Programmation Orientée Objet (POO)

EX 4 : Héritage

Objectifs

Comprendre les mécanismes de l'héritage d'attributs et de méthodes
Connaître les notions de méthode polymorphe et de résolution d'appel de méthode
Savoir concevoir et programmer une hiérarchie de classes

I. Gestion d'une bibliothèque

Pour la gestion d'une bibliothèque on vous demande d'écrire une application traitant des *documents* de natures diverses : des livres, des revues, des dictionnaires, etc.
Les *livres*, à leur tour, peuvent être des *romans* ou des *manuels*.
Tous les documents ont un numéro d'enregistrement (un entier) et un titre (une chaîne de caractères).
Les livres ont, en plus, un auteur (une chaîne) et un nombre de pages (un entier).
Les romans ont éventuellement un prix littéraire (un entier conventionnel, parmi : GONCOURT, MEDICIS, INTERALLIE, etc.).
Les manuels ont un niveau scolaire (un entier).
Les revues ont un mois et une année (des entiers).
Les dictionnaires ont une langue (une chaîne de caractères convenue, comme "anglais", "allemand", "espagnol", etc.).
Tous les objets en question ici (livres, revues, dictionnaires, romans, etc.) doivent pouvoir être manipulés en tant que documents.

1) Faire le schéma des classes en UML
ATTENTION : Il faut vérifier la logique de votre schéma notamment qu'il est possible de dire par exemple : "un Roman est-un Livre"

2) Définir les classes en Java
ATTENTION : ne pas redéclarer les attributs hérités dans les sous-classes !

II. Banque et héritage

Dupliquer votre projet banque en un projet banqueHeritee.

Modifier les différentes classes de manière à gérer de manière plus générale une classe Compte et ses sous-classes CompteEpargne et CompteCourant.

Vérifier que c'est logique de dire :

Un …… EST-UN ……..

Un …… EST-UN ……..

Un …… A-UN ……..

Un …… A-UN ……..

Quels sont les attributs que vous devez déclarer dans Compte ?

Faire un schéma UML des classes comme vu en cours avec : noms des classes, noms des attributs, flèches « est-un ».

Programmer une méthode créditer et débiter dans chaque classe Compte.

Dans la méthode debiter des comptes courants, prendre en compte le fait qu'il y a un seuil de découvert autorisé.

Pour la méthode crediter des comptes epargnes, prendre en compte le fait qu'il y a un taux d'intérêt à appliquer au montant crédité.

Mettre à jour votre classe Banque de manière à exploiter le polymorphisme et la résolution dynamique d'appel de méthode.

III. Gestion d'un poulailler

Un éleveur de volailles reçoit d'un fournisseur de jeunes canards et de jeunes poulets qu'il élève jusqu'à ce qu'ils aient la taille suffisante à leur commercialisation.

Une volaille est caractérisée par son poids et un numéro d'identification reporté sur une bague qu'elle porte sur sa patte pour des raisons de traçabilité. Les volailles arrivent à l'élevage à l'âge de deux semaines. Elles sont baguées et enregistrées.

Le prix du canard et celui du poulet sont deux prix différents, exprimés en "Euros par kilo". Le prix est le même pour tous les individus de la même espèce. Ce prix varie tous les jours.

Le poids auquel on abat les bêtes est différent pour les canards et les poulets, mais c'est le même pour tous les poulets et le même pour tous les canards.

1) Dessiner un schéma UML avec les classes, les attributs d'objets et attributs de classe.

2) Ecrivez les classes nécessaires. Il faut pouvoir enregistrer les prix du jours, les poids d'abattage, le poids d'une volaille donnée. Ecrivez une classe Elevage permettant de représenter l'ensemble des animaux de l'élevage au moyen d'un tableau. Des méthodes doivent permettent de

sélectionner les animaux à abattre et d'évaluer le prix obtenu pour ces animaux qui doivent être ensuite supprimés du tableau. Il faut également pouvoir enregistrer les jeunes animaux qui arrivent. On vous donne l'exemple du main() d'une classe Ferme qui utilise ces classes.

1. Quelles méthodes sont polymorphes ……..
2. Est-ce que vous avez mis des instructions dans la méthode getPrix() de la classe Volaille ?

   …….

   Est-ce logique ? ……...
3. Quelles méthodes sont surchargées ? …….

Vocabulaire

- Surcharge : même nom mais entête différente dans la même classe = overloading
- Redéfinition = même entête de méthode dans une sous-classe = overriding
- Type déclaré vs. type réel

```
package poulailler;

public class Ferme {

    //-----------------------------------------------------
    public static void main(String[] args){
       // Creer et ajouter quelques animaux
       Elevage laFerme = new Elevage();
       for    (int i=0; i<15; i++){
             laFerme.ajouter(new Poulet(0.250,150+i));
       }

       for(int i=0; i<15; i++){
             laFerme.ajouter(new Canard(0.250,380+i));
       }

       for (int i=0; i<10; i++){
             laFerme.ajouter(new Poulet(0.250,700+i));
       }
       laFerme.ajouter(new Canard(0.750,825));
```

```
        // Changer le poids de quelques animaux
        for (int i=0; i<8; i++){
              laFerme.changePoids(155+i,1.3);
              laFerme.changePoids(382+i,1.55);
        }

        // Afficher l'état du volailler
        laFerme.ecrire();
        System.out.println("Valeur des animaux a abattre: ");
        System.out.println(laFerme.evaluerBetesAAbattre());
        laFerme.envoyerALAbattoir();
        laFerme.ecrire();

        System.out.println("Valeur des animaux a abattre: ");
        System.out.println(laFerme.evaluerBetesAAbattre());
    }
}
//-------------------------------------------------------------------------------------------------
```

AFFICHAGE LORS DE L'EXECUTION DE Ferme.main() :

150 0.25 0.25

151 0.25 0.25

152 0.25 0.25

153 0.25 0.25

154 0.25 0.25

155 1.3 1.3

156 1.3 1.3

157 1.3 1.3

158 1.3 1.3

159 1.3 1.3

160 1.3 1.3

161 1.3 1.3

162 1.3 1.3

163 0.25 0.25

164 0.25 0.25

380 0.25 0.3

381 0.25 0.3

382 1.55 1.8599999999999999

383 1.55 1.8599999999999999

384 1.55 1.8599999999999999

385 1.55 1.8599999999999999

386 1.55 1.8599999999999999

387 1.55 1.8599999999999999

388 1.55 1.8599999999999999

389 1.55 1.8599999999999999

390 0.25 0.3

391 0.25 0.3

392 0.25 0.3

393 0.25 0.3

394 0.25 0.3

700 0.25 0.25

701 0.25 0.25

702 0.25 0.25

703 0.25 0.25

704 0.25 0.25

705 0.25 0.25

706 0.25 0.25

707 0.25 0.25

708 0.25 0.25

709 0.25 0.25

825 0.75 0.8999999999999999

Valeur des animaux a abattre:

25.279999999999998

++150 0.25 0.25

151 0.25 0.25

152 0.25 0.25

153 0.25 0.25

154 0.25 0.25

825 0.75 0.8999999999999999

709 0.25 0.25

708 0.25 0.25

707 0.25 0.25

706 0.25 0.25

705 0.25 0.25

704 0.25 0.25

703 0.25 0.25

163 0.25 0.25

164 0.25 0.25

380 0.25 0.3

381 0.25 0.3

702 0.25 0.25

701 0.25 0.25

700 0.25 0.25

394 0.25 0.3

393 0.25 0.3

392 0.25 0.3

391 0.25 0.3

390 0.25 0.3

Valeur des animaux a abattre:

0.0

IV. Gestion d'une société de secourisme

Une société médicale souhaite informatiser la gestion de ses véhicules de secours apportés par les secouristes lors qu'ils sont appelés en cas d'accident.

Vehicules

Chaque véhicule est caractérisé par un identificateur unique. Les identificateurs commençent à partir de 100.

Parmi les véhicules de secours que gère la société il y a des engins et des PC (Poste de Commandement).

Un PC (Poste de Commandement) sert de support pour l'organisation des secours. Il reste sur place durant l'intervention et ne peut donc pas transporter de personne blessée.

A chaque engin correspond un nombre de personnes qu'il peut transporter.

Tous les engins permettent de transporter au moins une personne.

Un VA (Véhicule d'Assistance) permet de transporter 1 personne

Un VS (Véhicule de Secours) permet de transporter 2 personnes

La société comporte actuellement 2 PC, 3 VS et 2 VA.

La société prévoit d'acheter dans les prochaines années d'autres types d'engins permettant notamment de transporter plus de deux personnes.

Les méthodes suivantes devront être définies (à vous de trouver dans quelle(s) classe(s)):

- affecter un numéro d'intervention et une distance en km par rapport à la société (pour tous les véhicules)
- embarquer (String nomPersonne1) pour tous les engins
- pour les VS : prévoir aussi une version de la méthode qui permette d'embarquer en même temps 2 personnes : embarquer (String nomPersonne1, String nomPersonne2)
- affecter un hopital de destination (uniquement pour les engins)
- calculer le cout
    - pour le PC, le cout est fixe : 1000 Euros
    - pour les engins, cela dépend :
        - VA : 300 Euros
        - VS : le cout est de 600 Euros pour les distances supérieures à 50 km, sinon le cout est de 150 Euros par personne transportée

Paquetages

Vous définirez vos classes dans des paquetages vehicules et vehicules.engins

La classe SocieteMed sera définie dans le paquetage par défaut.

Vous devez mettre les instructions package et import nécessaires pour toutes les classes (y compris SocieteMed pour laquelle vous founirez aussi attribut(s) et méthode(s) autres que main()).

Définir les classes nécessaires pour gérer les véhicules et la société.

Vous mettrez par défaut les attributs en private mais vous n'avez pas besoin d'écrire les accesseurs.

Cependant, dans le cas de classes héritées, on permettra aux sous-classes d'accéder directement aux attributs de la classe mère.

Questions sur ces classes

1. Donner un exemple de méthode polymorphe. Expliquez.
2. Donner un exemple de méthode surchargée. Expliquez.
3. Donner un exemple de polymorphisme appliqué au type d'un objet passé en paramètre. Expliquez.

Source de la méthode main() de la classe SocieteMed :

```
public static void main (String args []) {
       SocieteMed s = new SocieteMed();

       // Créer les véhicules
       PC pc1 = new PC ();
       PC pc2 = new PC ();
       VS vs1 = new VS ();
       VS vs2 = new VS ();
       VS vs3 = new VS ();
       VA va1 = new VA ();
       VA va2 = new VA ();

       // Ajouter les véhicules à la société
       s.ajouterVehicule (pc1);
       s.ajouterVehicule (pc2);
       s.ajouterVehicule (vs1);
       s.ajouterVehicule (vs2);
       s.ajouterVehicule (vs3);
       s.ajouterVehicule (va1);
       s.ajouterVehicule (va2);
```

```
        // Affecter qq véhicules à une intervention
        int numeroIntervention = 17 ;
        int distance = 35 ;
        pc1.affecterIntervention (numeroIntervention, distance);
        vs1.affecterIntervention (numeroIntervention, distance);
        vs2.affecterIntervention (numeroIntervention, distance);
        va2.affecterIntervention (numeroIntervention, distance);

        // Lors de l'intervention, embarquer quelques personnes
        vs1.embarquer("Dupont");
        vs2.embarquer("Martin", "Durand");

        // Afficher les informations sur tous les véhicules
        System.out.println (s);

        // Calculer le cout de l'intervention
        System.out.println ("Cout total de la journée : " +
s.calculerCout());
    }
```

Execution de la méthode main() de la classe SocieteMed :

```
SocieteMed [vehicules=[
PC [id=100, numeroIntervention=17, distance=35],
PC [id=101, numeroIntervention=0, distance=0],

VS [nomPersonne2=null, nbPlaces=2, hopital=null, nomPersonne1=Dupont,
nbPersonnesTransportees=1, id=102, numeroIntervention=17, distance=35],
VS [nomPersonne2=Durand, nbPlaces=2, hopital=null, nomPersonne1=Martin,
nbPersonnesTransportees=2, id=103, numeroIntervention=17, distance=35],
VS [nomPersonne2=null, nbPlaces=2, hopital=null, nomPersonne1=null,
nbPersonnesTransportees=0, id=104, numeroIntervention=0, distance=0],

VA [nbPlaces=1, hopital=null, nomPersonne1=null,
nbPersonnesTransportees=0, id=105, numeroIntervention=0, distance=0],
```

```
VA [nbPlaces=1, hopital=null, nomPersonne1=null,
nbPersonnesTransportees=0, id=106, numeroIntervention=17,
distance=35]]]

Cout total de la journée : 1750.0
```