Université Paris Diderot **Programmation Orientée Objet**
L2/L3 **Année 2014-2015**

TD - Séance n°5

Classes Abstraites - Interfaces

1 Cours

Exercice 1 [Vrai/Faux]

- On peut instancier une classe abstraite.
- Une classe abstraite peut contenir un constructeur avec un corps.
- Si `A` est une classe abstraite, et `B` une classe (non abstraite) qui hérite de `A` peut-on écrire le code suivant : `A a = new B()` ?
- une classe abstraite peut contenir des méthodes abstraites et non abstraites.
- une Interface peut avoir un champ privé.
- Toutes les méthodes d'une `Interface` doivent être délarées `public`. Même question pour `abstract`.
- Une interface peut hériter d'une autre interface.

Exercice 2 Pour chacun des code suivants, dire s'il est correct ou pas.

1)
```
public abstract class A{
        abstract int x;                                  2

        public abstract void f(int x);                   4

        public int g(int y);                             6

}                                                        8
```

2)
```
public class A{
        int x;                                           2
}

public abstract class B extends A{
}                                                        6
```

3)
```
public abstract class A{
        public abstract A(int x);                        2
}
```

4)
```
public abstract class A{                                 1
        public abstract void f(int x);
}                                                        3
```

```
public class B extends A{
        public void g(String s){
                System.out.println(s);
        }
}
```

5)

```
public abstract class A{
        public abstract void f(int x);
}

public abstract class B extends A{
        public void g(String s){
                System.out.println(s);
        }
}
```

6)

```
public interface A{
        final static int x;
}

public class C extends A{
        int x;
}
```

7)

```
public interface A{
        final static int x=42;
        void f();
}

public class C implements A{
        private int x;

        public void f(){
                System.out.println(this.x);
                System.out.println(x);
                System.out.println(C.x);
        }
}
```

Exercice 3 [Class `Object`]

1. Qu'affiche le code suivant ? Pourquoi peut on considérer qu'une erreur a été commise ? Comment la réparer ?

```
public class A{
        int x;
        public A(int x){
                this.x=x;
        }

        public boolean equals(A a){
                return this.x==a.x;
        }

        public static void main(String[] args){
```

```
                A a1 = new A(12);
                A a2 = new A(12);

                System.out.println(a1==a2);
                System.out.println(a1.equals(a2));

                ArrayList<A> liste = new ArrayList<A>();
                liste.add(a1);
                System.out.println(liste.contains(a2));
        }
}
```

2. Qu'affiche le code suivant ?

```
public class A{
        int x;

        public A(int x){
                this.x=x;
        }

        public String toString(){
                return "la valeur de x est"+x;
        }

        public static void main(String[] args){
                A a = new A(12);
                System.out.println(a.toString());
                System.out.println(a);
        }
}
```

2 Instruments de Musique

Dans cet exercice, on va tenter modéliser une structure de classes pour des instruments de musique.

Il existe plusieurs façons de classifier les instruments. Une première consiste a différencier selon le procédé qui permet de produire le son. Certains sont dits mécaniques, dans le sens ou le son provient d'une vibration mécanique d'une pièce ou d'une masse d'air (tous les instruments traditionnels, mais aussi la guitare électrique ou les pianos électriques type orgue hammond, rhodes, etc...) et d'autres, dits électroniques, dont le son est produit par un générateur électronique oscillant (les synthétiseurs). Ensuite le son peut être amplifié par une caisse de résonnance ou bien a l'aide d'un microphone. Une guitare électrique, par exemple, n'est pas un synthétiseur, le son provient bien de la vibration d'une corde, mais il est amplifié a l'aide de microphones qui transforment cette vibration en signal électrique.

Pour les instruments mécaniques on les sépare en trois grandes familles,

- Les Cordes, qui peuvent être pincées (guitare), frappées (piano) ou frottées (violon).
- Les Vents divisés en Bois (le son vient de la vibration de l'air sur une pièce mécanique), et Cuivres (le son vient de la vibration des lèvres à l'embouchure).

– Les Percussions (on frappe une peau ou une pièce de metal).

Exercice 4 Fournir une architecture de classes et/ou interfaces pour représenter ceci. Tous les types descendront du type `Instrument`. Donner alors la déclaration (l'en-tête) de la classe `Orgue`, de la classe `Saxophone`, et de la classe `GuitareElectrique`.

Exercice 5 On aimerait bien aussi pouvoir dire qu'un piano à queue, un orgue d'église, un synthétiseur électronique a clavier type piano, appartiennent tous a la famille des claviers. Comment faire cela ?

3 Quizz

Le but de cet exercice est de faire la structure des classes pour un programme permettant de poser des Quizz notés. Chaque questionnaire sera la donnée d'une liste de questions, chacune valant un certain nombre de points.

L'éxecution du programme dans un terminal donnera quelque chose comme :

```
Bonjour et bienvenue pour ce quizz.

1) Quel est le nom du prof de POO ?  (5 points)
Fauconnier                                                    4
Bonne reponse
-------------------------------------------------------       6
2) Quel age a le prof de POO? (a 1 pres) (5 points)
20                                                            8
Mauvaise reponse
-------------------------------------------------------       10
FINI : Vous avez marque 5 points sur 20 possibles
```

On voit dans cet exemple que les deux questions posées sont de types différents. La premiere demande une chaîne de caractères, la deuxième demande une valeur numérique, avec une marge d'erreur autorisée. On peut meme imaginer d'autres types de questions, donc il est naturel ici de penser une structure de classes avec de l'abstraction. On vous propose de définir un type `Question` et un type `Quizz`.

Exercice 6 Au vu de l'exemple ci-dessus et du déroulement du jeu (pour chaque question : on pose la question, on récupère une réponse, on vérifie la validité), de quelles méthodes aura-t-on besoin dans `Question` pour pouvoir écrire une méthode `void poser()` dans une classe `Quizz` sans avoir à se soucier de l'implémentation des questions ? Ecrivez la méthode `poser()` pour un déroulement dans un terminal comme dans l'exemple ci dessus.

Exercice 7 On veut que les sous-types de `Question` permettent de répondre au cahier des charges suivant :

– une question peut avoir une réponse textuelle ou numérique (et dans ce cas une marge d'erreur peut etre autorisée).
– une question peut avoir plusieurs bonne réponses (ex : "citer une ville de plus de 20 millions d'habitants"")

- on peut vouloir poser une question de façon ouverte, ou alors comme dans un QCM avec un certain nombre de propositions.
- on peut décider que selon le contexte une question vaut x points, mais que dans un autre quizz la même question comptera différemment (typiquement pour un QCM on met génréalement des points négatifs pour les mauvaises réponses)

Proposez une structure de classes/interfaces avec les méthodes pour pouvoir répondre à ce cahier des charges.

Exercice 8 On pourrait avoir envie que notre programme se déroule dans une autre interface utilisateur que l'écran du terminal. Il existe par exemple en Java des méthodes très simples (voir la classe `JOptionPane`) pour interagir avec des boîtes de dialogue plutôt qu'avec le terminal. Comment pourrait-on modifier la classe `Quizz` et sa méthode `poser` pour pouvoir se donner la possiblité d'avoir une évolution future de l'interface utilisateur ?