

INTRODUCTION AUX MÉTHODOLOGIES DE CONCEPTION

Mme Sfaxi Lilia

Mme Zoubeir Najet

MCOO-Chapitre 1

L2ARS/SIL - 2011/2012

Plan

- Introduction aux systèmes d'information
- Introduction à la conception
- Le langage UML
- Le paradigme Objet

Chp1: Introduction aux Méthodologies de Conception

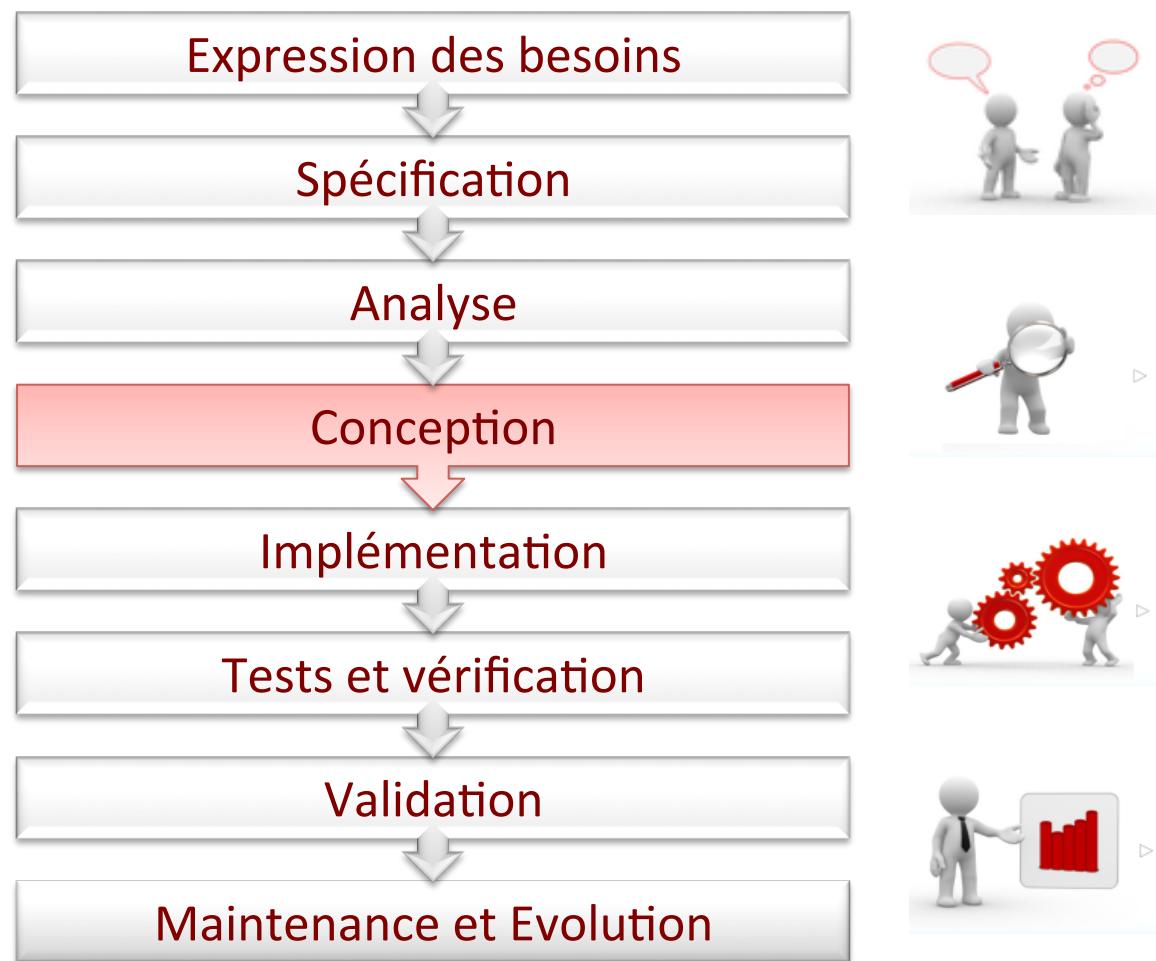
LES SYSTÈMES D' INFORMATION

Méthodologie de
Conception Orientée Objet

Introduction aux SI

- Un système d'information représente l'ensemble des éléments participant à la gestion, stockage, traitement, transport et diffusion de l'information au sein d'une organisation
- Rôle:
 - ✓ Collecte d'informations
 - ✓ Stockage de l'information
 - ✓ Traitement de l'information
 - ✓ Diffusion de l'information

Etapes de Réalisation d'un SI (1/3)



Etapes de Réalisation d'un SI (2/3)

- Expression des besoins
 - ✓ Définition d'un cahier des charges
- Spécification
 - ✓ Ce que le système doit être et comment il peut être utilisé
- Analyse
 - ✓ Éléments intervenant dans le SI, leurs structures et relations
 - ✓ A définir sur 3 axes
 - Savoir-faire de l'objet → *axe fonctionnel*
 - Structure de l'objet → *axe statique*
 - Cycle de vie de l'objet → *axe dynamique*
- Conception
 - ✓ Apport de solutions techniques: architecture, performance et optimisation
 - ✓ Définition des structures et des algorithmes

Etapes de Réalisation d'un SI (3/3)

- Implémentation
 - ✓ Réalisation et programmation
- Tests et vérification
 - ✓ Contrôles de qualité
 - ✓ Instaurés tout au long du cycle de développement
- Validation
 - ✓ Vérification de la correspondance avec le cahier des charges + discussion avec l'utilisateur
- Maintenance et Evolution
 - ✓ Maintenance corrective: traiter les erreurs (bugs)
 - ✓ Maintenance évolutive: intégration de nouveaux changements

Chp1: Introduction aux Méthodologies de Conception

ÉTAPE DE CONCEPTION

Méthodologie de
Conception Orientée Objet

Conception

- Processus créatif pour la mise au point d'un logiciel
- Permet de donner une architecture au logiciel en le découplant en briques, chacune en charge de fonctionnalités différentes
- 2 types de conception
 - ✓ Conception architecturale
 - Définition de la structure interne du logiciel
 - Décomposition en composants de taille maîtrisable
 - Définition des interfaces et interactions entre composants
 - ✓ Conception détaillée
 - Définition du rôle de chacun des composants
 - Définition des sous-composants

Modélisation

- Support de la conception
- Formalisation de la solution, en utilisant des notations ou des règles connues
- Permet de réduire la complexité d'un phénomène
 - ✓ Éliminer les détails non significatifs
 - ✓ Réfléter ce qui est important pour la compréhension et prédition du phénomène modélisé
- Création d'un **Modèle**
 - ✓ Représentation abstraite et simplifiée d'une entité du monde réel en vue de le décrire, de l'expliquer ou de le prévoir
 - ✓ Exemple : Plan

Méthode et Langage

- Méthode de conception
 - ✓ Description normative des étapes de la modélisation
 - ✓ Exemple: Merise
 - ✓ Problème:
 - Existence de plusieurs cas particuliers ➔ difficulté de représenter une méthode exhaustive
- Langage de modélisation
 - ✓ Langage graphique pour représenter, communiquer les divers aspects d'un système d'information
 - ✓ Possède un vocabulaire et des règles qui permettent de combiner les mots afin de communiquer

Chp1: Introduction aux Méthodologies de Conception

UML : UNIFIED MODELING LANGUAGE

Historique

- Années 80:
 - ✓ Méthodes pour organiser la programmation fonctionnelle (Merise)
 - ✓ Séparation des données et des traitements
- Début des années 90:
 - ✓ Apparition de la programmation objet: nécessite d'une méthodologie adaptée
 - ✓ Apparition de plus de 50 méthodes entre 1990 et 1995
- 1994
 - ✓ Consensus sur 3 méthodes
 - **OMT** de James Rumbaugh : représentation graphique des aspects statiques, dynamiques et fonctionnels d'un système
 - **OOD** de Grady Booch: concept de package
 - **OOSE** de Ivar Jacobson: description des besoins de l'utilisateur

UML

- Consensus entre les « 3 Amigos » pour créer une méthode commune:
 - ✓ UML : Unified Modeling Language (Langage de Modélisation Unifié)
- 1997: Définition de la norme UML comme standard de modélisation des systèmes d'information objet par l'**OMG** (*Object Management Group*)
- UML est employé dans l'ensemble des secteurs du développement informatique
 - ✓ Systèmes d'information
 - ✓ Télécommunication, défense
 - ✓ Transport, aéronautique, aérospatial
 - ✓ Domaines scientifiques
- Mais pas seulement : on peut modéliser des comportement mécaniques, humain, etc.

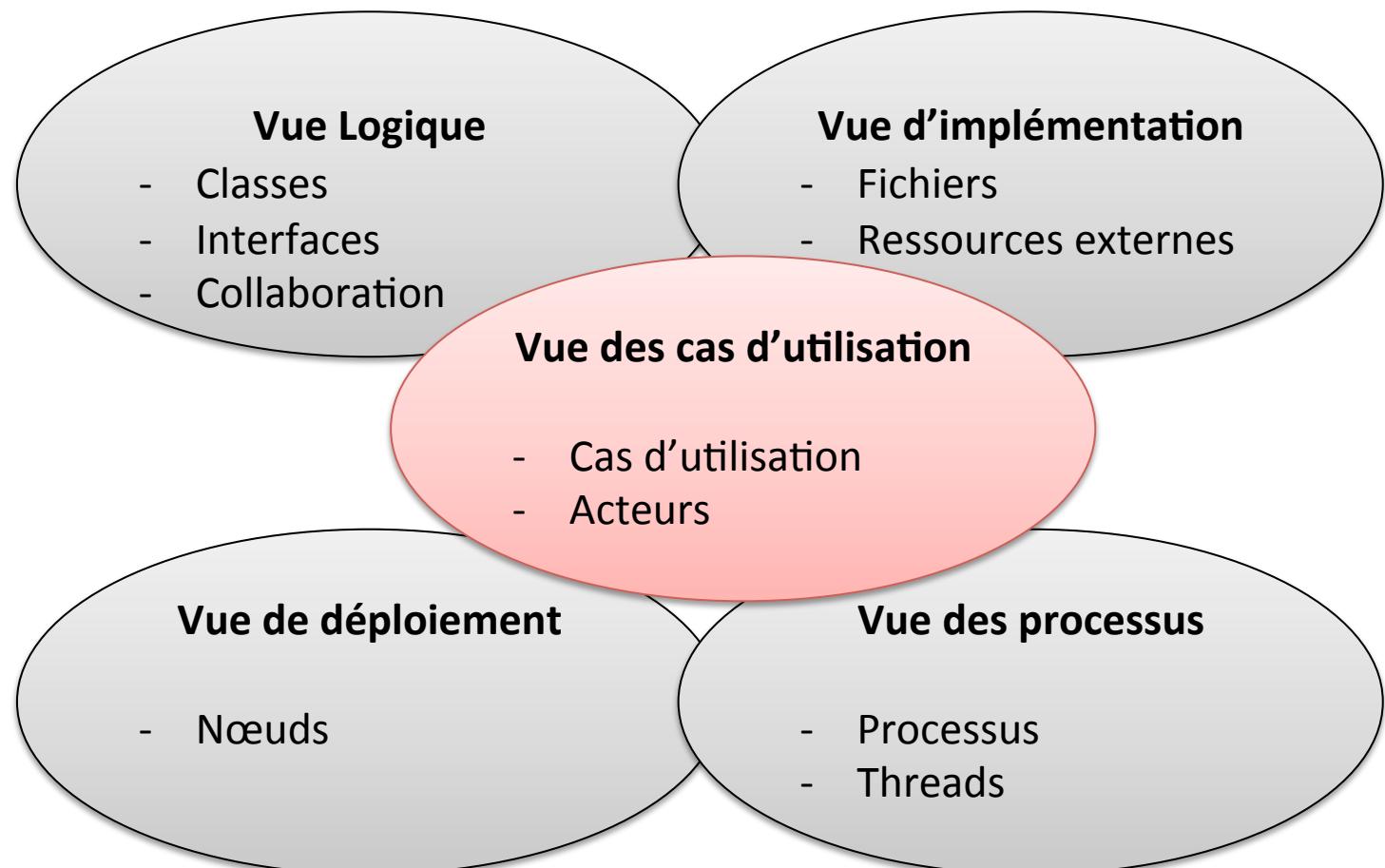
Briques de base d'UML

- *Les éléments*
 - ✓ Ce sont les abstractions essentielles au modèle.
- *Les relations*
 - ✓ Les relations expriment les liens existants entre les différents éléments.
- *Les diagrammes*
 - ✓ Un diagramme est une représentation visuelle de l'ensemble des éléments qui constituent le système
 - ✓ Ils servent à visualiser un système sous différents angles (utilisateur, administrateur par ex.)
 - ✓ Dans les systèmes complexes, un diagramme ne fournit qu'une vue partielle du système
 - L'ensemble des diagrammes réunis permet d'obtenir une vue globale du système à concevoir
 - Chaque diagramme va permettre de modéliser ou spécifier une vue (spécificité) du système à concevoir

Les 4+1 Vues

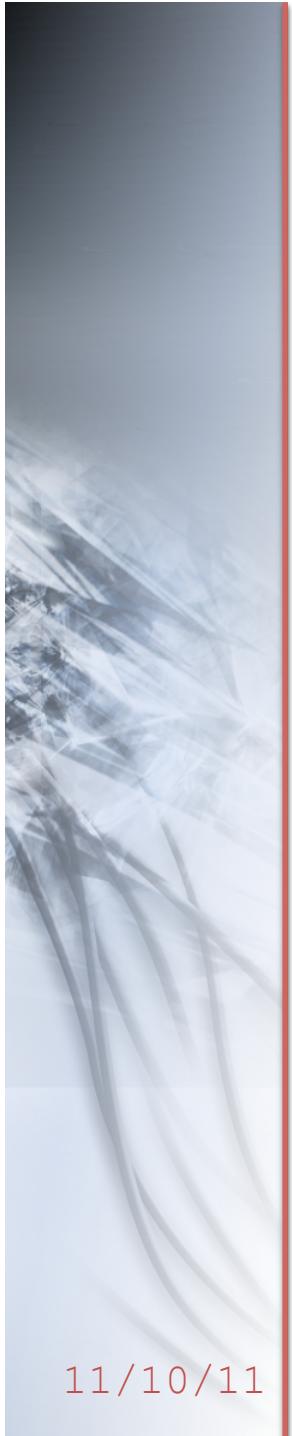
- Vue des cas d'utilisation
 - ✓ Description du modèle vu par les acteurs du système
 - ✓ Besoins attendus pour chaque acteur
 - ✓ Le **QUOI** et le **QUI**
- Vue logique
 - ✓ Définition du système vu de l'intérieur
 - ✓ **COMMENT** satisfaire les besoins des acteurs
- Vue d'implémentation
 - ✓ Dépendances entre les modules
- Vue des processus
 - ✓ Vue temporelle et technique
 - ✓ Mise en œuvre des notions de tâches concurrentes, synchronisation...
- Vue de déploiement
 - ✓ Position géographique et architecture physique de chaque élément
 - ✓ Le **OÙ**

Les 4+1 Vues



Diagrammes d' UML 2

- Diagrammes structurels / statiques (UML Structure)
 - ✓ diagramme de classes (Class diagram)
 - ✓ diagramme d'objets (Object diagram)
 - ✓ diagramme de composants (Component diagram)
 - ✓ diagramme de déploiement (Deployment diagram)
 - ✓ diagramme de paquetages (Package diagram)
 - ✓ diagramme de structures composites (Composite structure diagram)
- Diagrammes comportementaux / dynamiques (UML Behavior)
 - ✓ diagramme de cas d'utilisation (Use case diagram)
 - ✓ diagramme d'activités (Activity diagram)
 - ✓ diagramme d'états-transitions (State machine diagram)
 - ✓ diagrammes d'interaction (Interaction diagram)
 - diagramme de séquence (Sequence diagram)
 - diagramme de communication (Communication diagram)
 - diagramme global d'interaction (Interaction overview diagram)
 - diagramme de temps (Timing diagram)



Chp1: Introduction aux Méthodologies de Conception

LE PARADIGME ORIENTÉ OBJET

Introduction à l'approche Orientée Objet

- Evolution foudroyante du matériel
 - ✓ Premier ordinateur :
 - 50 tonnes, 25 Kwatts, quelques milliers de positions de mémoire
 - Quelques composants par circuit
 - ✓ Actuellement : Processeurs avec 2, 4 et jusqu'à 6 cœurs
 - Quelques grammes, 17 watts, jusqu'à 16 Go de RAM, environs 20 000 MIPS (millions d'instructions par seconde)
 - 400 millions de transistors
 - ➡ Concept clef : la *Réutilisation*
- Evolution lente du logiciel
 - ✓ Les projets informatiques repartent de zéro!
- Solution : Exploiter le concept de réutilisation pour le logiciel
 - ✓ Approche orientée objet

Objet...?

- Définitions :
 - ✓ Entité cohérente rassemblant des données et du code travaillant sur ces données
 - ✓ Structure de données valuées qui répond à un ensemble de messages
- Caractérisé par :
 - ✓ son comportement : que peut-on faire avec cet objet?
 - Méthodes
 - ✓ son état : comment réagit l'objet quand on applique ces méthodes?
 - Attributs (Champs)
 - ✓ son identité : comment distinguer les objets qui ont le même état et le même comportement?
 - Identifiant
- A les mêmes réactions et la même modularité que le monde réel
 - ✓ L'objet informatique est une projection de l'objet du monde réel

Classe

- Composant de base
- Contient la description d'un objet
 - ✓ Modèle de l'objet effectif
- Correspond à l' « *idée* » qu'on se fait d'un objet
 - ✓ Analogie avec la philosophie platonnienne idéaliste :
 - « *Vous vous promenez dans la campagne, vous croyez avoir rencontré des troupeaux de chevaux. Quelle erreur! (...) Car le Cheval-Modèle, le Cheval-Idée, n'est ni noir ni blanc, il n'est daucune race chevaline. Il est cheval pur et vos sens ne vous le montreront jamais...* » [Civilisation Grecque – A.Bonnard]
 - ✓ Voilà, la classe, c'est l' « *idée* » du cheval
 - ✓ Un pur sang arabe de couleur noire, dont le nom est ASWAD et qui boîte légèrement, est un **objet** instancié à partir de cette classe!

Exemples

- Classe

Voiture
marque
couleur
immatriculation
démarrer
conduire
arrêter

```
class Voiture {  
    // attributs  
    String marque;  
    String couleur;  
    String immatriculation;  
    // méthodes  
    void démarrer( ){ }  
    void conduire( ){ }  
    void arrêter( ){ }  
}
```

- Objet

twingo : Voiture
marque : Renault
couleur : grise
immatriculation : 102 102

```
Voiture twingo = new Voiture( );
```



Concepts fondamentaux de l'approche OO

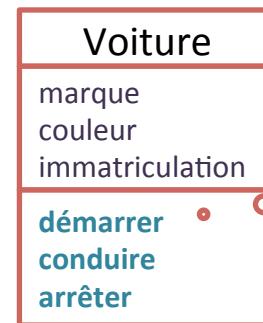
- Caractéristiques de l'approche objet :
 - ✓ Encapsulation
 - ✓ Héritage
 - ✓ Polymorphisme
 - ✓ Agrégation

Encapsulation

- Mécanisme consistant à rassembler, au sein d'une même structure, les données et les traitements
 - ✓ Définition des attributs et méthodes au niveau de la classe
- L'implémentation de la classe est cachée pour l'utilisateur
 - ✓ Définition d'une interface : vue externe de l'objet
- Possibilité de modifier l'implémentation sans modifier l'interface
 - ✓ Facilité de l'évolution de l'objet
- Préservation de l'intégrité des données
 - ✓ L'accès direct aux attributs est interdit
 - ✓ L'interaction entre les objets se fait uniquement grâce aux méthodes

Encapsulation : Exemple

Concepteur



Affiche :
*La voiture est
démarrée*

Utilisateur

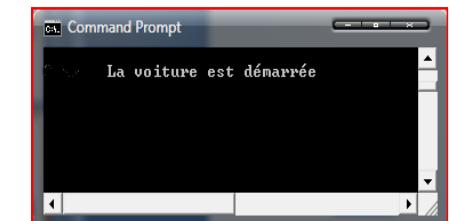


J'aimerais créer une
nouvelle twingo

Voiture twingo = new Voiture();

Que se passe-t-il si je
démarrer ma twingo?

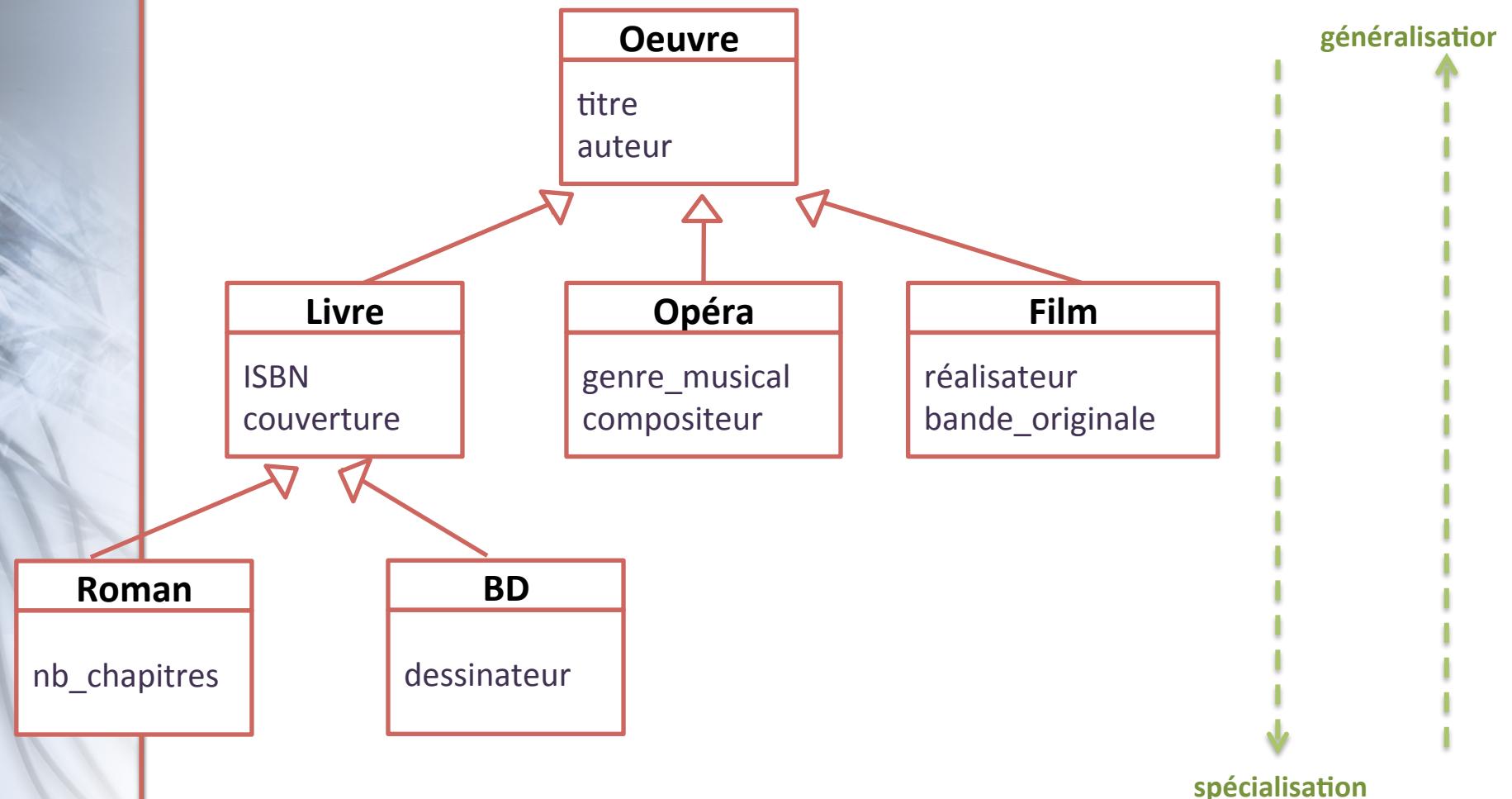
twingo.démarrer();



Héritage

- Un objet spécialisé bénéficie ou hérite des caractéristiques de l'objet le plus général, auquel il rajoute ses éléments propres
 - ✓ Création de nouvelles classes basées sur des classes existantes
 - ✓ Transmission des propriétés (attributs et méthodes) de la classe mère vers la classe fille
- Traduit la relation « est un ... »
- Deux orientations possibles
 - ✓ Spécialisation : Ajout / adaptation des caractéristiques
 - ✓ Généralisation : Regroupement des caractéristiques communes
- Possibilité d'héritage multiple
- Avantages
 - ✓ Éviter la duplication du code
 - ✓ Encourager la réutilisation du code

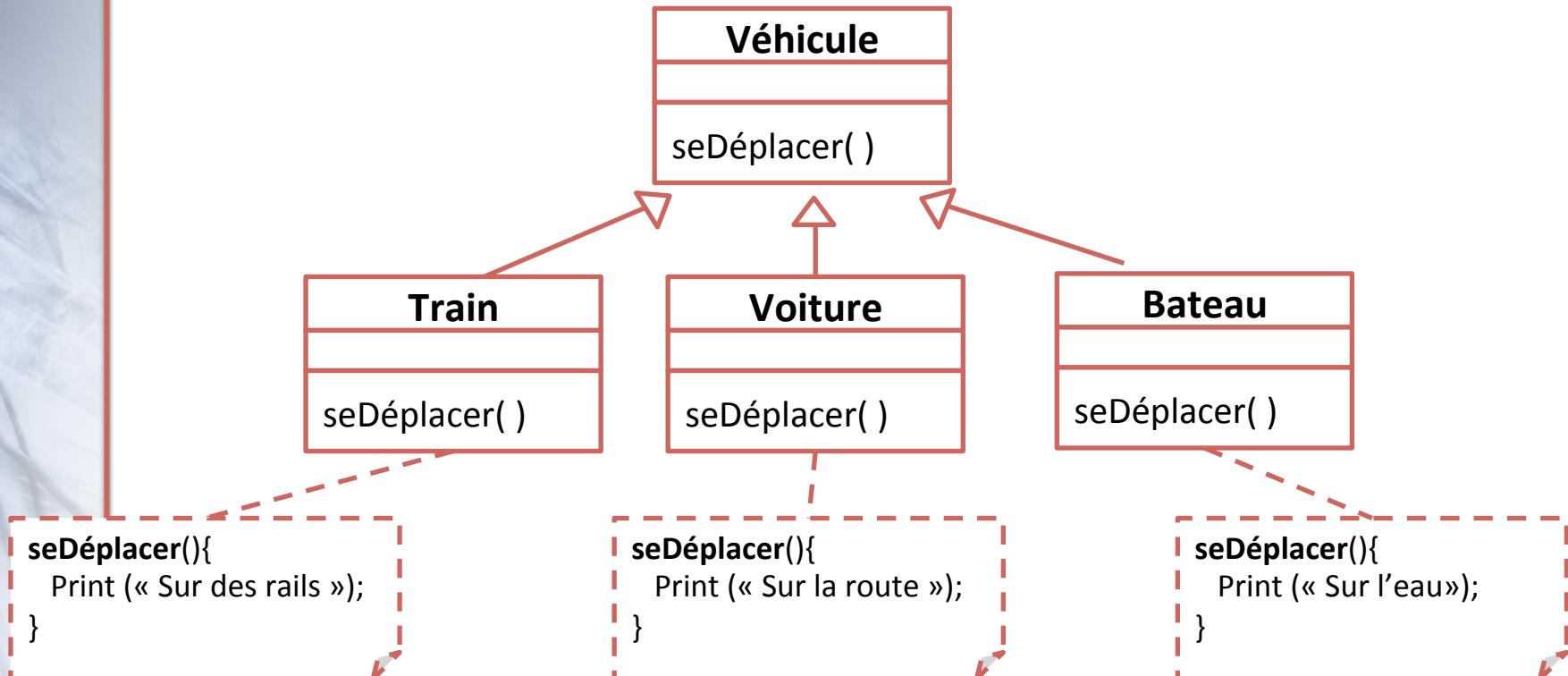
Héritage : Exemple



Polymorphisme

- Définition :
 - ✓ *Poly* : plusieurs
 - ✓ *Morphisme* : Forme
- Faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes
- Capacité d'une classe à redéfinir une méthode héritée à partir d'une classe mère
 - ✓ Surcharge
- Avantages
 - ✓ Lisibilité du code
 - ✓ Généricité du code

Polymorphisme : Exemple



Agrégation

- Relation entre deux classes, de sorte que les objets de l'une soient des composants de l'autre
- Traduit la relation « *est composé de...* » ou « *a ...* »
- Toute agrégation est caractérisée par une *cardinalité*
 - ✓ Combien définit la classe contenante d'instances de la classe contenue?
 - ✓ À combien de classes peut appartenir un objet?
- Avantage
 - ✓ Partir d'objets de base pour construire des objets complexes

Agrégation : Exemple

