Algorithmique

Laurent Debize

TIIS2

Programmation Orientée Objet

Définition

- Un objet est quelque chose de **concret** de la vraie vie.
- Il possède des caractéristiques, qu'on appelle **attributs**
- Il peux aussi réaliser des actions, qu'on appelle **méthodes**

Exemple

- Objet : Animal
- Attributs : couleur et poids
- Méthodes : manger et seDeplacer

Les classes

Pour créer des objets, nous avons besoin d'un plan de fabrication : ce plan de fabrication s'appelle la **classe**.
Dans une classe, on définira :

- Les attributs
- Les méthodes

Exemple

Les classes

Définition d'une classe :

```
class Animal
{

}
```

L'encapsulation

L'encapsulation permet de **protéger** les attributs et méthodes de votre classe, pour qu'ils ne soient pas accessibles aux autres développeurs.
Il existe trois types de mot-clé pour définir la **visibilité** d'un attribut ou méthode :

- **private** : seul le code de votre classe peut voir et accéder à cet attribut ou méthode.
- **public** : toutes les autres classes peuvent voir et accéder à cet attribut ou méthode.
- **protected** : seul le code de votre classe et de ses sous-classes peut voir et accéder à cet attribut ou méthode.

L'encapsulation

Exemple

```
class Animal
{
private:
    string couleur = "gris";
    float poids = 10;

public:
    void ajouterUnKilo()
    {
        poids = poids + 1;
    }
};
```

Utilisation d'une classe

- Définir la classe avant le `main`
- **Instancier**, c-à-d créer un objet, de la classe Animal

Exemple

```
class Animal
{
private:
    string couleur = "gris";
    float poids = 10;

public:
    void ajouterUnKilo()
    {
        poids = poids + 1;
    }
};

int main() {
    Animal chien;
    return 0;
}
```

Accesseurs et mutateurs

D'après le principe de l'encapsulation :

- Tous les attributs sont *privés*
- On crée des méthodes *publiques* pour lire (**accesseurs**) ou écrire (**mutateurs**) dans ces attributs

Par convention, ces méthodes portent le nom de l'attribut, préfixé de `get` pour les accesseurs ou de `set` pour les mutateurs.

Accesseurs et mutateurs

Exemple

```
class Animal
{
private:
    string couleur = "gris";
    float poids = 10;

public:
   float getPoids()
    {
        return poids;
    }

    void setPoids(float nouveauPoids)
    {
        poids = nouveauPoids;
    }
};
```

Accesseurs et mutateurs

Utilisation

```
int main() {
    Animal chien;

    cout << "poids du chien : " << chien.getPoids() << " kg" << endl;
    chien.ajouterUnKilo();
    cout << "poids du chien : " << chien.getPoids() << " kg" << endl;
    chien.setPoids(15);
    cout << "poids du chien : " << chien.getPoids() << " kg" << endl;

    return 0;
}
```

Exercice

Créer une classe *Personne* contenant :

- Les attributs *nom*, *prénom*, *age*.
- Les accesseurs et mutateurs
- La fonction *vieillir()* qui incrémente l'âge
- La fonction *presentations()* qui affiche le nom et le prénom et l'age de la personne.
- Ajouter une autre fonction de votre imagination...

Testez cet objet en instanciant quelques personnes, vérifier que toutes les fonctions sont opérationnelles.

Vous m'enverrez votre travail par email en respectant les conditions suivantes :

- Nommer votre fichier de la façon suivante **NOM_Prénom.cpp**
- Objet : **TIIS2**
- Envoyez à l'heure : chaque jour de retard sera pénalisé de 2 points.