Exercices : POO et classe

Exercice 1 : Création et Manipulation d'une Classe Personne

Objectif : Créer une classe `Personne` avec des propriétés publiques et privées, des méthodes, des getters et setters.

Instructions :

1. Créez une classe `Personne` avec les propriétés suivantes : - nom (publique) - #age (privée) - adresse (publique)

2. Ajoutez un constructeur pour initialiser les propriétés nom, #age, et adresse.

3. Ajoutez les méthodes suivantes à la classe `Personne` :

   - `afficherInfos()` : affiche les informations de la personne (nom, age, adresse).
   - Getter et setter pour `age` :
     - `get age()` : retourne l'âge.
     - `set age(nouvelAge)` : modifie l'âge.

4. Créez des instances de la classe `Personne`, testez les méthodes, getters, setters.

Exercice 2 : Classe CompteBancaire

Objectif : Créer une classe `CompteBancaire` pour gérer les informations et les opérations d'un compte bancaire.

Instructions : 1. Créez une classe `CompteBancaire` avec les propriétés suivantes : - titulaire (publique) - solde (privée)

2. Ajoutez un constructeur pour initialiser les propriétés titulaire et solde (défaut à 0).

3. Ajoutez les méthodes suivantes à la classe `CompteBancaire` :

   - `afficherSolde()` : affiche le solde actuel du compte.
   - `deposer(montant)` : ajoute le montant spécifié au solde.
   - `retirer(montant)` : retire le montant spécifié du solde si le solde est suffisant, sinon affiche un message d'erreur.

4. Ajoutez des getters et setters pour le solde (`solde`).

5. Créez des instances de la classe `CompteBancaire`, testez les méthodes, getters, setters.

Exercice 3 : Classe Produit

Objectif : Créer une classe `Produit` pour gérer les informations d'un produit avec des propriétés publiques et privées, des méthodes, des getters et setters, et une méthode statique pour calculer la TVA.

Instructions : 1. Créez une classe `Produit` avec les propriétés suivantes : - nom (publique) - prixHT (privée) - quantite (publique)

2. Ajoutez un constructeur pour initialiser les propriétés nom, prixHT (privée) , et quantite.

3. Ajoutez les méthodes suivantes à la classe `Produit` :

   - `afficherInfos()` : affiche les informations du produit (nom, prixHT, quantite).
   - Getter et setter pour `prixHT` :
     - `get prixHT()` : retourne le prix hors taxes.
     - `set prixHT(nouveauPrixHT)` : modifie le prix hors taxes.
   - `calculerPrixTTC()` : calcule et retourne le prix toutes taxes comprises (TTC) en ajoutant une TVA de 20% au prix hors taxes.

4. Ajoutez une méthode statique `calculerTVA` qui prend un montant HT et retourne le montant de la TVA (20%).

5. Créez des instances de la classe `Produit`, testez les méthodes, getters, setters, et la méthode statique.

Exercice 4 : Classe `Voiture`

Objectif : Créer une classe `Voiture` pour gérer les informations d'une voiture avec des propriétés publiques et privées, des méthodes, des getters et setters, et une méthode statique pour créer une voiture de base.

Instructions : 1. Créez une classe `Voiture` avec les propriétés suivantes : - marque (publique) - kilometrage (privée) - annee (publique)

2. Ajoutez un constructeur pour initialiser les propriétés marque, kilometrage, et annee.

3. Ajoutez les méthodes suivantes à la classe `Voiture` :

- `afficherInfos()` : affiche les informations de la voiture (marque, kilometrage, annee).
- Getter et setter pour `kilometrage` :
  - `get kilometrage()` : retourne le kilometrage.
  - `set kilometrage(nouveauKilometrage)` : modifie le kilometrage.
- `ajouterKilometres(km)` : ajoute les kilomètres spécifiés au kilométrage actuel.

4. Ajoutez une méthode statique `creerVoitureDeBase` qui crée et retourne une instance de `Voiture` avec les valeurs suivantes : marque = “Renault”, kilometrage = 0, annee = 2022.

5. Créez des instances de la classe `Voiture`, testez les méthodes, getters, setters, et la méthode statique.