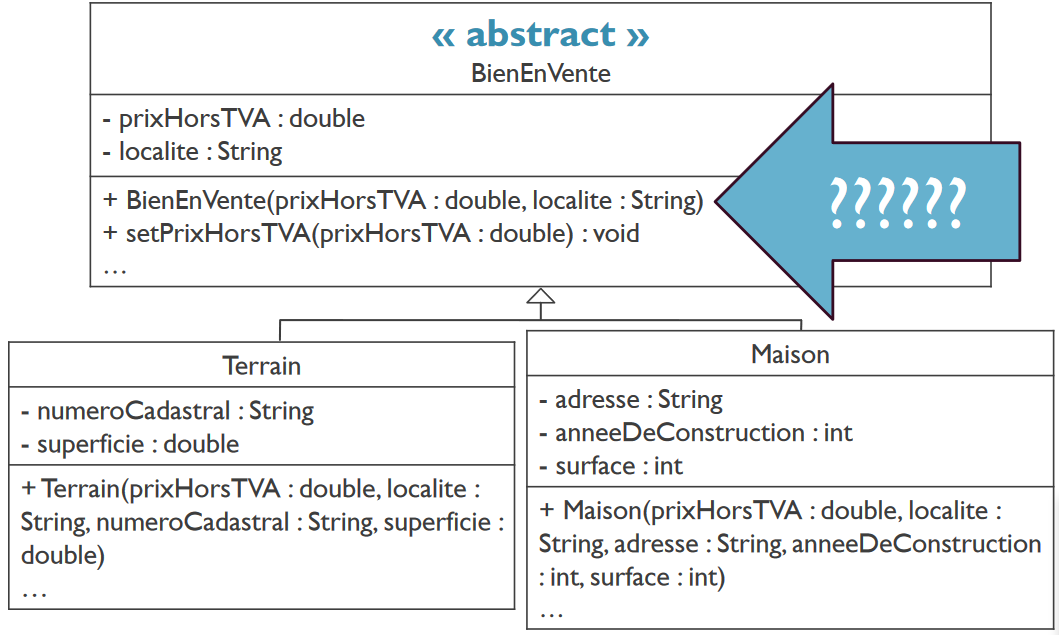
Classe abstraite :

* Pas destinée à être instanciée
* Permet de définir des propriétés (attributs et méthodes)   
  communes à différentes classes une seule fois.

1. UML :



1. Java :

Caractérisé par le mot clé  « abstract »



1. Remarques :
   * + Impossible d’invoquer un constructeur d’une classe abstraite   
       autrement qu’en utilisant l’instruction super(...)
     + Pour le reste, tout fonctionne comme dans l’héritage classique.
     + Une classe peut être abstraite même si elle hérite d’une   
       classe concrète

Méthode abstraite :

* Une méthode qui n’a pas de code, uniquement l’entête avec paramètres, nom de la méthode

1. UML :

Une image contenant table

Description générée automatiquement

1. Java :

Une image contenant texte

Description générée automatiquement

1. Remarques :
   * Si une classe contient une méthode abstraite, cette classe doit aussi être abstraite

Remarques générales :

* Les méthodes static, final ou private ne peuvent pas être abstraites.
* Une classe qui ne fournit pas une implémentation(dans la classe elle-même ou par héritage) de toutes les méthodes abstraites, héritées ou non, doit être elle-même abstraite.
* Dans une méthode d’une classe, on peut faire appel à une méthode abstraite de la classe.

Interface :

* Contient une liste de méthodes non implémentées
* Contient une liste de constantes

Mais

* Elle peut aussi contenir une liste de méthodes implémentées par défaut
* Une liste de méthodes de classe
  1. UML :

Une image contenant texte

Description générée automatiquement

* 1. Java :

Caractérisé par le mot clé « interface »



* 1. Remarques :
* Contient pas d’attribut d’instance
* Contient pas de constructeur
* Toutes les méthodes d’une interface sont public
* Toutes les variables sont public static final

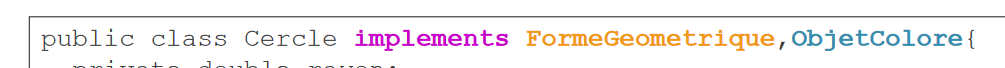
Pour implémenter la classe interface en Java on utilise le mot clé implements :



Tandis que en UML, c’est une fléche en pointillé vide.



Une classe peut implémenter plusieurs interfaces



Utilisation :

* On peut déclarer des variables de type interface
* Pour instancier une variable de type interface, il faut utiliser une classe qui implémente l’interface
* On ne peut utiliser que les méthodes qui sont déclarées dans l’interface

Les méthodes equals et hashCode sont à écrire dans chaque classe implémentant l’interface

Une image contenant table

Description générée automatiquement

Interface Iterable :

* Interface implémentée lorsqu’on a une classe possédant une «liste» d’objets si on veut pouvoir utiliser un for each pour les parcourir.
* ′ Méthode imposée par l'interface Iterable<T>
* Permet de parcourir les objets qui se trouvent dans la liste

1. UML :

Une image contenant texte

Description générée automatiquement

1. Java :

Une image contenant texte

Description générée automatiquement

1. Méthodes :
   * hasNext() : Signale s’il reste un élément dans la liste qui n’a pas encore été consulté, retourne un boolean
   * next() : Renvoye l’élément suivant, présent dans la liste parcourue par itérateur, retourne un objet
   * remove() : Supprimer l’élément de la collection qui a été envoyé par le dernier next(), retourne void