Programmation Orientée Objet (C++) : Synthèse des concepts de l'orienté objets

Jamila Sam

Laboratoire d'Intelligence Artificielle
Faculté I&C

EPFL

Objectifs de la semaine

Nous voici arrivés au terme de l'introduction des concepts de bases de l'orienté-objets.

Pour une étude de cas récapitulative de ces concepts :

https ://www.coursera.org/learn/programmation-orientee-objet-cpp/home/week/7

- ☞ Semaine 7

Jamila Sam
& Jean-Cédric Chappelier

Etat des lieux

Vous avez abordé jusqu'ici :

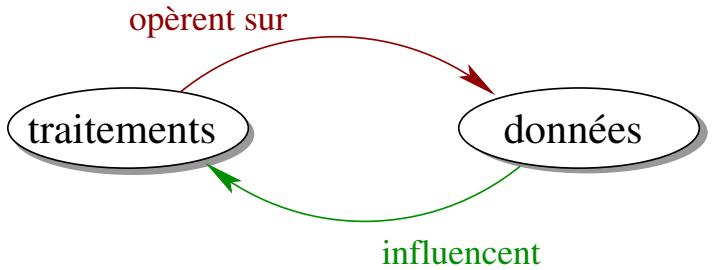
1. les bases de la programmation procédurale ;
2. les bases de la programmation orientée objets.

☞ Il nous reste à aborder quelques thèmes d'approfondissement : structures de données et «templates» ainsi qu'un survol de la librairie standard

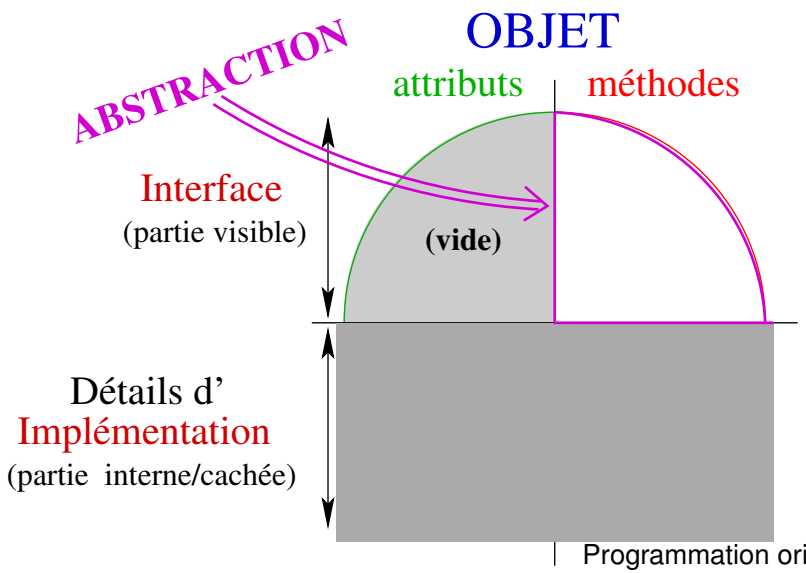
Jamila Sam
& Jean-Cédric Chappelier

Qu'avons nous vu en programmation ?

programmer c'est décomposer une **tâche** à automatiser en une séquence d'instructions (**traitements**) et des **données**



en programmation orientée objets, on regroupe dans le même objet **les traitements et les données** qui lui sont spécifiques (principe d'**encapsulation**)



Qu'avons nous vu en programmation ?

programmer c'est décomposer une **tâche** à automatiser en une séquence d'instructions (**traitements**) et des **données**

Algorithme	S.D.A.
Traitements	Données
Expressions & Opérateurs Structures de contrôle Fonctions	Variables Portée Chaînes de caractères Tableaux statiques Tableaux dynamiques Structures Pointeurs
Entrées/Sorties	

Jamila Sam
& Jean-Cédric Chappelier

Qu'avons nous vu en programmation ?

en programmation orientée objets, on regroupe dans le même objet **les traitements et les données** qui lui sont spécifiques (principe d'**encapsulation**)

Objet	
Encapsulation et Abstraction Classes Héritage simple/multiple Polymorphisme Classes abstraites/virtuelles Résolution des collisions de noms	
Traitements	Données
Méthodes Constructeurs & Destructeurs Const Virtuelles (pures) Surcharge d'opérateurs(interne/externe)	Attributs Appels aux constructeurs des attributs (hérités) Statiques
Privés/protégés/publiques Hérités/cachés (: :)	

FONDAMENTAUX

1. **déclarez avant d'utiliser**
   - variables
     ```
     int i;
     vector<double> v;
     ```
   - fonctions ☞ prototype
     ```
     double sin(double x);
     bool cherche_valeur(Listechainee l, Valeur v);
     ```
   - classes ☞ Attributs et prototypes des méthodes
2. **modularisez** / décomposez / pensez « atomique » et « **objet** »
   - 2.1 conception (qu'est ce qu'on veut ?)
   - 2.2 implémentation (comment ça se réalise ?)
   - 2.3 syntaxe (comment ça s'écrit ?)
   - 2.4 tests (où sont mes fautes, comment pourrais-je les tester ?)

« fondamentaux » de la POO

1. **encapsulation** Objet = attributs + méthodes

```
class Rectangle {
public:
  double surface() { ... };
  ...
private:
  double hauteur;
  double largeur;
}
```

   Attributs et méthodes publiques ☞ Interface de la classe (abstraction)

2. **héritage**

```
class RectangleColore : public Rectangle {
   Couleur couleur;
   //...};
```

3. **polymorphisme** le choix du type se fait à l'exécution, en fonction de la nature réelles des instances (typage dynamique)
   Ingrédients : Pointeurs/Références + méthodes virtuelles