Université de Paris **Programmation Orientée Objet**
L2 **Année 2021-2022**

TD - Séance n°6

Interfaces et classes abstraites

Exercice 1 *Interfaces vs classes abstraites*

1. Peut-on instancier directement une interface ? Une classe abstraite ?
2. Peut-on y mettre un constructeur ? Un constructeur avec un corps ?
3. Est-ce que le code suivant est valide : `A a = new B();`
   — si `A` est une classe abstraite, étendue par la classe `B` ?
   — si `A` est une interface, implémentée par une classe `B` ?
4. Une interface/classe abstraite peut-elle contenir des attributs ? Avec quels modificateurs ? Doivent-ils être initialisés ?
5. Une interface/classe abstraite peut-elle contenir des méthodes abstraites ? non-abstraites ? statique et abstraite ?
6. Une interface peut-elle hériter d'une autre interface ? d'une classe abstraite ?
7. Une classe abstraite peut-elle hériter d'une autre classe abstraite ? d'une interface ?

Exercice 2 *Instruments de musique*

Dans cet exercice, on va tenter de modéliser une structure de classes pour des instruments de musique. Il existe plusieurs façons de les classifier.

Une première consiste à différencier selon le procédé qui permet de produire le son. Certains sont dits mécaniques, dans le sens où le son provient d'une vibration mécanique d'une pièce ou d'une masse d'air (tous les instruments traditionnels) et d'autres, dits électroniques, dont le son est produit par un générateur (les synthétiseurs).

Une seconde consiste à différencier selon la manière dont le son est amplifié. L'amplification peut à nouveau être mécanique, par une caisse de résonnance, ou bien électrique à 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.

Les productions mécaniques de son sont séparées en trois grandes familles,

— Les Cordes, qui peuvent être pincées (guitare), frappées (piano) ou frottées (violon).
— Les Vents divisés en Bois et Cuivres
— Les Percussions (on frappe une peau ou une pièce de bois ou de métal).

Tous les objets descendront d'un type Instrument ; ils ont en commun de pouvoir être joués, on munira donc Instrument d'une méthode abstraite **void** play(), qui devra être implémentée par chaque instrument. Ils ont aussi en commun d'avoir un nom : Instrument disposera donc d'un champ String name. Les instruments à cordes disposent d'un champ indiquant le nombre de cordes. Les instruments à amplification électrique disposent d'une méthode retournant le type de prise.

1. Écrivez les classes et/ou interfaces permettant de modéliser cette hiérarchie. Donner la déclaration (l'en-tête) de la classe Piano, de la classe Saxophone, de la classe GuitareElectrique et de la classe PianoSynthetiseur.
2. On aimerait bien aussi pouvoir dire qu'un piano, un orgue, un piano synthétiseur, appartiennent tous à la famille des claviers. Comment faire cela ?

Exercice 3 *Java for Rocket Scientists*

On a les classes suivantes :

```
public interface Spationef {
    public default int equipageMax() {
       return 0;
    }
    public String typeSpationef();
}
```

```
public interface Propulsion {
    public String typePropulsion();
}
```

```
public abstract class Navette implements Spationef, Propulsion {
    public String typeSpationef() {
       return "navette spatiale";
    }

    public abstract String typePropulsion();
}
```

```
public class NavetteAmericaine extends Navette {
    public int equipageMax() {
      return 8;
    }
    public String typeSpationef() {
      return "navette spatiale americaine";
    }
    public String typePropulsion() {
      return  "par portage, puis ergol liquide"
    }
}
```

```
public class NavetteDiscovery extends NavetteAmericaine {
    public int equipageMax() {
      return 7;
    }
    public String typeSpationef() {
      return "navette Discovery";
    }
}
```

```
public class NavetteRusse extends Navette {
    public String typeSpationef() {
       return "navette spatiale russe";
    }

    public int equipageMax() {
       return 0;// en effet : il s'agissait de la premiere
                // a etre totalement automatisee
    }

    public String typePropulsion() {
       return "par portage, puis ergol liquide";
    }
}
```

```
public class SatelliteMeteo implements Spationef {
    public String typeSpationef() {
       return "satellite meteo";
    }
}
```

Dans le code suivant, dites quelles sont les lignes qui compilent, qui s'exécutent. En cas d'erreur, expliquez pourquoi. Si l'on commente les lignes qui posent problème, qu'affiche l'exécution du code ?

```
  Spationef u1 = new Navette();
  Navette u2 = new NavetteDiscovery();
  Propulsion u3 = new NavetteDiscovery();
  NavetteRusse u4 = new NavetteDiscovery();
  Spationef u5 = new NavetteRusse();
  SatelliteMeteo u6 = new SatelliteMeteo();
```

```
  System.out.println(u2.equipageMax());
  System.out.println(u3.equipageMax());
  System.out.println(u5.equipageMax());
  System.out.println(u6.equipageMax());
  System.out.println(u2.typeSpationef());
  System.out.println(
    ((NavetteAmericaine)u2).typeSpationef());
```

```
8   System.out.println(u3.typePropulsion());
    System.out.println(u6.typePropulsion());
```

```
1   Navette u7 = u5;
    SatelliteMeteo u8 = (SatelliteMeteo) u5;
3   NavetteRusse u9 = (NavetteRusse) u5;
    Spationef u10 = u2;
5   Spationef u11 = u3;
    Spationef u12 = (Navette) u3;
```