Université Paris Diderot **Programmation Orientée Objet**
L3 **Année 2012-2013**

TD - Séance n°2

Révisions Classes, Encapsulation, Listes

Dans cette seance vous devriez reussir a terminer les 4 premiers exercices. Le 5 eme est un complement. En condition d'examen les 4 premiers exercices doivent être fait en moins de 40 minutes, entraînez-vous jusqu'a ce que ce soit le cas.

Exercice 1 *Révision. Une classe simple accompagnée d'une classe Test*

On considere les deux classes suivantes :

```
public class Personne {
    private String nom;
    private String prenom;
    public int age;

    public Personne(String nom, String prenom, int age){
        this.nom = nom;
        this.prenom = prenom;
        this.age = age;
    }
    public void setPrenom(String p){
        this.prenom = p;
    }
    public void anniversaire(){
        this.age ++;
    }
    public void presenteToi(){
        System.out.println("Je m'appelle : " + this.prenom +" "+ this.nom+".");
        System.out.println("J'ai "+this.age+" ans.");
    }
}

public class Test {
    public static void main(String[] args){
        Personne martin = new Personne("Dupont","Jean",15);
        martin.presenteToi();
        martin.setPrenom("Martin");
        martin.presenteToi();
    }
}
```

1. Qu'obtient-on dans le terminal a l'execution de `Test.class` ?
2. Peut-on executer les lignes suivantes dans le `main` pour changer le nom d'une Personne ?

   ```
   martin.nom = "Scorsese";
   martin.presenteToi();
   ```

   Si non, proposez une facon de faire. Discutez de la nature `public` du champ `age`.
3. Si le main avait ete ecrit dans la classe `Personne` et non dans la classe `Test` aurait-on eu le droit d'ecrire `martin.nom = "Scorsese";` ?

Exercice 2 *Petites manipulations*

1. Modi ez la classe `Personne` de l'exercice precedent, en ajoutant des proprietes, de sorte que chacun puisse avoir une certaine quantite de monnaie dans sa poche.
2. Les deux methodes suivantes doivent permettre a deux personnes de se transmettre une certaine somme. Le boolean retourne correspond au succes (ou non) du payement.

   ```
   public static boolean donne (Personne p1, Personne p2, int montant) { ... }
   public boolean donne(Personne p,  int montant){ ... }
   ```

   Ecrivez ces methodes, et un exemple d'appel pour chacune. En commentaire, completez la speci cation en precisant qui paye a qui.

Exercice 3 *Encapsulation*

Un compte en banque se caracterise par son solde, et par son titulaire (qui est une personne).

1. Ecrivez une classe Compte, ainsi qu'un constructeur.
2. Ecrivez des methodes `getSolde, credite, debite` qui respectivement donnent le montant present sur un compte, credite et debite un compte d'un certain montant.
3. Ecrivez une methode qui permette au titulaire de retirer une certaine somme pour la mettre dans sa poche. (Pas de decouvert)
4. Ecrivez une methode qui permette au titulaire d'e ectuer un depôt, a partir de l'argent qu'il a en poche.

Exercice 4 *Liens croisés*

Dans notre modelisation, un compte est lie a son titulaire, mais une personne n'est pas au courant qu'elle possede un compte. Nous proposons ici une solution simple.

1. Modi ez la classe Personne, en ajoutant un champs de type `Compte[]` qui contiendra l'ensemble des comptes associes a une personne.

2. Modi ez le constructeur de Personne, en ajoutant un parametre `int n` : le constructeur se chargera de fabriquer `n` comptes di erents de solde nul pour cette personne.
3. Ecrivez dans votre `main` quelques manipulations de credit sur ces comptes.
4. Quand on cherche a retirer une somme importante, on peut decider de vider chacun de nos comptes tant qu'on n'a pas atteint cette somme (ou que tous nos comptes soient a sec). Ecrivez cette methode `retrait`.

Exercice 5 *Complément*

Cette modelisation a quelques faiblesses. En e et dans la realite une personne ne na^t pas avec un stock de comptes, et n'est pas limitee dans le nombre de comptes qu'elle peut ouvrir. D'autre part, c'est au moment de la creation d'un compte que la personne se voit informee de l'existence de ce compte.

Proposez une modelisation avec des listes cha^nees qui soit plus souple que celle des exercices precedents. Nous vous demandons une solution avec une implementation des listes qui n'utilise pas de bibliotheques java. Vous porterez toute votre attention sur le cha^nage de vos objets, et a une de nition propre des listes vides.