Le modificateur `final`

Ce modificateur permet d'indiquer les éléments du programme qui ne doivent pas être modifiés/redéfinis/étendus

- Possible pour les classes, méthodes, attributs, variables
- Utile surtout pour les variables
- Moins courant pour les méthodes et les classes

Méthodes finales

Si l'on ajoute `final` à une méthode :

☞ Impossible de la redéfinir dans une sous-classe

Exemple : on aimerait toujours appliquer la méthode `vieillir` de `Personnage`

```
class Personnage
{
  //...
   final void vieillir() {
      --dureeVie;
   }
}
```

☞ message d'erreur du compilateur si la classe `Sorcier` essaie de redéfinir la méthode `vieillir`

Classes finales

Si l'on ajoute `final` à une classe :

☞ Impossible d'étendre la classe par une sous-classe

Exemple : on aimerait que la classe `Sorcier` n'ait jamais de sous-classe

```
final class Sorcier extends Magicien {
   //...
}
```

```
class MageNoir extends Sorcier { ..}
// illicite!!
```

Classes et méthode finales

Les méthodes et classes finales peuvent être à priori « agaçantes » :

- Exemple: la classe prédéfinie `String` est finale
- Aucune possibilité de définir

  `class MyString extends String`

  afin d'améliorer certaines méthodes par redéfinition!!

  ☞ Mais, permet de fixer une fois pour toute le comportement d'une classe ou méthode

Variables finales et objets référencés (1)

Si l'on ajoute `final` à une variable d'instance, une variable locale ou un paramètre :

- ☞ il devient impossible de lui affecter une valeur plus d'une fois

Un attribut `final` peut être initialisé dans le constructeur mais ne doit plus être modifié par la suite

Attention : `final` empêche l'affectation d'une nouvelle valeur à une variable, mais n'empêche pas de modifier l'éventuel objet référencé par cette variable :

Un exemple ...

Variables finales et objets référencés (2)

```
class Conteneur {
   private int valeur;
   public void setValeur(int val) { valeur = val; }
}

class Test {
   public static void main(String[] args) {
      Conteneur c = new Conteneur();
      c.setValeur(42);
      modifier(c);
   }
   static void modifier(final Conteneur c) {
      c.setValeur(-1); // modifie l'objet référencé !!
      //c = new Conteneur(); //FAUX
   }
}
```