Ministère de l'enseignement supérieur et de la recherche scientifique
Université Djillali BOUNAAMA - Khemis Miliana (UDBKM)
Faculté des Sciences et de la Technologie
Département de Mathématiques et d'Informatique

Chapitre 3

Programmation orientée objet

MI-GLSD-M1-UEM213 : Paradigmes de langages de Programmation

Noureddine AZZOUZA

n.azzouza@univ-dbkm.dz

Plan du Cours

1. Principe et Motivation
2. Historique et Présentation
3. Concepts de Base
4. Importance du paradigme objet

Principe & Motivation

Principe

- ✓ La **programmation orientée objet** (p.o.o) repose sur le concept d'objet : chaque entité manipulée est autonome « **objet** » et utilise des requêtes « **messages** » pour interagir avec d'autres objets.
- ✓ Un mécanisme de définition et de création des objets permet de créer des modèles d'objets « **classes** » décrivant la structure des objets et leurs comportement lors de l'arrivée des requêtes.
- ✓ Regroupement des entités sous une description structurelle et comportementale.

Besoin

✓ Communauté de Génie Logiciel

- ❑ Augmentation de la demande de nouveaux logiciels (applications)
- ❑ Augmentation de la complexité de ces logiciels.
- ❑ Résoudre les problèmes de : Développement, maintenance, limite de la réutilisation

✓ Communauté des programmeurs

- ❑ Besoin d'encapsulation
- ❑ Problème de la programmation procédurale et modularité

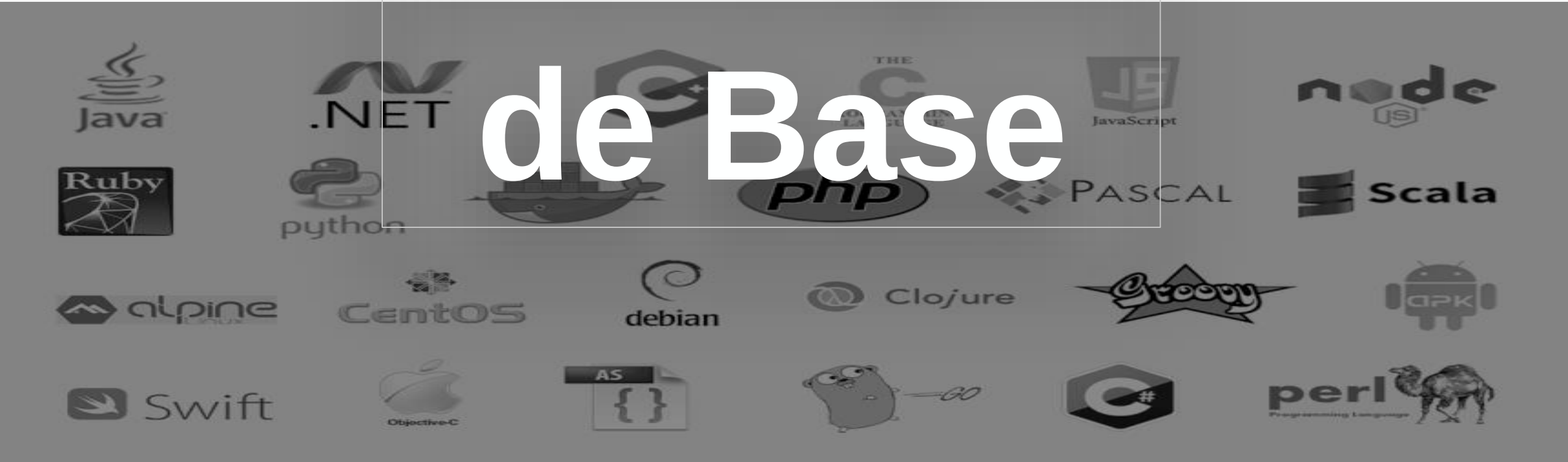
Le problème du monde réel est caractérisé par des objets et leurs interactions : Un système développé en utilisant la programmation orientée objet doit produire un système proche de la représentation du monde réel que celui de la programmation procédurale

Historique et Présentation

Historique et présentation

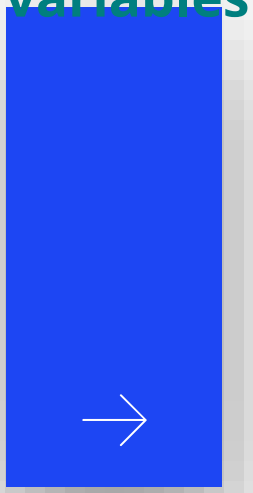
- ✓ **SIMULA 1963** : premier langage mariant les données et les procédures pour les systèmes de simulation.
- ✓ **SIMULA 1967** : classifie les concepts d'objet et de classe.
- ✓ **SMALTALK 72** : généralise la notion d'objet et introduit l'interaction entre objets par les messages.
- ✓ **SMALTALK 80** : définit une forme et uniformise le modèle en perfectionnant la définition de la classe « méta classe » ; et depuis, une vague de langages de manipulation d'objets s'est répandu tels que FORMES, XOOL, LOOPS, CEYX, ACT1 et ACT2, C++, GLIPS, CLOS, JAVA.

Concepts de Base



Objet

- ✓ **l'objet** est une collection d'opérations qui partagent un état.
- ✓ Les opérations déterminent les **messages** (appels) auxquels l'objet doit répondre.
- ✓ **L'état** partagé est caché de l'extérieur est accessible uniquement aux **opérations** de l'objet.
- ✓ Les **variables** représentant l'état interne de l'objet sont appelées **variables d'instances** et ses opérations sont appelées méthodes.
- ✓ La collection de **méthodes** détermine son **interface** et son **comportement**.



Objet

- ✓ Un objet est une abstraction d’une donnée caractérisée par un identifiant unique et invariant, un état représenté par une valeur simple ou structurée et une classe d’appartenance.

Objet = Identité + Etat + Comportement

Objet = une instance de sa classe

Exemple

objet : voiture_identité

Attribut	valeur	
Puissance	12 CV	état
Couleur	rouge	
Type	coupe	
Marque	FORD	

Opération		
Démarrer 0		comportements
Accélérer 0		
Arrêter 0		
Calculer vitesse		
Calculer la consommation		

Classe

- ✓ Une **classe** d'objets décrit le domaine de définition d'un ensemble d'objets ayant une structure commune, un comportement similaire, une même sémantique et des relations communes avec les autres objets.
- ✓ c'est un mécanisme qui permet de regrouper sous une même définition **des objets** similaires.
- ✓ Chaque objet d'une classe est construit par un certain processus appelé **Instanciation**.

Classe = type abstrait de données qui définit le type et le comportement commun à des objets.

Exemple

Exemple : la classe véhicule

Attributs : puissance : entier.

Couleur : alpha numérique.

Type : alpha numérique.

Marque : alpha numérique.

Opérations : Démarrer / Accélérer / Arrêter / calculer vitesse…….etc.

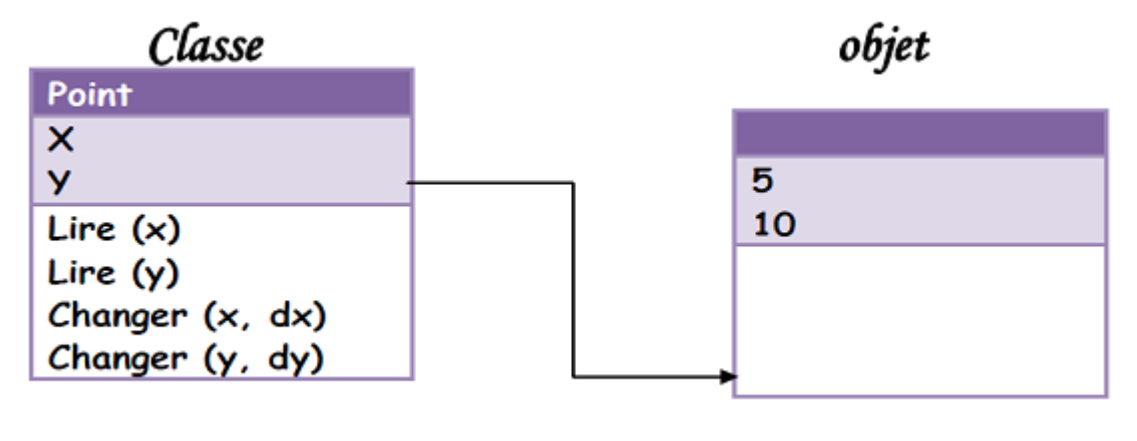
Classe vs. Objet

- ✓ La différence se situe dans la façon dont les attributs et méthodes sont traités dans les objets et les classes.

1. Une classe est une définition sur les objets, les attributs et méthodes dans la classe sont des déclarations et ne contiennent pas de valeurs.

2. Les objets sont traités comme instance de classe.

3. Les valeurs de l'ensemble des attributs décrivent l'état des objets



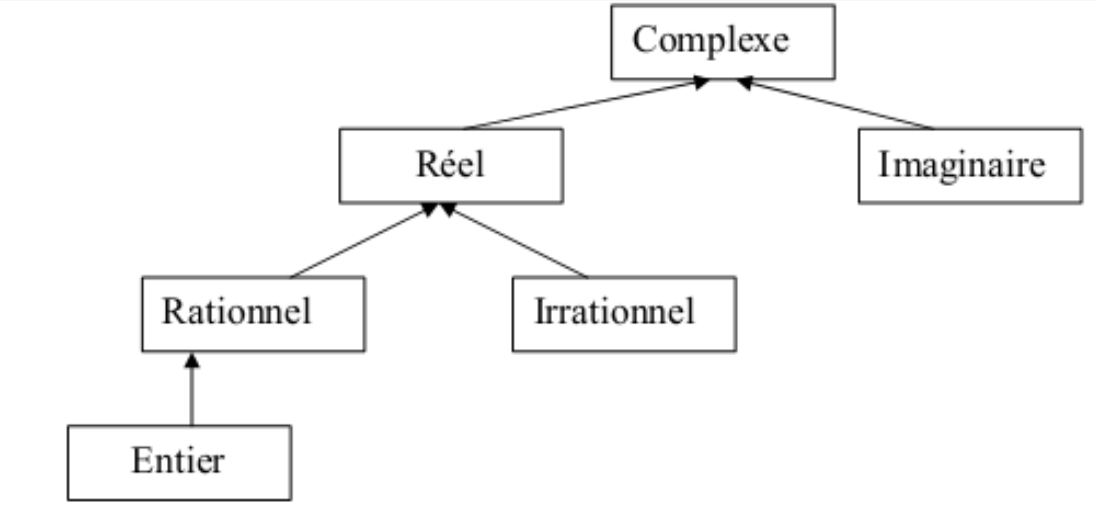
Héritage

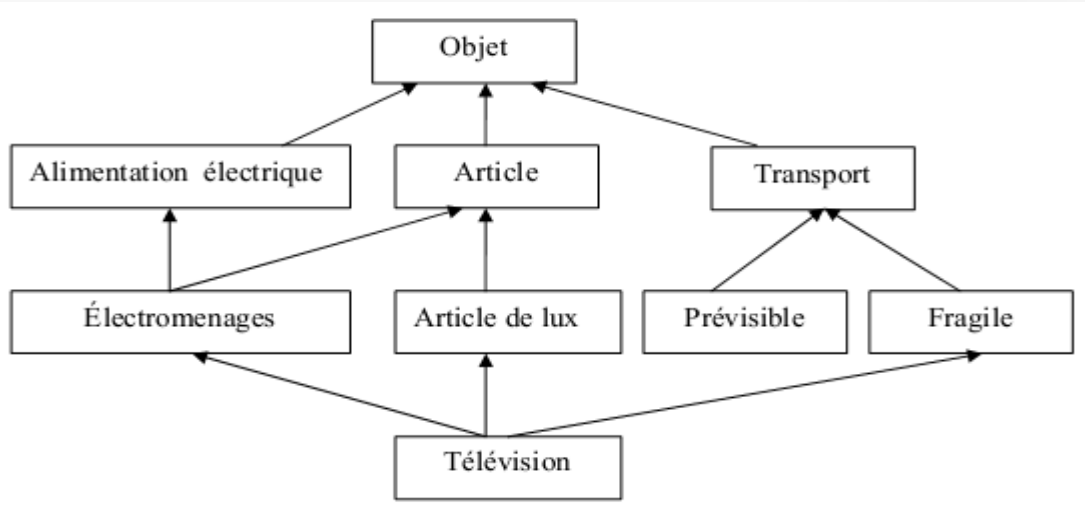
- ✓ La construction d'une hiérarchie de classes par raffinement et spécialisation.
- ✓ Permet aux classes d'hériter du comportement de leurs ancêtres (superclasse).
- ✓ La hiérarchie de classe représente un modèle du problème à résoudre.
- ✓ L'héritage permet de réutiliser le comportement d'une classe dans la définition de nouvelles classes. La sous-classe d'une classe hérite les opérations de la classe parente et doit avoir de nouvelles opérations et de nouvelles variables d'instances.

Héritage = Assure le mécanisme de réutilisation

Formes d'héritage

1. Héritage **pur** (pas de redéfinition du code des méthodes)
2. Héritage avec **surcharge** (overloading)
3. Héritage **simple** (hiérarchie d'héritage = arbre)
4. Héritage **multiple** (hiérarchie d'héritage = graphe)

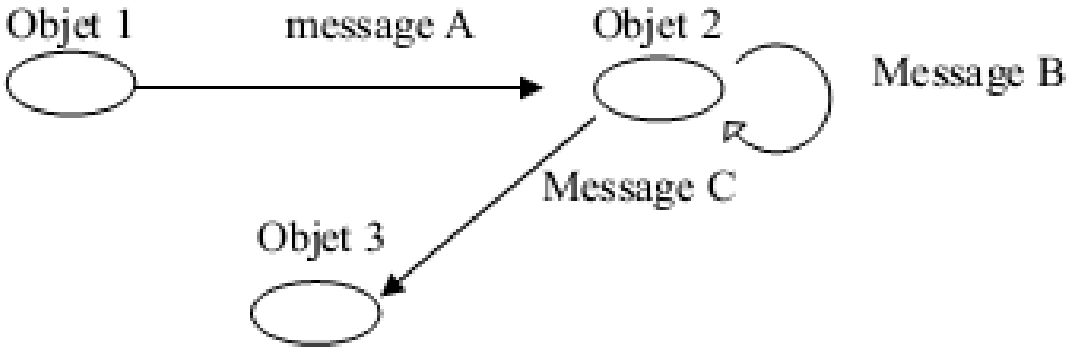




Message

- ✓ Le message est le support de la relation de communication entre objets.
- ✓ Le message relie de façon dynamique les objets.
- ✓ Il permet de restituer et de faire exécuter une fonction de l'application en mettant en collaboration un ensemble d'objets.

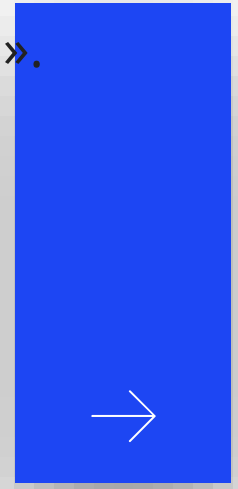
Exemple



Catégories des objets

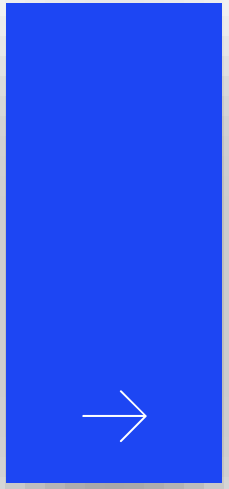
Les objets sont de trois catégories

1. Objet **acteur** : toujours à **l'origine** de l'interaction.
2. Objet **serveur** : toujours **receveur** de messages. « **Destinataire** »
3. Objet **agent** : cumule les comportements des types « **acteur** » et « **serveur** ».
   - ✓ il assure dynamiquement le routage des messages.
   - ✓ les messages sont représentés graphiquement par des flèches.



Encapsulation

- ✓ L'encapsulation spécifie que les objets sont des données encapsulées avec les méthodes (opérations) correspondantes.
- ✓ Le comportement de l'objet est alors décrit par l'interface extérieure qui se résume à un ensemble de messages auxquels l'objet est apte à répondre et il est indépendant de la structure interne de l'objet en raison de protéger les données de l'objet.
  - ❑ L'accès est coordonné par l'emploi de messages de l'interface.
  - ❑ Facilité de modularité, de réutilisation et d'entretien.
  - ❑ Protection des données par encapsulation.
  - ❑ Modularité et maintenabilité «facilité de modification ».
  - ❑ Réutilisabilité par instanciation, héritage et composition.
  - ❑ Lisibilité.



Importance

Importance du paradigme objet

- ✓ Le concept d'objet a progressivement été identifié comme un paradigme permettant de répondre non seulement à des besoins de programmation, mais aussi d'analyses et de conception.
- ✓ La classe est une synthèse de deux moyens d'abstraction classiques : Module et Type.
- ✓ L'héritage est un moyen de délégation permettant d'hiérarchiser (classifier) les classes et fournit aussi un outil de maîtrise supplémentaire de la complexité.
- ✓ La paradigme objet est donc un moyen très général pour modéliser des concepts abstraits aussi que des entités concrètes du monde réel