La "programmation objet" expliquée en 45 minutes.

La **programmation objet** c'est :
- une idée
- trois concepts

C'est très simple et très puissant !

L'idée c'est de modéliser un objet et des particularités de cet objet dans le code d'un programme.
L'exemple type d'objet est une **fenêtre** avec des fenêtres particulières telles que des boutons, des champs textes, des labels, des cases à cocher, des barres de défilement, etc.
Un deuxième exemple d'objet est une **figure géométrique** avec comme figures particulières, le quadrilatère, le trapèze, le parallélogramme, le losange, le carré, etc.
Un troisième exemple d'objet est **un vertébré** avec comme vertébrés particuliers, les mammifères, les oiseaux, les reptiles, les poissons, etc.

Un objet est caractérisé par :
° **ses propriétés** :
Sa couleur, sa position, sa forme, son nom, etc.
Cela correspond à des *variables*.

° **ses méthodes** :
L'objet peut être déplacé, sa couleur peut être modifiée, son nom peut être affiché, etc.
Cela correspond à des *fonctions*.

Les trois concepts.

1) L'encapsulation
Toutes les propriétés et les méthodes sont regroupées dans une structure, dans le but de bien isoler l'objet du reste du programme.
Cela fait un peu d'ordre, mais il n'y a rien de révolutionnaire là-dedans.

2) L'héritage
À partir d'un objet, un objet plus particulier peut être défini. C'est un objet descendant de l'objet parent, qui hérite de toutes ses propriétés et de toutes ses méthodes, auxquelles de nouvelles propriétés et de nouvelles méthodes peuvent être ajoutées.
Exemples :
° À partir d'une fenêtre, on peut définir une fenêtre particulière qui est un bouton. Le bouton a une méthode particulière de réagir à un clic, il a une couleur particulière et un nom à lui.
° À partir d'une figure géométrique, on peut défini un quadrilatère, qui a la particularité d'avoir 4 côtés, puis un autre descendant peut être un trapèze, qui une paire de côtés parallèle, etc.
° À partir d'un vertébré, un mammifère peu être défini, qui a la particularité d'avoir des poils et d'allaiter. Un chien peut ensuite être un objet descendant d'un mammifère, avec ses propriétés particulières, puis un labrador, etc.
C'est déjà un concept intéressant, qui permet de récupérer du code pour l'utiliser dans la description d'un objet plus spécifique.

3) Le polymorphisme
Un enfant hérite des méthodes de son parent (souvent dans les langages de programmations, il ne peut y avoir qu'un seul parent). Le polymorphisme est la possibilité pour un enfant de redéfinir la méthode héritée de son parent. Il peut utiliser la méthode de son parent, puis lui ajouter des fonctionnalités. Par exemple, si le parent possède une méthode qui affiche son nom, l'enfant peut afficher le nom, puis afficher d'autres caractéristiques qui lui sont propre.
C'est le polymorphisme qui donne toute sa puissance à la programmation objet !

Un exemple

L'exemple type est la fenêtre avec tous ses descendants. On trouve de tels exemples sur le Web.
Ici, on développera en langage Javascript l'exemple des vertébrés, avec les descendances qui sont des vertébrés particuliers.

Il faut encore distinguer la **classe** de tous les vertébrés, d'une **instance** de cette classe, qui est un **objet**, c'est-à-dire un vertébré en particulier, vous par exemple, ou votre chien ou votre chat.

La structure que l'on pourrait illustrer est la suivante :
Seul ce qui est en gras a été implémenté dans l'exemple : page355_prog_objet.html
c.f. : http://www.juggling.ch/zgisin/a2019_oc3/page355_prog_objet.html

- **classe des vertébrés**
    - **classe des mammifères**
        - **classe des chiens**
            - classe des labradors
            - classe des caniches
        - **classe des chats**
    - classe des poissons
        - classe des poissons rouges
    - classe des oiseaux
        - classe des rouge-gorges
        - classe des corbeaux

L'extension complète peut devenir énorme. L'exemple sert juste à comprendre l'idée.

Exemple de programme similaire écrit en langage Python :
http://www.juggling.ch/gisin/python/python_code.html#ex_XXX01a_Classe_vertebres

Deux derniers détails.
1) Toutes classes a un **constructeur**, pour créer et initialiser une instance de cette classe, c'est-à-dire pour créer un objet ayant les caractéristiques de cette classe.
Souvent la classe a aussi un destructeur, pour libérer de la mémoire lorsque l'objet est détruit.

2) Tout objet a une propriété particulière qui est une référence à lui-même.
Cette propriété peut s'appeler : "*this*" ou "***self***".
Elle est nécessaire lorsque l'objet veut s'envoyer lui-même en paramètre à une fonction ou une méthode d'un autre objet.
Elle est aussi nécessaire pour distinguer le nom d'une variable ou d'un paramètre du nom d'une propriété de l'objet.