Encapsulation

# Encapsulation

L'écriture de classes offre d'autres avantages que le simple regroupement de données et de traitements. Parmi ceux-ci figure la possibilité de restreindre l'accès à certains éléments de la classe. C'est ce que l'on appelle **l'encapsulation**.

**DEFINITION** : **l'encapsulation** est l'un des principes fondamentaux de la POO. Il consiste à restreindre l'accès à certains éléments d'une classe (le plus souvent ses attributs). L'objectif de l'encapsulation est de ne laisser accessible que le strict nécessaire pour que la classe soit utilisable.

## Avantages

L'encapsulation offre de nombreux avantages :

* Diminution des risques de mauvaise manipulation d'une classe.
* Création de classes "boîtes noires" par masquage des détails internes.

## Définition

Les mots-clés public et private permettent de modifier le **niveau d'encapsulation** (on parle aussi de **visibilité** ou **d'accessibilité**) des éléments de la classe (attributs et méthodes):

* + Un élément **public** est librement utilisable depuis le reste du programme.
  + Un élément **privé** est uniquement utilisable depuis les méthodes de la classe elle- même
  + **Par défaut** aucun modificateur : visibilité depuis toutes les classes du même package

## Exemple d'utilisation

On souhaite qu'un compte bancaire créé ne puisse pas changer de solde ni de devise. Cela est possible en définissant les attributs solde et devise comme étant **privés**.

**public** **class** CompteBancaire

{

**private** **double** solde;

**private** String devise;

public String adresse

**public** CompteBancaire(**double** SOLDE, String devise, String ADRESSE)

{

solde = SOLDE;

devise = DEVISE;

adresse=ADRESSE;

}

}

**public** **class** TestCompteBancaire {

**public** **static** **void** main(String[] args)

{

CompteBancaire comptePierre = **new** CompteBancaire(1000, "dollar","5552 jarry 7h7 7j5 qc", );

comptePierre.adresse="9854 rue jean talon 1g4 2g2 qc"; //OK

comptePierre.solde=500; // Erreur

comptePierre.devise="dollars"; // Erreur

}

}

# Accesseurs

## Définition

**DEFINITION** : un **accesseur** est une méthode le plus souvent *publique* qui permet d'accéder à un attribut *privé*.

* + un accesseur en lecture (***getter***) permet de **lire** la valeur d'un attribut.
  + un accesseur en écriture (***setter***) permet de **modifier** la valeur d'un attribut.

**Une bonne pratique de la programmation**

* restreindre l'accès direct aux données avec le spécificateur private
* fournir des méthodes accesseurs / mutateurs (méthodes "get/set" ) qui lisent et changent la donnée