TD 3 – DÉVELOPPEMENT JAVA : PROGRAMMATION ORIENTÉE OBJET

Exercice 1 : Gestion d'une bibliothèque

Contexte : Vous développez un système de gestion pour une bibliothèque municipale.

Consignes :

1. Créez une classe Livre avec les attributs suivants :
    - titre (String)
    - auteur (String)
    - anneePublication (int)
    - disponible (boolean)
2. Dans une classe main, créez 3 objets de type Livre avec les données suivantes :
    - Livre 1 : "Le Petit Prince", "Antoine de Saint-Exupéry", 1943, true
    - Livre 2 : "1984", "George Orwell", 1949, false
    - Livre 3 : "L'Étranger", "Albert Camus", 1942, true
3. Affichez les informations de chaque livre sous la forme :
   "Le Petit Prince par Antoine de Saint-Exupéry (1943) - Disponible : true"

Exercice 2 : Gestion d'un parc automobile

Contexte : Une entreprise de location de voitures souhaite gérer son parc automobile.

Consignes :

1. Créez une classe Voiture avec les attributs suivants :
    - marque (String)
    - modele (String)
    - kilometrage (int)
    - prixLocationJour (double)
2. Créez 4 voitures dans la méthode main :
    - Voiture 1 : "Renault", "Clio", 45000, 35.50
    - Voiture 2 : "Peugeot", "3008", 12000, 55.00
    - Voiture 3 : "Citroën", "C3", 78000, 30.00
    - Voiture 4 : "Toyota", "Yaris", 5000, 40.00
3. Parcourez toutes les voitures et affichez uniquement celles dont le kilométrage est inférieur à 50 000 km.

4. Calculez et affichez le prix total si on loue toutes les voitures affichées pour 7 jours.

Exercice 3 : Système de réservation

Contexte : Un restaurant souhaite gérer ses tables via un système simple.

Consignes :

1. Créez une classe TableRestaurant avec les attributs suivants :
    - numeroTable (int)
    - nombrePlaces (int)
    - estReservee (boolean)
    - nomClient (String) – **peut être vide si pas réservée**
2. Dans la méthode main, créez 5 tables :
    - Table 1 : numéro 1, 2 places, non réservée, ""
    - Table 2 : numéro 2, 4 places, réservée, "Dupont"
    - Table 3 : numéro 3, 6 places, non réservée, ""
    - Table 4 : numéro 4, 2 places, réservée, "Martin"
    - Table 5 : numéro 5, 8 places, non réservée, ""
3. Affichez toutes les tables disponibles (non réservées) avec leur nombre de places.
4. Calculez et affichez le nombre total de places disponibles dans le restaurant.

PARTIE 2 : CONSTRUCTEURS

Objectifs

- Implémenter les trois types de constructeurs : par défaut, paramétré, par recopie
- Comprendre l'initialisation d'objets

Exercice 4 : Gestion de comptes bancaires

Contexte : Une banque souhaite créer un système de gestion de comptes clients.

Consignes :

1. Créez une classe CompteBancaire avec les attributs suivants :
    - numeroCompte (String)
    - nomTitulaire (String)

- solde (double)
- typeCompte (String) – ex: "Courant", "Épargne"

2. Implémentez trois constructeurs :
    - Constructeur par défaut : initialise un compte avec :
        - numeroCompte = "000000"
        - nomTitulaire = "Anonyme"
        - solde = 0.0
        - typeCompte = "Courant"
    - Constructeur paramétré : accepte les 4 paramètres et initialise l'objet
    - Constructeur par recopie : accepte un objet CompteBancaire et copie tous ses attributs
3. Dans la méthode main, testez les trois constructeurs :
    - Créez un compte avec le constructeur par défaut → affichez ses infos
    - Créez un compte avec le constructeur paramétré (n° "FR123456", "Sophie Leblanc", 2500.00, "Épargne") → affichez ses infos
    - Créez une copie du deuxième compte → modifiez le nom du titulaire de la copie en "Jean Leblanc" → affichez les deux comptes pour vérifier l'indépendance

Exercice 5 : Gestion de produits e-commerce

Contexte : Une plateforme e-commerce gère un catalogue de produits électroniques.

Consignes :

1. Créez une classe ProduitElectronique avec les attributs suivants :
    - reference (String)
    - nom (String)
    - prix (double)
    - stock (int)
    - garantieAnnees (int)
2. Implémentez trois constructeurs :
    - Constructeur par défaut : référence = "REF000", nom = "Produit générique", prix = 0.0, stock = 0, garantieAnnees = 1
    - Constructeur paramétré complet : tous les paramètres
    - Constructeur paramétré partiel : seulement (reference, nom, prix) → stock = 10 par défaut, garantieAnnees = 2 par défaut
3. Créez une méthode afficherInfos() qui affiche toutes les informations du produit de manière formatée.
4. Dans la méthode main :

- Créez 3 produits avec les trois types de constructeurs
- Affichez les informations de chaque produit
- Créez un 4ème produit en copiant le 2ème (via constructeur par recopie à implémenter)
- Modifiez le stock du produit copié et vérifiez que l'original n'est pas affecté

PARTIE 3 : ENCAPSULATION

Objectifs

- Utiliser les modificateurs d'accès (private, public)
- Créer des getters et setters
- Valider les données avec les setters

Exercice 6 : Gestion de réservations d'hôtel

Contexte : Un hôtel souhaite sécuriser les données de réservation.

Consignes :

1. Créez une classe ReservationHotel avec les attributs privés suivants :
    - numeroReservation (String)
    - nomClient (String)
    - nombreNuits (int)
    - prixParNuit (double)
2. Créez un constructeur paramétré qui initialise tous les attributs.
3. Créez les getters pour tous les attributs.
4. Créez les setters suivants avec validation :
    - setNombreNuits(int nuits) : accepte uniquement si nuits >= 1 et nuits <= 30
    - setPrixParNuit(double prix) : accepte uniquement si prix >= 50.0 et prix <= 1000.0
    - Les autres attributs ne doivent pas avoir de setters (lecture seule après création)
5. Ajoutez une méthode calculerCoutTotal() qui retourne nombreNuits × prixParNuit.
6. Dans la méthode main :
    - Créez une réservation : "RES2025001", "Marie Dubois", 3, 120.00
    - Affichez le coût total

- Tentez de modifier le nombre de nuits à 35 (doit être refusé)
- Modifiez le nombre de nuits à 5 (doit être accepté)
- Affichez le nouveau coût total

Exercice 7 : Système de gestion d'abonnements

Contexte : Une plateforme de streaming gère les abonnements de ses utilisateurs.

Consignes :

1. Créez une classe Abonnement avec les attributs privés suivants :
    - idAbonnement (String)
    - email (String)
    - formule (String) – "Basic", "Standard", "Premium"
    - prixMensuel (double)
    - actif (boolean)
2. Créez un constructeur paramétré pour tous les attributs.
3. Créez les getters pour tous les attributs.
4. Créez les setters suivants avec validation et logique métier :
    - setEmail(String email) : accepte uniquement si l'email contient "@" et "."
    - setFormule(String formule) :
        - accepte uniquement "Basic", "Standard", "Premium"
        - Met automatiquement à jour prixMensuel : Basic = 9.99, Standard = 14.99, Premium = 19.99
    - setActif(boolean actif) : setter simple
5. Ajoutez les méthodes suivantes :
    - afficherDetails() : affiche toutes les infos de l'abonnement
    - calculerCoutAnnuel() : retourne prixMensuel × 12 si actif, sinon 0
6. Dans la méthode main :
    - Créez 3 abonnements (un de chaque formule, tous actifs)
    - Affichez les détails de chaque abonnement
    - Désactivez le 2ème abonnement
    - Tentez de changer la formule du 1er abonnement en "VIP" (doit échouer)
    - Changez la formule du 1er abonnement en "Premium" (doit réussir et changer le prix)
    - Affichez le coût annuel total de tous les abonnements actifs

PARTIE 4 : HÉRITAGE

Objectifs

- Créer des hiérarchies de classes (classe mère/fille)
- Utiliser le mot-clé extends
- Réutiliser et spécialiser le comportement

Exercice 8 : Système de gestion de véhicules

Contexte : Une société de transport gère différents types de véhicules.

Consignes :

1. Créez une classe mère Vehicule avec les attributs protected :
    - immatriculation (String)
    - marque (String)
    - annee (int)
2. Ajoutez dans Vehicule :
    - Un constructeur paramétré pour initialiser les 3 attributs
    - Une méthode afficherInfos() qui affiche immatriculation, marque et année
3. Créez une classe fille Camion qui hérite de Vehicule avec :
    - Un attribut supplémentaire : chargeMaximale (int) en kg
    - Un constructeur qui appelle le constructeur parent et initialise chargeMaximale
    - Une méthode afficherInfosCamion() qui appelle afficherInfos() puis affiche la charge maximale
4. Créez une classe fille Moto qui hérite de Vehicule avec :
    - Un attribut supplémentaire : cylindree (int) en cm^3
    - Un constructeur qui appelle le constructeur parent et initialise cylindree
    - Une méthode afficherInfosMoto() qui appelle afficherInfos() puis affiche la cylindrée
5. Dans la méthode main :
    - Créez 2 camions et 2 motos avec des données réalistes
    - Affichez les informations complètes de chaque véhicule

Exercice 9 : Système de gestion de personnel

Contexte : Une entreprise gère différents types d'employés avec des spécificités.

Consignes :

1. Créez une classe mère Personnel avec les attributs protected :
    - matricule (String)
    - nom (String)
    - prenom (String)
    - salaireBase (double)
2. Ajoutez dans Personnel :
    - Un constructeur paramétré
    - Une méthode calculerSalaire() qui retourne salaireBase
    - Une méthode afficherFiche() qui affiche matricule, nom, prénom et salaire calculé
3. Créez une classe fille Commercial qui hérite de Personnel avec :
    - Attributs supplémentaires : chiffreAffaires (double), tauxCommission (double – ex: 0.05 pour 5%)
    - Constructeur approprié
    - Redéfinir calculerSalaire() pour retourner : salaireBase + (chiffreAffaires × tauxCommission)
    - Une méthode afficherFicheCommercial() qui appelle afficherFiche() puis affiche CA et taux
4. Créez une classe fille Technicien qui hérite de Personnel avec :
    - Attributs supplémentaires : nombreHeuresSupp (int), tauxHoraire (double)
    - Constructeur approprié
    - Redéfinir calculerSalaire() pour retourner : salaireBase + (nombreHeuresSupp × tauxHoraire)
    - Une méthode afficherFicheTechnicien() qui appelle afficherFiche() puis affiche heures supp et taux
5. Créez une classe fille Manager qui hérite de Personnel avec :
    - Attribut supplémentaire : prime (double)
    - Constructeur approprié
    - Redéfinir calculerSalaire() pour retourner : salaireBase + prime
    - Une méthode afficherFicheManager() qui appelle afficherFiche() puis affiche la prime
6. Dans la méthode main :
    - Créez 2 commerciaux, 2 techniciens, 1 manager avec des données réalistes
    - Affichez la fiche complète de chaque employé
    - Calculez et affichez la masse salariale totale de l'entreprise (somme de tous les salaires calculés)