Algorithmique et UML

CM1-1 : Initiation à la programmation

Mickaël Martin Nevot

V1.9.0

Algorithmique et UML

Programme (informatique)

- Enchaînement d'**instructions**
- Écrit dans un **langage de programmation**
- Exécuté par des appareils informatiques
- Pour effectuer une tâche donnée
- **Cycle de vie** :
  - Construction
  - Distribution
  - Utilisation (installation ?)
  - Abandon (évolution rapide du marché, des modes, etc.)

Programmation informatique

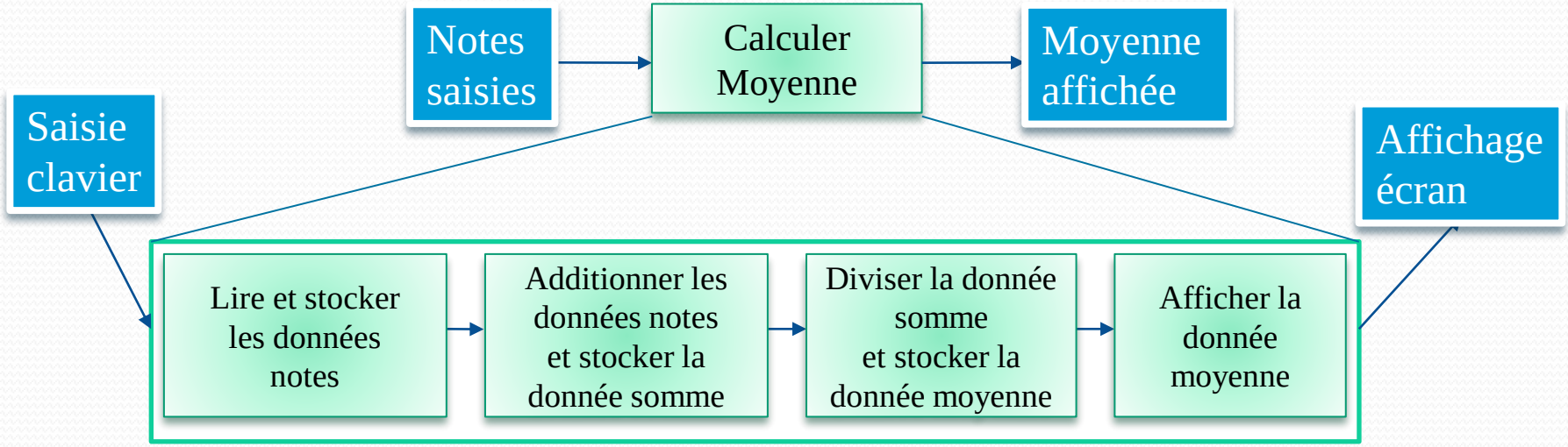
- Conception :
  - Données à traiter
  - Méthode employée (algorithme)
  - Résultat (données en sortie)

Programmation informatique

- Codage :
  - Contraintes d'architecture (choix du langage ?)
  - Langage de programmation
- Transformation du code source :
  - **Compilation** ou interprétation
- Test (important !)

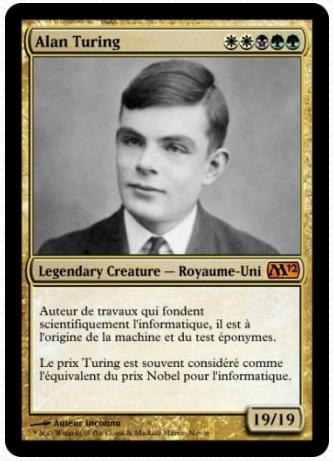
Programmation informatique

- Conception
- Codage
- Transformation du code source
- Test



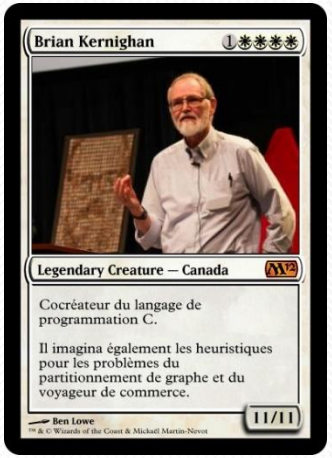
Programmation : historique

- **IX^e siècle** : Al Khuwarizmi (algorithmique)
- **1842** : Charles Babbage et Ada Lovelace (programmation)
- **1847** : Boole (logique)
- **1936** : Turing (machine de)



Programmation : historique

- **1954** : système d'exploitation et Fortan
- **1959** : Grace Murray Hopper (compilateur/bogue)
- **1960 - 1980** : cartes perforées
- **1970** : programmation structurée
- **1980 – 1990** : programmation orientée objet (POO)

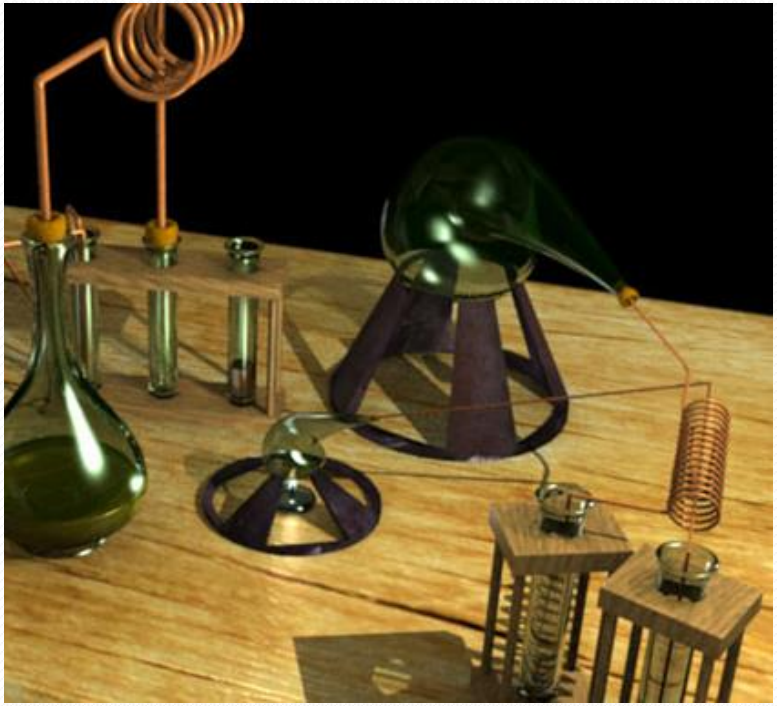


Environnement de programmation

- Éditeur
- Compilateur :
  - Vérifier la syntaxe
  - Traduire en langage machine

Qu'est-ce qu'un objet ?

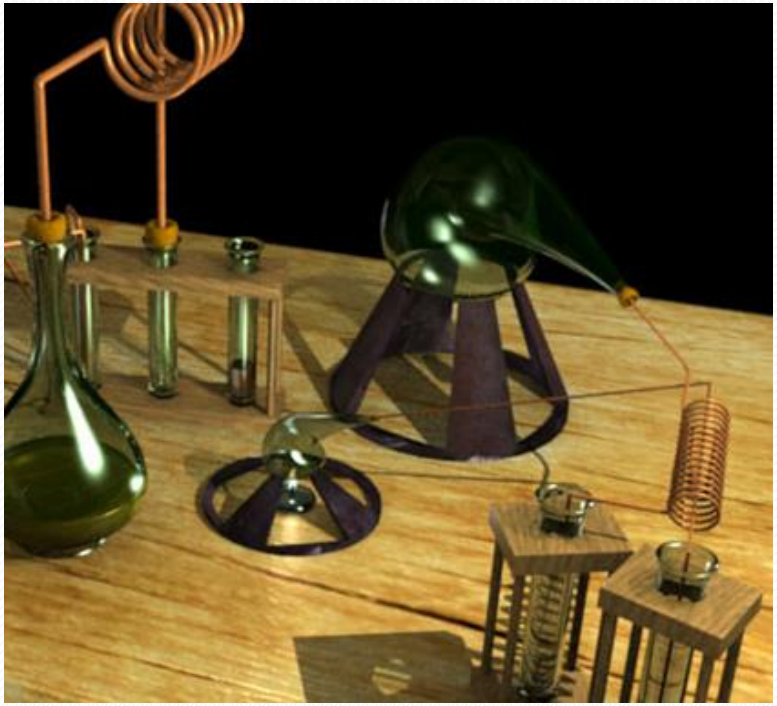
Code source traditionnel



Code source orienté objet

Qu'est-ce qu'un objet ?

Code source traditionnel



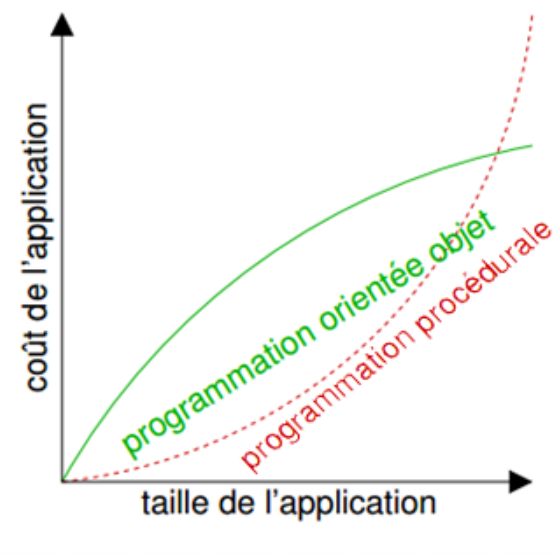
Code source orienté objet

« Un objet est une capsule logicielle oblative avec un tropisme conatif dont l'hétéronomie est la marque de la durée, de l'éphémère et de la hoirie »

— Serge Miranda

Pourquoi la POO ?

- **Diminuer le coût** d'un logiciel
- Faciliter la **conception** et **l'évolution** du code
- **Augmenter la durée de vie** d'un logiciel
- Augmenter la **réutilisabilité** d'un logiciel
- Augmenter la **facilité de maintenance** d'un logiciel



Conception d'un logiciel à la manière de la fabrication d'une voiture

Principe de la POO

- Communication inter-objets par messages
- À la réception d'un message :
  - Appel d'une méthode qui modifie son état
  - Appel d'une méthode qui envoie un message à son tour

POO

- **Paradigme** (≠ méthodologie)
- Concept :
  - Objet :
    - **État** (attributs)
    - **Comportement** (méthodes)
    - Identité
  - **Encapsulation**
  - **Héritage**
  - **Polymorphisme**
- Langage de programmation :
  - **Java**, C++, Ada, PHP, C#, Objective C, Python, etc.

Objet et classe

- **Objet** : un concept, une idée/entité du monde physique
  - Exemples : voiture, étudiant, chat, fenêtre, forme, etc.
- **Classe** : regroupe les objets de mêmes comportements
- **Instancier** : fabriquer un exemplaire d'un élément à partir d'un modèle qui lui sert en quelque sorte de moule
- Un objet est une instance de **classe**
- **Réification** : permet de transformer ou à transposer un concept en une entité informatique

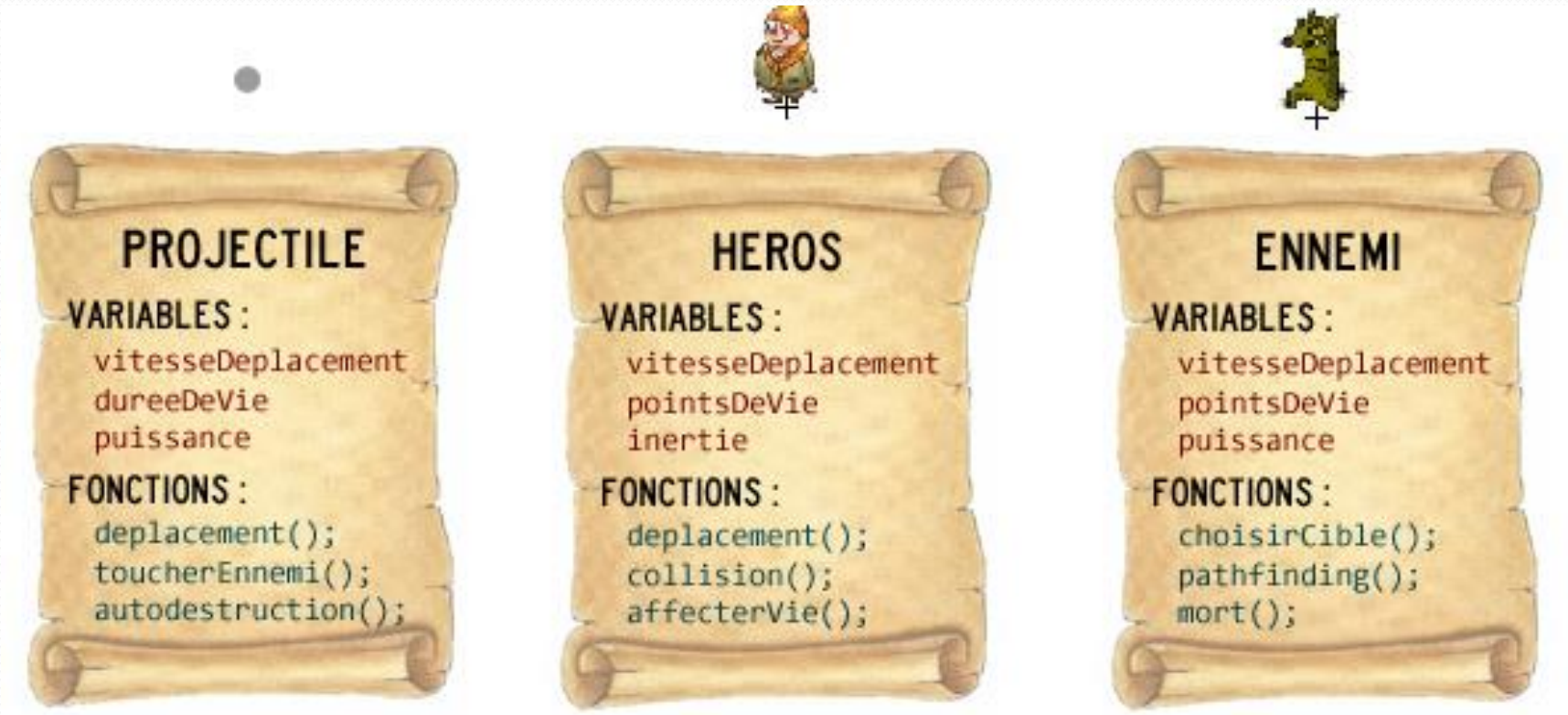
Une classe représente une responsabilité

État et comportement

- État :
  - Défini par l'ensemble des attributs
  - **Attribut**, variable d'instance, donnée membre :
    - Variable spécifique à l'objet
- Comportement :
  - Défini par l'ensemble de méthodes
  - **Méthode** : fonctions spécifique à l'objet
  - Méthodes :
    - **Constructeur** : appelé à la création de l'objet
    - **Destructeur** : appelé à la destruction d'un objet
    - Méthode abstraite : méthode sans code
    - Accesseurs et mutateurs : sert de mandataire d'accès à l'état de l'objet depuis l'extérieur de celui-ci

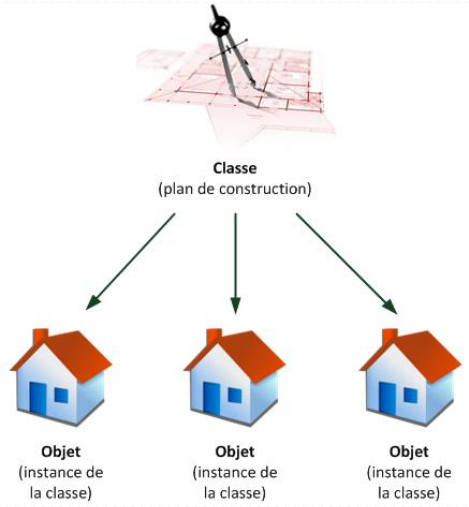
POO et magie

POO et magie



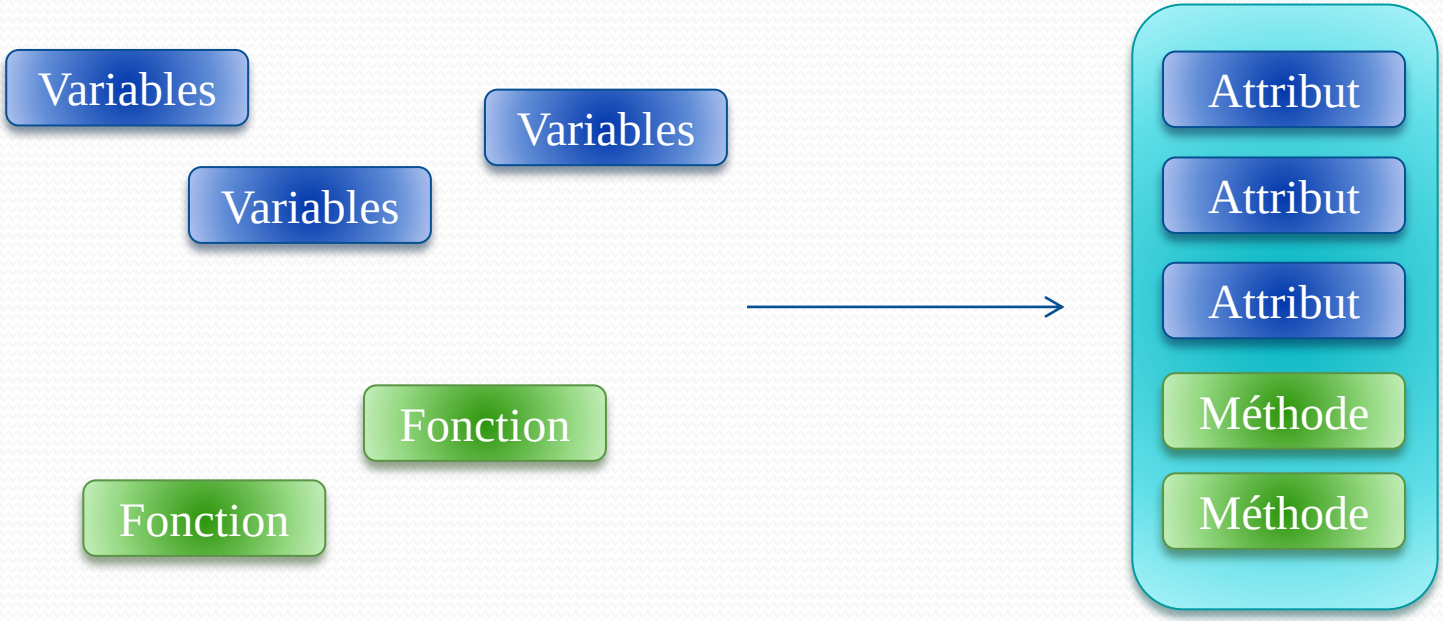
POO : deux aspects

- Programme en cours d'**écriture** :
  - Ensemble de **classes**
  - Chaque classe a des **attributs** et des **méthodes**
- Programme en cours d'**exécution** (processus) :
  - Ensemble d'**objets**
  - Chaque **objet** a son **état** courant et un **comportement**



Encapsulation

- Association de variables et fonctions dans une même entité
- L'objet est vu de l'extérieur comme une boîte noire ayant certaines propriétés et ayant un comportement spécifié

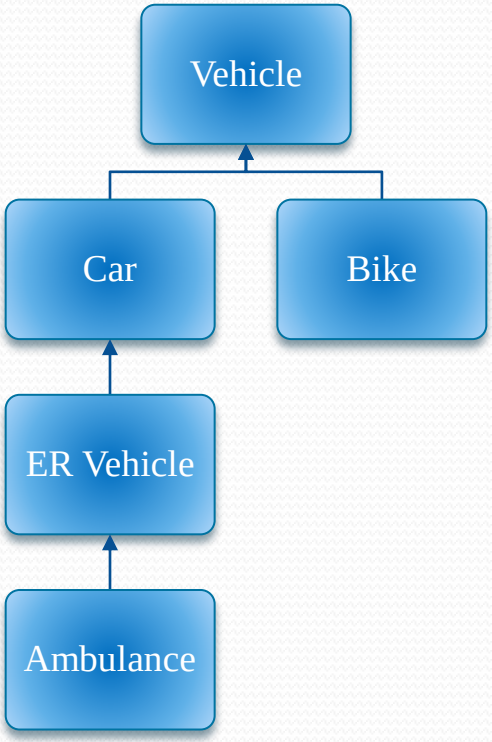


Encapsulation : accessibilité

- Concerne : classe, constructeur, attribut et méthode
- Visibilité :
  - **Publique** : accessible de partout et sans aucune restriction
    Protégée : accessible aux classes du module (ou paquetage) et à ses classes filles
  - **Privée** : accessible uniquement au sein de sa classe
- Accesseurs (*getters*) / mutateurs (*setters*) :
  - Permet de récupérer/modifier la valeur d'un attribut avec une visibilité restreinte grâce à une méthode

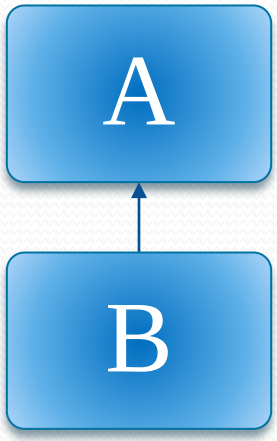
Héritage

- Construction d'une classe à partir d'autres classes en partageant leurs attributs et méthodes
- **Spécialisation** (ou généralisation), **enrichissement**
- **Réutilisation**
- **Redéfinition**



Classes et sous-classes

- B **hérite** de A
- A est la **classe mère**, B la **classe fille**
- A est la **super-classe** de B
- B est une **sous-classe** de A
- Un objet de type B **est** aussi un objet de type A
- Un objet de type A **n'est pas** un objet de type B



Notions générales

- **Opérateurs**
- Données :
  - **Variables** ≠ **attributs**
  - Constantes
  - **Valeurs littérales** (`"hello!"`, `3`, `2.75`, `true`)
  - Expressions (et expressions « parenthésées »)
- **Affectation** ≠ **condition**
- **Instructions**
- **Fonction** ≠ procédure ≠ **méthode** (signature)
- **Paramètres** (formels) ≠ **arguments** (paramètres effectifs)
- **Exécution séquentielle**

Aller plus loin

- Prototype
- Mixin
- Trait

Crédits

Auteur

Mickaël Martin Nevot
mmartin.nevot@gmail.com

Carte de visite électronique

Relecteurs

Cours en ligne sur : www.mickael-martin-nevot.com