Concept du paradigme de la programmation orientée objet

Dans ce chapitre, nous allons essayer de décrire ce qu'est la programmation orientée objet (POO)

En fait, on peut dire que la POO est une façon de développer une application qui consiste à représenter (**modéliser**) une application informatique sous la forme d'objets, ayant des propriétés et pouvant interagir entre eux. La modélisation orientée objet est proche de la réalité ce qui fait qu'il sera relativement facile de modéliser une application de cette façon. De plus, les personnes non-techniques pourront comprendre et éventuellement participer à cette modélisation.

Cette façon de modéliser les choses permet également de découper une grosse application, généralement floue, en une multitude d'objets interagissant entre eux. Cela permet de découper un gros problème en plus petits afin de le résoudre plus facilement.

Un autre avantage de la POO est la réutilisabilité. Des objets peuvent être réutilisés ou même étendus grâce à l'héritage.

Il faut savoir que la POO, c'est beaucoup plus que ça et nous en verrons des subtilités plus loin, mais comprendre ce qu'est un objet est globalement suffisant pour une grande partie du tutoriel.

Qu'est-ce qu'un objet ?

Alors qu'est-ce qu'un objet ?

Si on prend le monde réel, nous sommes entourés d'objets : une chaise, une table, une voiture, etc. Ces objets forment un tout.

- Ils possèdent des propriétés (la chaise possède 4 pieds, elle est de couleur bleue, etc.).
- Ces objets peuvent faire des actions (la voiture peut rouler, klaxonner, etc.).
- Ils peuvent également interagir entre eux (l'objet conducteur démarre la voiture, l'objet voiture fait tourner l'objet volant, etc.).

On peut donc avoir plusieurs objets chaises : on parle également d'instances. Les objets chaises, ce sont bien celles concrètes que l'on voit devant nous autour de l'objet table pour démarrer une partie de carte.

Sachant qu'un objet en programmation c'est comme un objet du monde réel mais ce n'est pas forcément restreint au matériel. Un chien est un objet. Des concepts comme l'amour ou une idée sont également des objets, tandis qu'on ne dirait pas cela dans le monde réel.

En conclusion:

- La définition (ou structure) d'un objet est un concept abstrait, comme une définition dans le dictionnaire. Cette définition décrit les caractéristiques d'un objet (la chaise a des pieds, l'homme a des jambes, etc.). Cette définition est unique comme une définition dans le dictionnaire.
- Un objet ou une instance est la réalisation concrète de la structure de l'objet. On peut avoir de multiples instances, comme les 100 voitures sur le parking devant

chez moi. Elles peuvent avoir des caractéristiques différentes (une voiture bleue, une voiture électrique, une voiture à 5 portes, etc.)

L'encapsulation

Le fait de concevoir une application comme un système d'objets interagissant entre eux apporte une certaine souplesse et une forte abstraction.

Prenons un exemple tout simple : la machine à café du bureau. Nous insérons nos pièces dans le monnayeur, choisissons la boisson et nous nous retrouvons avec un gobelet de la boisson commandée. Nous nous moquons complètement de savoir comment cela fonctionne à l'intérieur et nous pouvons complètement ignorer si le café est en poudre, en grain, comment l'eau est ajoutée, chauffée, comment le sucre est distribué, etc.
Tout ce qui nous importe c'est que le fait de mettre des sous dans la machine nous permet d'obtenir un café qui va nous permettre d'attaquer la journée.

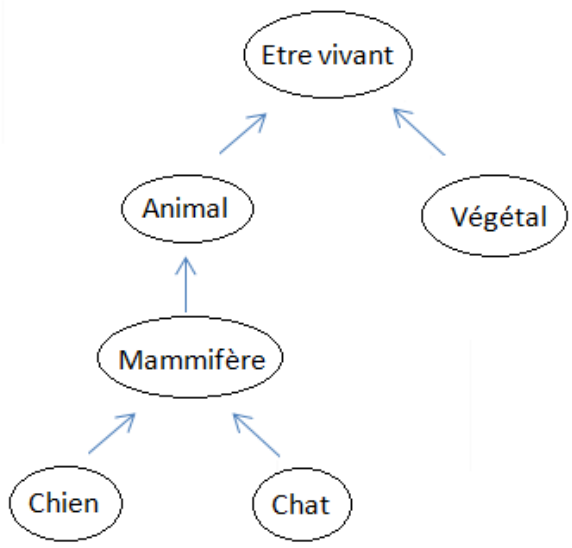
C'est ce qu'on appelle l'**encapsulation**. Cela permet de protéger l'information contenue dans notre objet et de le rendre manipulable uniquement par ses actions ou propriétés. Ainsi, l'utilisateur ne peut pas accéder au café ni au sucre ou encore moins à la monnaie. Notre objet est ainsi protégé et fonctionne un peu comme une boite noire. L'intérêt est que si la personne qui entretient la machine met du café en grain à la place du café soluble, c'est invisible pour l'utilisateur qui se soucie simplement de mettre ses pièces dans la machine.

Héritage

C'est-à-dire qu'un objet dit « père » peut transmettre certaines de ses caractéristiques à un autre objet dit « fils ».

Pour cela, on pourra définir une relation d'héritage entre eux. S'il y a une relation d'héritage entre un objet père et un objet fils, alors **l'objet fils hérite de l'objet père**. On dit également que l'objet fils est une **spécialisation** de l'objet père ou qu'**il dérive de l'objet père**.

Exemple:



Soit l'objet « chien » et imaginons ses caractéristiques tirées du monde réel en utilisant l'héritage :

- L'objet « chien » est une sorte d'objet « mammifère »
- L'objet « mammifère » est une sorte d'objet « animal »
- L'objet « animal » est une sorte d'objet « être vivant »

Pour bien comprendre cet héritage de comportement, empruntons à nouveau les exemples du monde réel.

- L'être vivant peut par exemple faire l'action « vivre ».
- Le mammifère possède des yeux.
- Le chien, qui est une sorte d'être vivant et une sorte de mammifère, peut également faire l'action « vivre » et aura des yeux.
- Le chat qui est une autre sorte d'être vivant peut lui aussi faire l'action « vivre » et aura également des yeux.

Polymorphisme - Substitution

Polymorphisme

Le mot **polymorphisme** suggère qu'une chose peut prendre plusieurs formes.

En fait, on peut dire qu'une manifestation du polymorphisme est la capacité pour un objet de faire une même action avec différents types d'intervenants. C'est ce qu'on appelle le polymorphisme « ad hoc » ou le polymorphisme « paramétré ». Par exemple, notre objet voiture peut rouler sur la route, rouler sur l'autoroute, rouler sur la terre si elle est équipée de pneus adéquats, rouler au fond de l'eau si elle est amphibie, etc. …
Concrètement ici, je fais interagir un objet « voiture » avec un objet « autoroute » ou un objet « terre »… par l'action qui consiste à « rouler ».

Substitution

La **substitution** est une autre manifestation du polymorphisme. Il s'agit de la capacité d'un objet fils à redéfinir des caractéristiques ou des actions d'un objet père.

Prenons par exemple un objet mammifère qui sait faire l'action « se déplacer ». Les objets qui dérivent du mammifère peuvent potentiellement avoir à se déplacer d'une manière différente. Par exemple, l'objet homme va se déplacer sur ses deux jambes et donc différemment de l'objet dauphin qui se déplacera grâce à ses nageoires ou bien encore différemment de l'objet « homme accidenté » qui va avoir besoin de béquilles pour s'aider dans son déplacement.
Tous ces mammifères sont capables de se déplacer, mais chacun va le faire d'une manière différente. Ceci est donc possible grâce à la substitution qui permet de redéfinir un comportement hérité. Ainsi, chaque fils sera libre de réécrire son propre comportement, si celui de son père ne lui convient pas.

Interfaces

L'interface est un contrat que s'engage à respecter un objet. Il indique en général un comportement.

Prenons un exemple dans notre monde réel et connu : les prises de courant. Elles fournissent de l'électricité à 220V avec deux trous et (souvent) une prise de terre. Peu importe ce qu'il y a derrière, du courant alternatif de la centrale du coin, un transformateur, quelqu'un qui pédale,... nous saurons à coup sûr que nous pouvons brancher nos appareils électriques car ces prises s'engagent à nous fournir du courant alternatif avec le branchement adéquat.
Elles respectent le contrat ; elles sont « branchables ». Ce dernier terme est un peu barbare et peut faire mal aux oreilles.

Conclusion

- L'approche orientée objet permet de modéliser son application sous la forme d'interactions entre objets.
- Les objets ont des propriétés et peuvent faire des actions.
- Ils masquent la complexité d'une implémentation grâce à l'encapsulation.
- Les objets peuvent hériter de fonctionnalités d'autres objets s'il y a une relation d'héritage entre eux.