**INTERFACE**

Une interface en PHP est un type spécial de classe qui définit un contrat pour les classes qui l'implémentent. Elle **spécifie une liste de méthodes** **que ces classes doivent obligatoirement implémenter, sans en fournir le code**.   
  
Les interfaces servent principalement à :  
  
1. Assurer une certaine ***structure commune*** entre différentes classes, en forçant l'implémentation de méthodes spécifiques. Cela **facilite le polymorphisme** *(permet à des classes différentes d être traitées de façon similaire)* **et le couplage faible** entre les classes *(peu ou pas de dépendance entre les classe)*.   
  
2. Définir des **contrats de programmation** que les classes doivent respecter, améliorant ainsi la maintenabilité et la modularité du code.   
  
3. Permettre l'**héritage multiple**, puisqu'*une classe peut implémenter plusieurs interfaces* à la fois, contrairement à l'héritage classique limité à une seule classe parente.   
  
4. **Séparer les interfaces des implémentations,** ce qui permet de modifier les implémentations sans affecter le code utilisant ces interfaces.  
  
Les interfaces définissent un contrat de programmation que les classes doivent respecter, favorisant ainsi la réutilisation, la modularité et la maintenabilité du code orienté objet en PHP.

**LES DIFFERENCES AVEC CLASSE ABSTRACT**

une interface est différente d'une classe abstraite , même si elles partagent certaines similitudes :

* **Une interface ne peut contenir que des méthodes abstraites** (sans implémentation), tandis qu'une classe abstraite peut contenir à la fois des méthodes abstraites et des méthodes implémentées.
* **Toutes les méthodes d'une interface doivent être publiques**, alors que les méthodes d'une classe abstraite peuvent avoir différents niveaux de visibilité (public, protected, private).
* **Une classe peut implémenter plusieurs interfaces,** mais ne peut hériter que d'une seule classe abstraite.
* **Les interfaces ne peuvent pas avoir de propriétés**, contrairement aux classes abstraites qui le peuvent.
* **Une interface ne peut pas être instanciée**, elle définit uniquement un contrat que les classes qui l'implémentent doivent respecter. Une classe abstraite ne peut pas non plus être instanciée directement, mais **elle peut contenir une implémentation partielle**.
* Les interfaces favorisent la **composition et le couplage faible** entre les classes, tandis que l'héritage de classes abstraites **promeut une hiérarchie de classes plus rigide**.

**Les interfaces définissent un contrat purement abstrait que les classes doivent implémenter, tandis que les classes abstraites fournissent une implémentation partielle pouvant être étendue. Les interfaces encouragent la modularité et la réutilisabilité du code, alors que les classes abstraites facilitent l'héritage et le partage de fonctionnalités communes.**