## 1. Introduction à la Programmation Orientée Objet en PHP

La Programmation Orientée Objet (POO) est un paradigme de programmation qui permet de modéliser des entités du monde réel sous forme d'objets, organisant ainsi le code de manière plus structurée et réutilisable. En PHP, la POO repose sur la création de classes, qui définissent des propriétés (attributs) et des méthodes (comportements), et sur l'instanciation de ces classes pour créer des objets. Les classes servent de modèles à partir desquels on peut créer plusieurs objets, chacun ayant ses propres données tout en partageant les mêmes comportements définis par la classe.

## 2. Encapsulation et Modificateurs d'Accès

L'encapsulation est un principe clé de la POO, qui consiste à regrouper les données et les méthodes qui manipulent ces données dans une même unité, la classe. Ce mécanisme permet de protéger les données internes de la classe en restreignant leur accès depuis l'extérieur. En PHP, cela est contrôlé par l'utilisation des modificateurs d'accès :

- **Public** : Les attributs et méthodes déclarés comme public sont accessibles de n'importe où, que ce soit à l'intérieur ou à l'extérieur de la classe.
- **Private** : Les attributs et méthodes déclarés comme private sont accessibles uniquement à l'intérieur de la classe qui les définit. Aucun autre objet ou classe ne peut y accéder directement.
- **Protected** : Les attributs et méthodes déclarés comme protected sont accessibles uniquement à l'intérieur de la classe et dans les classes qui en héritent.

Cela permet de gérer l'accès aux données et de contrôler comment elles peuvent être modifiées ou utilisées.

## 3. Héritage et Polymorphisme

L'héritage est un concept fondamental de la POO qui permet à une classe d'hériter des propriétés et des comportements d'une autre classe, appelée classe parente. Cela permet de réutiliser du code et de créer des classes plus spécialisées à partir de classes génériques. Par exemple, une classe "Voiture" peut être une classe parente, et une classe "VoitureDeCourse" peut hériter de ses propriétés et méthodes tout en ajoutant des caractéristiques spécifiques.

Le polymorphisme, quant à lui, permet à une même méthode de se comporter différemment en fonction de l'objet qui l'appelle. Cela est généralement réalisé par la redéfinition (overriding) des méthodes héritées, permettant à des objets de classes

dérivées d'avoir un comportement unique pour des méthodes identiques définies dans la classe parente.

En résumé, l'héritage et le polymorphisme facilitent la réutilisation du code et permettent de créer des systèmes plus flexibles et modulaires, tout en respectant le principe de substitution des objets, où un objet d'une classe dérivée peut être utilisé à la place d'un objet de la classe parente sans altérer le comportement du programme.