Classes abstraites

Classe “normales”

Une classe “normale”

- ✔ déclare un comportement : déclare des méthodes publiques.
- ✔ implémente un comportement : ses méthodes publiques ont du code.

Classes abstraites

Une **classe abstraite** déclare un comportement mais se différencie d'une classe "normale" :

- ✗ certaines de ses méthodes peuvent ne pas être implémentées. Les méthodes qui ne sont pas implémentées sont appelées des **méthodes abstraites** déclarées avec le mot-clef **abstract** .
- ☛ Pour définir une classe abstraite, on utilise le mot-clef **abstract**

> **Attention**. Une classe abstraite est une classe :
>
> - Une classe abstraite peut déclarer des attributs comme une classe "normale".
> - Une classe abstraite doit posséder au moins un constructeur comme une classe "normale" mais **on ne peut pas créer d'instances d'une classe abstraite.**
> - Une classe contenant une méthode abstraite est obligatoirement abstraite.

Classes abstraites

```
abstract public class Forme {
  private Color couleur;
  public Forme(Color couleur) {this.couleur =
    couleur;}
  public Color couleur() {return couleur;}
  abstract public double surface();
}
```

Par opposition aux méthodes abstraites, on appelle méthode concrète une méthode “normale” (qui a du code).

Héritage

Une classe qui hérite d'une classe abstraite et qui implémente toutes les méthodes abstraites héritées est une classe "normale" :

```
public class Cercle {
   print double rayon;
   public Cercle(Color couleur,double rayon) {
     super(couleur);
     this.rayon = rayon;
   }
   @Override
   public double surface() {
     return Math.PI * rayon * rayon;}
}
```

Types possible d'une référence

S'il n'est pas possible de créer des instances d'une classe abstraite ni d'une interface, le type d'une référence peut être une classe "normale" ou une classe abstraite.

```
Forme f;
```

La référence `f` peut être utilisée pour enregistrer l'identité d'une instance d'**une sous-classe** de la classe `Forme`.

```
f = new Cercle(Color.RED,4.0);
```

Tableaux / ArrayList

Le type des références d'un tableau ou d'une arraylist peut être une classe abstraite :

```
Forme[] tableau;
AttayList<Forme> arraylist;
```

La création du tableau ou de l'arraylist est fait comme avec des références dont le type est une classe "normale" :

```
tableau = new Forme[5];
arraylist = new ArrayList<Forme>();
```

Héritage

Une classe qui hérite d'une classe abstraite et qui n'implémente pas **toutes** les méthodes abstraites héritées doit être déclarée abstraite :

```
abstract class Quadrilatere extends Forme {
  public Quadrilatere(Color couleur) {
    super(couleur);
  }
  /*
  la classe n'implémente pas la méthode abstraite
  héritée surface()
  */
  }
```

Héritage

Une classe abstraite peut-être une sous-classe d'une classe normale :

```
 public class Person {
  private String name;
  public Person(String name) {this.name = name;}
  public String name() {return name;}
}

abstract public class AbstractStudent extends
    Person {
  public AbstractStudent(String name) {
    super(name);}

  public abstract String grade();
}
```