Université Ibn Zohr
Faculté des sciences – Département Informatique

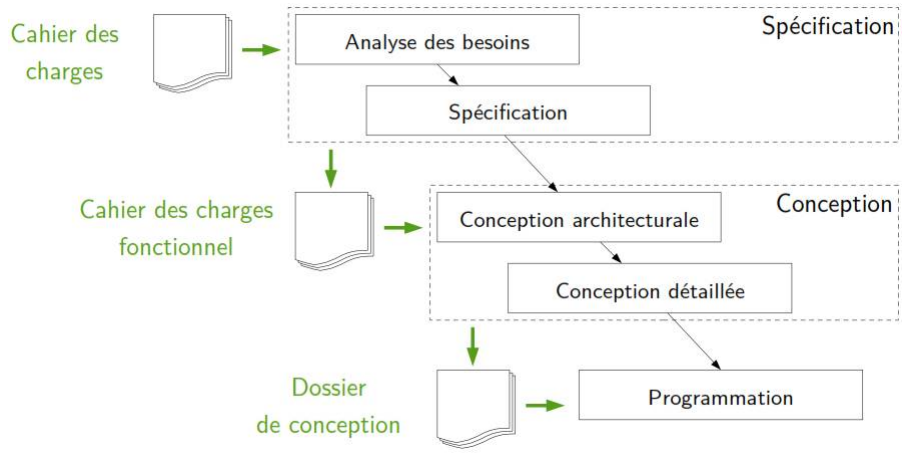
Génie Logiciel
« Ch03. Conception »

Ayoub SABRAOUI

Master OTI –S1

Année universitaire 2016/2017

Documentation du processus de GL



2

Conception

Objectif Fournir une vue interne de la structure du logiciel :

Le dossier de conception détaillée

- Proposer une architecture logicielle/matérielle
- Définir les différents composants du logiciel
- Associer les fonctionnalités spécifiques aux composants

Enjeu Contrat entre les développeurs et le chef de projet

- Chef de projet : la solution proposée répond à la spécification
- Développeurs : la conception structure la réalisation

3

Contenu du document de conception

Conception générale

- Description de la structure globale du logiciel
- Description succincte de la dynamique du logiciel

Conception détaillée

- Structure détaillée du logiciel
- Réalisation des cas d'utilisation
- Nouvelle version de l'interface
- Réalisation des contraintes de performance, sécurité…

4

Architecture logicielle

Structure du logiciel en termes de composants

- Fonctionnalités associées aux composants
- Interfaces entre les composants
- Dépendances entre les composants
- Décomposition des composants en sous-composants

Objectif

- Éviter les oublis/redondances de fonctionnalités
- Décentraliser le développement
- Faciliter l'intégration

5

Réalisation des exigences

Réalisation des cas d'utilisation

- Diagrammes de séquence instanciés sur le diagramme de classes : Réalisation d'une fonctionnalité comme suite d'interactions entre des objets du système
- Algorithmes principaux en pseudo-code

Réalisation des exigences non fonctionnelles Solutions adoptées pour répondre aux contraintes de performance, sécurité…

Objectif Validation de la conception par rapport au cahier des charges fonctionnel

6

Objets et classes

Conception orientée objet : Représentation du système comme un ensemble d'objets interagissant

Diagramme de classes

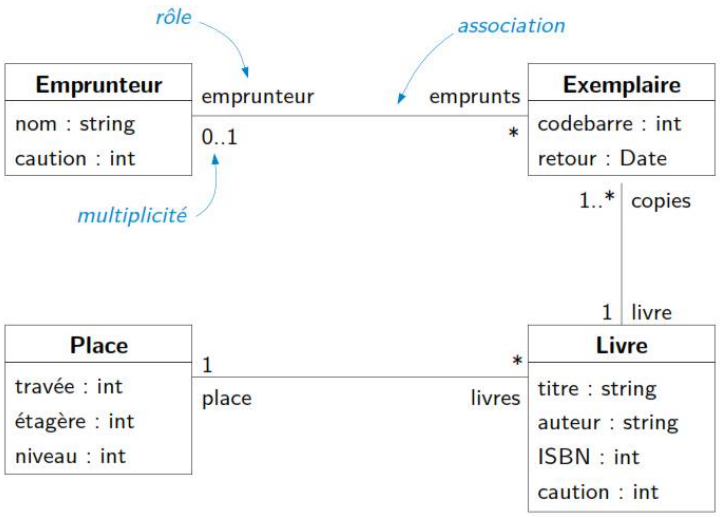
- Représentation de la structure interne du logiciel
- Utilisé surtout en conception mais peut être utilisé en analyse

Diagramme d'objets

- Représentation de l'état du logiciel (objets + relations)
- Diagramme évoluant avec l'exécution du logiciel
  - Création et suppression d'objets
  - Modification de l'état des objets (valeurs des attributs)
  - Modification des relations entre objets

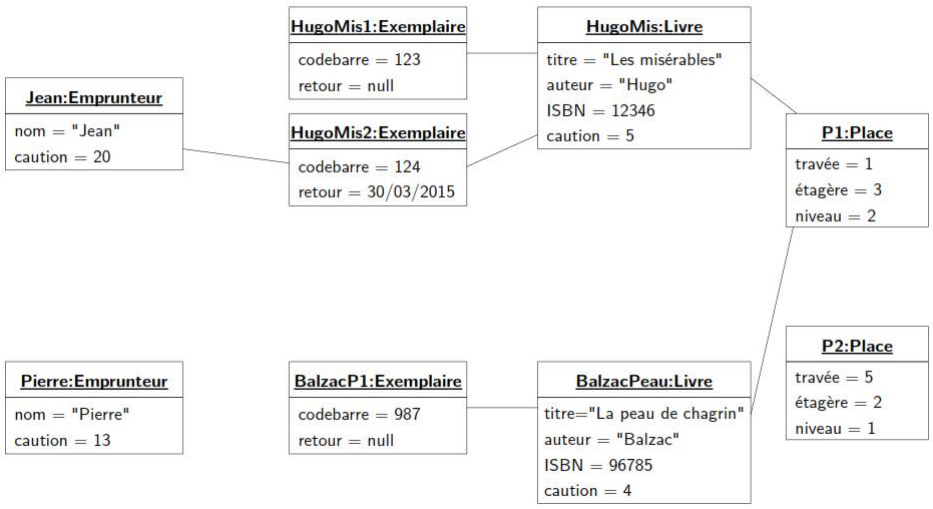
7

Diagramme de classes



8

Diagramme d'objets



9

Contraintes associées au diagramme

Contraintes non exprimées dans le diagramme de classes

- Le code barre d'un exemplaire et l'ISBN d'un livre sont uniques
- Un utilisateur ne peut pas emprunter plus d'un exemplaire d'un même livre
- Si un exemplaire est emprunté, sa date de retour n'est pas null
- Si un exemplaire n'est pas emprunté, sa date de retour est null
- La caution et l'ISBN d'un livre, les numéros de travée, étagère et niveau sont des entiers positifs

Invariants à ajouter explicitement dans la documentation, à prendre en compte dans l'implantation pour respecter le cahier des charges

10

Objets et classes

Objet : Entité concrète ou abstraite du domaine d'application

Décrit par : identité (adresse mémoire)
+ état (attributs)
+ comportement (opérations)

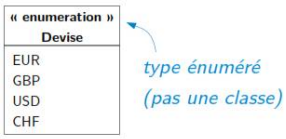
Classe : Regroupement d'objets de même nature (mêmes attributs + mêmes opérations)

Objet = instance d'une classe

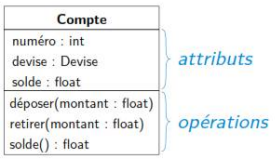
Classes

Attributs

- Caractéristique partagée par tous les objets de la classe
- Associe à chaque objet une valeur
- Type associé simple (int, bool,…), primitif (Date) ou énuméré
- Valeur des attributs = état de l'objet



Objets différents (identités différentes) peuvent avoir les mêmes attributs



Opérations

- Service qui peut être demandé à tout objet de la classe
- Comportement commun à tous les objets de la classe

Ne pas confondre avec une méthode = implantation de l'opération

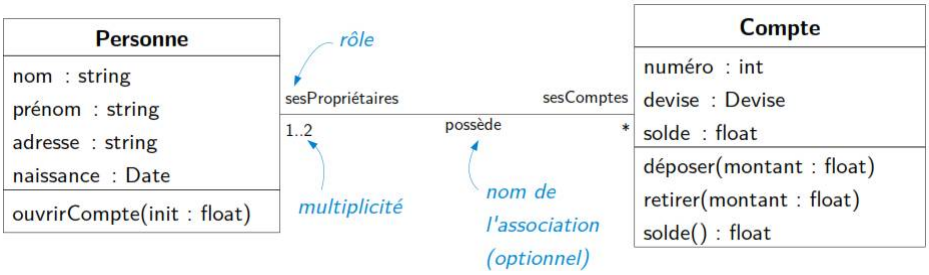
12

Relations entre classes

Association entre classes : Relation binaire (en général)

Rôle: Nomme l'extrémité d'une association, permet d'accéder aux objets liés par l'association à un objet donné

Multiplicité: Contraint le nombre d'objets liés par l'association

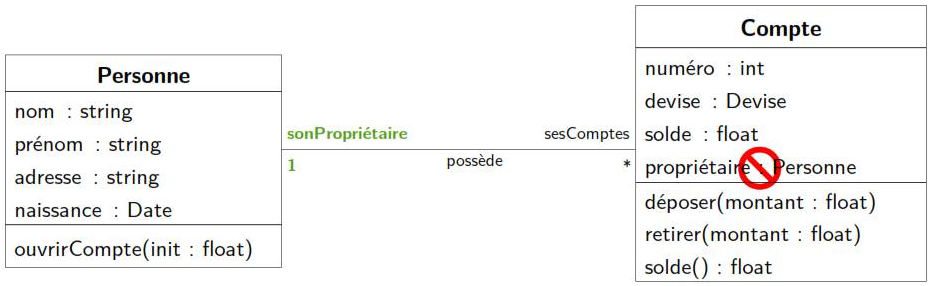


13

Attribut et association

Rappel : Types des attributs simple, primitif ou énuméré

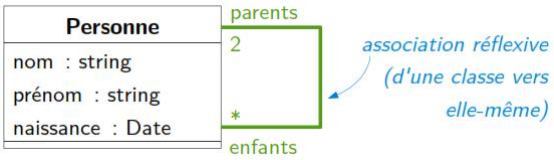
En particulier, pas d'attribut dont le type est une classe du diagramme
Mais association vers cette classe



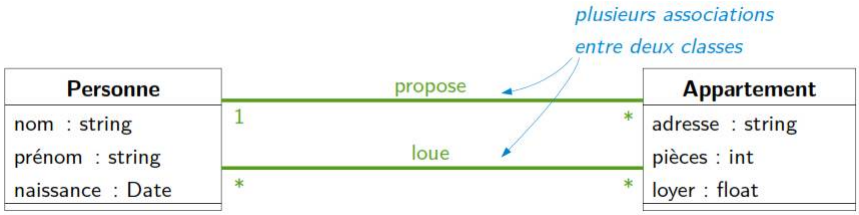
14

Associations particulières

Association réflexive



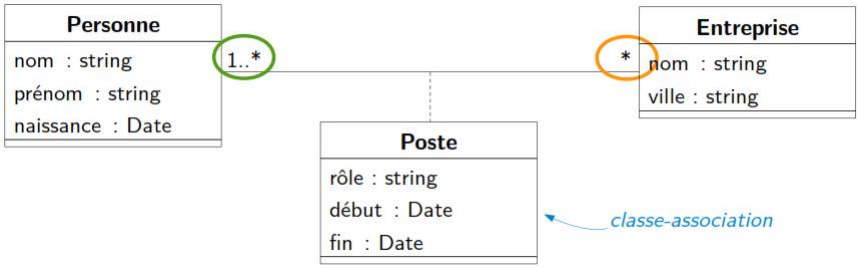
Associations multiples



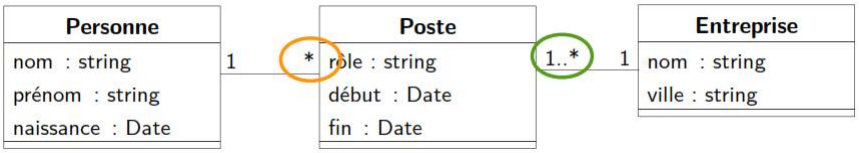
15

Classe-association

Permet de paramétrer une association entre deux classes par une classe

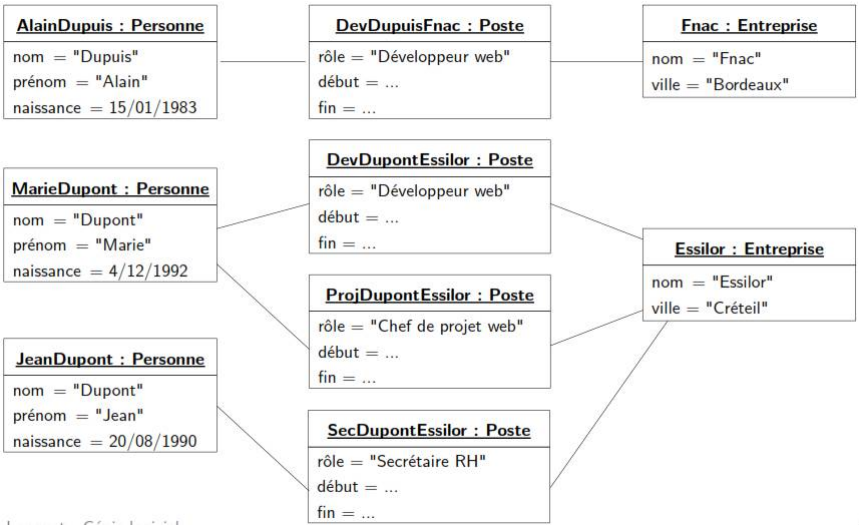


Équivalence en termes de classes et d'associations



Classe-association

Exemple de diagramme d'objets



Agrégation

Association particulière entre classes

- Dissymétrique: une classe prédominante sur l'autre
- Relation de type composant-composite

Deux types d'agrégation

- Agrégation faible
- Composition

Exemple

Lecteur de contenu audio permettant de créer des listes de lecture

18

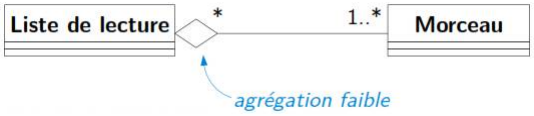
Agrégation faible

Agrégation par référence

- Le composite fait référence à ses composants
- La création ou destruction du composite est indépendante de la création ou destruction de ses composants
- Un objet peut faire partie de plusieurs composites à la fois

Exemple

- Une liste de lecture est composée d'un ensemble de morceaux
- Un morceau peut appartenir à plusieurs listes de lecture
- Supprimer la liste ne supprime pas les morceaux



19

Composition

Agrégation par valeur

- Le composite contient ses composants
- La création ou destruction du composite entraîne la création ou destruction de ses composants
- Un objet ne fait partie que d'un composite à la fois

Exemple

- Un morceau n'appartient qu'à un album
- La suppression de l'album entraîne la suppression de tous ses morceaux

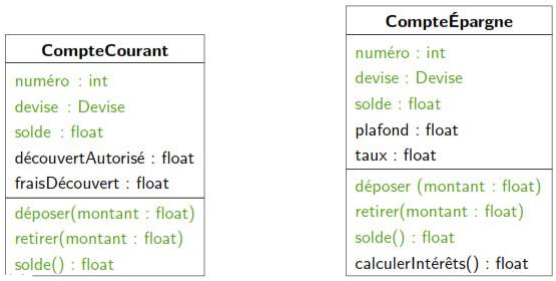
20

Hiérarchie de classes

Principe: Regrouper les classes partageant des attributs et des opérations et les organiser en arborescence

Spécialisation: raffinement d'une classe en une sous-classe

Généralisation: abstraction d'un ensemble de classes en super-classe



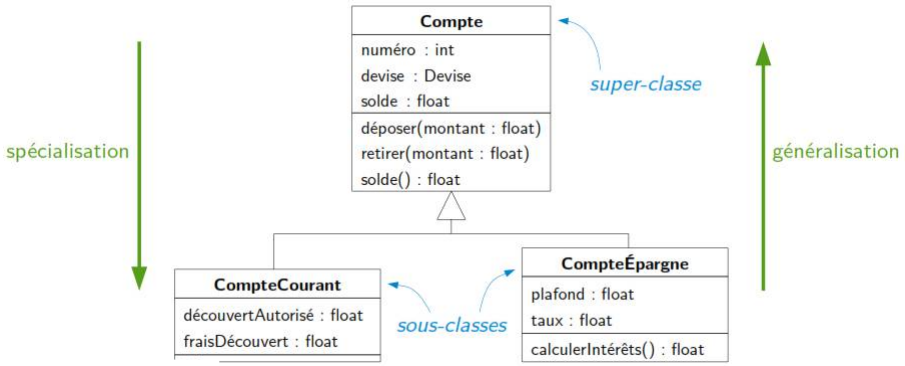
21

Hiérarchie de classes

Principe : Regrouper les classes partageant des attributs et des opérations et les organiser en arborescence

Spécialisation: raffinement d'une classe en une sous-classe

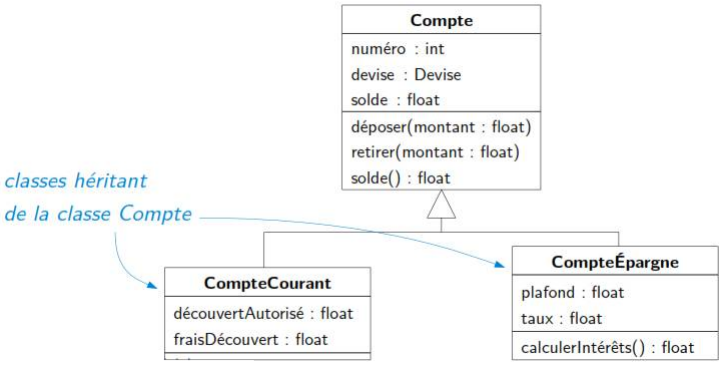
Généralisation: abstraction d'un ensemble de classes en super-classe



Hiérarchie de classes

Principe : Regrouper les classes partageant des attributs et des opérations et les organiser en arborescence

Héritage: Construction d'une classe à partir d'une classe plus haute dans la hiérarchie (partage des attributs, opérations, contraintes...)

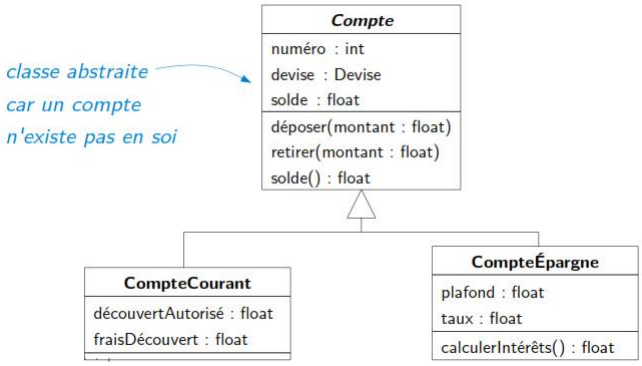


23

Classe abstraite

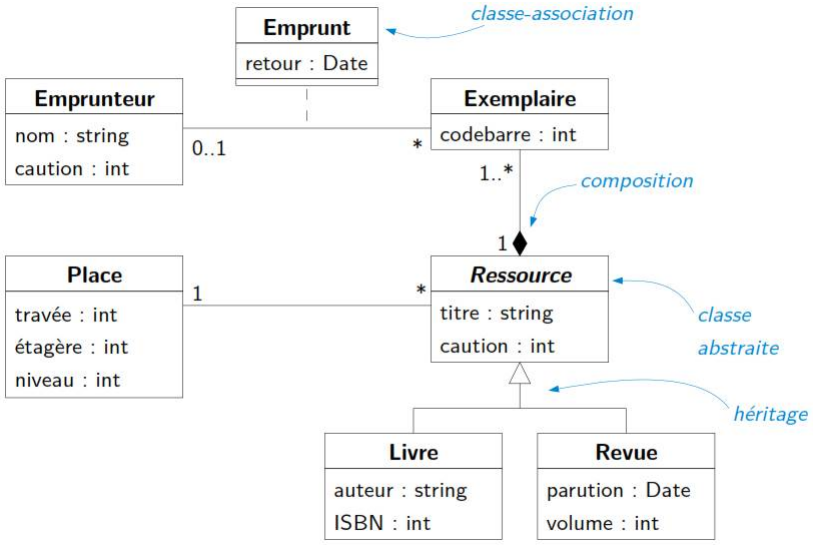
Classe sans instance, seulement une base pour les classes héritées

Notation : nom de la classe en italique (ou stéréotype « abstract »)



24

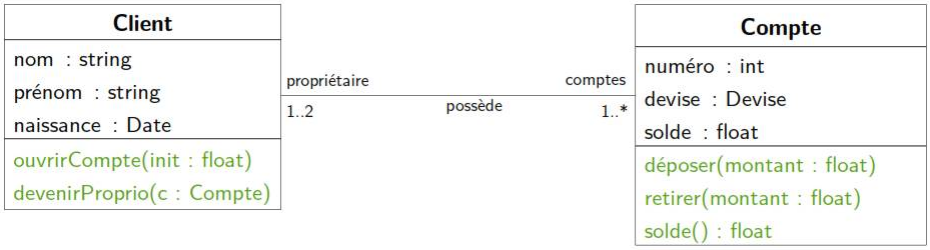
Diagramme de classes revu



Opérations

Opérations

- Service qui peut être demandé à tout objet de la classe
- Comportement commun à tous les objets de la classe

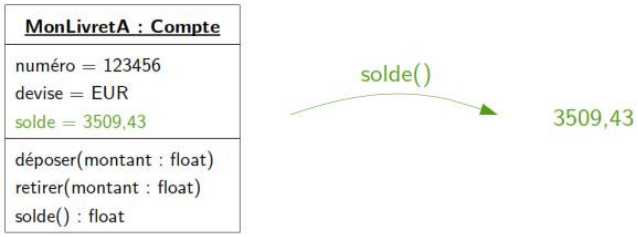


26

Opérations

Effets possibles d'une opération

- Renvoyer le résultat d'un calcul



27

Opérations

Effets possibles d'une opération

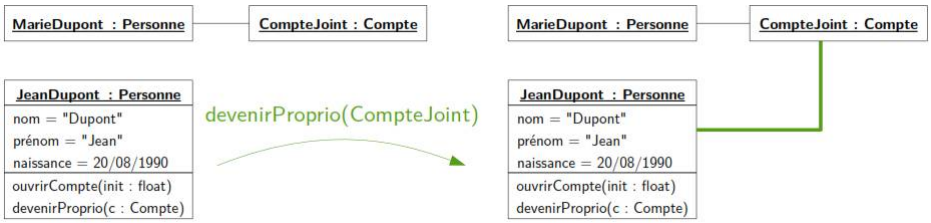
- Renvoyer le résultat d'un calcul
- Modifier l'état du système
  - modification de la valeur des attributs

28

Opérations

Effets possibles d'une opération

- Renvoyer le résultat d'un calcul
- Modifier l'état du système
  - modification de la valeur des attributs
  - ajout/suppressions de liens entre objets

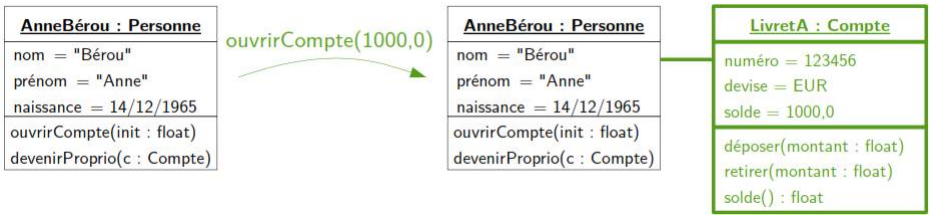


29

Opérations

Effets possibles d'une opération

- Renvoyer le résultat d'un calcul
- Modifier l'état du système
  - modification de la valeur des attributs
  - ajout/suppressions de liens entre objets
  - création/destruction d'objets

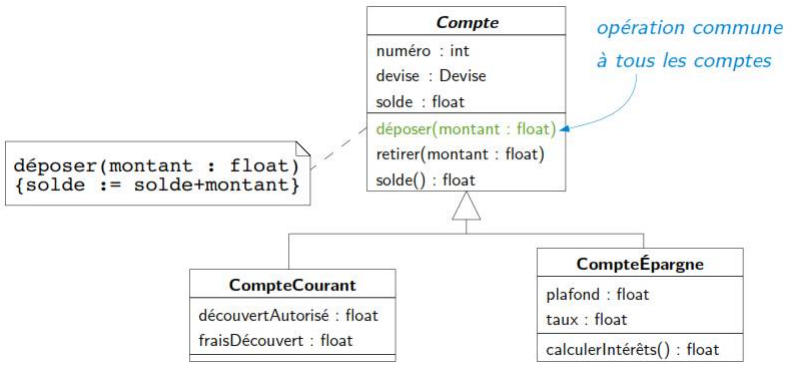


30

Héritage d'opération

Opération commune aux sous-classes :

- Définition dans la super-classe

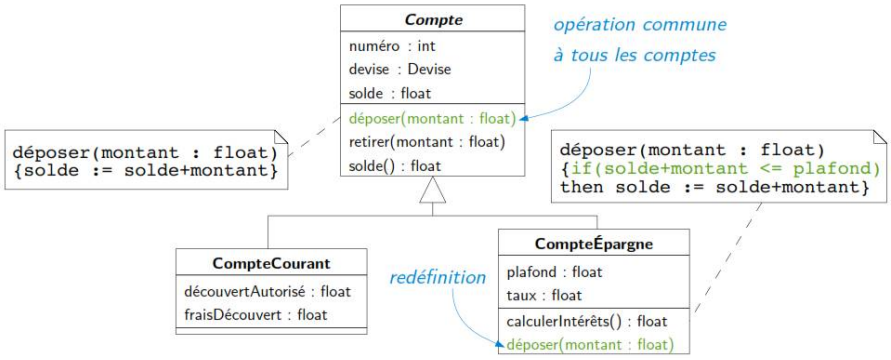


31

Redéfinition d'opération

Opération commune aux sous-classes :

- Définition dans la super-classe
- Possibilité de redéfinition locale de l'opération dans une sous-classe pour prendre en compte un cas particulier



Classe abstraite

Classe sans instance car certaines opérations non définies

- Opérations non définie en italique
- Nom de la classe en italique (ou stéréotype « abstract »)

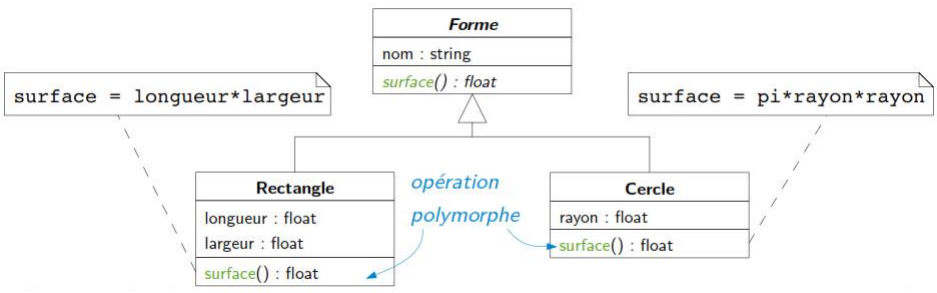
Exemple : On ne peut pas calculer la surface d'une forme sans savoir de quelle forme il s'agit

33

Polymorphisme

Contexte : Définition d'une opération abstraite dans les classes héritant d'une classe abstraite

Opération polymorphe : Opération définie dans différentes sous-classes mais opération spécifique à la sous-classe

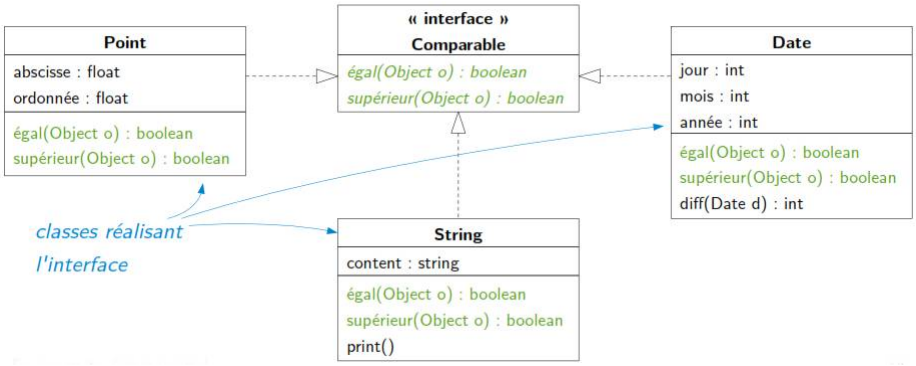


34

Interface

Liste d'opérations constituant un contrat à respecter par les classes réalisant l'interface

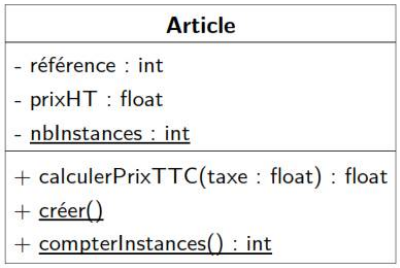
- Pas une classe, ne peut pas servir à créer des objets
- Toutes les opérations sont abstraites



Notations avancées

Attributs et opérations de classes

- Attribut de classe : valeur commune à toutes les instances
- Opération de classe : opération sur la classe elle-même (création de nouvelles instances par exemple)
- Soulignés dans la classe



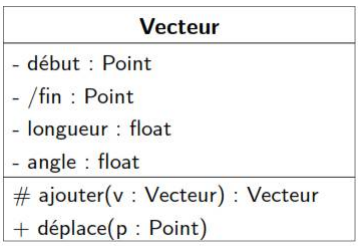
36

Notations avancées

Visibilité

+ : public, accessible à toutes les classes
# : protégé, accessible uniquement aux sous-classes
- : privé, inaccessible à tout objet hors de la classe

Pas de visibilité par défaut

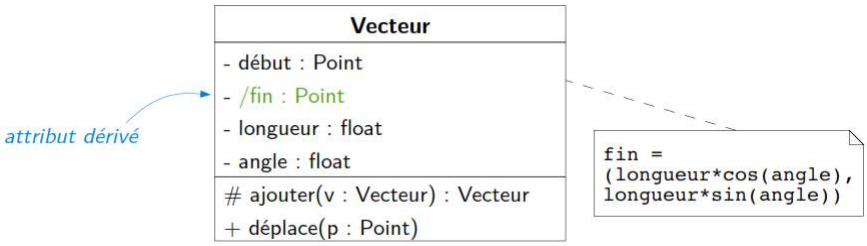


37

Attribut dérivé

Peut être calculé à tout moment à partir d'autres informations du système

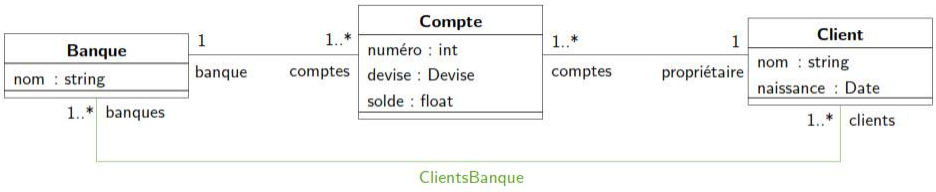
- Notation : /attribut
- Peut nécessiter des informations de plusieurs classes



38

Association dérivé

Redondance d'une association : Association n'apportant pas d'information supplémentaires, se demander si nécessaires

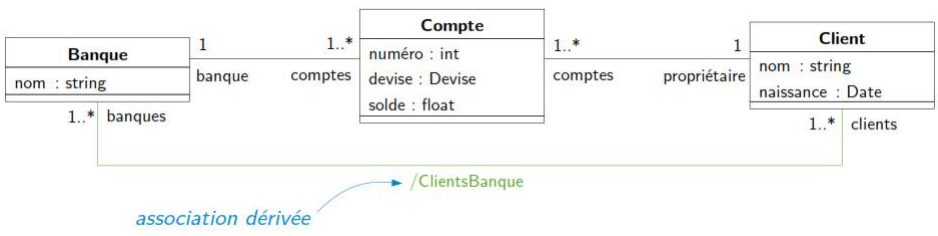


39

Association dérivé

Redondance d'une association : Association n'apportant pas d'information supplémentaires, se demander si nécessaires
Si nécessaire, la faire apparaître comme association dérivée

Association dérivée : Peut être calculée à tout moment à partir d'autres informations du système
Notation : /association



Limites du diagramme de classes

Diagramme de classes représente la structure du système en termes d'objets et de relations entre ces objets

Ne permet pas de représenter :

- Valeurs autorisées des attributs
- Conditions sur les associations
- Relations entre les attributs ou entre les associations

Expression des contraintes liées au diagramme :

- Notes dans le diagramme
- Texte accompagnant le diagramme
- OCL : langage de contraintes formel associé à UML

41

Contraintes, invariants

Propriétés:

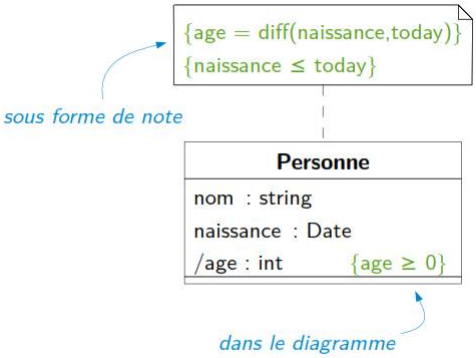
- Portant sur les éléments du modèle
- Doivent être vérifiées à tout instant
- En général, restriction sur les diagrammes d'objets possibles à partir du diagramme de classes
- Héritage des contraintes de la super-classe vers les sous-classes

Contraintes présentes dans le diagramme :

- Type des attributs
- Multiplicités des associations

42

Contraintes sur les attributs



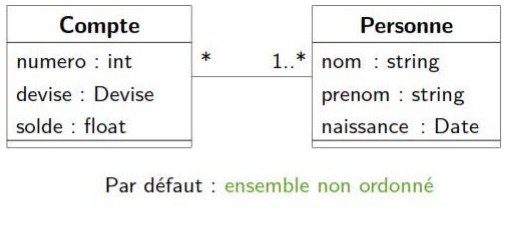
Contraintes sur la classe Personne :

- L'âge est toujours positif
- L'âge est calculé comme la différence de la date de naissance et la date d'aujourd'hui
- La date de naissance est inférieure ou égale à la date d'aujourd'hui

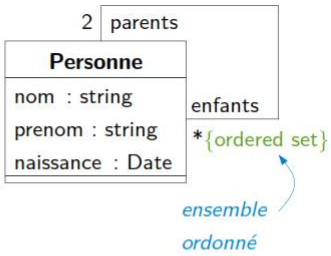
dans un document annexe

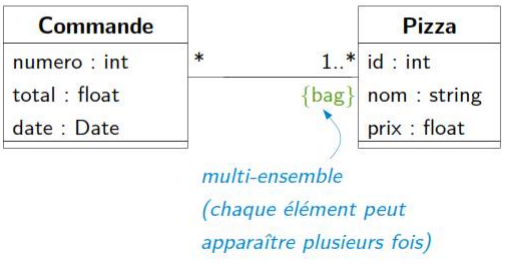
43

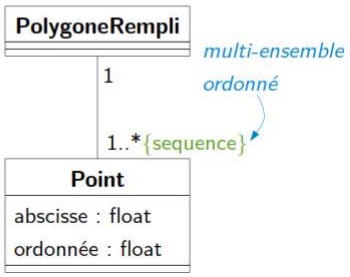
Contraintes associées à la multiplicité



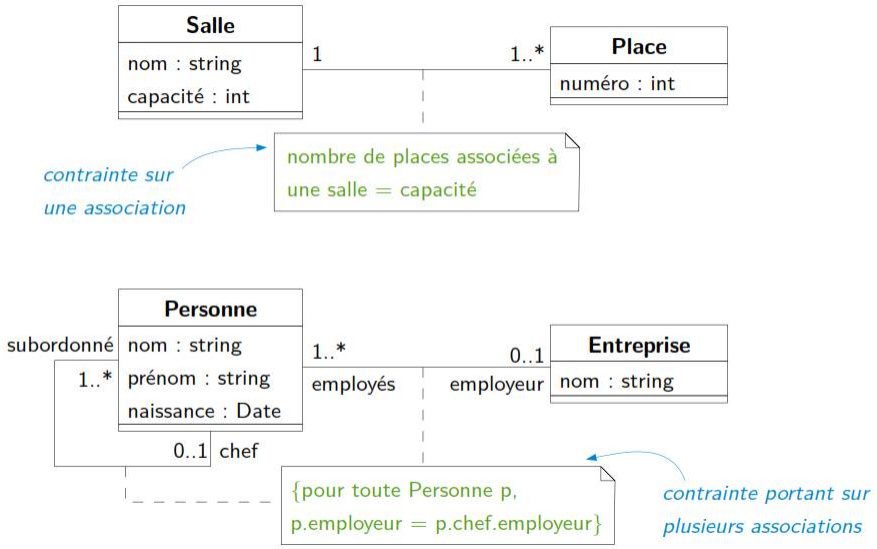
Par défaut : ensemble non ordonné







Contraintes sur les associations



Contraintes, invariants