Programmation Orientée Objet

TD2: Héritage et polymorphisme

Exercice 5.

A/ Classe Personne

1. Créez la classe *Personne* représentant une personne qui est caractérisée dans le cadre de notre exercice par un nom, un prénom et un âge.
2. Dotez la classe Personne d'un constructeur exhaustif initialisant tous ses champs et d'une méthode *afficher* pour afficher les caractéristique d'un objet de type *Personne.*
3. Créez une classe MainApp comportant la méthode *main* qui crée un objet de type *Personne* en initialisant ses attributs à des valeurs entrées par l'utilisateur de votre programme (*la lecture des valeurs se fait dans le main pas dans le constructeur de la classe Personne)*.

B/ Héritage, redéfinition et polymorphisme

1. Créez une classe nommée *Enseignant* héritant de la classe Personne et ayant deux champs : Numéro de sécurité sociale de type `long,` et `modules` un tableau de chaines de caractères représentant les modules qu'il enseigne.
2. Créez un constructeur pour la classe *Enseignant* appelant celui de la classe mère et initialisant les champs propres à la classe *Enseignant*.
3. Créez de même une classe *Etudiant* héritant de la classe Personne et ayant comme attributs : `matricule` de type entier, `notes` un tableau de réels (de type `double)` indiquant les notes de l'étudiant dans les différents modules qui sont au nombre de 3.
4. Créez un constructeur pour la classe *Etudiant* appelant celui de la classe mère et initialisant les champs propres à la classe *Etudiant*.
5. Dotez la classe *Etudiant* de la méthode `double claculMoyenne(double[] notes)` qui calcule la moyenne de l'étudiant en supposant que tous les modules ont un coefficient de 1.
6. Redéfinissez la méthode *afficher* de *Personne* dans les deux classes Enseignant et Etudiant afin d'afficher les champs qui leurs sont propres.
7. Créez au niveau de la méthode *main* un tableau de 5 éléments de type *Personne* contenant deux éléments de type *Enseignant* et 3 de type *Etudiant*.
8. Affichez pour chaque élément du tableau les informations le concernant.

C/ Classes abstraites et interface Comparable

On désire obliger les programmeurs qui conçoivent des classes héritant de la classe Personne à doter ces classes de certaines méthodes :

1. Rendez la classe *Personne* abstraite et y ajouter la méthode abstraite de signature *afficherType()* qui affiche le type de l'objet.
2. Implémenter cette méthode dans les classes dérivées *Enseignant* et *Etudiant*.
3. Tester le programme modifié dans la méthode *main*.
4. Modifiez le programme afin que les classes *Enseignant* et *Etudiant* implémentent l'interface `Comparable`. La méthode `compareTo` sera redéfinie dans la classe *Enseignant* afin de pouvoir classer les noms et les prénoms des enseignants par ordre alphabétique et sera redéfinie dans la classe *Etudiant* afin de classer les étudiants suivant l'ordre décroissant de leurs moyennes.
5. Modifier la méthode *main* de sorte à avoir un tableau d'étudiants trié et un autre tableau d'enseignants trié. (Utilisez la méthode statique `sort` de la classe `Arrays` : Consultez la documentation Java).

Programmation Orientée Objet

TD2: Héritage et polymorphisme

Exercice 7

NetMusic est une entreprise de vente d'albums de musique sur Internet. Les clients peuvent consulter le catalogue des albums disponibles et choisir ceux qui les intéressent puis les acheter en ligne. Le catalogue comporte un ensemble d'albums définis chacun par son titre, le nom de son auteur, le nom de sa maison de production, l'année de son édition, la liste des titres qu'il comporte ainsi que son prix. Pour faciliter la consultation du catalogue aux clients, NetMusic classe les albums dans des catégories suivant leur style de musique (Classique, Jazz, Pop, Rock, Rap, RnB). Certains albums sont disponibles en version physique (la quantité est spécifiée à la création) alors que d'autres sont disponibles en version numérique. Ces derniers disposent d'une URL à partir de laquelle le client peut les télécharger après l'achat. NetMusic offre une réduction de 5% sur le prix des albums numériques, tandis que le prix d'un album physique est calculé en ajoutant les frais de livraison qui constituent 10% du prix de l'album. NetMusic informe ses clients que seuls les albums physiques peuvent être échangés ou remboursés sous un délai de 7 jours. Chaque client est caractérisé par son nom, son prénom et son adresse. Pour effectuer ses achats en ligne, le client doit disposer d'un compte sur le site de NetMusic qui est caractérisé par un nom d'utilisateur, un mot de passe, un solde qui peut être créditer par une valeur, et qui est débité après un achat, et aussi un panier. Ce dernier sert à conserver la liste des albums sélectionnés par le client avant de lancer l'opération d'achat.

1. Proposez une modélisation orientée objet : Tracez un diagramme représentant les classes en précisant pour chacune d'elles si elle est abstraite, si c'est une interface ou si c'est une énumération. Indiquez les relations entre les différentes classes (utilisation, héritage, implémentation). Détaillez les attributs et les méthodes de chaque classe dans un tableau (voir tableau ci-dessous) en spécifiant leur mode d'accès, leur type ou type de retour et leur rôle.

Nom classe	Rôle
Liste attributs	
Méthode d'accès Type_attribut nom_attribut	Rôle
…	…
Liste des méthodes	
Signature méthode	Rôle
…	…

NetMusic diversifie ses produits en offrant à ses clients la possibilité de commander des magazines culturels relatifs à la musique. Ces derniers peuvent également êtres échangés ou remboursés sous un délai de 7 jours.

2. Apportez à votre conception les modifications nécessaires pour s'adaptez à ces nouvelles offres de service.

3. ***L'implémentation sera traitée lors du TD3.*** Toutefois, vous pouvez commencer à implémentez vos classes de sorte à pouvoir compléter le *main* suivant :

Programmation Orientée Objet

TD2: Héritage et polymorphisme

```
public static void main(String[] args) {
      //Création de NetMusic
      NetMusic site=new NetMusic(…);

       /*** I. Gestion des clients ***/
      //A. Création de Compte
             //1. Création d'un client qui n'existe pas
             …….
             //2. Test de création d'un client qui existe
             ….
             //3. Test de création d'un compte avec un nom d'utilisateur existant
             …..
             //4. Création d'un compte avec un nom d'utilisateur différent

      //B. Authentification

             //1. Test d'authentification d'un client qui n'existe pas
             …….
             //2. Test d'authentification d'un client qui existe mais le mot de passe est incorrecte
             …….
             //3. Test d'authentification d'un client qui existe avec mot de passe correcte
             …….

      /*** II. Gestion du catalogue d'Album***/
      //A. Création et affichage des albums

             //1. Ajouter des albums physiques et numériques au catalogue
             …….
             //2. Consulter le catalogue d'album
             …….

      //B. Achat et Remboursement des albums

             //1. Achat de deux albums physique
             …….
             //2. Achat d'un album numérique
             …….
             //3. Remboursement d'un album physique dans un délai >7 jours
             …….
             //4. Remboursement d'un album physique dans un délai de 7 jours
             …….

      /*** III. Gestion du catalogue de magazines***/
      //A. Création et affichage des magazines

             //1. Ajouter des magazines au catalogue
             …….
             //2. Consulter le catalogue de magazines
             …….

       //B. Achat et Remboursement des magazines

             //1. Achat d'un magazine sans avoir le solde suffisant pour le faire
             …….
             //2. Créditer le compte avec 500 DA
             …….
             //3. Achat de deux magazines
             …….
             //4. Remboursement d'un magazine dans un délai >7 jours
             …….
```

```
        //5. Remboursement d'un magazine dans un délai de 7 jours
        …….

}
```

Programmation Orientée Objet

TD2: Héritage et polymorphisme

Annexe 1. Compilation et exécution en mode console

Ici nous allons compiler et exécuter un programme Java en utilisant une console de commandes (pas d'IDE). Pour cela, nous allons utiliser la classe Point vue en cours.

1. Notez le chemin d'installation du répertoire bin du JDK. Sous Windows, il est souvent sous la forme : C:\ProgramFiles\Java\jdk1.8.0\bin
2. Pour être sûr que Windows reconnaisse le compilateur java, spécifier ce chemin comme variable d'environnement :
    a. Allez vers Ordinateur -> Propriétés -> Avancé -> Variable d'environnement -> Variable système
    b. Ajoutez le chemin du répertoire bin du jdk déjà noté à la valeur de la variable PATH
    c. Cliquez sur OK
3. Copiez le code de la Point du cours dans un fichier avec un éditeur texte (ex. blocNote) et ajoutez-y le code suivant

```
public static void main (String args[]){
Point p = new Point();
p.afficher();
}
```

4. Enregistrez le fichier avec le nom Point.java en vous assurant qu'il a l'extension « .java » et pas «.txt » ou « .java.txt »
5. Ouvrez la console de commande et placez-vous dans le dossier contenant le fichier Point.java. Pour compiler ce fichier, il faut utiliser la commande ***javac*** suivie du nom du fichier.java

```
javac Point.java
```

Remarque : Si la commande *javac* n'ai pas reconnue par Windows, vous pouvez toujours utilisez la commande DOS *set PATH* pour mettre en place la variable d'environnement pendant cette session de console (à refaire à chaque fois):

```
set PATH=%PATH%; C:\Program Files\Java\jdk1.8.0\bin
```

S'il n'y a pas d'erreurs, vous verrez apparaître dans le même dossier le fichier Point.class généré par le compilateur. Il contient le bytecode java qui sera utilisé par la JVM au moment de l'exécution.

6. Pour exécuter un programme Java, il faut utiliser la commande ***java*** suivie du nom du fichier

```
java Point
```

Programmation Orientée Objet

TD2: Héritage et polymorphisme

Annexe 2. Les énumérations

Une énumération est un type de données particulier qui permet à une variable d'être définie par un ensemble de constantes prédéfinies. Les énumérations héritent de la classe java.lang.Enum Une énumération se déclare comme une classe, mais en remplaçant le mot clé class par enum.

Exemple : une énumération des jours de la semaine se déclare comme suit :

```
public enum Jour {
    DIMANCHE, LUNDI, MARDI, MERCREDI, JEUDI, VENDREDI, SAMEDI
}
```

Voici un code qui montre comment utiliser l'énumération Jour définie ci-dessus:

```
class EssaiJour {
public static void main(String[] arg) {

/*Déclaration d'une énumération*/
Jour jour = Jour.valueOf(arg[0]);
if (jour == Jour.SAMEDI) System.out.print("fin de semaine : ");
switch(jour){
    case VENDREDI :
    case SAMEDI :System.out.println("se reposer");
    break; default :
    System.out.println("travailler");
    break;}
}
}
```

Pour la commande « java EssaiJour SAMEDI », on obtient : `fin de semaine : se reposer`