Ecole nationale Supérieure d'Informatique (ESI) Le 21/05/2011

2ème année CPI Nom : Prenom : Groupe :

# Contrôle Intermédiaire

## UEF4.3. Programmation Orientée Objet

**Durée 02 heures**

**Documents interdits**

**Nom :**

**Prénom :**

**Groupe :**

**Questions de cours :**

1. Qu'est ce que l'encapsulation ?

**L'encapsulation est un principe clé dans la POO qui consiste à regroupe des données (attributs) et un comportement (méthodes) dans une même classe et à réglementer l'accès aux données de l'extérieur.**

2. Citer les deux autres concepts de la Programmation Orientée Objet .

**Héritage et polymorphisme.**

3. Quelle est la différence entre une interface et une classe abstraite ?

- **Une interface est une classe dont toutes les méthodes sont abstraites.**
- **Une classe qui implémente une interface doit donner un corps à toutes ses méthodes. Alors qu'une classe qui dérive d'une classe abstraite n'est pas obligée de redéfinir toutes ses méthodes et peut même n'en redéfinir aucune.**
- **Une interface peut être instanciée alors qu'une classe abstraite ne peut qu'être dérivée.**
- **Une classe peut implémenter plusieurs interfaces mais ne peut dériver que d'une seule classe abstraite**

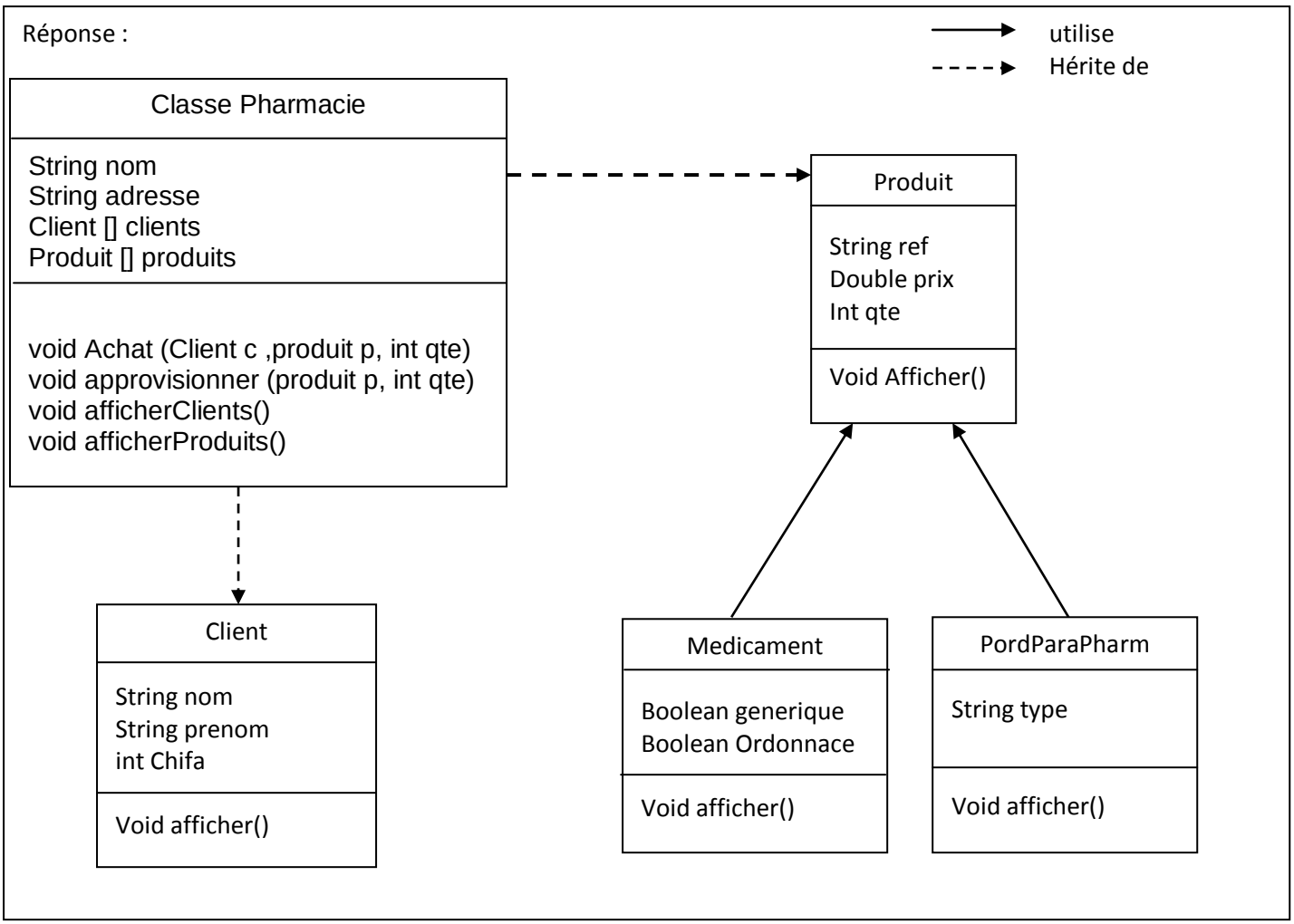
**Exercice 1. Gestion d'une pharmacie**

On s'intéresse à la gestion d'une pharmacie qui gère des clients et des produits. Une pharmacie est caractérisée par son nom (de type String) , son adresse (de type String), ses clients ( un tableau de clients) et la liste de ses produits (un tableau de produits). Un client est caractérisé par son nom (de type String), son prénom (de type String), le numéro de sa carte CHIFA ( de type int). Un produit est caractérisé par sa référence (de type String), son prix (de type double) et sa quantité en stock (de type int). Les produits que vend cette pharmacie sont soit des médicaments, soit des produits de parapharmacie. Les médicaments sont caractérisés par le fait qu'ils peuvent être génériques ou pas, et par le fait qu'ils peuvent être délivrés sans ordonnance ou pas. Les produits de parapharmacie sont quant à eux caractérisés par leur type (produit de beauté, cosmétique ou diététique). Le programme doit permettre de gérer :

- L'opération d'achat caractérisée par un produit, un client et une quantité.
- L'opération d'approvisionnement du stock caractérisée par un produit et une quantité.
- L'affichage de la liste des clients de la pharmacie et la liste des produits qu'elle vend.
- L'affichage des informations concernant le client.
- L'affichage des informations concernant un produit.
- L'affichage des informations concernant un médicament.
- L'affichage des informations concernant un produit de parapharmacie.

Question : Proposer une modélisation orientée objet à ce problème (Donner les classes, leurs attributs et leurs méthodes) sous forme d'un schéma, en précisant les relations entre les classes (comme dans l'environnement BlueJ). Il n'est pas demandé de donner le corps des méthodes

Réponse :



**Exercice 2**

Ecrire, en langage Java, deux classes A et B telles que :
– La classe B hérite de la classe A ;
– la classe A possède :
- Un attribut entier x visible seulement par ses classes dérivées ;
- Un attribut entier y visible par ses classes filles et les classes du même package
- Un constructeur affectant une valeur à l'attribut x ;

– La classe B possède :
- un attribut entier z visible uniquement dans la classe B ;
- un constructeur affectant une valeur aux attributs x et z ;

Réponse :

```
Class A {

protected int x ;

protected int y ;

public A(int x) {

this.x = x;

}
```

```
class B {

        private int z;

        public B(int x, int z) {

            super (x);

        this.z = z;

    }

}
```

**Exercice 3.** Qu'affiche le programme suivant ?

```
public class UneClasse{
    public static int x = 7;
    public int y = 3;
}

Public class Exercice2 {
Public static void main(String args[]) {
UneClasse a = new UneClasse ();
UneClasse b = new UneClasse ();
a.y = 5;
b.y = 6;
a.x = 1;
b.x = 2;
System.out.println("a.y = " + a.y);
System.out.println("b.y = " + b.y);
System.out.println("a.x = " + a.x);
System.out.println("b.x = " + b.x);
}
}
```

Réponse : le programme affiche :

**a.y = 5**

**b.y = 6**

**a.x = 2 ;**

**b.x = 2 ;**

**Exercice 4.**

1. Le programme suivant comporte des erreurs. Trouver ces erreurs et les expliquer.

```
1.      class Exercice3 {
2.        public static void main(String[] args) {
3.         A x = new A();
4.         B y = new B();
5.         C z = new C();
6.         y.b = 2;
7.         z.c = 3;
8.      }
9.      abstract class A {
10.     int a;
11.     }
12.     class B extends A {
13.     int b;
14.        public B(int b){
15.        this.b = b;
16.     }
17.     }
18.     class C extends A {
18.         final double c = 1;
20.     }
21.     abstract class D extends A {
22.     double d;
23.     int operation(int a) {
24.     return (a * 2);
25.     }
26.     abstract int calcul(int b) {
27.     }
28.     abstract void afficher();
29.     }
```

**Réponse :**

**Ligne 3 : La classe A est abstraite elle ne peut pas être instanciée.**

**Ligne 4 : La classe B possède un constructeur avec argument, donc on ne peut pas utiliser le constructeur par défaut.**

**Ligne 7: la variable c de la classe C est déclarée final, elle ne peut pas être modifiée.**

**Lignes 26 et 27 :La classe calcul est abstraite, elle ne doit pas comporter d'accolades. Il suffit de mettre un point virgule après sa signature.**

**Il manque une accolade fermante pour la classe Exercice.**

NB: indiquer le numéro de la ligne comportant l'erreur et donner l'explication

**2. a-** La classe Erreur comporte une erreur, laquelle

**(5 mn)**

```
public class Erreur {
        String s;
        static class Inner {
                void testMethod() {
                  s = "Set from Inner";
                }
        }
}
```

Réponse :

**La classe Inner est déclarée static et accède à la variable s qui est une variable d'instance non static de sa classe englobante.**

**Une classe interne *static* ne peut pas accéder à une variable membre.**

**Exercice 5 :** Examiner le code suivant:

```
class A {

public void calculer (double x, int i)
{…
}

}

public class B extends A
{

public void calculer (int i, double x)
{…
}

}
```

Questions : Encadrer la bonne réponse

5.1) La méthode calculer est :

a. Redéfinie,
b. surdéfinie
c. Ni redéfinie, ni surdéfinie

5.2) Si i est un entier et x un double, et a et b sont des objets de types A et B respectivement. Est-ce que les appels suivant sont corrects ? Si oui, quelle méthode est appelée ?

a.calculer ( i, x ) ; // **Incorrect, erreur à la compilation**

a.calculer (x, i); // **OK : Appel de calculer(double, int) de A**

b.calculer (i, x); // **OK: appel de calculer(int , double) de B**

b.calculer (x, i); // **OK: Appel de calculer( double, int de A**

**Questions** : Répondre par **vrai** ou **faux.**

1. Si une classe B hérite d'une classe A, on dit que :
   - a- B spécialise A **Vrai**
   - **b-** B généralise A **Faux**
   - **c-** B possède au moins tous les champs et les méthodes de A **Vrai**
   - **d-** A possède au moins tous les champs et les méthodes de B **Faux**
   - **e-** Toute instance de B peut être considérée comme un A **Vrai**
   - f- Toute instance de A peut être considérée comme un B **Faux**

2. La signature d'une méthode c'est :
   - **a.** Son nom et le type de sa valeur de retour **Faux**
   - **b.** Son nom et les noms de ses paramètres **Faux**
   - **c.** Son nom et les types de ses paramètres **Vrai**
   - **d.** Son nom, les types de ses paramètres et le type de sa valeur de retour **Faux**

3. Si les classes Pomme et Orange dérivent de la classe Fruit et la classe Golden dérive de la classe Pomme alors on peut écrire :
   - **a.** Fruit [] tab = new Orange [10] ; **Vrai**
   - **b.** Fruit [] tab = new Fruit [10] ; **Vrai**
   - **c.** Golden [] tab = new Pomme [10] ; **Faux**
   - **d.** Golden [] tab = new Orange [10] ; **Faux**

- Parmi les quatre propositions ci-dessus, laquelle permet de créer un tableau pouvant contenir oranges, des pommes et des golden ( a, b, c ou d)? **b**