**Travaux Dirigés N° 6**

**Exercice 1 : Classe Rectangle**

**Classe Rectangle**

* **Attributs :**
  + longueur : float
  + largeur : float
* **Méthodes :**
  + initialise(float longueur, float largeur) : initialise les dimensions du rectangle avec les valeurs fournies.
  + perimetre() : renvoie le périmètre du rectangle.
  + surface() : renvoie la surface du rectangle.

**Classe TestRectangle**

Écrire une classe TestRectangle qui :

1. Demande à l'utilisateur de saisir la longueur et la largeur d'un rectangle.
2. Calcule et affiche le périmètre et la surface du rectangle en utilisant les méthodes de la classe Rectangle.

**Exercice 2 : Classe Voiture**

**Classe Voiture**

* **Attributs :**
  + marque : String (marque de la voiture).
  + modele : String (modèle de la voiture).
  + vitesse : float (vitesse actuelle de la voiture, initialisée à 0).
* **Méthodes :**
  + initialise(String marque, String modele) : initialise la marque et le modèle de la voiture.
  + accelerer(float valeur) : augmente la vitesse de la voiture de la valeur donnée.
  + freiner(float valeur) : réduit la vitesse de la voiture de la valeur donnée, sans descendre en dessous de 0.
  + afficher() : affiche les informations de la voiture (marque, modèle, vitesse).

**Classe TestVoiture**

Écrire une classe TestVoiture qui :

1. Crée une voiture en spécifiant sa marque et son modèle.
2. Fait accélérer la voiture de 30 km/h, puis de 20 km/h.
3. Fait freiner la voiture de 25 km/h.
4. Affiche les informations de la voiture à chaque étape.

**Exercice 3 : Classe CompteBancaire**

**Classe CompteBancaire**

* **Attributs :**
  + titulaire : String (nom du titulaire du compte).
  + solde : float (solde du compte, initialisé à 0).
* **Méthodes :**
  + initialise(String titulaire, float soldeInitial) : initialise le titulaire et le solde initial du compte.
  + deposer(float montant) : ajoute le montant donné au solde.
  + retirer(float montant) : retire le montant donné du solde si le solde est suffisant, sinon affiche un message d’erreur.
  + afficherSolde() : affiche le solde actuel du compte.

**Classe TestCompte**

Écrire une classe TestCompte qui :

1. Crée un compte bancaire pour un utilisateur.
2. Effectue un dépôt de 1000 €.
3. Effectue un retrait de 500 €, puis un autre retrait de 700 €.
4. Affiche le solde après chaque opération.

**Exercice 4 : Classe Livre**

**Classe Livre**

* **Attributs :**
  + titre : String (titre du livre).
  + auteur : String (nom de l'auteur).
  + prix : float (prix du livre).
  + quantite : int (quantité disponible).
* **Méthodes :**
  + initialise(String titre, String auteur, float prix, int quantite) : initialise les attributs avec les valeurs fournies.
  + vendre(int nb) : réduit la quantité disponible du nombre d’exemplaires vendus, si suffisant, sinon affiche un message d'erreur.
  + approvisionner(int nb) : augmente la quantité disponible du nombre d'exemplaires ajoutés.
  + afficherDetails() : affiche les informations du livre (titre, auteur, prix, quantité).

**Classe TestLivre**

Écrire une classe TestLivre qui :

1. Crée un livre avec un titre, un auteur, un prix, et une quantité donnée.
2. Vend 3 exemplaires du livre.
3. Ajoute 5 exemplaires à la quantité disponible.
4. Affiche les informations du livre après chaque opération.

**Exercice 5 : Classe Horloge**

**Classe Horloge**

* **Attributs :**
  + heures : int
  + minutes : int
  + secondes : int
* **Méthodes :**
  + initialise(int heures, int minutes, int secondes) : initialise les attributs avec les valeurs fournies.
  + ajouterSecondes(int secondesAjoutees) : ajoute un nombre de secondes à l'heure actuelle, en ajustant les minutes et heures si nécessaire.
  + afficherHeure() : affiche l'heure sous le format HH:MM:SS.

**Classe TestHorloge**

Écrire une classe TestHorloge qui :

1. Initialise une horloge avec une heure donnée.
2. Ajoute 70 secondes à l'heure actuelle.
3. Affiche l'heure après l'ajout.

**Exercice 6 : Classe Produit**

**Classe Produit**

* **Attributs :**
  + nom : String (nom du produit).
  + prixUnitaire : float (prix unitaire du produit).
  + quantite : int (quantité en stock).
* **Méthodes :**
  + initialise(String nom, float prixUnitaire, int quantite) : initialise les attributs avec les valeurs fournies.
  + achat(int nb) : réduit la quantité en stock du nombre d’articles achetés, ou affiche une erreur si le stock est insuffisant.
  + changerPrix(float nouveauPrix) : modifie le prix unitaire du produit.
  + afficherProduit() : affiche les informations du produit (nom, prix unitaire, quantité).

**Classe TestProduit**

Écrire une classe TestProduit qui :

1. Crée un produit avec un nom, un prix unitaire et une quantité donnée.
2. Effectue un achat de 5 unités.
3. Change le prix du produit.
4. Affiche les informations du produit après chaque modification.

**Exercice 7 : Classe Etudiant**

**Classe Etudiant**

* **Attributs :**
  + nom : String (nom de l’étudiant).
  + note1 : float (première note).
  + note2 : float (deuxième note).
* **Méthodes :**
  + initialise(String nom, float note1, float note2) : initialise le nom et les notes.
  + moyenne() : calcule et retourne la moyenne des deux notes.
  + afficherResultat() : affiche « Admis » si la moyenne est supérieure ou égale à 10, sinon affiche « Ajourné ».

**Classe TestEtudiant**

Écrire une classe TestEtudiant qui :

1. Crée un étudiant avec un nom et deux notes.
2. Calcule et affiche sa moyenne.
3. Affiche s'il est admis ou ajourné.

**Exercice 8 : Classe Personne**

**Classe Personne**

* **Attributs :**
  + nom : String (nom de la personne).
  + age : int (âge de la personne).
* **Méthodes :**
  + initialise(String nom, int age) : initialise le nom et l’âge de la personne.
  + estMajeur() : retourne true si la personne est majeure (18 ans ou plus), sinon retourne false.
  + afficherDetails() : affiche les informations de la personne (nom et âge) avec un message indiquant si elle est majeure ou mineure.

**Classe TestPersonne**

Écrire une classe TestPersonne qui :

1. Crée une personne en demandant son nom et son âge à l'utilisateur.
2. Affiche les informations de la personne et indique si elle est majeure ou mineure.

**Exercice 9 : Classe Banque**

**Classe Banque**

* **Attributs :**
  + nom : String (nom de la banque).
  + clients : List<String> (liste des noms des clients de la banque).
* **Méthodes :**
  + initialise(String nom) : initialise la banque avec un nom et une liste de clients vide.
  + ajouterClient(String client) : ajoute un client à la liste.
  + afficherClients() : affiche la liste des clients de la banque.

**Classe TestBanque**

Écrire une classe TestBanque qui :

1. Crée une banque avec un nom donné.
2. Ajoute trois clients.
3. Affiche la liste des clients.

**Exercice 10 : Classe LivreEmpruntable**

**Classe LivreEmpruntable**

* **Attributs :**
  + titre : String (titre du livre).
  + emprunte : bool (indique si le livre est emprunté, initialisé à False).
* **Méthodes :**
  + initialise(String titre) : initialise le titre du livre.
  + emprunter() : si le livre n’est pas encore emprunté, le marque comme emprunté et affiche un message de confirmation ; sinon affiche un message d’erreur.
  + rendre() : marque le livre comme non emprunté et affiche un message de confirmation.
  + afficherStatut() : affiche si le livre est emprunté ou disponible.

**Classe TestLivreEmpruntable**

Écrire une classe TestLivreEmpruntable qui :

1. Crée un livre avec un titre donné.
2. Emprunte le livre.
3. Tente de l'emprunter à nouveau.
4. Rend le livre.
5. Affiche le statut du livre à chaque étape.