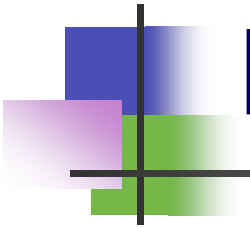
Chapitre 2 : modélisation Objet

M.BOUABID, 09-2013



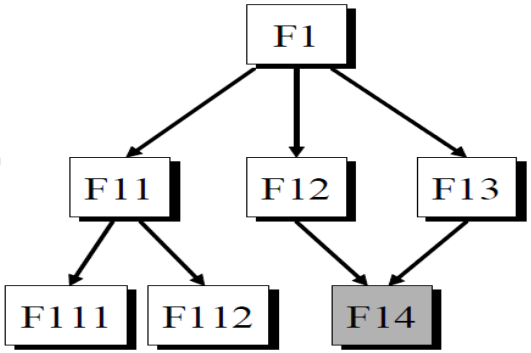
Plan

- Approche cartésienne
- Approche systémique
- Approche objet
- Concepts de l'approche objet

Classes de méthodes de conception

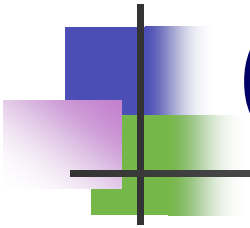
- Approches cartésiennes (première génération)
  - 70’s
- Approches systémiques (deuxième génération)
  - 80’s
- Approches objet (troisième génération)
  - 90’s et +

Les approches cartésiennes (1/2)



- Classique
  - Décomposition d'un problème en sous-problèmes
  - Décomposition d'une fonction en sous-fonctions jusqu'à atteindre un niveau facile à coder
- Méthodes
  - Méthodes de programmation structurée, SADT, Jackson,etc.

Les approches cartésiennes (2/2)



- Points forts
  - Simplicité et bon sens
  - Adéquation à capturer les besoins des utilisateurs
- Points faibles
  - Éffort sur les fonctions au détriment des données(redondance)
  - Règles de décomposition non explicites
  - Difficile de faire de la réutilisation

Les approches systémiques (1/2)

- Conception des SI
  - Structure & Comportement
  - Modélisation des données et des traitements
- Méthodes
  - Merise et Information Engineering

Les approches systémiques (2/2)

- Points forts
  - Grande cohérence des données
- Points faibles
  - Absence de règles pour assurer la cohérence entre modèle des données et modèle des traitements.
  - Faiblesse de la modélisation des traitements

Les approches objet (1/2)

- Évolution de l'approche systémique vers une plus grande cohérence entre les objets et leurs comportements
- Vision du SI comme un ensemble d'objets avec leurs opérations
- Méthodes
  - OMT, OOD, OOSE, etc.

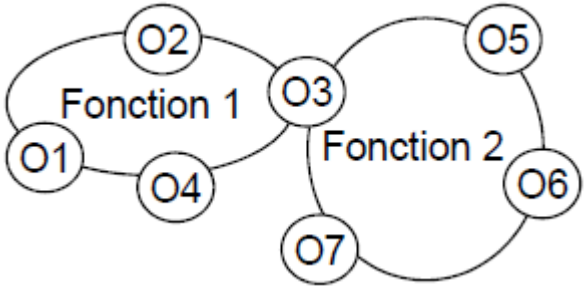
Les approches objet (2/2)

- Points forts
  - Grande capacité à modéliser les objets complexes.
  - Réduction des distorsions entre les le réel et le système informatique.
  - Grande capacité à intégrer la dynamique des objets.
- Points faibles
  - Difficulté de l'effort d'abstraction

Approche cartésienne vs. objet

- Approche cartésienne: approche descendante par décomposition de fonctions

- Approche objet: approche ascendante par composition d'objets



Approche systémique vs. objet

- Basées sur les mêmes concepts
  - SI = structure + comportement
- Approche systémique
  - Les langages de modélisation pour les données et les traitements sont incompatibles
  - Faut-il commencer par les données ou les traitements?
- • Approche objet
  - Réunir les traitements et les données dans la même unité (objet)

Introduction à l'approche Orientée Objet (1/2)

- 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

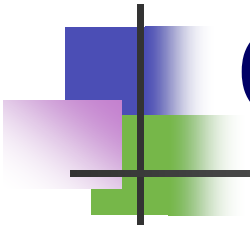
  Concept clef : la ***Réutilisation***

Introduction à l'approche Orientée Objet (2/2)

- Evolution lente du logiciel
  - Les projets informatiques repartent de zéro!
- Solution : Exploiter le concept de réutilisation pour le logiciel
  - Approche orientée objet

Objet…?

- Définition :
  - Entité cohérente rassemblant des données et du code travaillant sur ces données
- 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

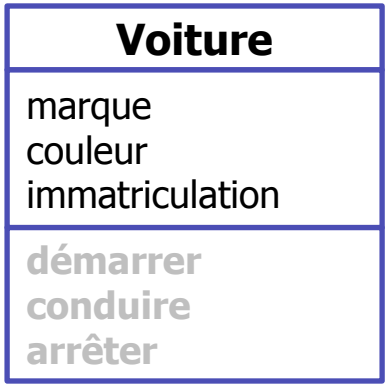


Classe

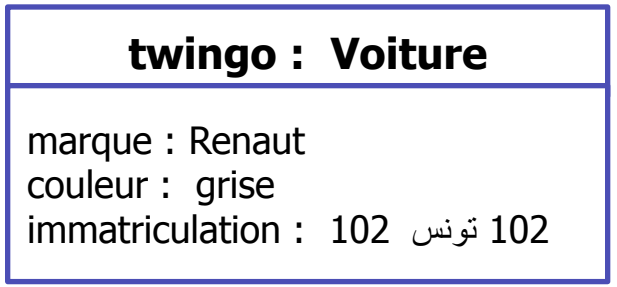
- Une classe est la description d'une collection d'objets ayant la même structure, le même comportement, les mêmes relations et la même sémantique

Exemples

- Classe



- Objet

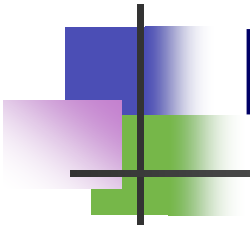



```
class Voiture {
    // attributs
       String marque;
       String couleur;
       String immatriculation;
    //méthodes
       void démarrer( ){ }
       void conduire( ){ }
       void arrêter( ){ }
}
```

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

Concepts fondamentaux de l'approche OO

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

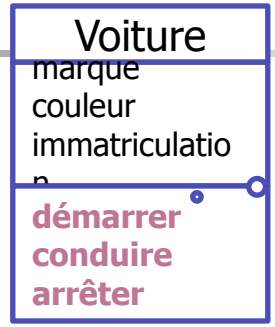


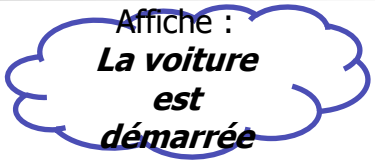
Encapsulation

- 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

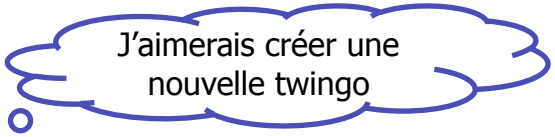
Encapsulation : Exemple

Concepteur

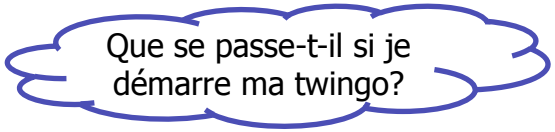




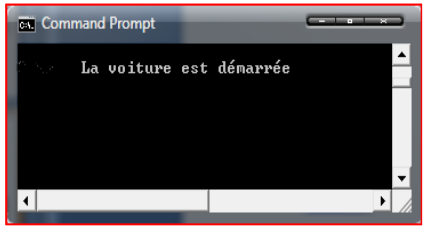
Utilisateur

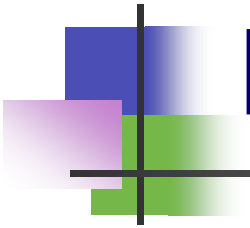


Voiture twingo = new Voiture( );



twingo.démarrer( );

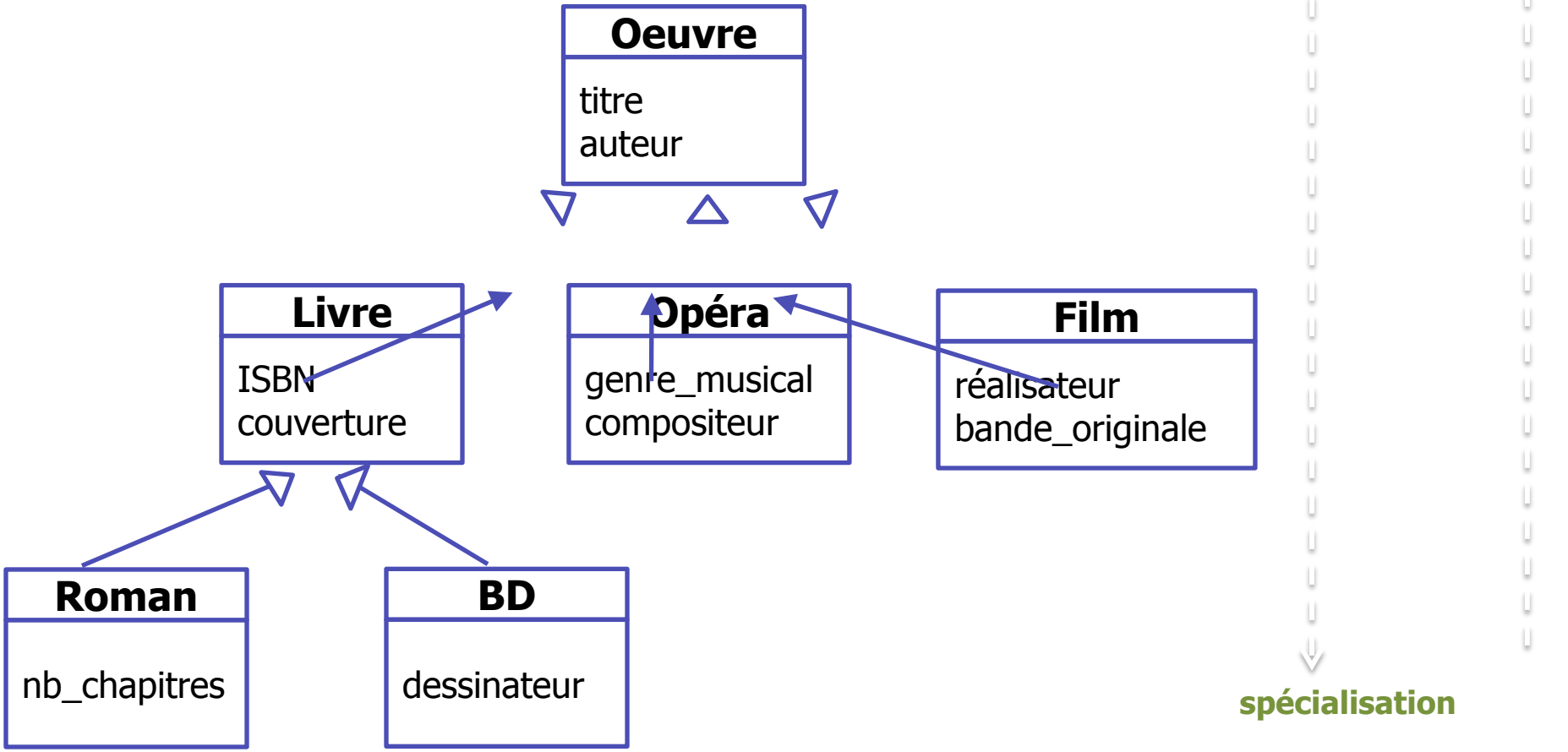




Héritage

- 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

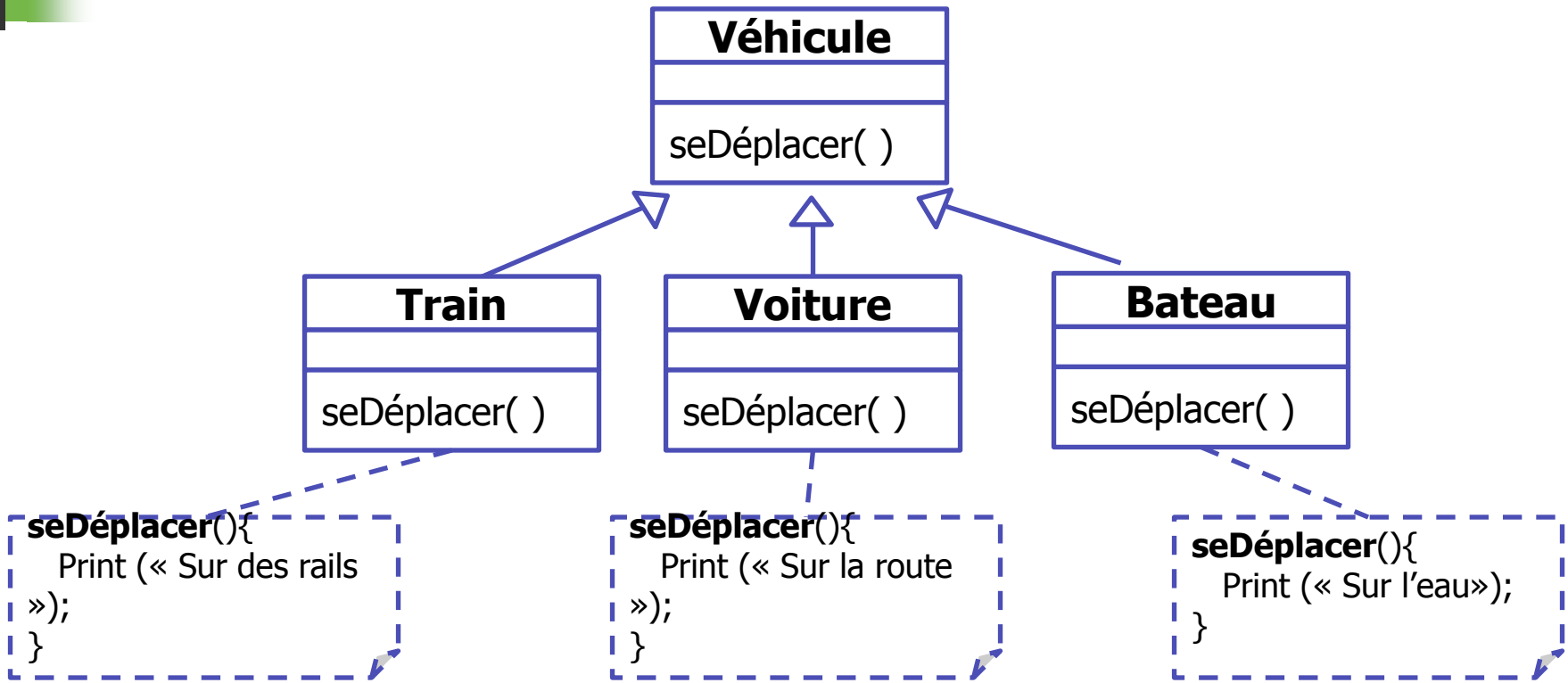
Héritage : Exemple



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



Abstraction

- L'abstraction est la caractérisation d'un objet par *une partie publique, une partie privée et une partie implémentation.*
  - L'accès public
  - L'accès privé
  - La partie implémentation

Abstraction

- Ce concept d'abstraction engendre deux catégories d'acteurs :
  - **les concepteurs des classes**
  - **les utilisateurs des objets**
- Ces derniers peuvent utiliser les méthodes d'une classe indépendamment de leurs structures internes.
- Ce qui permet aux concepteurs des classes d'objets de modifier la structure interne des méthodes des classes sans altérer le travail de leurs utilisateurs.