UTOPIOS
TRANSFORMATION DIGITALE INTELLIGENTE

Programmation orientée objet

Sommaire

UTOPIOS
TRANSFORMATION DIGITALE INTELLIGENTE

Introduction

Programmation informatique

- Un algorithme est un processus étape par étape
- Un programme informatique est un ensemble d'instructions étape par étape pour un ordinateur
- Tout programme informatique est un algorithme

Programmation informatique

L'histoire de la programmation informatique s'éloigne régulièrement des conceptions de la programmation axées sur les machines pour s'orienter vers des concepts et des métaphores qui reflètent plus étroitement la façon dont nous voyons et comprenons le monde.

Langage de programmation

- Les langages de programmation permettent aux développeurs de créer des logiciels.
- Les trois grandes familles de langages sont les suivantes
  - le langage machine
  - le langage d'assembleur
  - Le langage de haut niveau

Langage machine

- Le langage machine, ou code machine, est la suite de bits qui est interprétée par le processeur d'un ordinateur
- Il est composé d'instructions et de données à traiter codées en binaire
- Ce langage est difficile à lire ou programmer, une simple erreur de 0 ou de 1 peut faire planter le programme

```
1110000110101
1001010000101
```

Langage Assembleur

- le langage assembleur est le langage de plus bas niveau qui représente le langage machine sous une forme lisible par un humain
- Les combinaisons de bits du langage machine sont représentées par des symboles « mnémoniques »
- Un langage assembleur est propre à une famille de processeur (ex: x86, ARM)

```
movb $0x61,%al
```

Langage haut niveau

- Un langage de haut niveau est orienté autour du problème à résoudre, qui permet d'écrire des programmes en utilisant des mots usuels des langues naturelles et des symboles mathématiques familiers
- Il existe plus de 200 langages haut niveau

```
String firstName = "John"
String lastName = "Doe"
String fullname = firstName + lastName
```

Paradigme de programmation

- Historiquement on divise les langages de haut niveau en 2 catégories
  1. **Les langages procéduraux**
  2. **Les langages orientés objets**
- Aujourd'hui la plupart des langages ne peuvent pas être strictement classés dans un paradigme car ils comportent plusieurs caractéristiques de plusieurs paradigmes

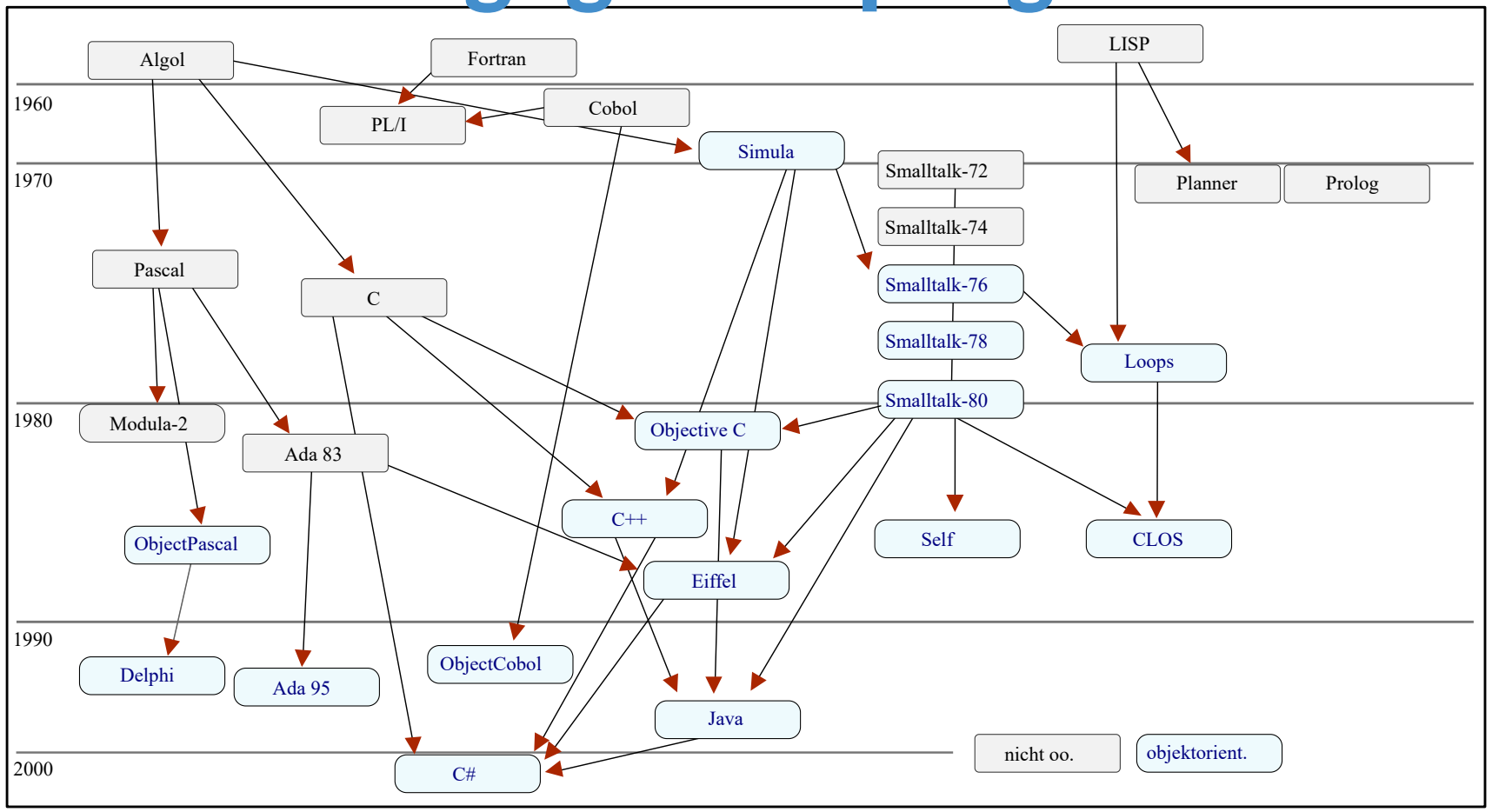
Langage procédural

- Les premiers langages de haut niveaux étaient appelés langages procéduraux
- Les langages procéduraux se caractérisent par des ensembles séquentiels de commandes linéaires. L'accent de ces langages est situé sur la structure du code
- Exemples de langages procéduraux : C, COBOL, Fortran, LISP, Pascal, BASIC

Langage Orienté Objet

- Les langages orientés objets ne sont pas axés sur la structure, mais sur la modélisation des données
- Les développeurs utilisent des "plans" pour concevoir des modèles de données que l'on appelle des **classes**
- Exemple de langages orientés objets : Java, C#, C++, ...

Histoire des langages de programmation



- Simula (Simple universal language) a été créé en 1962 sous la dénomination Simula I par Ole-Johan Dahl et Kristen Nygaard
- Le langage évolua en 1967 sous le nom de Simula 67 en implantant le premier le modèle de classe de Hoare
- Simula est le père de tous les langages à classe comme Smalltalk, C++, Java, Eiffel
- Smalltalk introduira la programmation orientée objet quelques années plus tard

UTOPIOS
TRANSFORMATION DIGITALE INTELLIGENTE

Concepts de base

Définition de la POO

- La POO est une philosophie de conception de programmes
- Elle consiste en la définition et l'interaction de briques logicielles appelées **objets**
- Un objet représente un concept, une idée ou toute **entité** du monde physique, comme une voiture, une personne ou encore une page d'un livre
- La POO permet de transcrire les éléments du réel sous forme virtuelle

Mécanisme de la POO

- En POO, on définit non seulement le **type de données** d'une structure de données, mais aussi les **types d'opérations/méthodes** qui peuvent être appliquées à la structure de données
- La structure de données devient un objet qui qui comprend à la fois des données et des fonctions (méthodes) en une seule unité
- En outre, les structures peuvent créer des **relations** différents objets (ex: héritage)

Citation

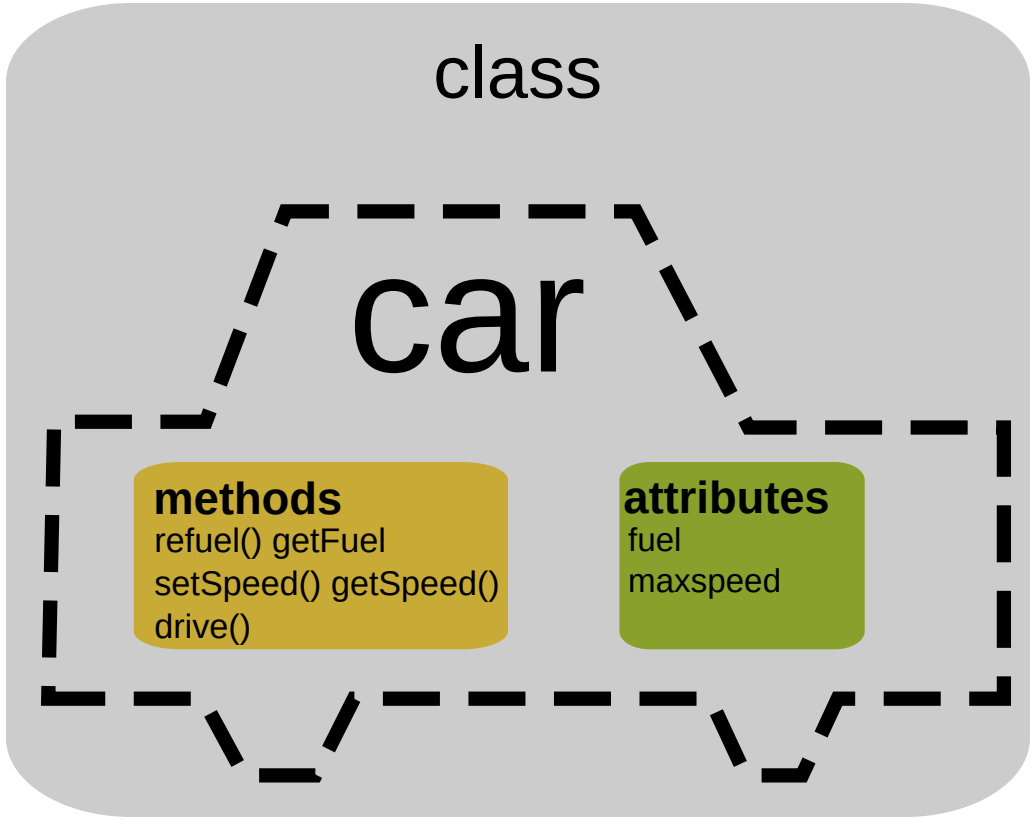
💡 L'idée clé derrière la programmation orientée objet:

> “ Le monde réel peut être décrit de manière "précise" comme une collection d'objets qui interagissent. ”

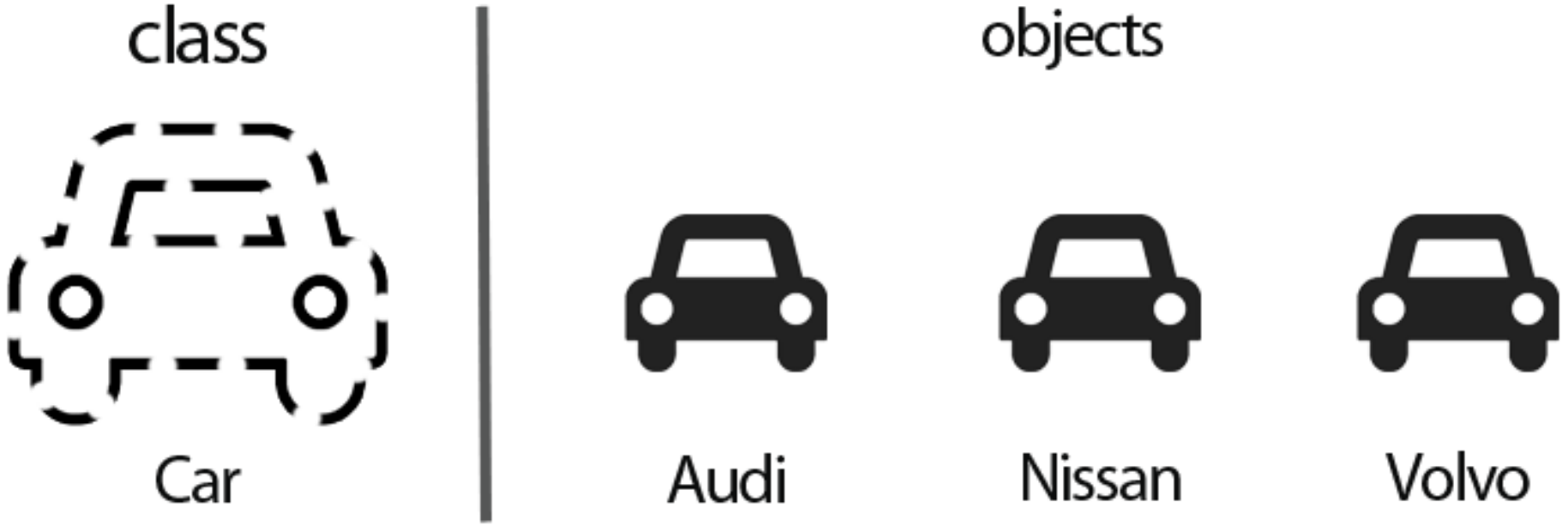
Terminologie

- **Objet**: instance d'une classe, c'est-à-dire un élément qui possède les caractéristiques et les comportements définis par la classe. Ex: Une Tesla rouge
- **Classe**: un modèle qui décrit les caractéristiques et les comportements communs d'un ensemble d'objets. Ex: Voiture
- **Méthode**: fonction associée à une classe ou à un objet, qui définit un comportement spécifique. Ex: Démarrer()
- **Propriété**: attribut associé à une classe ou à un objet, qui définit une caractéristique spécifique. Ex: Couleur

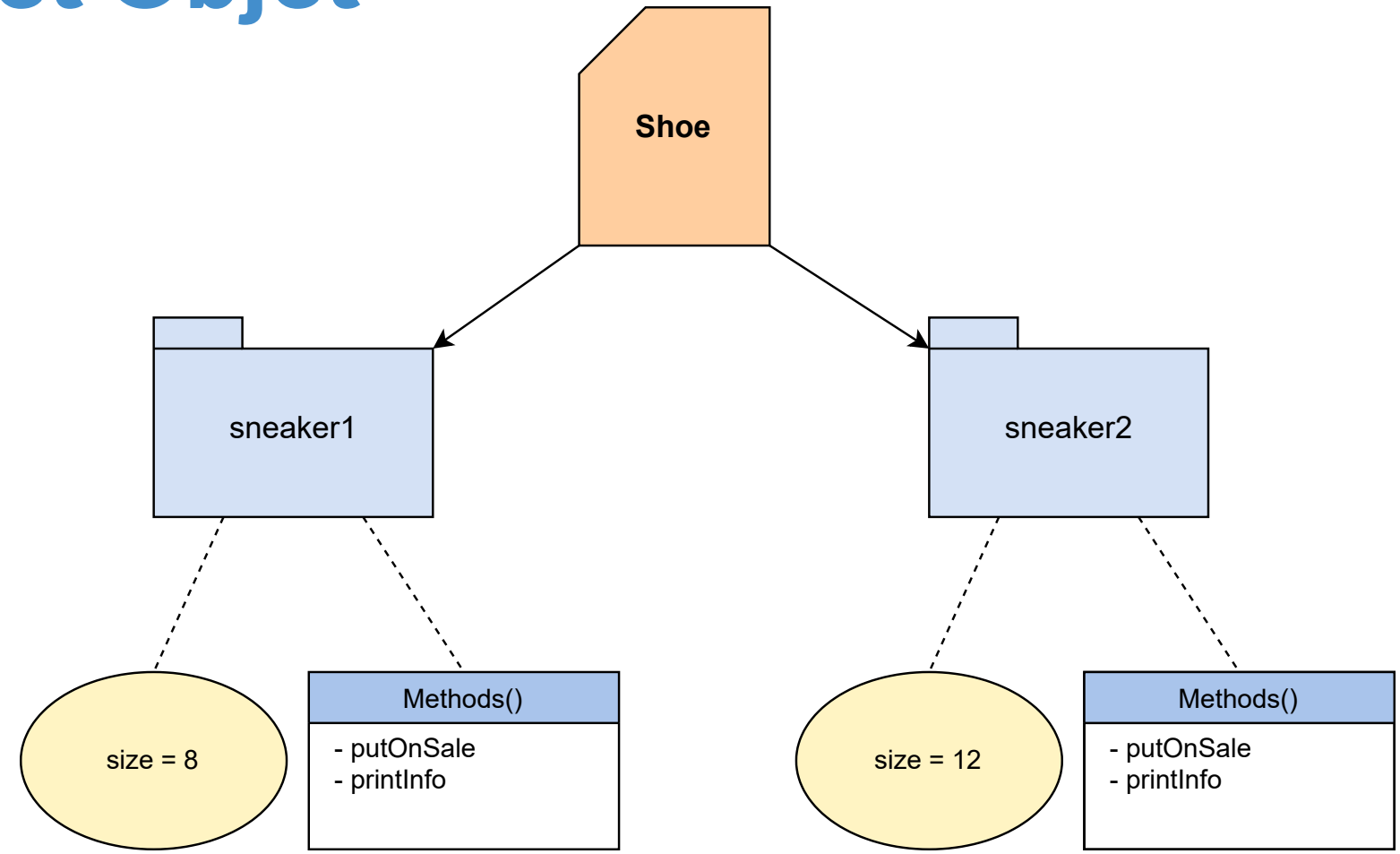
Representation d'une classe



Classe et Objet

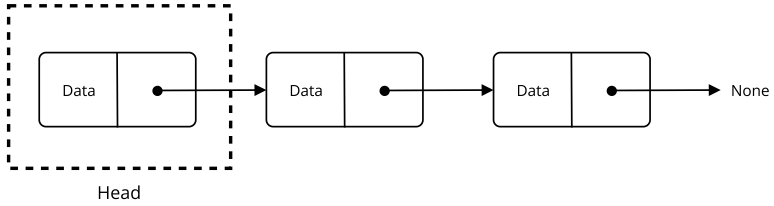


Classe et Objet



Détail sur les objets

De manière informelle, un objet représente une entité, qu'elle soit physique, conceptuelle ou logicielle



État d'un objet

Un objet est une entité informatique qui :

1. Encapsule un état
2. Est capable d'effectuer des actions, ou des méthodes, sur cet état
3. Communique avec d'autres objets par le passage de messages

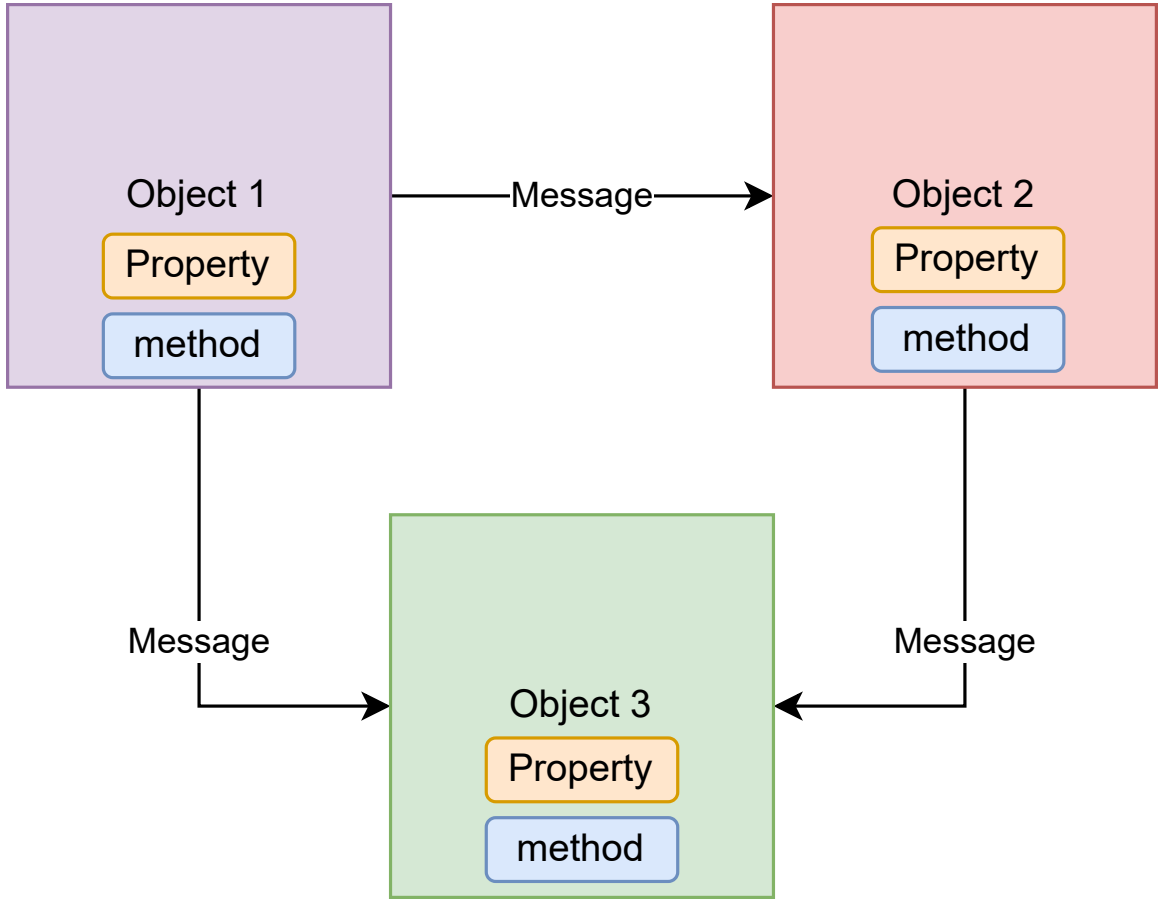
En résumé

- Un **objet** est une **classe** qui prend vie !
- Homo Sapien est une **classe**, John et Jack sont des des **objets**
- Animal est une **classe**, Félix le chat est un **Objet**
- Véhicule est une **classe** La Tesla Modèle 3 de mon voisin est un **objet**
- Galaxie est une **classe**, la Voie lactée est un **objet**

Différence Classe/Objet

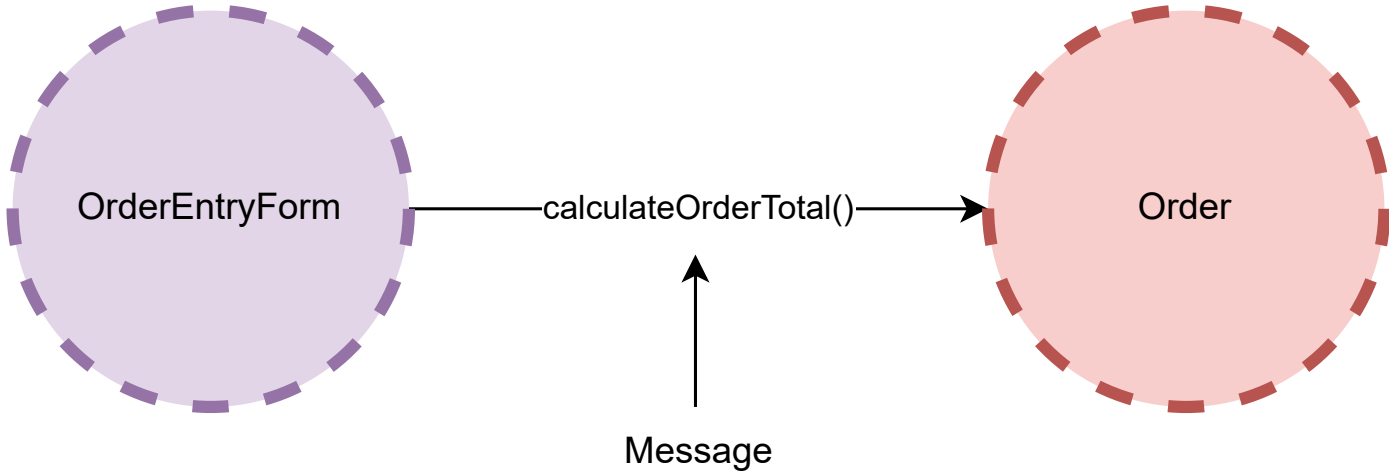
Classe	Objet
Une classe est un type de donnée	Un objet est une instance de classe
Génère des objets	Donne vie à une classe
N'occupe pas d'espaces en mémoire	Occupe de l'espace en mémoire
Ne peut pas être manipulé (sauf static)	Peut être manipulé

Interaction entre objets



Exemple d'interaction

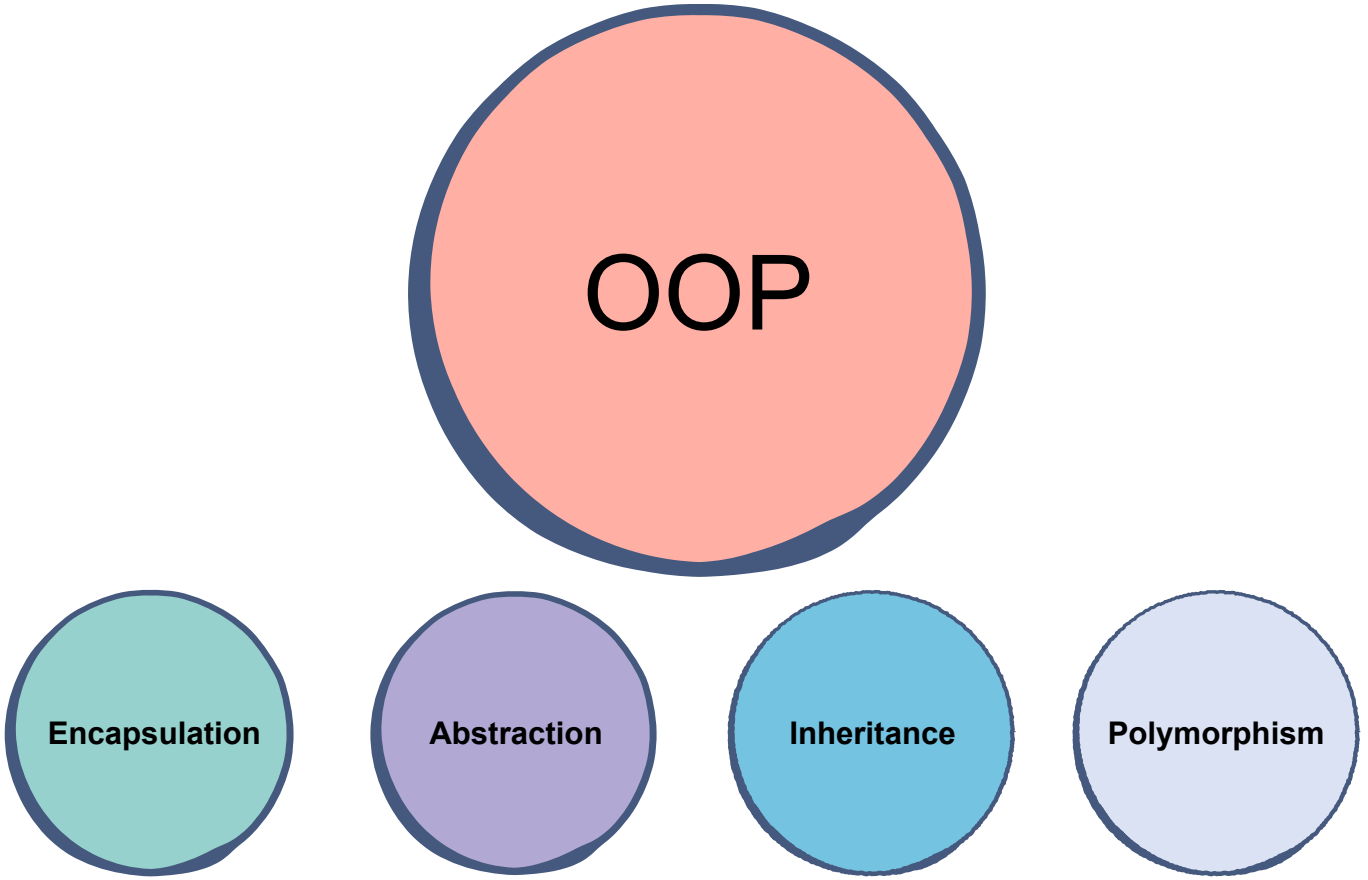
- Le formulaire OrderEntryForm souhaite que Order calcule la valeur totale de la commande
- La classe Order est responsable de calculer le total de la commande



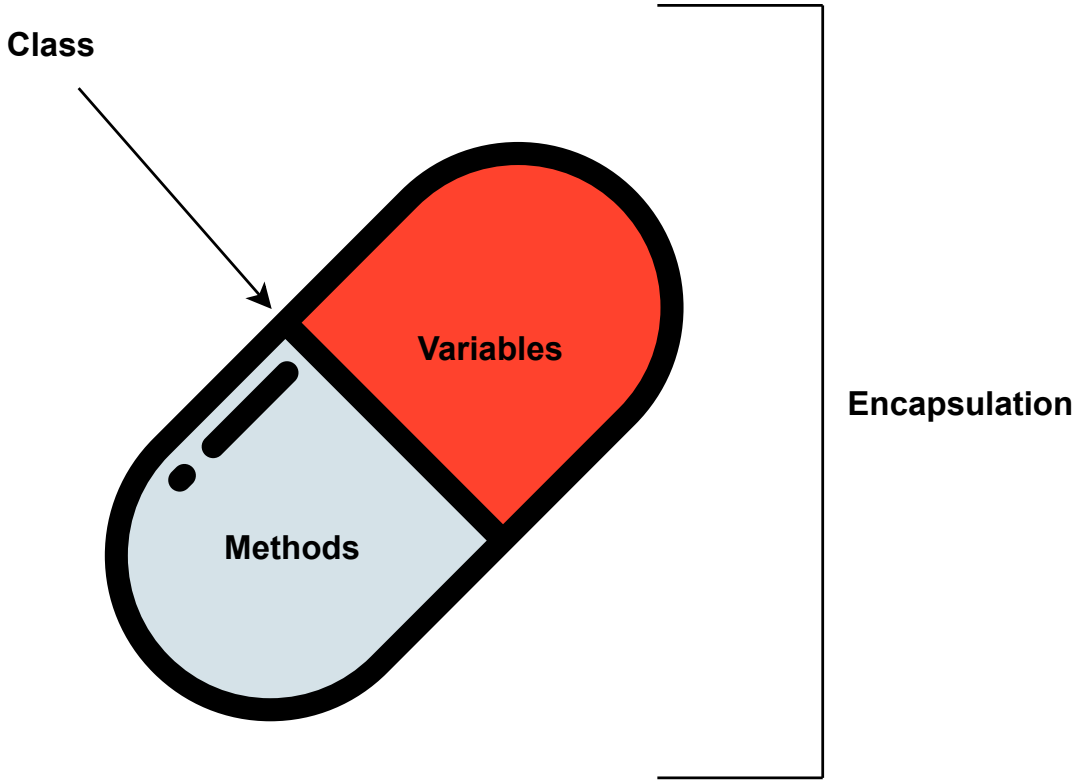
UTOPIOS
TRANSFORMATION DIGITALE INTELLIGENTE

Les 4 principes de la POO

Les 4 piliers de la POO



L'encapsulation



Définition de l'encapsulation

- L'encapsulation est un moyen de restreindre l'accès direct à certains composants d'un objet, de sorte que les utilisateurs ne peuvent pas accéder aux valeurs d'état de toutes les variables d'un objet particulier
- L'encapsulation peut être utilisée pour masquer à la fois les membres des données et les fonctions ou méthodes associées à une classe ou à un objet instancié

Visibilité

- L'encapsulation permet de définir des niveaux de visibilité des éléments de la classe
- On définit généralement 3 niveaux de visibilités :
  - **public** : accessible à l'intérieur et en dehors de la classe
  - **private** : accessible uniquement dans la classe
  - **protégée** : accessible dans la classe et les classes dérivées

Avantages de l'encapsulation

- **Favorise la modularité** et la réutilisabilité du code avec les classes génériques et l'héritage
- **Simplification de la maintenance**, en limitant les dépendances entre les classes et en facilitant le changement d'implémentation
- **Améliore la lisibilité**, en masquant les détails d'implémentation et en mettant en évidence l'interface publique de la classe

L'héritage

L'héritage est un concept de la programmation orientée objet qui permet de :

- Définir une classe à partir d'une autre classe existante, appelée **classe parent** ou **superclasse**
- **Hériter des attributs** et **des méthodes** de la classe parent, et les modifier ou les compléter selon les besoins
- Favoriser la **réutilisation du code** et la hiérarchisation des concepts

Concept de l'héritage

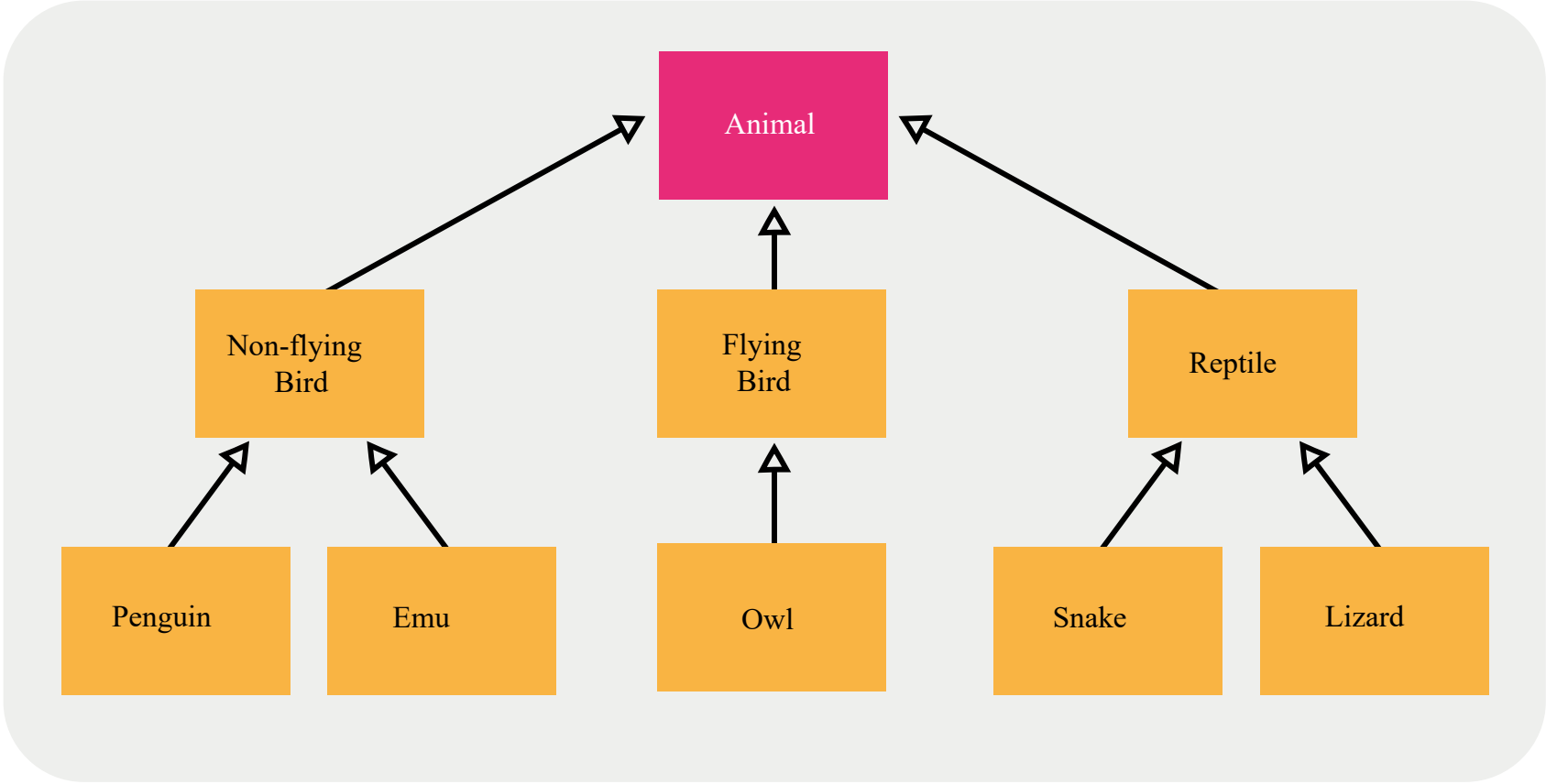
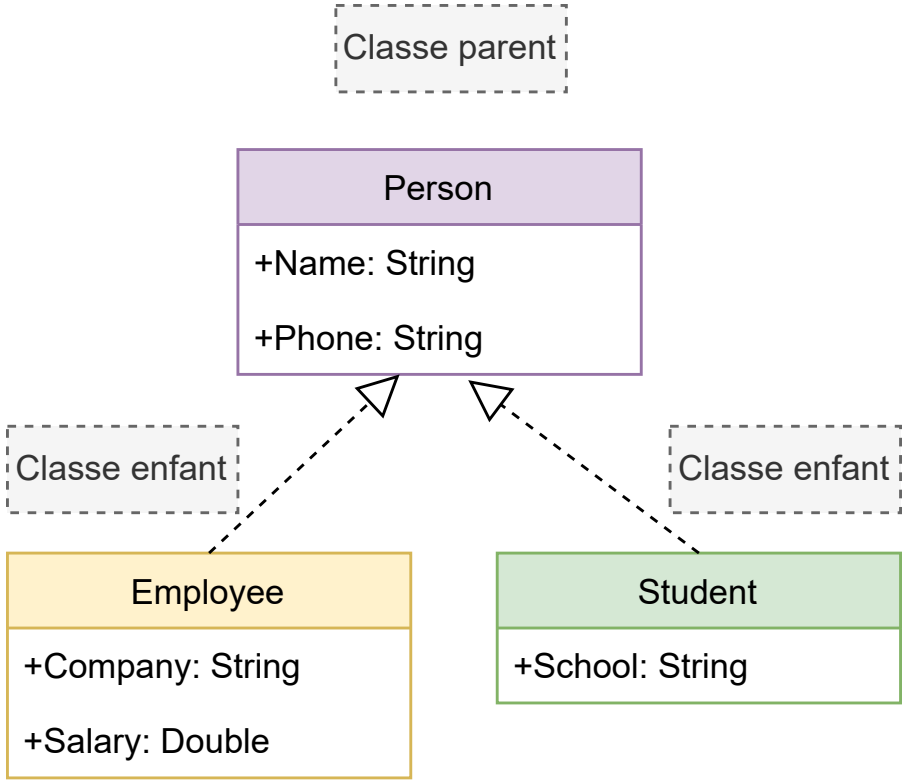
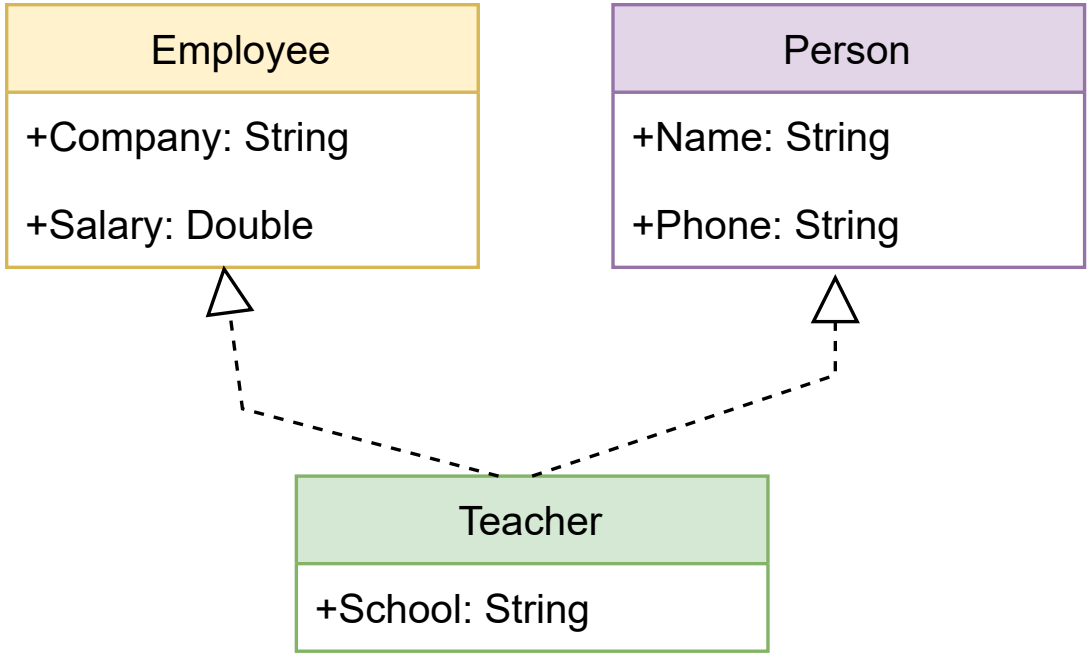


Diagramme de classe



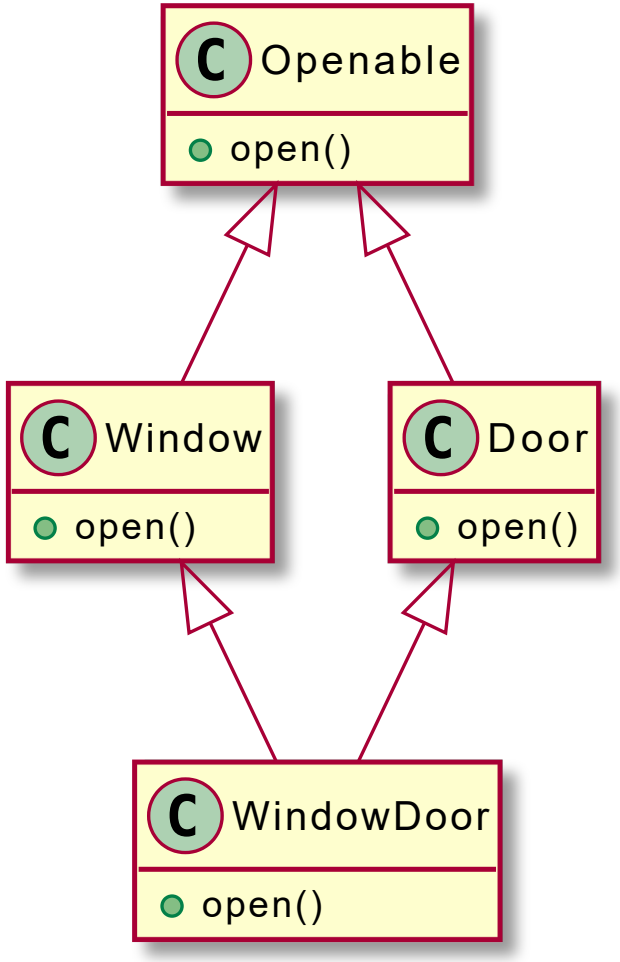
Héritage multiple

- Une classe peut également hériter de plusieurs classes
- ⚠️ la plupart des langages ne supportent pas l'héritage multiple



Héritage en diamant

- Le problème de l'héritage en diamant se produit lorsqu'une classe enfant hérite de deux classes parentes qui partagent toutes deux une classe grand-parent commune

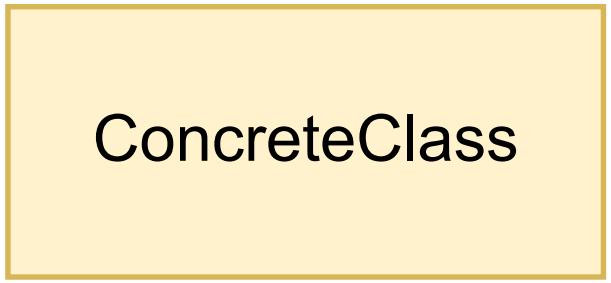


Classe abstraite

Une classe abstraite en POO est une classe qui :

- Ne peut pas être instanciée directement
- Peut contenir des méthodes abstraites, c'est-à-dire sans implémentation
- Peut contenir des attributs et des méthodes concrètes, c'est-à-dire avec implémentation
- Sert de modèle pour les classes dérivées qui doivent implémenter les méthodes abstraites

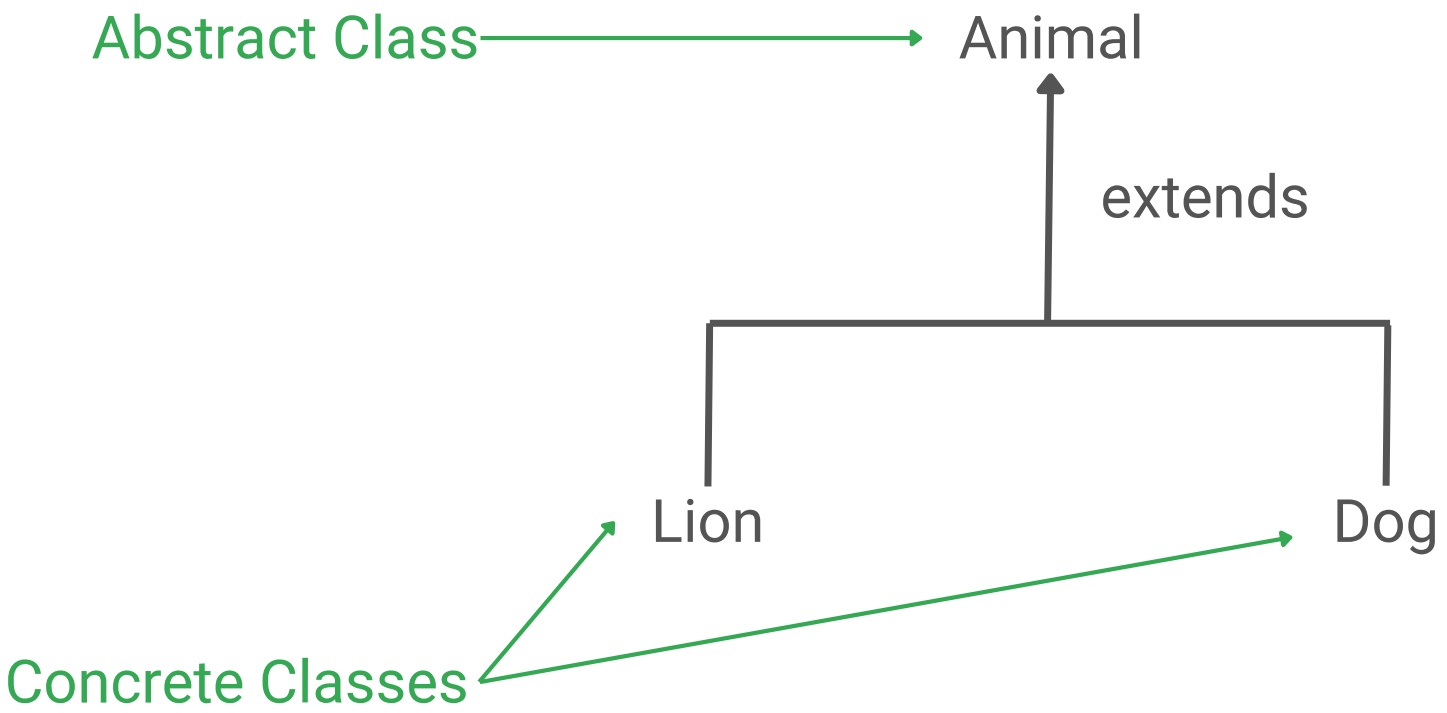
Type de classes



Peut être instanciée

Ne peut pas être instanciée

Exemple de classe abstraite



Le polymorphisme

- Le mot polymorphie vient du grec et signifie "qui peut prendre plusieurs formes"
- Le polymorphisme est utilisé en relation avec les fonctions, les méthodes et les opérateurs.
- Des fonctions et des méthodes de mêmes noms peuvent avoir des comportements différents ou effectuer des opérations sur des données de types différents

Les types de polymorphismes

Sorte	Nom	Description
polymorphisme ad hoc	surcharge	déclarations de plusieurs méthodes ayant le même nom, mais des paramètres différents
polymorphisme paramétré	programmation générique	consiste à définir des fonctions qui peuvent être appliquées à des types paramétrés
polymorphisme par sous-typage	polymorphisme par héritage	redéfinir une méthode dans des classes héritant d'une classe de base

La surcharge

- La surcharge est une possibilité offerte par certains langages de programmation qui permet de choisir entre différentes implémentations d'une même fonction ou méthode selon le nombre et le type des arguments fournis
- Surcharge **statique** : implémentation faite à la compilation en fonction du nombre d'arguments et de leur type statique déclaré
- Surcharge **dynamique** : l'implémentation faite à l'exécution en fonction du type dynamique des arguments

La redéfinition

- Grâce à la redéfinition, il est possible de **redéfinir une méthode** dans des classes héritant d'une classe de base
- Par ce mécanisme, une classe qui hérite des méthodes d'une classe de base peut **modifier le comportement** de certaines d'entre elles pour les adapter à ses propres besoins
- Contrairement à la surcharge, une méthode redéfinie doit non seulement avoir le **même nom** que la méthode de base, mais le **type et le nombre de paramètres** doivent être **identiques** à ceux de la méthode de base

UTOPIOS
TRANSFORMATION DIGITALE INTELLIGENTE

Remarques

Avantages de la POO

- La modularité
- L'abstraction
- La productivité et la ré-utilisabilité
- La sûreté
- La qualité des logiciels
- Rapidité de développement

Inconvénients de la POO

- **La complexité** : conception plus élaborée
- **La verbosité**: nombreuses classes, méthodes, attributs et interfaces
- **La performance**: la POO peut entraîner une surcharge de mémoire et de temps d'exécution
- **La rigidité** : la POO peut limiter la flexibilité et la créativité

Quand utiliser la POO ?

- Quand on veut modéliser un système complexe composé de nombreux éléments interagissant entre eux
- Quand on veut faciliter la maintenance et l'évolutivité du code en le rendant plus lisible et plus cohérent
- Quand on veut profiter des fonctionnalités avancées des langages orientés objet, comme les exceptions, les classes abstraites, les interfaces, etc

UTOPIOS
TRANSFORMATION DIGITALE INTELLIGENTE

Merci pour votre attention

Des questions ?