Classe abstraite et interface

1. Classe abstraite

1. Pourquoi ?

- Dans certains cas une classe contient une méthode dont son implémentation est inconnue, ce sont les sous classes qui ont une implémentation de cette méthode (exemple : FormeGeometriqueFermee a une méthode getSurface() mais sa formule est inconnue (les sous classe Rectangle, Carre et Cercle ont chacune, une implémentation de cette méthode).
- Dans d'autres cas une classe est une classification (ou groupement d'autre classe) mais cette classe n'a pas d'instances directes mais elle peut avoir des instances de ses sous classes (exemple :Animal est une classe générique pour tous les animaux mais il n'y a pas une instance directe de Animal, ce sont les sous classes Chien, Chat et Perroquet qui ont des instances.
- Pour modéliser ses deux types de cas de figure on utilise les classes abstraites.

2. Définition

A. Méthode abstraite

Une méthode abstraite est une méthode qui a une signature (visibilité, type de retour, nom et paramètres) mais pas un corps. La visibilité d'une méthode abstraite ne peut être que public ou protected.

B. Classe abstraite

Une classe abstraite est une classe qui a au moins une méthode abstraite ou une classe qui ne doit pas avoir des instances directes (même si elle n'a aucune méthode abstraite).

3. Syntaxe

```
public abstract class NomClasse {
      visibilité abstract typeRetour nomMethode();
}
ou
public abstract class NomClasse {

}
```

Exemple :

```
public abstract class Animal {
}
public abstract class FormeGeoFermee {
      public abstract float getSurface();
}
```

4. Déclaration

On peut déclarer des instances dont le type est une classe abstraite.

Exemple :

```
Animal a ;
FormeGeoFermee fgf ;
```

5. Instanciation

On ne peut pas instancier une classe abstraite.

```
a = new Animal();              //erreur de compilation
fgf = new FormeGeoFermee();    //erreur de compilation
a= new Perroquet();            //correct
fgf=new Rectangle();           //correct
```

2. Interface

1. Pourquoi ?

- Dans certains cas on a besoin de modéliser un comportement commun à plusieurs classes (exemple : Dessinable : afficher(), deplacer(), agrandir(), dubliquer())
- Comme java ne supporte pas l'héritage multiple, l'utilisation d'un héritage et d'une implémentation d'un comportement peut se rapprocher à un héritage multiple.

2. Définition

Une interface est une description abstraite d'un ensemble de comportement. Toutes les méthodes d'une interface sont abstraites et elle peut avoir seulement des attributs statiques.

3. Syntaxe

```
public interface NomInterface {
      visibilité abstract typeRetour nomMethode();
}
ou
public interface NomInterface {
       visibilité typeRetour nomMethode();
}
```

Exemple :

```
public interface Dessinable {
      public abstract float afficher();
      public abstract float deplacer();
      public abstract float agrandir(int coef);
      public abstract float dupliquer();
}
```

4. Relation Interface/Interface

Une interface peut être une sous-interface d'une autre.

```
public interface I1 {    }
public interface I11 extends I1 {   }
```

5. Relation Classe/Interface

Une classe peut implémenter une interface et dans ce cas elle hérite toutes ses méthodes abstraites.

```
public interface Dessinable {  }
public class Etoile extends FormeGeoFermee implements Dessinable{
      // Etoile doit redéfinir getSurface() héritée de FormeGeoFermee
      // Etoile doit redéfinir afficher(),deplacer(),agrandir(int coef) et
      // dupliquer() héritée de FormeGeoFermee
}
```

6. Déclaration

On peut déclarer des instances dont le type est une interface.

Exemple :

```
Dessinable d;
```

7. Instanciation

On ne peut pas instancier une classe interface.

```
d = new Dessinable();              //erreur de compilation
d= new Etoile();                   //correct
```