Conception par contrat

- Java possède différentes techniques de mise en place de contrats pour les classes

Exceptions et Assertions

- Les assertions permettent de vérifier des propriétés
- Le programme est interrompu en cas de non respect d'une assertion : `java.lang.AssertionError`
- Elles peuvent être inhibées et aident le débogage

Assertions

Syntaxe

```
assert conditionQuiDoitEtreVraie ;
assert conditionQuiDoitEtreVraie : objet ;
```

Exemples

```
assert (indice >= 0 && indice < 8) ;
assert (estTrie(tableau)) ;
assert (indice >= 0 && indice < 8) : '' ligne inexistante'';
```

Assertions

Bonnes pratiques

- invariant d'algorithme
  - *à la fin de l'itération i dans la recherche du minimum la variable min contient la plus grande valeur rencontrée entre 0 et i*
- invariant de flux
  - *ce point de programme ne peut être atteint*
- pré/posconditions
  - *à la fin d'une fonction de tri le tableau est trié*
- invariants de classe
  - *une voiture a 4 roues*

Assertions

Mauvaises pratiques

Il ne faut pas …

- vérifier les paramètres d'une méthode à l'aide d'assertions car ces paramètres viennent d'ailleurs et peuvent vraisemblablement être faux (donc ne pas sortir en échec, traiter plutôt le problème)
- vérifier les résultats d'interactions avec un utilisateur (vont souvent comporter des erreurs, à traiter)
- créer d'effets de bord, de modification de l'état du programme avec les assertions (dans aucune des deux expressions)

Assertions

Le mécanisme est désactivé par défaut en raison de son coût

Activation `-ea`

sur certaines classes ou packages

```
java –ea:monPackage1 AppliTest
```

Désactivation `-da`

Désactivation sur certaines classes ou packages

```
java –ea:monPackage1
     -da:monPackage1.Piece AppliTest
```

Assertions

Pour les activer sous eclipse ..