# Exploiter les annotations à l'exécution

## Objectifs

— Définir des annotations

— Utiliser des annotations pendant l'exécution du programme

**Exercice 1 : Lanceur de JUnit 4**

On se propose de développer un équivalent de JUnit 4 simplifié en partant d'un équivalent de JUnit 3 qui s'appuie sur l'introspection.

**1.1.** Définir les annotations `Avant`, `Apres` et `UnTest`.

**Attention :** Ce sont des annotations qui ressemblent à celles de JUnit, mais ce ne sont pas celles de JUnit. Il n'y a pas à utiliser JUnit : on le refait.

**1.2.** Adapter les programmes de test fournis (il est conseillé de commencer par adapter un premier test et ne faire les autres que plus tard). On changera les noms des méthodes de test (on supprime 'tester'), `preparer` et `nettoyer` et on vérifiera que le lanceur actuel ne fonctionne plus.

**1.3.** Reprendre la classe `LanceurIndependant` qui prend en arguments les noms des classes de test et les exécute.

**Remarque :** Pour exploiter les annotations pendant l'exécution, on utilise l'API d'introspection avec les méthodes supplémentaires liées aux annotations telles que `getAnnotation` qui prend en paramètre le type de l'association cherchée.

**1.4.** Ajouter un attribut sur l'annotation Test qui permet de décider si le test doit être exécuté ou non. L'attribut s'appellera `enabled` et sa valeur par défaut sera vrai. Modifier le Lanceur pour qu'il ignore les tests qui ne sont pas activé (`enabled` à faux).

**Remarque :** Une fois l'annotation trouvée, on peut utilisée les méthodes qui donnent accès aux attributs définis sur l'annotation.

**1.5.** Ajouter sur l'annotation `Test` un attribut `expected` qui prend comme valeur une exception et signale que pour réussir le test doit lever cette exception.