Introduction à la programmation orientée objet

Contenu

- Introduction
- La notion d’objet
- La notion de classe
- Les objets et les classes en C++
- Classes vs Structures en C++

Introduction

La programmation orientée objet, c'est quoi?

- De nombreux langages de programmation (dont le C++) supportent la programmation orientée objet (POO)
- Il s'agit d'un paradigme (approche) de programmation qui vient avec un vocabulaire et des concepts qui lui sont propres

Introduction

- **Les deux autres grands paradigmes de programmation qui existent aujourd'hui sont:**
  - La **programmation structurée**
  - La **programmation fonctionnelle**

Introduction

- **Jusqu'à maintenant, nous avons vu les concepts permettant d'exprimer la logique de programmation**
  - **Séquence** (les instructions s'exécutent une à la suite de l'autre)
  - **Sélection** (structures conditionnelles)
  - **Itération** (boucles)

Introduction

- **Les ajouts que la POO apportent à la programmation structurée concernent moins la logique du code que son organisation (architecture)**
  - Visent une plus grande modularité et réutilisabilité du code
- **Les paradigmes ne sont pas propres à des langages**
  - Par exemple, en C++ on peut faire autant de la programmation orientée objet que de la programmation structurée
  - Ces deux paradigmes sont disponibles dans de nombreux autres langages

La notion d’objet

Les objets

- **Un objet est une entité possédant**
  - des données
  - des opérations possibles, qui utilisent ces données et peuvent les modifier
- **Représente souvent une entité du monde réel, ex:**
  - Un livre
  - Un étudiant
  - Une voiture
- **ou un concept propre au programme, ex:**
  - Un bouton
  - Un formulaire
  - Un paquet TCP/IP

Les propriétés

- **Les attributs ou propriétés d'un objet désignent les données qui le décrivent**
- **Ex: propriétés d'une voiture**
  - Marque
  - Modèle
  - Année
  - Couleur
  - Kilométrage

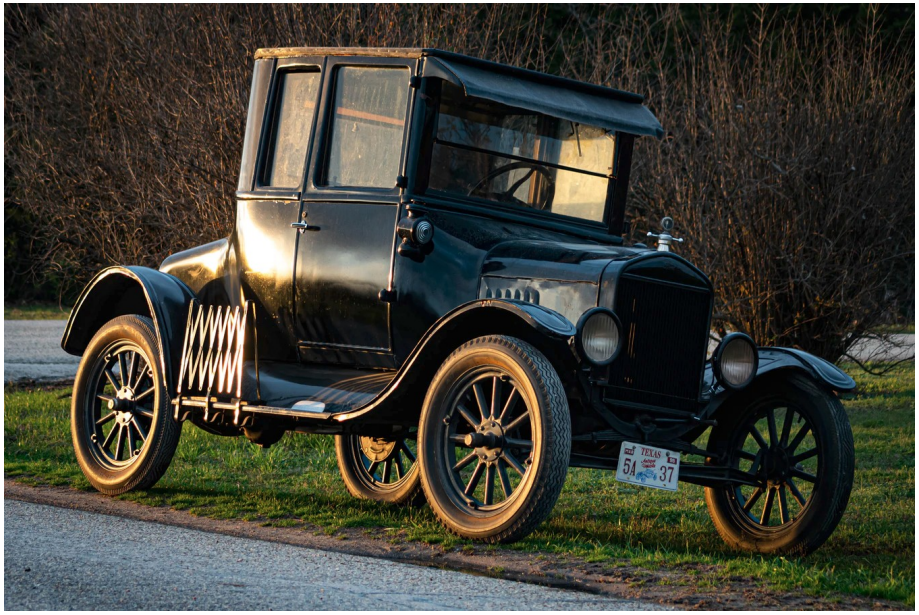


Photo par Andrew Lancaster sur Unsplash

Les méthodes

- **Les méthodes désignent les opérations qu'un objet peut effectuer**
  - Celles-ci ont parfois un impact sur les attributs de l'objet
- **Ex: méthodes d'une voiture**
  - Avancer
  - Reculer
  - Tourner
  - Freiner

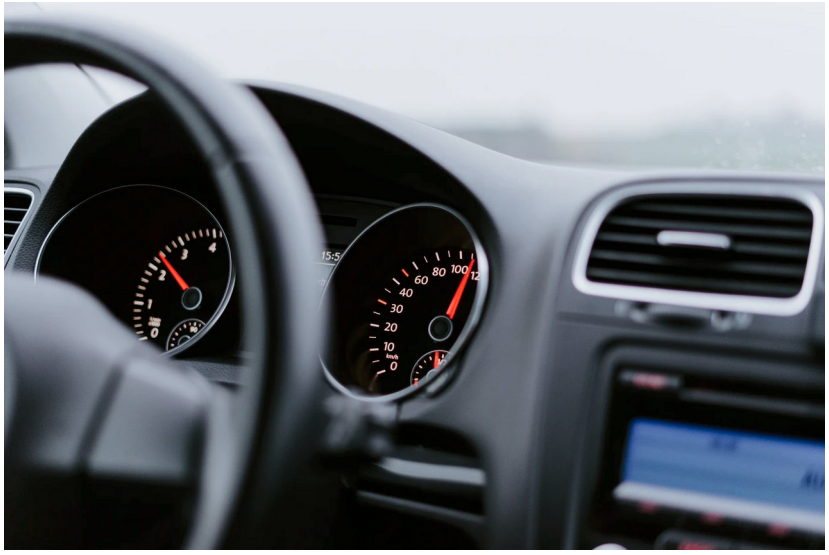


Photo par Julian Hochgesang sur Unsplash

Les méthodes

- En termes de programmation, une méthode est une fonction rattachée à un objet et ayant accès aux attributs de celui-ci

Les membres

- **Les attributs et les méthodes d'un objet sont collectivement appelés ses membres.**

La notion de classe

Les classes

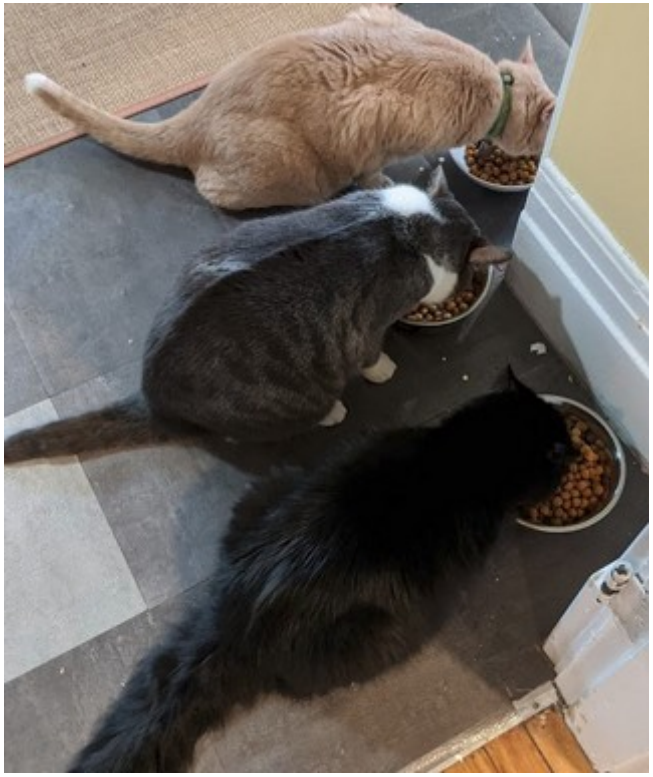
- **Une classe est le type d'un objet**
  - Définit la liste des propriétés communes à tous les objets de ce type
  - Définit le code des méthodes de ces objets
- **Les structures en C++ sont une version primitive des classes**

Les classes

- **On dit d'un objet appartenant à une classe qu'il est une instance de cette classe**
- **Ex:**
  - **La Hyundai Elantra de Pier-Luc** est un **objet** qui est une **instance** de la **classe** **Voiture**
  - **Robert De Niro** est une instance de la classe **Acteur**
  - ***Clean Code*** est une instance de la classe **Livre**
  - Vous êtes tous et toutes des instances de la classe **Étudiant**

Exemple

Voici trois instances de la classe Chat:



Analogie des gâteaux

- **Une analogie courante est de comparer les objets à des gâteaux, et les classes à des moules**

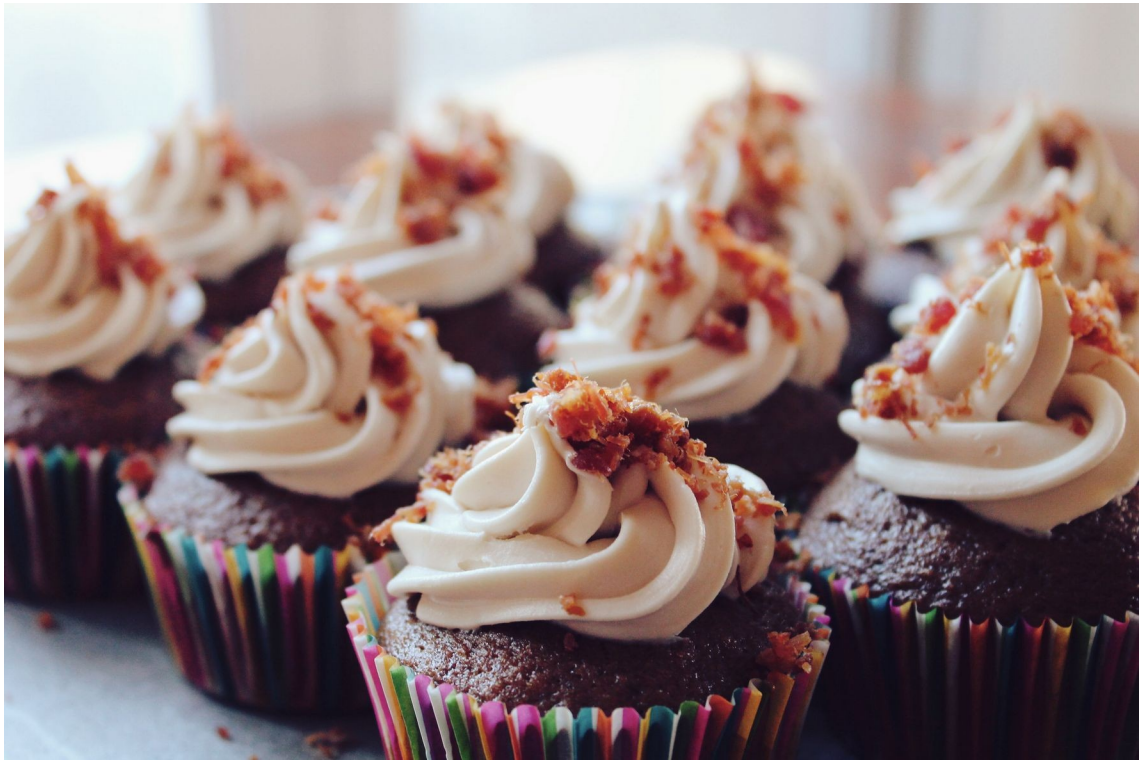


Photo par Brian Chan sur Unsplash

Les objets et les classes en C++

Déclaration d'une classe

Fichier livre.h

```
#pragma once

class Livre
{
private:
    std::string _titre;
    std::string _auteur;
    float _prix;
public:
    void initialiser(std::string titre, std::string auteur, float prix);
    void afficherTitre();
    void afficherPrix();
};
```

Définition des méthodes

Fichier livre.cpp

```
#include <iostream>
#include <iomanip>
#include "livre.h"

using namespace std;

void Livre::initialiser(string titre, string auteur, float prix) {
    _titre = titre;
    _auteur = auteur;
    _prix = prix;
}

void Livre::afficherTitre() {
    cout << _titre;
}

void Livre::afficherPrix() {
    cout << _prix << " $";
}
```

Instanciation et utilisation

Fichier main.cpp

```cpp
#include <iostream>
#include "livre.h"

using namespace std;

int main() {
    Livre harryPotter1;

    setlocale(LC_ALL, "french");

    harryPotter1.initialiser("Harry Potter à l'école des sorciers",
                             "J.K. Rowling", 16.95);

    cout << "Titre: ";
    harryPotter1.afficherTitre();

    cout << endl << "Prix: ";
    harryPotter1.afficherPrix();
}
```

Résultat

Affichage généré par l'exécution du programme:

```
Titre: Harry Potter à l'école des sorciers
Prix: 16.95 $
```

Spécificateurs d'accès

Fichier livre.h

```
class Livre
{
private:
    std::string _titre;
    std::string _auteur;
    float _prix;
public:
    void initialiser(std::string titre, std::string auteur, float
prix);
    void afficherTitre();
    void afficherPrix();
};
```

- **private:**
  - Membres privés, on NE PEUT PAS y accéder de l'extérieur de la classe
- **public:**
  - Membre publics, on PEUT y accéder de l'extérieur de la classe

Spécificateurs d'accès

Fichier livre.cpp

```
void Livre::initialiser(string titre, string auteur, float prix) {
    _titre = titre;
    _auteur = auteur;
    _prix = prix;
}

void Livre::afficherTitre() {
    cout << _titre;
}
```

Le code des méthodes de la classe Livre a accès aux attributs privés de la classe

Encapsulation

- **Principe d'encapsulation:**
  - Les attributs sont privés
  - Les méthodes sont publiques
- **L'utilisateur n'a pas à connaître l'implémentation de la classe**
  - Pas besoin de s'y connaître en mécanique pour savoir qu'appuyer sur l'accélérateur fait avancer la voiture!
- **Garantit l'intégrité des données contenues dans l'objet**

Constructeurs et destructeur

- **À la création d'un objet, le compilateur appelle automatiquement le constructeur de la classe**
  - Méthode spéciale qui initialise les attributs de l'objet
- **À la destruction de l'objet, le compilateur appelle le destructeur de la classe**
  - Sert à libérer la mémoire utilisée par l'objet (utile dans le cas d'allocation dynamique)

Constructeurs

- **Un constructeur est une méthode qui porte le même nom que la classe**
- **N'a pas de type de retour**
- **Peut prendre ou non des paramètres**
  - Exemple d'appel de constructeur avec un paramètre:
    - **ifstream fichier("etudiants.txt");**
- **Une classe peut avoir plusieurs constructeurs**

Exemple

Fichier livre.h

```
class Livre
{
private:
    std::string _titre;
    std::string _auteur;
    float _prix;
public:
    Livre(std::string titre, std::string auteur, float prix);

    void afficherTitre();
    void afficherPrix();
};
```

Exemple

Fichier livre.cpp

```
#include <iostream>
#include <iomanip>
#include "livre.h"

using namespace std;

Livre::Livre(string titre, string auteur, float prix) {
    _titre = titre;
    _auteur = auteur;
    _prix = prix;
}
```

Exemple

Fichier main.cpp

```
int main() {
    Livre harryPotter1(
        "Harry Potter à l'école des sorciers", "J.K. Rowling", 16.95
    );
    Livre harryPotter2(
        "Harry Potter et la chambre des secrets", "J.K. Rowling", 17.95
    );

    setlocale(LC_ALL, "french");

    cout << "Titre Harry Potter 1: ";
    harryPotter1.afficherTitre();

    cout << endl << "Titre Harry Potter 2: ";
    harryPotter2.afficherTitre();
}
```

Constructeur sans paramètre

- **L'inconvénient d'avoir ajouté un constructeur avec paramètres, c'est qu'on ne peut plus déclarer un Livre sans passer de paramètre**
  - Cette ligne de code provoque maintenant une erreur: **Livre monLivre;**
- **Solution: déclarer un autre constructeur, qui ne prend pas de paramètre**
  - En profiter pour initialiser les attributs à des valeurs par défaut

Exemple

Fichier livre.h

```
class Livre
{
private:
    std::string _titre;
    std::string _auteur;
    float _prix;
public:
    Livre();
    Livre(std::string titre, std::string auteur, float prix);

    void afficherTitre();
    void afficherPrix();
};
```

Exemple

Fichier livre.cpp

```
#include <iostream>
#include <iomanip>
#include "livre.h"

using namespace std;

Livre::Livre() {
    _titre = "(Titre inconnu)";
    _auteur = "(Auteur inconnu)";
    _prix = 0;
}

Livre::Livre(string titre, string auteur, float prix) {
    _titre = titre;
    _auteur = auteur;
    _prix = prix;
}
```

Exemple

Fichier main.cpp

```
int main() {
    Livre harryPotter1(
        "Harry Potter à l'école des sorciers", "J.K. Rowling", 16.95
    );
    Livre harryPotter2(
        "Harry Potter et la chambre des secrets", "J.K. Rowling", 17.95
    );
    Livre livreInconnu;

    setlocale(LC_ALL, "french");

    cout << "Titre Harry Potter 1: ";
    harryPotter1.afficherTitre();

    cout << endl << "Titre Harry Potter 2: ";
    harryPotter2.afficherTitre();
}
```

Destructeur

- **Le destructeur de Livre est une méthode sans type de retour dont le nom est ~Livre**
- **Si la mémoire utilisée par l'objet était allouée dynamiquement, on devrait obligatoirement la nettoyer dans le destructeur**
  - Afin de prendre de bonnes habitudes, nous effacerons toujours les valeurs de nos attributs dans nos destructeurs

Exemple

Fichier livre.h

```
class Livre
{
private:
    std::string _titre;
    std::string _auteur;
    float _prix;
public:
    Livre();
    Livre(std::string titre, std::string auteur, float prix);
    ~Livre();

    void afficherTitre();
    void afficherPrix();
};
```

Exemple

Fichier livre.cpp

```
Livre::~Livre() {
    _titre = "";
    _auteur = "";
    _prix = 0;
}
```

Classes vs Structures en C++

Classes vs Structures en C++

- **La principale différence entre une structure et une classe est la suivante :**
  - Les membres d'une **classe** sont **privés** par défaut.
  - Les membres d'une **struct** sont **publics** par défaut.

Classes vs Structures en C++

- **Si vos objets doivent avoir des méthodes, alors vous voulez de l'encapsulation**
  - Vous devriez donc opter pour une classe
- **Les structures devraient être réservées pour des objets très simples qui ne contiennent que quelques propriétés et qui ne font pas d'opérations sur celles-ci**

Fin de la présentation

Des questions?