Créer une classe Voiture avec une propriété marque et afficher sa valeur.

Ajouter une méthode demarrer() qui affiche "La voiture démarre".

Créer plusieurs objets Voiture avec des marques différentes.

Créer une classe Chien avec une méthode aboyer().

Ajouter un constructeur à la classe Voiture qui prend en paramètre marque.

Ajouter une propriété privée vitesse et une méthode accelerer() qui augmente la vitesse.

Créer une méthode getVitesse() pour accéder à la vitesse (getter).

Créer une méthode setVitesse($v) avec une vérification (pas de valeur négative).

Créer une classe Rectangle avec largeur et hauteur, calculer l'aire.

Créer une classe Cercle avec un rayon et une méthode getSurface().

Créer une classe Personne avec nom, âge, et une méthode sePresenter().

Créer une classe Etudiant qui hérite de Personne et ajoute niveau.

Surcharger sePresenter() dans Etudiant.

Créer une classe Employe qui hérite de Personne et ajoute un salaire.

Ajouter un constructeur qui appelle parent::__construct().

Créer une classe Animal puis Chien et Chat qui héritent de Animal.

Ajouter une méthode parler() différente dans Chien et Chat.

Créer une classe CompteBancaire avec dépôt et retrait.

Créer une classe CompteEpargne qui hérite de CompteBancaire et ajoute un taux d'intérêt.

Implémenter une méthode calculerInteret() dans CompteEpargne.

Créer une classe abstraite Forme avec méthode abstraite aire().

Implémenter Rectangle et Cercle qui héritent de Forme.

Créer une interface Deplacable avec méthode deplacer($x, $y).

Implémenter Deplacable dans Voiture et Avion.

Créer une méthode polymorphe qui accepte un tableau de Forme et affiche leur aire.

Créer une interface AnimalInterface avec méthode parler().

Implémenter AnimalInterface dans Chien et Chat.

Créer une classe Zoo qui stocke plusieurs AnimalInterface et les fait "parler".

Créer une classe Vehicule abstraite avec méthode rouler().

Implémenter Voiture et Velo à partir de Vehicule.

Créer une classe MathUtils avec méthode statique factorielle($n).

Ajouter une constante PI dans Cercle.

Créer un compteur d'instances dans Personne avec une propriété statique.

Créer un Singleton simple (classe Database).

Créer un Trait Logger avec une méthode log($msg) et l'utiliser dans plusieurs classes.

Créer une exception personnalisée SoldeInsuffisantException dans CompteBancaire.

Gérer l'exception avec try/catch.

Créer une classe Panier avec des produits, lever une exception si le produit n'existe pas.

Créer une classe Utilisateur avec validation d'email (exception si invalide).

Créer une classe Fichier qui ouvre/écrit/lit un fichier avec gestion d'exceptions.

Implémenter le pattern Factory pour créer des objets Voiture selon la marque.

Implémenter le pattern Observer (un Journal notifie des Lecteurs).

Implémenter le pattern Strategy pour différents modes de paiement (Carte, PayPal).

Implémenter le pattern Decorator sur une classe Boisson avec Sucre, Lait.

Implémenter le pattern Adapter pour convertir une ancienne API vers une nouvelle.

Implémenter le pattern Composite pour gérer des fichiers et dossiers.

Implémenter le pattern Iterator sur une classe Collection.

Implémenter le pattern Dependency Injection (classe Service injectée dans Controleur).

Créer une classe ORM simple qui mappe une table SQL en objets PHP.

Implémenter un petit **mini framework MVC** (très basique).