جامعة مولاي إسماعيل UNIVERSITÉ MOULAY ISMAÏL المدرسة العليا للأساتذة ÉCOLE NORMALE SUPÉRIEURE	**TP3**: PROGRAMMATION ORIENTÉE OBJET **JAVA** ➢ **Classes abstraites** ➢ **Interfaces**	**LEI (S5)** **PAR MR** : M.MOUKHAFI

Les Interfaces

Une interface est une collection nommée de déclarations de méthodes (sans les implantations). Une interface peut aussi déclarer des constantes.

Exemple:

```
interface  MonInterface {
     public int x=2;
     public  void f();      // methode a implementer. Non static
}
```

NB.
- Une interface ne peut avoir de méthode static. Compile error: `"modifier static not allowed here"`
- Pas de méthode implementée non plus. Compile error: `"interface methods cannot have body"`
- *x* considérée comme `final variable` (ne peut être modifiée.)

Voici une classe qui implante cette interface:

```
class MonImplement implements MonInterface{
    public  void f() {
    // Implementation de f()
        System.out.println("Ok  Interface  ");
    }
}
```

Noter le mot clé ***implements*** .

```
class Test{
    static public void main(String args[]){
        MonImplement o = new MonImplement();
        o.f();
        System.out.println(o.x);
    }
}
```

Compiler les deux classes. Exécuter le teste.

Résultat:

```
Ok Interface
```

Exercices:

1) Faire deux interfaces *A* et *B* et une classe *C* qui implémente ces deux interfaces.
2) Que se passe t-il si les deux interfaces *A* et *B* déclarent une même méthode *f*()? Une même constante *x*? Indication: pour le savoir, créer une classe *Test* qui utilise *f*() et *x*. Pourquoi ne peut-on utiliser *x*? (réponse: La définition de *x* est *dans* les interfaces. Non celle de *f*().)

Les Classes Abstraites

Une classe abstraite est une classe qui peut contenir des méthodes sans implantations, dites méthodes abstraites. L'implantation est laissée aux éventuelles sous classes de la classe abstraite.
Une classe abstraite n'est pas obligée de contenir des méthodes abstraites. Mais une classe qui contient une méthode abstraite ou qui ne fournit pas l'implantation d'une méthode abstraite déclarée dans une classe mère, doit être déclarée abstraite.

Exemple:

```
abstract class MonAbstract {
     public int x=2;            // x variable d'instance (non considere' static)
     abstract public void f();  // methode abstraite a implementer par les sous-classes
     public void g(){
         System.out.println("Methode non abstraite");
     }
}
```

NB. Une méthode sans corps doit toujours être déclarée abstraite. Par ailleurs, une méthode abstraite ne peut être déclarée `static` (pourquoi?)

Implantation:

```
class MonConcrete extends MonAbstract{
    // Noter 'extends' au lieu de implements
    public  void f() {
       // Implementation de f()
        System.out.println("Methode abstraite implementee par sous-classe");
    }
}

class Test{
    static public void main(String args[]){

        MonConcrete o = new MonConcrete();
        o.g();
        o.f();
        o.x++;
        System.out.println(o.x);
    }
}
```

NB. Compiler d'abord la classe `MonAbstract`.

Résultat obtenue:

```
$ java Test
Methode non abstraite
Methode abstraite implementee par sous-classe
3
```

Exercices: Faire une hiérarchie de plus d'une classes abstraites. Vérifier les règles énoncées précédemment.

Classe Abstraite vs Interface

- Une interface ne peut implanter une méthode alors qu'une classe abstraite peut. A.
- Une classe peut implanter plusieurs interfaces mais ne peut avoir qu'une seule superclasse.
- Une interface n'appartient pas à la hiérarchie des classes. Des classes sans rapport entre elles peuvent implanter la même interface.
- Plus important, avec Interface on est sûr qu'un objet a implementé une méthode (il ne peut différer cette implantation).
- Une interface peut faire *extends* de +r autres interfaces
  (Le vérifier sur un exemple `interface C extends A , B {}`)
-

Extrait de la documentation d'oracle

- *Implementing an interface allows a class to become more formal about the behavior it promises to provide. Interfaces form a contract between the class and the outside world, and this contract is enforced at build time by the compiler. If your class claims to implement an interface, all methods defined by that interface must appear in its source code before the class will successfully compile.*
  (http://java.sun.com/docs/books/tutorial/java/concepts/interface.html)