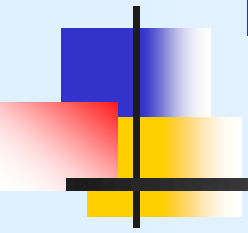


Programmation Objet

Bernard Archimède
Professeur de Universités

EC0908SI0401

Simulation d'un petit écosystème

Prédateur

Vision du prédateur qui balaie l'écran.

Plante

L'eau

Proie

Vision de la proie qui balaie l'écran.

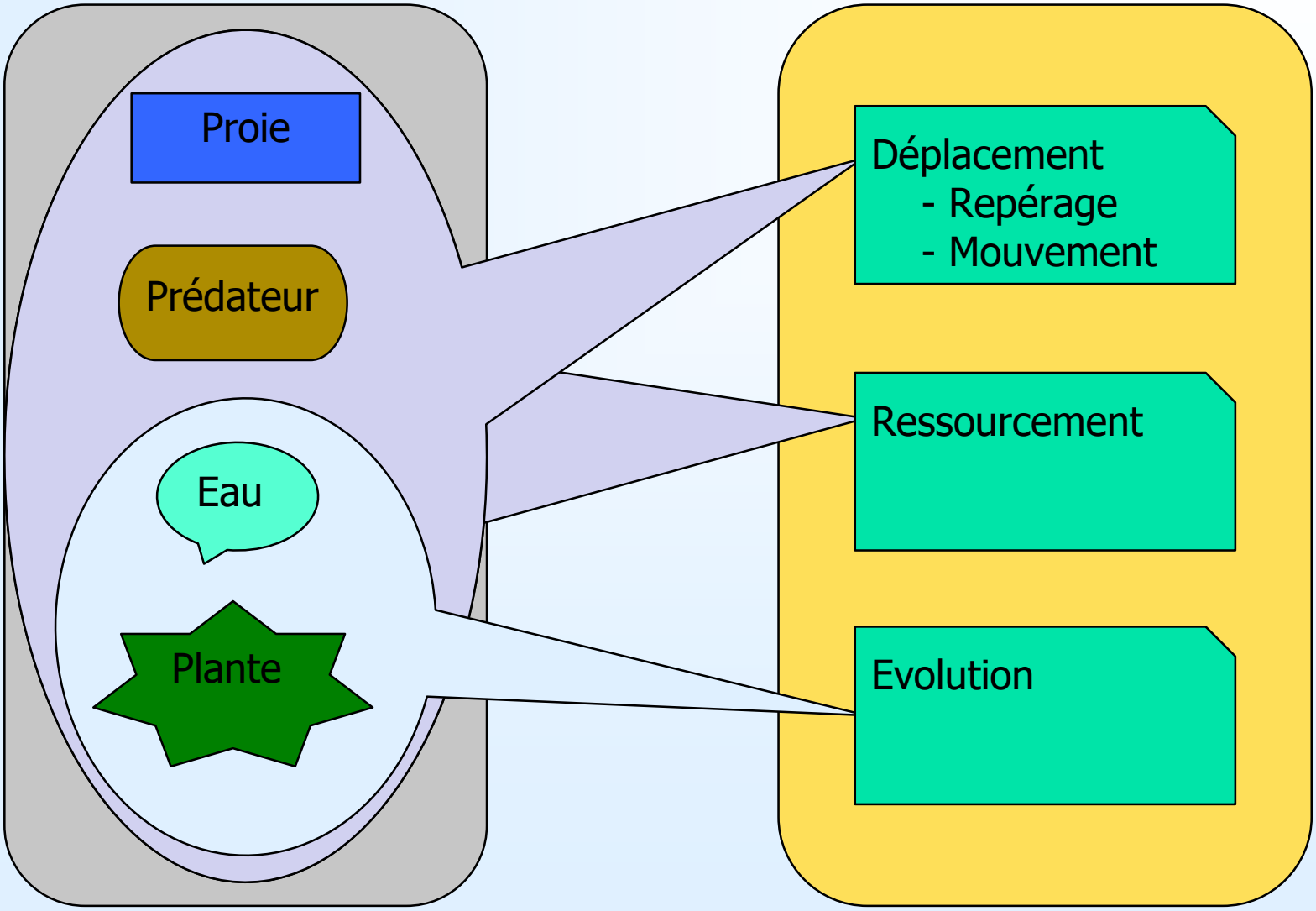
Scénario de la simulation

- La proie se déplace vers l'eau, vers la plante, ou fuit le prédateur ; elle agit en fonction du premier de ces objets qu'elle repère.
- Quand la ligne de vision traverse un objet, quel qu'il soit, l'objet est considéré comme repéré, la vision ne quitte plus l'objet, et l'animal se dirige vers la cible ou la fuit.
- Le prédateur se déplace vers l'eau ou poursuit la proie en fonction du premier des deux objets perçus.
- L'énergie selon laquelle les deux animaux se déplacent décroit au fur et à mesure des déplacements, et conditionne leur vitesse de déplacement.
- Si le prédateur rencontre la proie, il la mange.
- Dès que le prédateur ou la proie rencontre l'eau, ils se ressourcent (leur énergie augmente) et l'eau diminue de quantité (visuellement, la taille de la zone d'eau diminue).
- Dès que la proie rencontre la plante, elle se ressource (son énergie augmente également) et la plante diminue de quantité (sa taille diminue).
- Enfin, la plante pousse lentement avec le temps, alors que la zone d'eau, au contraire, s'évapore.

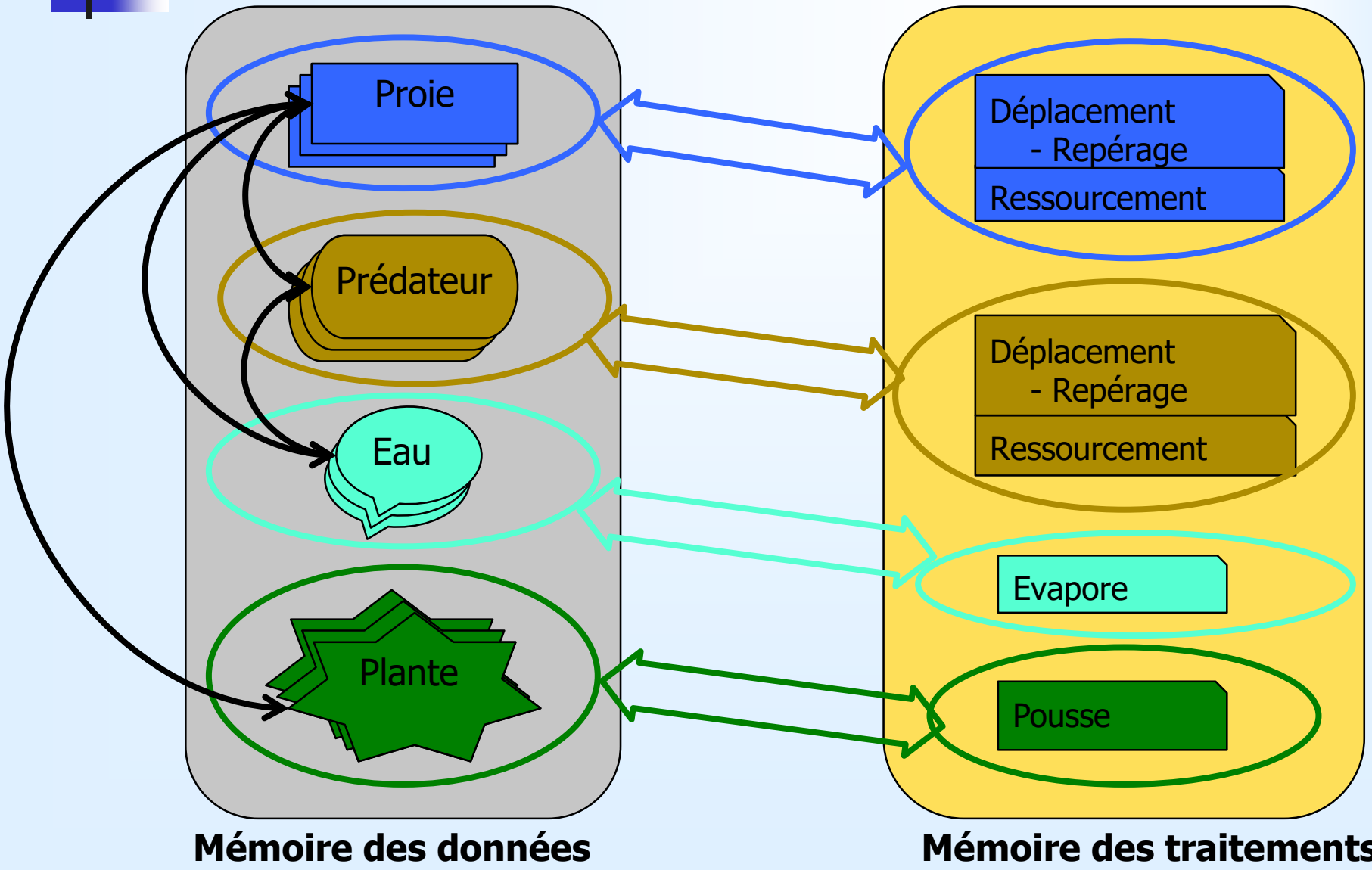
Approche procédurale (Langage C)

- Variables globales
  - Eau, Proie, Prédateur, Plante
- Fonctions
  - Déplacement
    - Repérage
    - Mouvement
  - Ressourcement
  - Evolution

Découpage logiciel par les traitements



Découpage logiciel par les données



Séparation données - traitements : 2 approches

- ***Priorité aux traitements***.
  - La conception fonctionnelle.
  - Sous-programmes imbriqués -> factorisation des traitements.
  - Sous-programmes classés par bibliothèque ou par couches logicielles fonctionnelles.
  - Données accessibles par paramètres et variables globales.
  - Difficile de parvenir à une décomposition naturelle du problème en des modules relativement indépendants du fait du partage des données et de l'interpénétration des fonctions
  - Accroissement des difficultés de maintenance et de mise à jour du logiciel

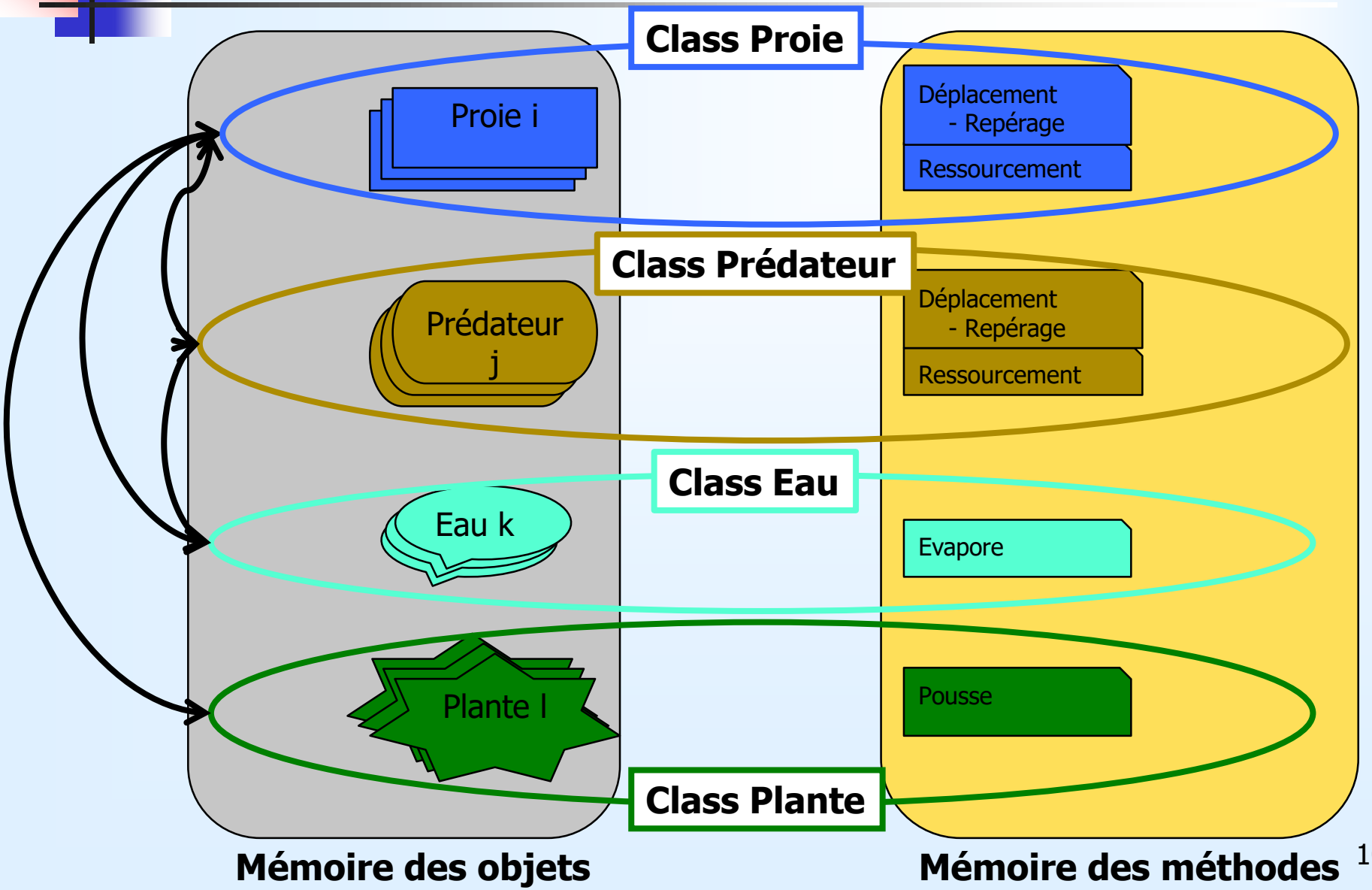
Séparation données - traitements : 2 approches

- ***Priorité aux données***.
  - La conception est dite "dirigée par les données".
  - Sous-programmes -> identification des traitements associés à une donnée.
  - Données aussi accessibles par paramètres et variables globales.
  - Il est plus intuitif de séparer le programme à réaliser vis à vis des acteurs qu'au regard des grandes activités
  - Le changement de la description structurelle d'une donnée, ne concerne qu'un nombre limité de fonctions.

Programmation Objet

- Issue de la deuxième approche
- Données (***attributs***) accessibles par ***encapsulation*** des données dans un "cadre" appelé ***classe***.
- Traitements encapsulés dans la classe sous la forme de ***méthode***.
- Règles strictes d'accès aux attributs et aux méthodes.
- La notion de classe --> l'héritage des classes.

Programmation Objet

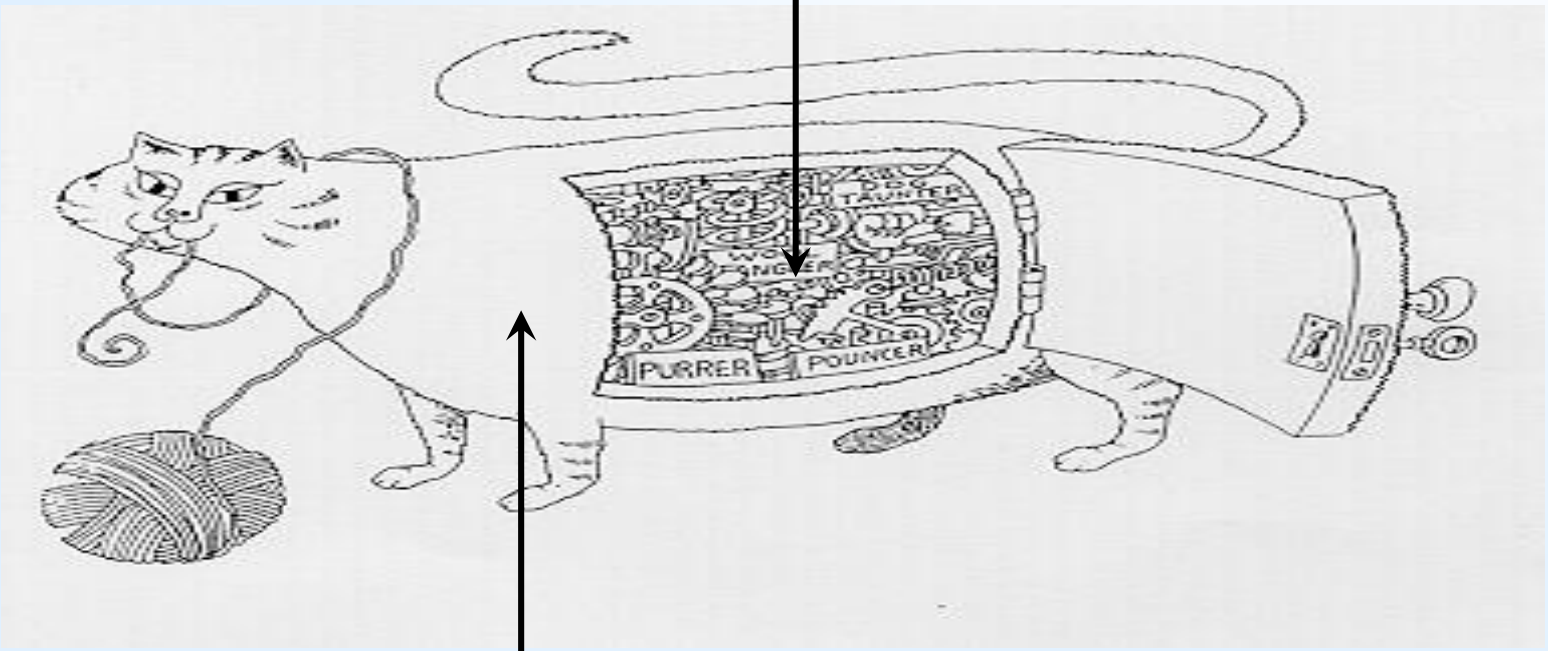


Encapsulation de l'information

- Une **encapsulation** est un regroupement sous un même nom des données et des procédures qui les manipulent et elles seules.
- Toute information concernant un module doit être privée (locale) sauf si elle est explicitement déclarée publique.
- Tout module est connu du reste du monde à travers une description officielle ou interface
- Tout le reste du module est privé

Encapsulation de l'information

Partie Privée



Partie Publique

Encapsulation de l'information (2)

- Avantages :
  - un objet ne dépend que de l'interface des autres objets avec lesquels il est en relation et non du codage des autres objets.
  - les risques d'erreurs sont considérablement réduits, et les conséquences sont mieux maîtrisables.
- Le principe de masquage de l'information met l'accent sur la nécessité de séparer
  - Fonction
  - et représentation
- Outre la continuité, il est également lié aux exigences de :
  - décomposabilité,
  - de composabilité,
  - et de compréhensibilité

Programme Objet : une structure différente

- programme « classique » :
  - composé de traitements qui gèrent des données déclarées globalement (ou localement mais passées en paramètre),
- programme « objet » :
  - composé d'objets dans lesquels se trouvent les traitements.
  - niveau de granularité des traitements beaucoup plus importante
    - entraîne globalement une baisse des performances d'exécution dans le cas de traitement algorithmique représentatif d'un calcul scientifique complexe.
    - possibilité de traitements écrits dans un autre langage, et intégrés dans le langage objet.
      - au détriment de la portabilité de Java sur différentes plates-formes
      - solution dans le système informatique réparti

L'Objet

- Entité concrète, fabriquée à l'aide d'une classe
  - Un objet est n'importe quoi ayant une limite bien définie (Cox)
  - Un objet représente un élément individuel, identifiable, unité ou entité, soit réel soit abstrait, avec un rôle bien défini dans le domaine du problème (Smith)
  - Un objet est une entité du monde physique et/ou social,
  - située dans l'espace et le temps,
  - individualisante par certaines caractéristiques
- Règle de cohérence
  - Les modalités d'interaction à travers la frontière doivent se référer aux attributs de l'objet.
  - tous les attributs retenus doivent être directement ou indirectement concernés par des échanges avec d'autres objets

L'Objet : Définition

- Identité d'un objet
  - propriété d'un objet qui le distingue de tous les autres
- État d'un objet
  - englobe toutes ses propriétés (statiques) et leurs valeurs courantes (dynamiques)
  - aucun objet n'existe isolément. On agit sur les objets, et eux-mêmes agissent sur d'autres objets

Caractéristiques fondamentales

- État : regroupe les valeurs instantanées de tous les attributs d'un objet
  - évolue dans le temps
  - conséquence de ses comportements passés
- Comportement : regroupe toutes les compétences d'un objet
  - décrit les actions et réactions de cet objet
  - opération = chaque atome de comportement
  - diagrammes : interactions entre objets
    - liens interaction entre les objets
    - signifie qu'un objet connaît ou voit un autre objet.
    - les messages naviguent le long des liens.
- Identité : existence propre de l'objet
  - distinguer tout objet de façon non ambiguë
  - indépendamment de son état.

Un Objet

- Est une unité atomique formée de l'union d'un état et d'un comportement
- Fournit une relation d'encapsulation qui assure à la fois
  - une cohésion interne très forte
  - et un faible couplage avec l'extérieur
- Révèle son vrai rôle et sa vraie responsabilité lorsque par l'intermédiaire de l'envoi de messages, il s'insère dans un scénario de communication
- Contient
  - un état interne qui lui est propre
  - un comportement accessible aux autres objets

Comme les êtres vivants, les objets naissent, vivent et meurent

Messages et Méthodes

- Messages
  - Déclenche une activité de l'objet destinataire
- comporte :
  - identité objet destinataire
  - nom de la méthode à activer
  - valeurs pour chacun des arguments spécifiés dans la signature de la méthode
  - le code d'une méthode ne doit manipuler comme données que les attributs de l'objet et respecter le principe de l'encapsulation
- Déclenche une activité de l'objet destinataire
- La liste des messages auxquels un objet peut répondre constitue son **INTERFACE**. (partie publique de l'Objet)

Messages et Méthodes

- Opérations sur un objet
  - fonction de classe activable par message
    - constructeur : crée un objet, initialise son état
    - destructeur : libère l'état d'un objet, détruit l'objet
    - modificateur, transformateur : altère l'état d'un objet (variables d'état)
    - sélecteur, accesseur : accède à l'état d'un objet mais ne le change pas (attributs)

De la programmation structurée à la programmation objet

- **Le programme objet est plus long que le programme C**
  - dû à la création des classes correspondantes aux données nécessaires à la réalisation du programme objet (constructeur, méthodes d'accès aux attributs, …)
- **Programmer par les données :** déclarer (la classe) et de créer (new) des données pour utiliser les traitements.
- **Re-utilisation :** avantage essentiel dans l'utilisation des langages objet
- **Spécialisation :** concept fort de la programmation objet
- **L'écriture "naturelle" :** se rapproche du besoin initial du problème tel qu'un "client" aurait pu l'énoncer

Classes et Objets

- Les objets

- Les classes
  - Boletus Edulis
  - Amanita phalloides
  - Boletus erythropus
  - Amanita muscaria

Classe et héritage

Créer des **classes** dans une conception objet permet d'utiliser une propriété essentielle dans la justification de l'utilisation des langages objet :

l'**héritage**

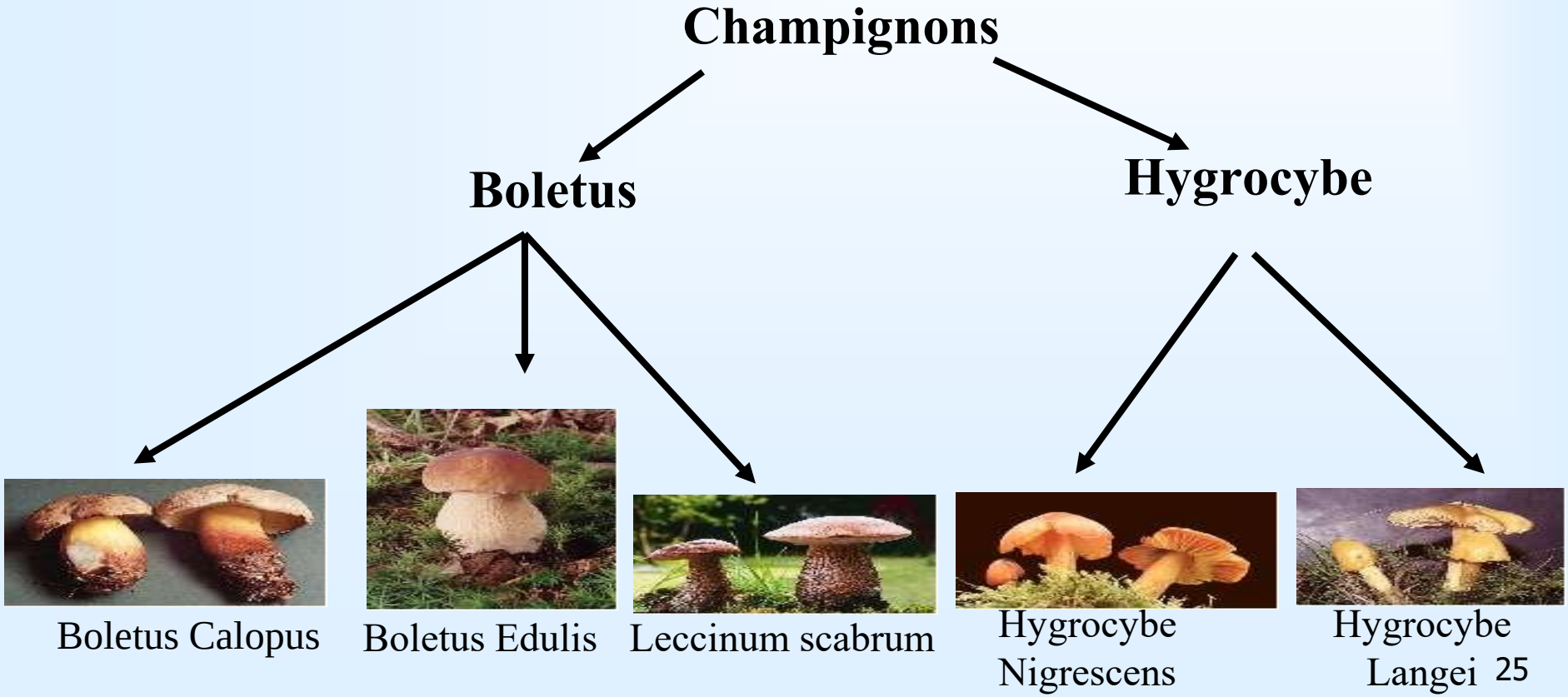
- permet de créer de nouvelles classes qui héritent des précédentes sans modifier le code des classes héritées.
- permet de répondre à un nouveau besoin qui serait d'utiliser un nouveau type d'objet.
- Dans un programme C, il faudrait modifier la fonction dans le code pour toujours

L'héritage

- notion essentielle dans une conception objet
- permet de créer une nouvelle classe "proche" d'une classe existante qui :
  - aura des traitements spécifiques
  - mais qui pourra tout naturellement utiliser les attributs et les méthodes de la classe d'héritage.
  - peut définir de nouvelles méthodes comme des méthodes déjà existantes dans les classes héritées.

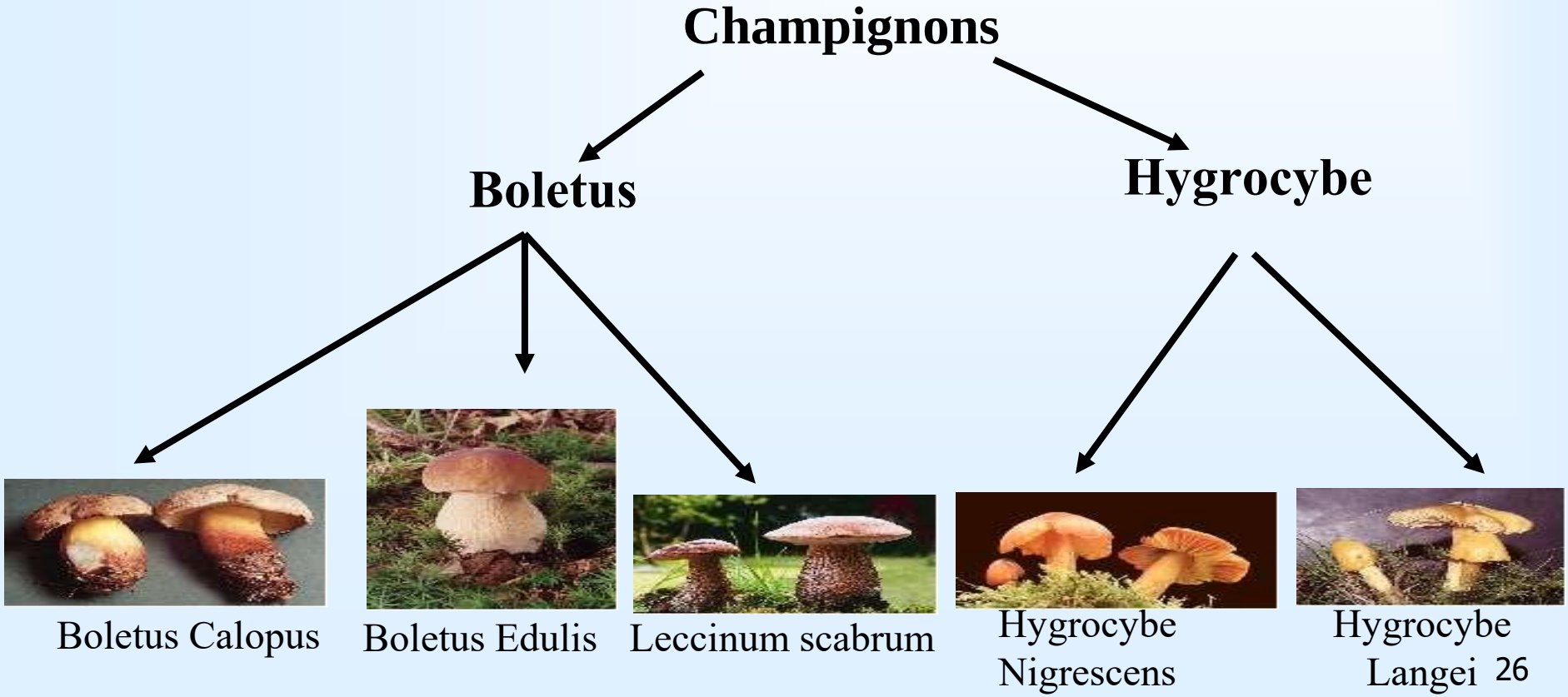
Spécialisation

- La spécialisation est une relation entre une classe et une ou plusieurs versions plus raffinées de celle-ci



Généralisation

- La généralisation est le mécanisme inverse qui consiste à lier des classes ayant des caractéristiques communes à une classes ayant ces caractéristiques



Exemples

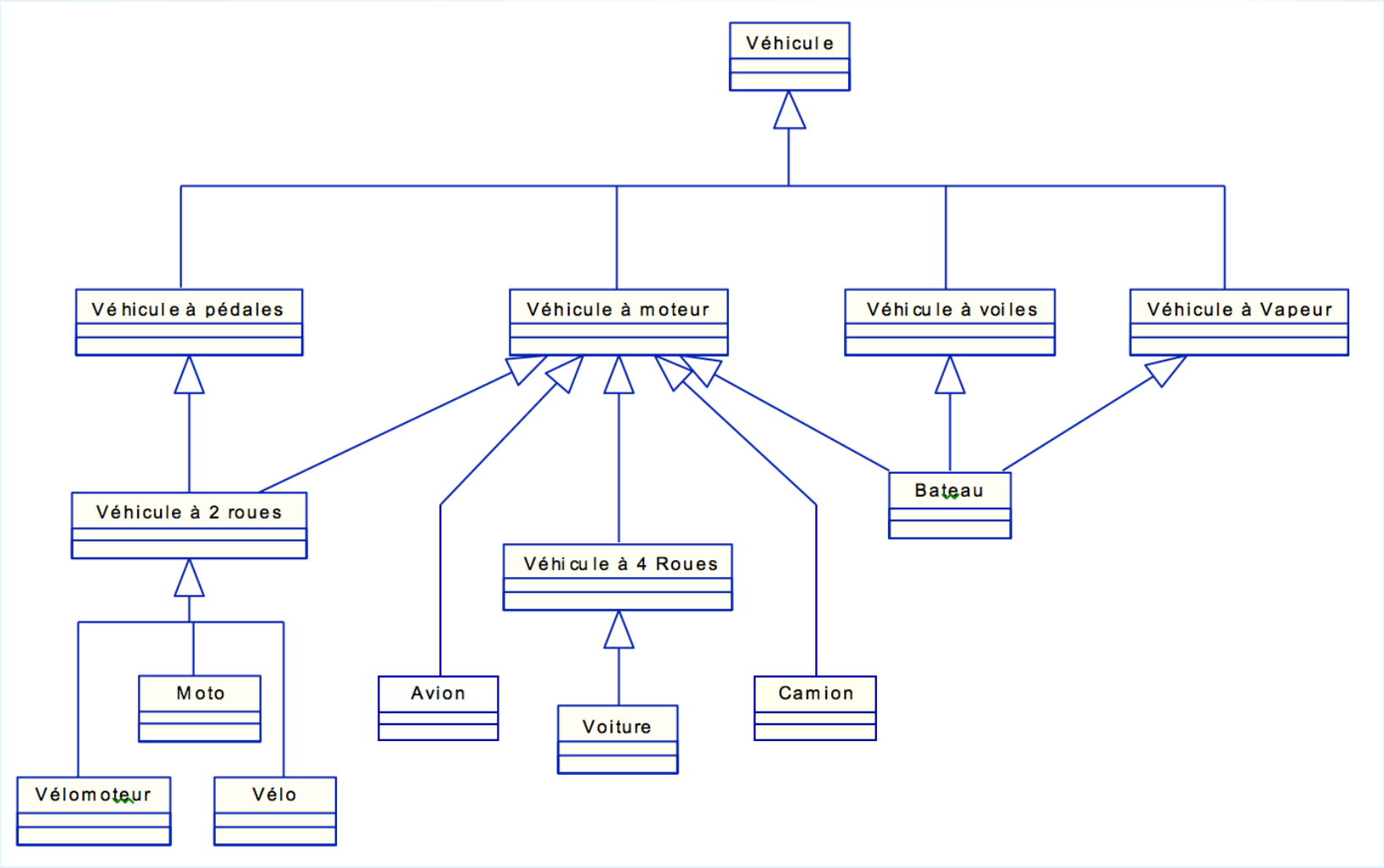
- Un rectangle et un carré héritent du parallélogramme

- Un véhicule à moteur, un véhicule à pédales héritent d'un véhicule
- Une voiture, un avion, un camion, un bateau héritent du véhicule à moteur
- Une voiture, un camion, un vélo héritent du véhicule terrestre
- Un vélo, un vélomoteur et une moto héritent d'un véhicule à 2 roues

- Un mammifère, un amphibien, un oiseau, un invertébré héritent de l'animal

- Question : de quelles propriétés, comportements, héritent chacun de ces objets?

Exemple


Véhicule
Véhicule à pédales
Véhicule à moteur
Véhicule à voiles
Véhicule à Vapeur
Véhicule à 2 roues
Véhicule à 4 Roues
Bateau
Moto
Avion
Voiture
Camion
Vélomoteur
Vélo

Généricité et **Polymorphisme**

- un traitement **générique** est un traitement algorithmique généralisé à tout un groupe de type de données. Par exemple :
  - un algorithme de tri, de gestion de pile, de file, ….
  - la manipulation des éléments d' un corps, d' un anneau en mathématique
- Dans les langages objets, il est plus simple de créer ses traitements génériques, indépendants du type des données que l' on manipule
- **Le polymorphisme :** possibilité de créer des "collections" de données de types différents.
  - La majorité des langages répondent à ce besoin en intégrant deux notions :
    - le pointeur
    - le "casting" ou conversion de type.
    - En langage C, on implémente le polymorphisme avec :
      - un tableau de pointeur
      - une structure d'"union" permettant de couvrir tous les types de données
  - En langage objet, on implémente naturellement avec :
    - un tableau d'objet de classe "Nom"
    - toutes les données pouvant être mises dans le tableau peuvent appartenir à n'importe quelle classe du moment que celle-ci hérite de la classe "Nom "

Terminologie

- classe mère
  - immédiatement supérieure au niveau hiérarchique (surclasse)
- classe fille
  - immédiatement inférieure au niveau hiérarchique (sous-classe)
- métaclasse
  - classe de plus haut niveau
- classe dérivée
  - obtenue à partir d'une autre par un processus d'héritage
- classe différée
  - certaines fonctions n'ont pas encore de code
- classe abstraite
  - ne peut pas créer d'objets.
  - contient au moins une fonction générique

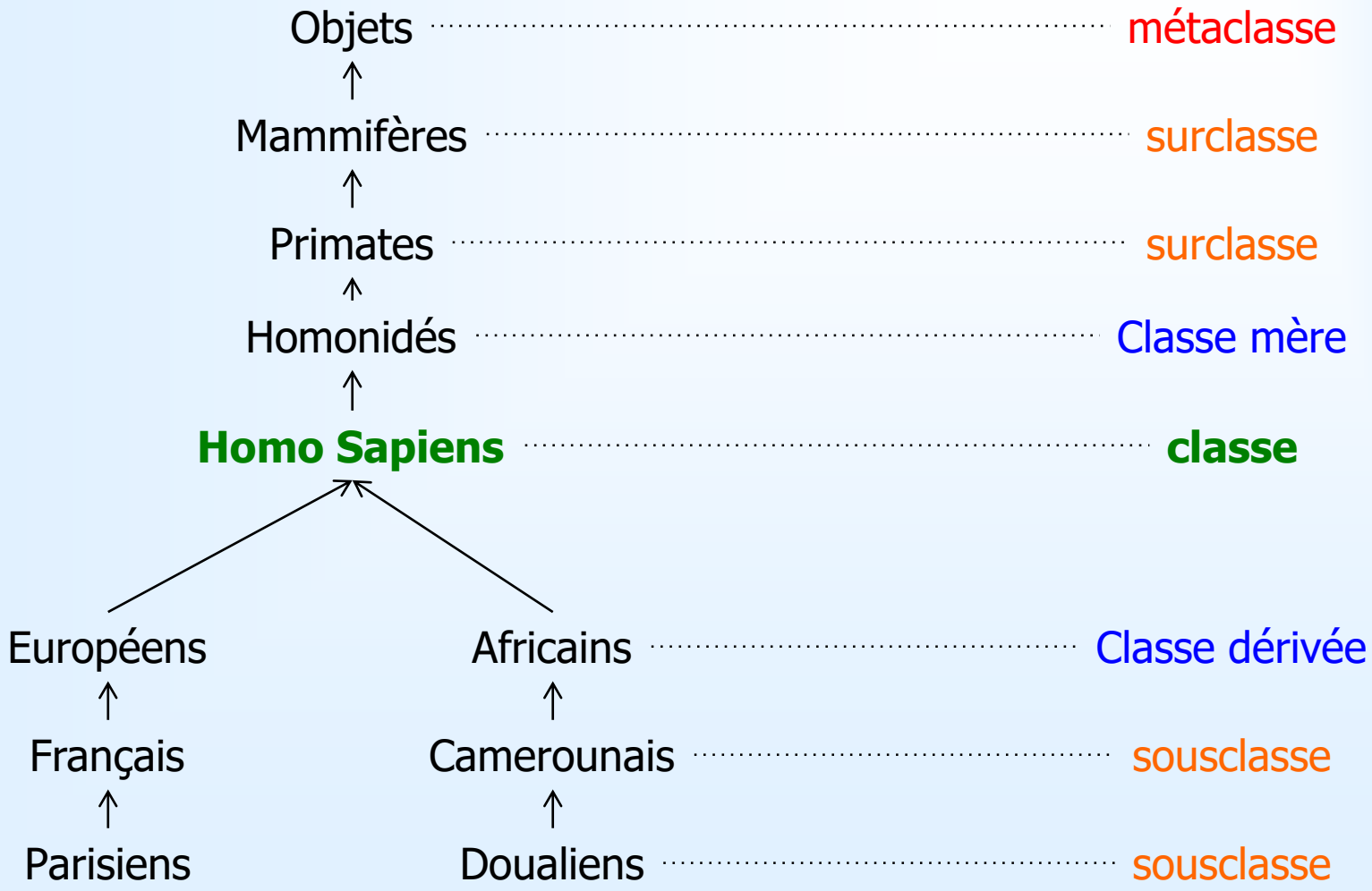
Terminologie

- classe générique
  - modèle pour d'autres classes
  - objets de la classe non prédéfinis
  - instancier un objet -> instancier la classe -> définir le type des objets manipulés

- Exemple : classe pile, type d'éléments non connu

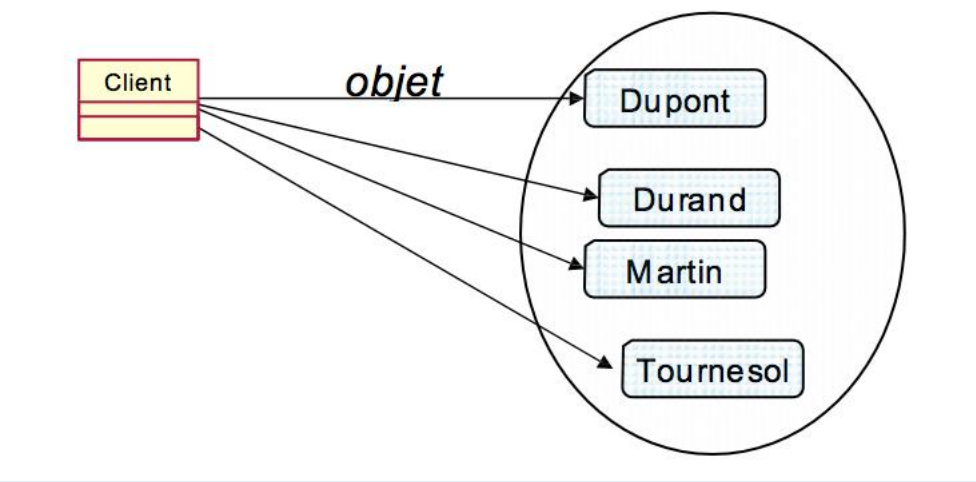
  - classe Pile[X] (classe génératrice d'autres classes)
  - **instanciation d'objets**
    - var Pile_Entiers as Pile[integer]
    - var Pile_Livres as Pile[livre]

Terminologie



Notion d'instanciation

- Mécanisme qui permet, à partir de la classe, la création d'un objet ou instance
  - Une classe peut être instanciée plusieurs fois
  - Une instance d'une classe est un objet
  - Les instances partagent les définitions des attributs mais non leur valeur
  - Les instances partagent les méthodes de la classe



Modélisation

- Modéliser des propriétés statiques et dynamiques de l'environnement ou **domaine du problème**
- Formaliser notre perception du monde et des phénomènes qui s'y déroulent
- Mettre en correspondance l'espace du problème et l'espace de la solution, en préservant la structure et le comportement du système analysé
- formes de collaboration entre les objets qui composent le système.
- **qualité fondamentale des objets** :
  - distinction des composants et **intégration**
  - organisation harmonieuse des composants plus élémentaires → démarche d'**intégration**
- L'approche objet tire sa force de sa capacité
  - à regrouper ce qui a été séparé
  - à construire le complexe à partir de l'élémentaire
  - intégrer statiquement et dynamiquement les constituants d'un système

Contraintes de réalisation

- Persistance des objets :
  - capacité d'un objet à transcender le temps ou l'espace
    - les objets non persistants :
    - transitoires ou éphémères. ← par défaut
- Transmission des objets d'un processus vers un autre.
  - l'objet initial est analysé lors de l'émission
  - la description de l'objet est transmise au travers du support de communication
  - un clone de l'objet est reconstruit lors de la réception
  - l'objet initial est supprimé
- Les objets miroirs
  - alternative à la transmission des objets
  - se comporte comme un autre objet avec lequel il est synchronisé

Communication entre objets

Comportement global d'une application

- 3 catégories de comportement :
  - acteurs : objets à l'origine d'une interaction.
    - Objets actifs
    - Possèdent un fil d'exécution (thread)
    - Passent la main aux autres objets
  - serveurs : destinataires des messages
    - Objets passifs
    - Le flot de contrôle est passé au serveur par l'objet qui envoie le message, et est récupéré après exécution du service
  - agents : acteurs + serveurs
    - peuvent interagir avec les autres objets à tout moment
- Message
  - unité de communication entre objets
  - appel de procédure, événement discret, interruption, datagramme UDP, recherche dynamique, ….

Langages & objets

- Langage à objet
  - expression générique pour langages d'acteurs, centrés objet et orientés objet.
- Langage centré objet
  - pas de distinction entre concepts
    - de classe, d'instance, d'envoi de messages et d'héritage
  - mais le concept de structuration objet existe (package Ada)
- Langage basé sur objet (LBO)
  - notions d'encapsulation et d'identité d'objet
- Langage basé sur classe (LBC)
  - LBO + notions d'abstraction
- Langage orienté objet (LOO)
  - LBC + héritage, + auto récursion + envoi de message
- Langage orienté classe
  - LOO dans lesquels chaque instance appartient à une classe mais où les classes peuvent ne pas avoir de surclasses

Tout est objet

- Propriété propre aux langages qui
  - peuvent définir tous les constituants du langage comme des objets
  - possèdent la notion de méta-classe (Java)
- Cette propriété confère au langage
  - une plus grande puissance due à la modifiabilité aisée de tout ou partie du système,
  - et un caractère encore plus dynamique que celui proposé par les langages d'acteurs.
- En général, les langages où tout est objet sont basés sur Lisp et fondés sur le modèle mathématique ensembliste.

Langages de l ’ Orienté Objet

C++

Eiffel

Java

SmallTalk

Caractéristique	SmallTalk	Eiffel	C++	JAVA
Typage	-	++	+	+
Encapsulation	+	+	+	+
Héritage	+	+	+	+
Héritage multiple	-	+	+	-
Polymorphisme	+	+	+	+
Persistance	-	+/-	-	-
Concurrence	-	+/-	-	+/-
Garbage collector	+	+	-	-
Généricité	-	+	+	-
Bibliothèques	+	+	+	+