<C++>

Constructeurs et attributs

Dorian.H Mekni

19 | APRIL | 2020

class | def

- Une classe peut representer une entité ou un concept
- Cette dernière peut posséder des capacités
- Dans le cas d'un cyborg, cela peut être le nom, la taille, le métier, le code robo-génétique, la mission, les armes, le mode opératoire etc

method | Rappel

Une méthode [method] est une fonction dans une classe.

attributs | Rappel

Les attributs sont des variables qui appartiennent à une classe

private

**private est un accessible privée. Ce qui veut dire que les attributs
ainsi que les méthodes qui y seront logés ne seront accessibles et utilisables que par la classe**

#include <string>

Bibliothéque qui permet la création de chaîne de caractères

C++

Human() | std::string name | std::string name, int age

Dans le cas illustré ici, le constructeur std::string name renseigne la classe juste pas son nom [name] Toute autre valeur comme l'age, on rajoute une valeur par défaut.

Le constructeur complet est le constructeur qui va initialiser toutes les propriétés de la classe.

C++

Human() | std::string name | std::string name, int age

```hpp
#ifndef class_hpp
#define class_hpp
#include<string>

class Human
 {

    public:
     Human(); // Il est préférable de garder un constructeur par defaut.
     Human(std::string name); // Contructeur qui renseigne la classe juste par son nom
     Human(std::string name, int age); // Constructeur complet
     ~Human();

    private:
     std::string a_name;// code protocolaire de prefixe pour nommer les attributs.
     int a_age;

 };


#endif /* class_hpp */
```

C++

Initialisation

Voilà une autre écriture au moment de l'initialisation d'une méthode.

```cpp
#include "class.hpp"
#include <iostream>
#include<string>

                                        /*
                            Autre maniére d'implémenter la méthode
                                        */

Human::Human() : a_name("Not yet defined"),a_age(1)
{
   // Le constructeur par défaut
}



Human::Human(std::string name) : a_name(name), a_age(3)
{
  // Le constructeur dont le [name] initialise la classe
}

Human::Human(std::string name, int age) : a_name(name), a_age(age)
{
  // Le constructeur complet qui initialise toutes les prop de la classe
}



                                    /*
                            Déconstructeur par défaut donc nullement
                            besoin d'initialiser cette méthode ici.
                                    */
```

C++

Constructeur de copie | &

Constructeur de copie qui permet de cloner un objet pour en créer un autre.

```hpp
#ifndef class_hpp
#define class_hpp
#include<string>

class Human
 {

    public:
     Human(); // Il est préférable de garder un constructeur par defaut.
     Human(std::string name); // Contructeur qui renseigne la classe juste par son nom
     Human(std::string name, int age); // Constructeur complet
     Human(const Human &other);// Contructeur de copie -> const car il n'y aura pas de
         modification

    private:
     std::string a_name;// code protocolaire de prefixe pour nommer les attributs.
     int a_age;

 };


#endif /* class_hpp */

```

C++