Programmation orientée objet

Exercice 1

1. Créer une classe Gateau
2. Ajouter les attributs suivants et les initialiser dans le constructeur :
    - nom gâteau : str
    - temps cuisson : int
    - liste ingrédients : list de str
    - étapes recettes : list de str
    - nom du créateur : str
3. Ajouter une méthode qui affiche les ingrédients de la recette
4. Ajouter une méthode qui affiche les étapes de la recette
5. Instancier un objet gâteau qui affiche les ingrédients ainsi que les étapes de préparation du gâteau

Exercice 2

1. Créer une classe nommée CompteBancaire qui représente un compte bancaire, ayant pour attributs :
    - numeroCompte (type numérique)
    - nom (nom du propriétaire du compte du type chaîne)
    - solde
2. Créer un constructeur ayant comme paramètres : numero_compte, nom, solde
3. Créer une méthode Versement() qui gère les versements
4. Créer une méthode Retrait() qui gère les retraits
5. Créer une méthode Agios() permettant d'appliquer les agios à un pourcentage de 5 % du solde
6. Créer une méthode afficher() permettant d'afficher les détails sur le compte

Exercice 3

1. Créer une classe WaterTank qui possédera les attributs d'instance suivants :
    - Poids de la citerne à vide : float
    - Capacité maximale : float
    - Niveau de remplissage : float
2. Créer les méthodes suivantes propres à chaque instance de classe :
    - Méthode indiquant le poids total
    - Méthode pour remplir la citerne avec un nombre de litres d'eau
    - Méthode pour vider la citerne d'eau d'un nombre de litres d'eau

3. Créer un attribut de classe qui contiendra la totalité des volumes d'eau des citernes
4. Testez votre programme

Exercice 4

1. Créer une classe Personne, contenant le nom de la personne, son prénom, son numéro de téléphone et son email. Une méthode `__toString` pour afficher les données de la personne.
2. Créer une classe Travailleur, qui hérite de la classe Personne et étend avec les attributs nom d'entreprise, adresse entreprise et téléphone professionnel. Une méthode `__toString` pour afficher les données et qui réutilise celle de Personne.
3. Créer une classe Scientifique qui hérite de la classe Travailleur et étend avec les attributs de type list disciplines (physique, chimie, mathématique, ...) et types du scientifique (théorique, expérimental, informatique...) Une méthode `__toString` pour afficher les données et qui réutilise celle de Travailleur.

Exercice 5

1. Créer une interface Forme avec les signatures suivantes :
    - getPerimeter(): float
    - getArea(): float
2. Créer une classe Cercle qui a une propriété rayon et qui implémente Forme
3. Créer une classe Carré qui a une propriété côté et qui implémente Forme
4. Instancier un objet Cercle et Carré puis afficher leurs aires et périmètres

Exercice 6: Traits

1. Créez un trait nommé "Dumper" qui a une méthode dump() qui prend une variable comme argument et utilise la fonction var_dump() pour afficher les détails de la variable.
2. Ensuite, créez deux classes, User et Post, utilisez le trait Dumper dans ces classes, et essayez de dump() différentes variables.

Exercice 7 : Namespaces

1. Créez deux fichiers PHP, chacun avec une classe nommée User.
2. Mettez chaque classe dans un espace de noms différent (par exemple, Blog et Admin).
3. Essayez ensuite d'instancier les deux classes User dans un troisième fichier PHP.
4. N'oubliez pas d'utiliser la directive use pour importer les espaces de noms corrects.

Exercice 8 : Exceptions

1. Créez une classe UserManager avec une méthode createUser() qui prend un tableau comme argument.
    - Cette méthode doit vérifier si le tableau contient les clés 'username' et 'email'.
    - Si ce n'est pas le cas, elle doit lancer une InvalidArgumentException.
2. Dans votre script principal, essayez d'appeler createUser() avec un tableau incorrect, puis utilisez un bloc try / catch pour attraper l'exception et afficher un message d'erreur.

Exercice 9 : Tout ensemble

Essayez de combiner tout ce que vous avez appris. Créez des classes avec des espaces de noms, utilisez des traits dans ces classes et utilisez des exceptions pour gérer les erreurs.

1. Créez un trait Logger qui a une méthode pour écrire des messages dans un fichier journal.
2. Créez une classe UserManager dans l'espace de noms App\Users qui utilise le trait Logger.
3. Dans UserManager, créez une méthode createUser() qui lance une exception si les arguments sont incorrects.
4. Dans votre script principal, utilisez un bloc try / catch pour gérer ces exceptions et utilisez la méthode de logging pour enregistrer les détails des erreurs dans un fichier.