classe abstraite & Interface

Module : POO
Enseignante: BOUZAROURA Ahlam
Année universitaire: 2019-2020

Méthode abstraite

Une méthode déclarée **abstract** **ne peut être exécutée**. En fait, elle **n'a pas d'existence réelle.** Sa déclaration indique simplement que les classes dérivées **doivent la redéfinir.**

Les méthodes **abstract** présentent les particularités suivantes :

- Une classe qui contient une méthode abstract doit être déclarée abstract.
- Les méthodes abstract n'ont pas d'implémentation. Leur déclaration doit être suivie d'un point-virgule.

```
Abstract Public class formes{
Abstract Public float surface();
Abstract Public float perimetre();
}
Public class rectangle extends formes{
Public float surface(){
return Longueur*Largeur
}
Public float perimetre{
return 2*(Longueur+Largeur)
};
}
```

Classe abstraite

Une classe contenant au moins une méthode abstraite est appelée une classe **abstraite** et cela doit être explicitement précisé dans la déclaration avec : **abstract class.**

Les classes abstract présentent les particularités suivantes :

- Une classe abstract ne peut pas être instanciée.
- Une classe peut être déclarée abstract, même si elle ne comporte pas de méthodes abstract.
- Pour pouvoir être instanciée, une sous-classe d'une classe abstract doit redéfinir toute les méthodes abstract de la classe parente.
- Si une des méthodes n'est pas redéfinie de façon concrète, la sous-classe est elle-même abstract et doit être déclarée explicitement comme telle.

Interface-1-

Une **interface** est un prototype de classe. Elle définit la signature des méthodes qui doivent être implémentées dans les classes construites à partir de ce prototype.

Une interface est une “classe” **purement abstraite** dont toutes les méthodes sont abstraites et publiques. Les mots-clés **abstract** et **public** sont optionnels.

Une interface peut avoir des méthodes et des variables comme la classe, mais les méthodes déclarées dans l'interface sont par défaut abstraite (seule signature de méthode, pas de corps). De plus, les variables déclarées dans une interface sont **public, static** et **final** par défaut.

Interface-2-

Les interface présentent les particularités suivantes :

- Comme les classes abstraites, les interfaces ne sont pas instanciables.
- Une interface ne possède pas d'attribut instance.
- Une interface n'a pas de constructeur.
- Toutes les méthodes sont abstract. Elles ne définissent pas les mécanismes internes
- Tout leurs attributs de classe sont des constante, définit comme des attributs déclarées comme public static final et ayant une valeur constante . :

public static final float PI = 3.14;

Utilité des interface

Puisque les méthodes dans les interfaces sont abstraites, elles doivent être mises en œuvre par la classe avant d'y accéder. La classe qui implémente l'interface doit implémenter toutes les méthodes de cette interface. En outre, le langage de programmation java ne supporte pas l'héritage multiple, en utilisant des interfaces, nous pouvons y parvenir car une classe peut mettre en œuvre plus d'une interface.

Java n’autorise que l’héritage simple. Une classe ne peut avoir qu’une seule superclasse. Une certaine forme d’héritage multiple est obtenue grâce aux interfaces.

Déclaration des interfaces

Interfaces are declared by specifying a keyword interface:

```
interface MonInterface
{
public void methode1();
public void methode2();
}
```

Implémentation d'Interface

Lorsque une classe **implémente** une interface elle doit fournir le corps de toutes les méthodes qui sont déclarées en interface.

Syntaxe:

```
class Maclasse implements inter1,inter2
class Maclasse implements MonInterface
```

Remarque importante:

la classe implémente l'interface mais une interface **extends** une autre interface.

```
Interface MonInterface1 extends MonInterface2
```

```
interface MonInterface{
public void méthode1();
public void méthode2();
}
class Maclasse implements MonInterface{
public void méthode1(){
System.out.println("implémentation méthode1");
}
public void méthode2(){
System.out.println("implémentation méthode2");
}
public static void main(String arvg[]){
MonInterface obj;
obj = new Maclasse();
obj. méthode1();
}
```

résultat

```
implémentation méthode1
```

Différence entre une classe abstraite et un interface:

Classe abstraite	interface
peut extend une seule classe ou une classe abstraite à la fois	Peut extend plusieurs interface à la fois
La classe abstraite peut avoir à la fois des méthodes abstraites et concrètes	Interface ne peut avoir que des méthodes abstraites
Dans la classe abstraite, le mot-clé abstract est obligatoire pour déclarer une méthode	Dans un d'interface, le mot-clé abstract est facultatif de déclarer une méthode

```
Class Exemple1{
public void affiche1(){
System.out.println("affiche1 méthode");}
}
abstract class Exemple2{
public void affiche2(){
System.out.println("affiche2 méthode");}
}
abstract class Exemple3 extends Exemple1{
abstract void affiche3();}
}
class Exemple4 extends Exemple3{
public void affiche3(){
System.out.println("affiche3 méthode");; }
}
class Test{
public static void main(String args[]){
Exemple4 obj=new Exemple4();  obj.affiche3();}
}
```

```
affiche3 méthode
```

```
interface Exemple1{
public void affiche1();}
interface Exemple2 {
public void affiche2();}
interface Exemple3 extends Exemple1,Exemple2{}
class Exemple4 implements Exemple3{
public void affiche1(){
System.out.println("affiche1 méthode");}
public void affiche2(){
System.out.println("affiche2 méthode");}}
class Exemple5 implements Exemple3{
public void affiche1(){
System.out.println("affiche1 méthode exemple5");}
public void affiche2(){
System.out.println("affiche3 méthode");}
}
```

```
class Exemple4 implements Exemple3{
public void affiche1(){
System.out.println("affiche1 méthode");}
public void affiche2(){
System.out.println("affiche2 méthode");}}
class Exemple5 implements Exemple3{
public void affiche1(){
System.out.println("affiche1 méthode exemple5");}
public void affiche2(){
System.out.println("affiche3 méthode");}
class Test{
public static void main(String args[]){
Exemple4 obj4=new Exemple4();
obj4.affiche1();
Exemple5 obj5=new Exemple5(); obj5.affiche2();
}
```

```
affiche1 méthode
affiche3 méthode
```