**Atelier 4 : Programmation Orientée Objet Java (2)**

Entrainement à la POO Java avec des mini exercices.

**Contexte :**

1. **Constructeur**

Écrire une nouvelle classe Employe possédant les propriétés : nom, prenom, adresse, anneeNaissance et anneeEncours, et ayant deux méthodes, la première "calculerAge" permettant de calculer l’âge d’un employé et la deuxième "afficherInfosClient" permettant d’afficher ses informations.

|  |
| --- |
| **Employe** |
| - nom : String  - prenom : String  - adress : String  - anneeNaissance : int  - anneeEncours : int |
| calculerAge() : int  afficherInfosClient() : void  constructeur() |

Créer une classe de test dans le package "test" qui permet de créer trois instances :

El alami Mohammad, né le 1963 habitant à Casablanca

El wahabi Khalid, né le 1990 habitant à Safi

Sbai Fouad, né le 2006 habitant à Salé

1. **Héritage**

**Exercice 1 :**

Une classe **Article** possède deux champs privés : nom et prix, et une méthode polymorphe afficher() pour afficher le prix d’un article.

Une classe **ArticleEnSolde** dérive de la classe Article. Cette sous-classe comprend le pourcentage de réduction sur le prix d'origine et également la redéfinition de la méthode affiche() afin d’afficher le pourcentage de remise sur le prix d'origine.

**Exercice 2 :**

Au sein d’une banque, chaque adhérent pourra avoir un compte bancaire qui peut être un compte épargne ou un compte courant. La classe compte bancaire comporte les données membres protégées suivantes : numéro du compte (type long) et le solde (type double) et les méthodes publiques suivantes :

* **Ajouter()** : ajout de l’argent dans le compte
* **Retirer()** : retirer de l’argent à partir d’un compte
* **Afficher()** : afficher les caractéristiques d’un compte.

Au sein d’un compte épargne, la valeur minimale du solde est 200 MAD et contient un taux d'intérêt annuel et une méthode permettant de calculer l'intérêt annuel.

Définir les trois classes et tester par la suite ces méthodes dans un programme principal.

**Exercice 3 :**

Dans un établissement Scolaire, on trouve trois sortes de personnes : Secrétaires, enseignants et étudiants. Chaque personne est caractérisée par son nom et prénom, sa date de naissance, son adresse et sa ville qui sont des attributs communs.

On définit les méthodes public suivantes de la classe **Personne** :

* Le constructeur **Personne** (String nom, String prenom, String adresse, String ville, dateNaissance) : crée et initialise un objet de type Personne.
* String **toString** () : fournit une chaîne de caractères correspondant aux caractéristiques (attributs) d’une personne.
* **modifiePersonne** (String adresse, String ville) : modifie l’adresse et la ville d’une personne
* **ecrirePersonne**() : pourrait afficher les caractéristiques d’une personne. Elle est déclarée **abstraite**.

**Secretaire** (String nom, String prenom, String adresse, String ville, String numeroBureau):

* Le constructeur **Secretaire** doit fournir les caractéristiques pour construire une Personne, plus les spécificités de la classe Secretaire (numeroBureau).
* String **toString ()** : fournit une chaîne contenant les caractéristiques d’une Secretaire.

De même, un **Enseignant** est une Personne enseignant une spécialité (mathématiques, informatique, anglais, gestion, etc.). Un **Étudiant** est une Personne préparant un diplôme (diplome). Les méthodes pour Enseignant et Étudiant sont similaires à celles de Secretaire. Une variable privée static dans chaque classe compte le nombre de personnes créées dans chaque catégorie.

**Questions :**

1. Proposez un diagramme de classes UML dans lequel vous préciserez les relations d’héritage.
2. Réalisez le code Java
3. **Polymorphisme**

**Exercice 1 :**

Une classe Personne avec :

* Un constructeur **Personne (**nom, prénom, adresse, ville et date de naissance)
* Une méthode polymorphe Afficher() pour afficher les données de chaque personne

Une classe Employe qui dérive de la classe Personne, avec un champ salaire, un constructeur et la redéfinition de la méthode Afficher().

Une classe Chef qui dérive de la classe Employe, avec un champ service, un constructeur et la redéfinition de la méthode Afficher().

Une classe Directeur qui dérive de la classe Chef, avec un champ societe, un constructeur et la redéfinition de la méthode Afficher.

**Questions :**

1. Écrire les classes Personne, Employe, Chef et Directeur.
2. Créez un programme de test qui comporte un tableau de huit personnes avec cinq employés, deux chefs et un directeur (8 références de la classe Personne dans lesquelles ranger 5 instances de la classe Employé, 2 de la classe Chef et 1 de la classe Directeur).
3. Affichez l'ensemble des éléments du tableau.

**Exercice 2**

On désire implémenter un système de gestion des emprunts d’ouvrages dans une bibliothèque. Un ouvrage doit être nécessairement soit un livre, soit une vidéo.

Une analyse a permis d’identifier les classes suivantes :

* **Ouvrage** est caractérisé par un titre (String), une date de création (String) et un indicateur d’existence (booléen : vrai si l’ouvrage est disponible dans la bibliothèque et faux s’il est emprunté). Une méthode polymorphe afficher() pour afficher les données de chaque Ouvrage
* **Livre** est un ouvrage caractérisé par son auteur (String) et une redéfinition de la méthode afficher()
* **Video** est un ouvrage caractérisé par son éditeur (String) et sa durée (Réel) et une redéfinition de la méthode afficher()
* **Abonne** est caractérisé par son numéro d’identité (entier), son nom (String), le numéro de son abonnement (entier) et l’ouvrage pris de la bibliothèque (un abonné peut prendre à la fois un seul ouvrage).
* **Bibliotheque** est caractérisée par un ensemble d’ouvrages, où chaque ouvrage est présent en un seul exemplaire et un ensemble d’abonnés.

Cette classe doit offrir les méthodes suivantes :

* + **ajoutOuvrage** : qui permet d’ajouter un ouvrage dans la Liste, s’il n’existe pas déjà (la comparaison doit se faire sur le titre)
  + **getOuvrage** : qui prend comme paramètre un titre et retourne l’ouvrage correspondant.
  + **ajoutAbonne** : qui permet d’ajouter un abonné, s’il n’existe pas
  + **getAbonne** : qui prend comme paramètre un numéro d’identité et retourne l’abonné correspondant

**Questions :**

Écrire en Java les classes décrites ci-dessus. Implémenter toutes les méthodes nécessaires même si elles ne sont pas mentionnées.

**Modalités pédagogiques :**

* Durée 2 jours
* Le travail individuel

Voici les concepts que vous devez appréhender :

* Les constructeurs
* Héritage
* Polymorphisme

**Livrable :**

Un dépôt Git dans lequel on retrouve chaque exercice dans un fichier séparé.

**Critères de performance**

Le formateur peut exécuter le code sur sa propre machine.

Le code est livré avant la date et l’heure limite.

**Modalités d'évaluation**

Une présentation de la démonstration.

**Compétences visées :**

Développer la partie back-end d’une interface utilisateur web