  - le sens de l'argument de l'opération est non retranscrit

Transcription des Associations

- Les associations sont traduites par des attributs dans les classes reliées
  - cardinalité ≤ 1 : une variable
  - cardinalité > 1 : un ensemble de variables

Transcription des Associations

- Les associations sont traduites par des attributs dans les classes reliées
  - cardinalité ≤ 1 : une variable
  - cardinalité > 1 : un ensemble de variables
- Un attribut est ajouté uniquement si l'association est navigable

Transcription des Associations

- Les associations sont traduites par des attributs dans les classes reliées
  - cardinalité ≤ 1 : une variable
  - cardinalité > 1 : un ensemble de variables
- Un attribut est ajouté uniquement si l'association est navigable
- Le nom de l'attribut est le nom de l'extrémité navigable

Transcription des Associations

- Les associations sont traduites par des attributs dans les classes reliées
  - cardinalité ≤ 1 : une variable
  - cardinalité > 1 : un ensemble de variables
- Un attribut est ajouté uniquement si l'association est navigable
- Le nom de l'attribut est le nom de l'extrémité navigable
- Contraintes en général non retranscrite

Les ensembles de variables

- L'ensemble de variables sera traduit par RSA en différents ensemble suivant les propriétés de l'attribut/association.

Les ensembles de variables

- L'ensemble de variables sera traduit par RSA en différents ensemble suivant les propriétés de l'attribut/association.
  - aucune propriété supplémentaire :
    - traduit en *Collection*.

Les ensembles de variables

- L'ensemble de variables sera traduit par RSA en différents ensemble suivant les propriétés de l'attribut/association.
  - aucune propriété supplémentaire :
    - traduit en *Collection*.
  - possède la propriété ordered :
    - traduit en *List*.

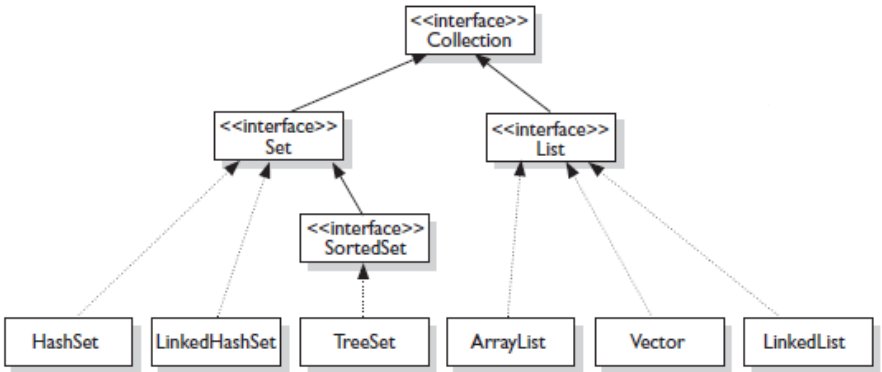
Les ensembles de variables

- L'ensemble de variables sera traduit par RSA en différents ensemble suivant les propriétés de l'attribut/association.
  - aucune propriété supplémentaire :
    - traduit en *Collection*.
  - possède la propriété ordered :
    - traduit en *List*.
  - possède la propriété unique :
    - traduit en *Set*.

Les ensembles de variables

- L'ensemble de variables sera traduit par RSA en différents ensemble suivant les propriétés de l'attribut/association.
  - aucune propriété supplémentaire :
    - traduit en *Collection*.
  - possède la propriété ordered :
    - traduit en *List*.
  - possède la propriété unique :
    - traduit en *Set*.
  - possède les propriétés ordered et unique :
    - traduit en *SortedSet*.

Les ensembles en Java



L'Héritage et les classes abstraites

Transcription des Généralisations/Spécialisations

- La relation de généralisation est traduite par une relation d'héritage en Java

L'Héritage et les classes abstraites

Transcription des Généralisations/Spécialisations

- La relation de généralisation est traduite par une relation d'héritage en Java
  - Attention : problème si héritage multiple

L'Héritage et les classes abstraites

Transcription des Généralisations/Spécialisations

- La relation de généralisation est traduite par une relation d'héritage en Java
  - Attention : problème si héritage multiple
- L'abstraction est traduite directement d'UML vers Java