

# Test Driven Development ou la programmation pilotée par les tests en Java

## Référence : TDDJ

Les objectifs de test ambitieux fixés en début de projet ne résistent pas aux fréquents retards. Le développement piloté par les tests (TDD) propose de garantir radicalement le niveau de l'effort de test : les tests systématiques sont écrits avant le code à tester ! Il existe autour de Java de nombreux outils supportant le TDD. Le retour sur investissement peut être important (qualité du code, non régression, évolutivité, maintenabilité). Encore faut-il éviter les écueils d'une mise en oeuvre inadaptée de la démarche.

Après ce cours, vous saurez utiliser des tests automatisés comme moyen de spécification, de conception et bien sûr de test. Vous saurez développer des tests pour du code existant hérité. Vous connaîtrez les techniques et outils, tels les bouchons, pour développer en TDD du code comprenant des bases de données et des IHM.

Vous verrez le TDD à l'oeuvre au travers d'exemples concrets. Une étude de cas réaliste vous permettra d'acquérir les réflexes du TDD, d'aborder les divers problèmes qui se posent aux développeurs en TDD et de mettre en oeuvre les bonnes pratiques, des plus simples aux plus élaborées.

### **Vous allez apprendre à :**

- Découvrir les principes fondamentaux et les bonnes pratiques du TDD
- Utiliser JUnit dans une approche TDD
- Mettre en oeuvre les divers types de tests automatisés au sein d'un processus agile
- Utiliser des techniques avancées d'écriture de tests
- Mettre en oeuvre le TDD en présence de code hérité (legacy)
- Appliquer le TDD dans des contextes spécifiques (bases de données, IHM)
- Pratiquer le Refactoring d'un code développé en TDD

**Durée :** 3.0 jours - 21.0 heures

**Audience :**

Développeurs Java, responsables tests, chefs de projet, responsables qualité

**Pré-requis :**

Pratique de la conception objet

Pratique du développement avec Java ou avoir suivi le cours JOD ou IJOP

**Méthode pédagogique :** 60% de travaux pratiques

**Programme détaillé :**

**Le test dans le processus de développement**

- Processus, qualité, tests
- Tests et agilité
- Tests et spécifications

**Tests automatisés avec le Framework JUnit**

- Le besoin d'un Framework de test
- Le Framework JUnit
- Bonnes pratiques associées à JUnit

**Principes fondamentaux du TDD**

- Le cycle de développement du TDD
- Test First
- Refactoring

**Stratégies de Test First**

- Tests comme moyen de spécification
- Tests comme moyen de conception
- Tests indépendants

**Écrire du code testable**

- Composition plutôt qu'héritage
- Éviter le code statique
- Isoler les dépendances
- Inverser les dépendances

**Mocks et doublures**

- Quand les utiliser
- Types de Mock
- Bibliothèques de Mocks

**Techniques d'écriture de tests**

- Tests basés sur la responsabilité

- Tests basés sur l'implémentation
- Styles de TDD

**Couverture des tests**

- Les axiomes sur la couverture des tests
- Types de couverture
- Combien de tests faut-il écrire ?
- Tests de régression
- Outils de couverture

**Test de code hérité**

- Qu'est-ce que du code hérité ?
- Cycle d'évolution du code hérité

**Tests fonctionnels avec Fit et FitNesse**

- Tests fonctionnels et TDD
- Écriture de tests fonctionnels exécutables avec FitNesse

**TDD dans des situations particulières**

- Tests en présence de bases de données
- Tests d'interface utilisateur

**Le Refactoring en TDD**

- Quelques "mauvaises odeurs"
- Techniques de Refactoring en TDD