**La Programmation Orientée Objet**

**Définition :** dans le langage informatique, la Programmation Orientée Objet (POO) consiste en la mise en relation d’objets, c’est-à-dire d’éléments de programmation, avec un langage spécifique qui permet aux objets de communiquer entre eux.

Avant les langages de POO, il y avait d’abord les langages de programmation linéaire. Chaque ligne de code est lue et exécutée dans l’ordre, du début à la fin du script. Il y a la possibilité de faire des sauts ou encore des boucles, mais le principe reste fondamentalement le même. Dans ce type de langage, il est impossible de réutiliser son code. Cette approche peut, du coup, difficilement s’appliquer à de gros programmes complexes.

Ensuite sont apparus les langages de programmation modulaire. Grâce à ces derniers, on peut éviter la redondance et on peut donc réutiliser son code via les fonctions ou les procédures. Chaque tâche exécutée par un programme représente un nombre variable d’instructions réunies en « bibliothèques » afin de pouvoir segmenter le code et favoriser sa réutilisation.

Les langages de POO sont, en quelque sorte, une évolution des langages de programmation modulaire et leur apportent quatre aspects primordiaux :

* **l’encapsulation :** permet de réunir des variables et des fonctions au sein d’une même entité appelée une classe. Les variables sont des données membres (ou encore attributs de classe), les fonctions, des méthodes. L’accès aux attributs et aux méthodes peut être réglementé.
* **l’héritage :** permet de définir une hiérarchie de classes. Chaque classe « enfant » hérite des attributs et des méthodes de son/ses parent(s). En pratique, la classe de base est générique, et plus on descend dans la hiérarchie, plus cette classe est spécialisée.
* **le polymorphisme :** vient du grec et signifie « qui peut prendre plusieurs formes ». Cette caractéristique permet de définir plusieurs fonctions de même nom avec des paramètres différents. La bonne fonction sera choisie en fonction de ses paramètres lors de l’appel.
* **l’abstraction :** son objectif principal est de gérer la complexité en masquant les détails inutiles à l’utilisateur. Cela permet à l’utilisateur d’implémenter une logique plus complexe sans comprendre, ni même avoir à penser à toute la complexité cachée.