POO

Programmation Orientée Objet

Christian Lisangola

Objectifs de la formation

Découvrir le concept de programmation orientée objet

Table des matières

S’initier aux concepts de base de la POO.

- Programmation procédurale vs programmation orientée objet.
- Découvrir la classe en programmation.
- Connaître les objets en programmation.
- Faire le point sur l'abstraction.
- Approcher la notion d'encapsulation.
- Savoir ce qu'est l'héritage.
- Comprendre le polymorphisme.

I.

S'initier aux concepts de base de la conception orientée objet.

Dans ce chapitre, découvrons les bases indispensables de la programmation orientée objet.

Avantages et inconvénients

1.

Procédural vs POO

Avantages et inconvénients

Programmation procédurale

Consiste à écrire son code en suivant une procédure logique, en suivant de façon séquentielle une suite de fonctions qui sont appelées dans le code avec parfois des fonctions qui s'appellent entre elles.

Ces fonctions sont appelées dans le code source en leur passant ou non des paramètres. Ces fonctions renvoies parfois des résultats exploitables dans la suite de la procédure.

Cette façon de coder est plus abordable aux débutants et demande moins d'analyse avant le développement.

Cependant le code source sera souvent plus long, difficilement réutilisable et plus difficile à maintenir.

Cela dit, certains langages comme le C sont prévus pour écrire du code en procédural.

Programmation orientée objet

Cette façon de coder s'approche un peu plus proche du raisonnement humain. Dans cette approche on crée des objets qu'on appel des classes.

Chaque classe va avoir ses propres attributs et ses propres méthodes.

L'héritage et le polymorphisme vont nous permettre de factoriser certaines opérations entre les différents modèles de données similaires.

De plus nous allons pouvoir aller du général vers le particulier en créant des classes parents et des classes enfants.

La POO donne un code plus lisible car il est constitué de petites entités qui vont interagir.

Enfin les classes enfants héritent des modifications apportées au classes mères ce qui limite les erreurs et oublis plus fréquents en procédural.

En conclusion

Procédural

- A base de fonctions qui ne contiennent pas les données.
- Code difficilement réutilisable tel quel.
- Chaque fonction ou procédure est associée à un traitement particulier.
- Chaque fonction peut être utilisée à différents emplacements du programme.
- Plus simple à conceptualiser.

POO

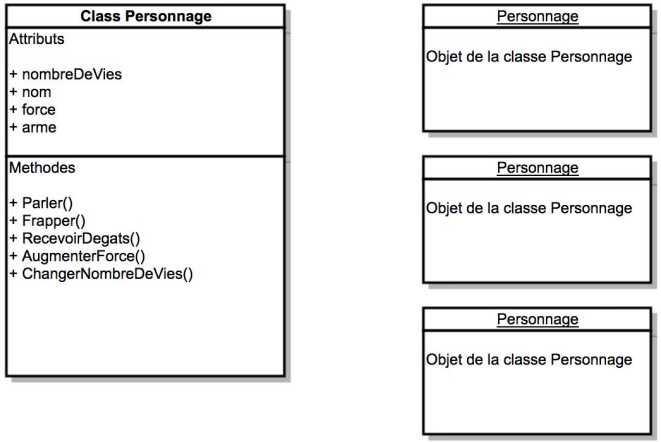
- A base d'objets qui contiennent les données
- Code réutilisable.
- Code scindé en entités autonomes (classes)
- Les données sont protégées par encapsulation.

2.

La notion de Classes

Qu’est ce qu’une classe

Une classe est un plan qui nous permet de créer plusieurs objets à partir de ce plan. Exemple : Plusieurs personnages d’un jeux vidéo. On parle alors d’instances de la classe Personnage.



Comment créer une classe

- Nom : c’est quoi ?

  Employé, compte bancaire, joueur, document, album…
- Attributs : ce qui la décrit.

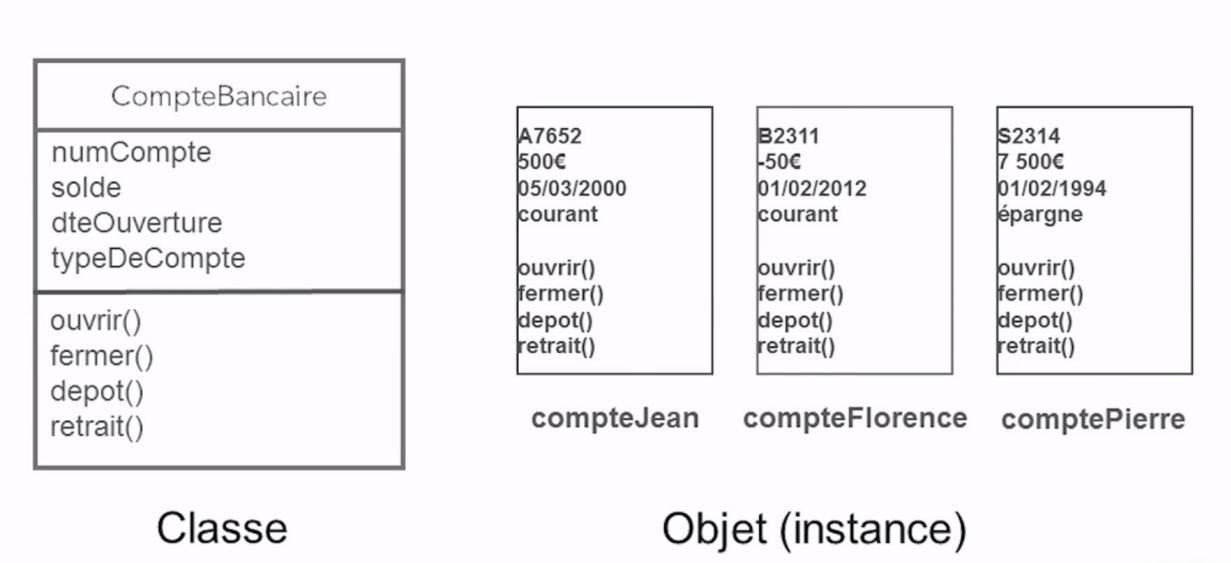
  Largeur, hauteur, couleur, type de fichier, score…

  On les appel aussi des propriétés.
- Comportements : que peut elle faire ?

  Jouer, ouvrir, chercher, enregistrer, imprimer…

  On les appel le plus souvent des méthodes.

Classe / Objet



Les class native du langage

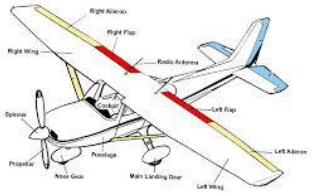
- La plupart des langages aujourd'hui proposent des classes pré-conçues prêtes à l'emplois.
- On peut utiliser ces classes et créer les nôtres.
- String(), Date(), Array()....

3.

Connaître les objets en programmation.

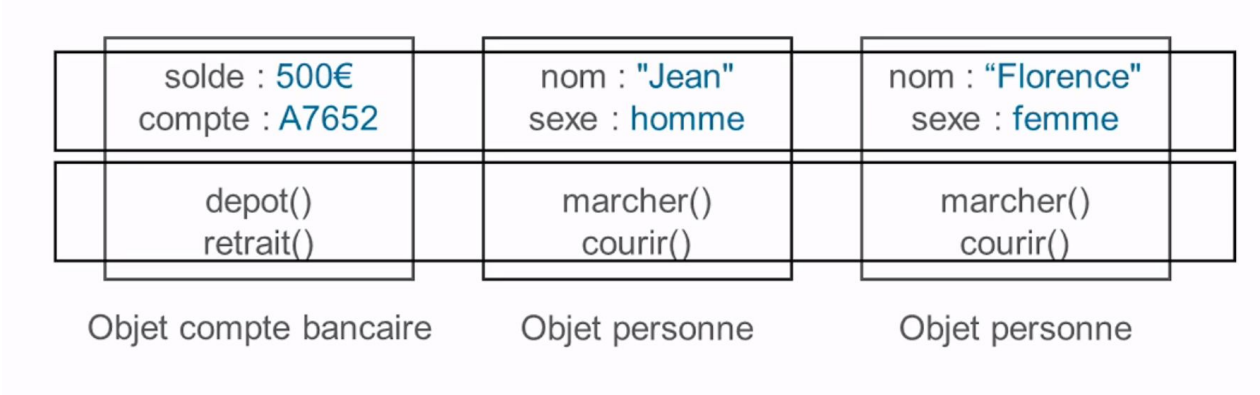
Qu’est ce qu’un Objet

- Chaque objet à une vie qui lui est propre (si je casse une tasse, l’autre reste intacte).
- Chaque objet peut avoir des propriétés différentes (une tasse peut être vide et l’autre à moitié pleine, une pomme peut avoir un nombre de feuilles différent ou une couleur différente).
- Chaque objet à ses propres comportements (un avion vole alors qu’un téléphone sonne)



Exemple d’objet informatique

- Un compte en banque est un objet informatique.
- Chaque objet est indépendant des autres.
- Chaque objet dispose de ses propres caractéristiques.
- Les objets peuvent interagir entre eux.



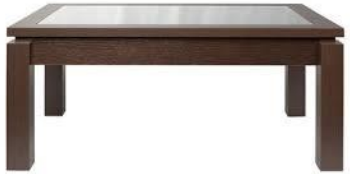
4.

Faire le point sur l’abstraction.

4 notions fondamentales en POO

- L’Abstraction.
- Encapsulation.
- Héritage.
- Polymorphisme.

La notion d’abstraction fait référence à un objet que l’on visualise sans pour autant que celui-ci soit parfaitement décrit. Exemple : une table est un objet que l’on visualise, pourtant il reste abstrait car l’on ne connaît pas ses propriétés. Le principe de l’abstraction en programmation est de se focaliser sur l’essentiel en ignorant tous les détails.



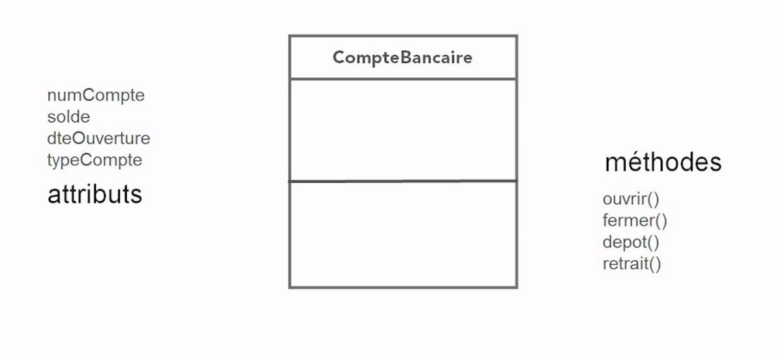
5.

L’encapsulation.

4 notions fondamentales en POO

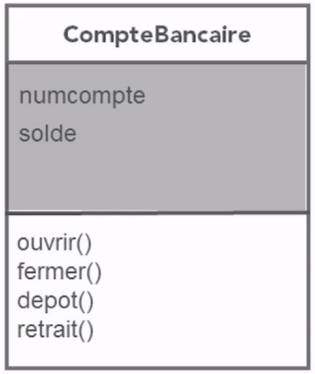
- L'Abstraction.
- Encapsulation.
- Héritage.
- Polymorphisme.

Consiste en premier lieu à encapsuler nos méthodes et nos attributs au sein d'une même unité : la classe.



Respect de l'encapsulation

- Un objet ne doit révéler de lui-même que le stricte nécessaire aux autres parties du programme. Exemple du compte bancaire : le solde n'est accessible que via une méthode de la classe. Sans utiliser cette méthode, on ne peut pas accéder au solde.
- On peut modifier les données d'une classe sans pour autant modifier les données de tous les objets.
- Les attributs sont privés.



6.

L’héritage.

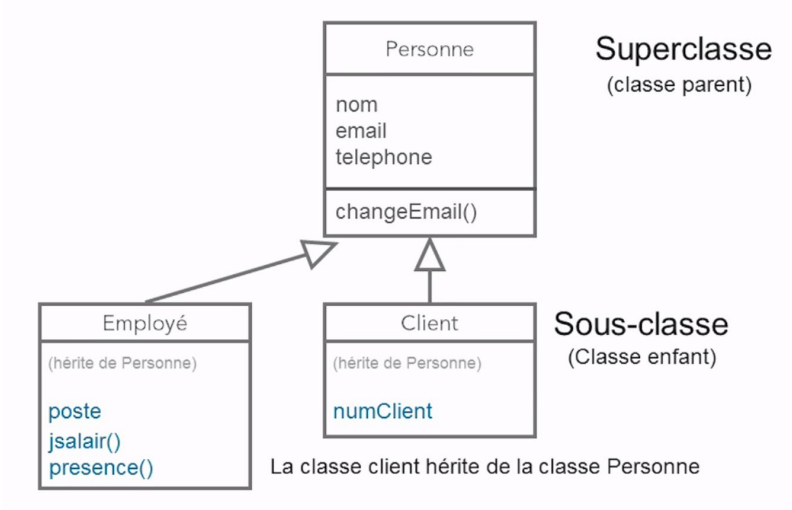
4 notions fondamentales en POO

- L'Abstraction.
- Encapsulation.
- Héritage.
- Polymorphisme.

L'héritage est une manière très pratique de réutiliser du code.

Les classes enfants héritent des attributs et méthodes de la classe parent.

Si on modifie la classe parent, toutes les classes enfants bénéficient de ces modifications.



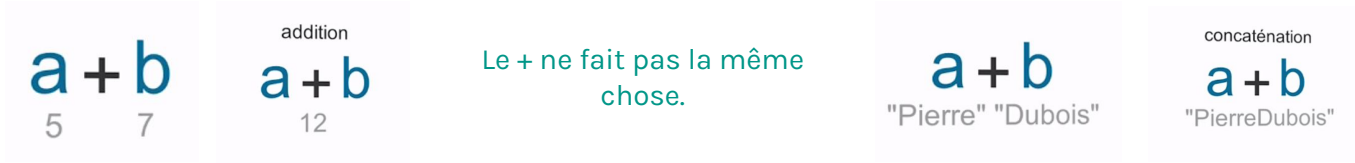
7.

Le polymorphisme.

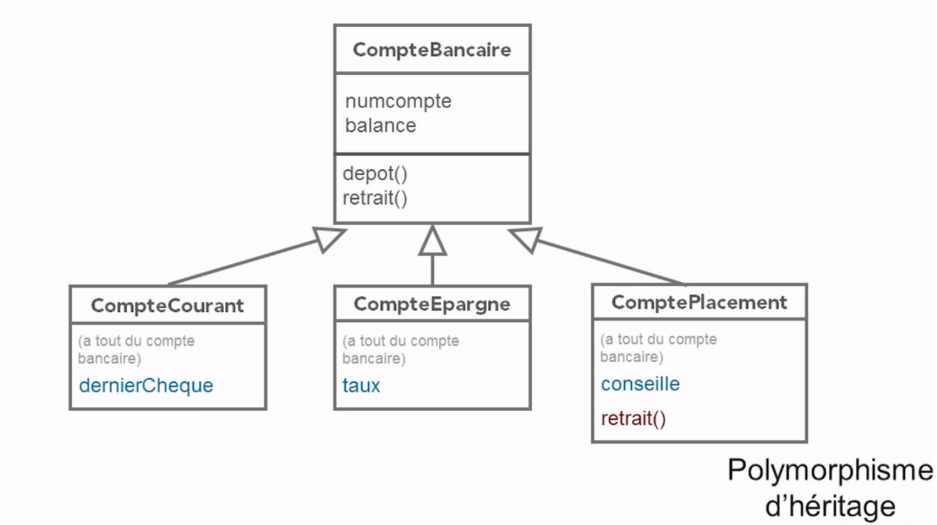
4 notions fondamentales en POO

- L'Abstraction.
- Encapsulation.
- Héritage.
- Polymorphisme.

Polymorphisme : qui peut prendre plusieurs formes.



Cas concret de Polymorphisme



- La méthode **retrait()** de la classe **ComptePlacement** se comporte différemment.
- Le polymorphisme est rarement utilisé.