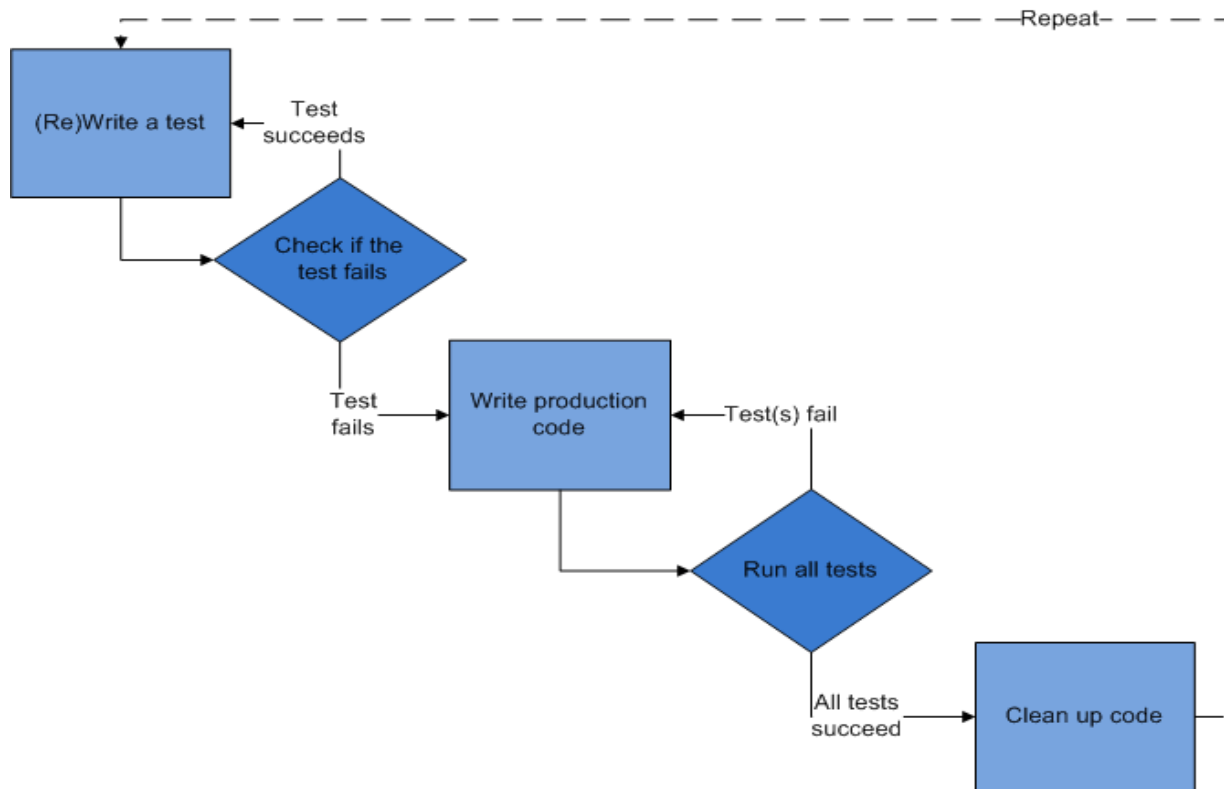


Test Driven Development

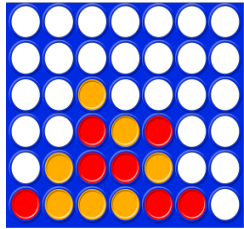
Le Test Driven Development (TDD) ou en français « développement piloté par les tests » est « une technique de développement de logiciel qui préconise d'écrire les tests unitaires avant d'écrire le code source d'un logiciel » (définition [wikipedia](https://fr.wikipedia.org/wiki/Test_driven_development)). Le cycle de développement « idéal » du Test-Driven Development est le suivant :



Cette façon de faire implique que les testeurs et les développeurs travaillent de concert, de façon à ce que les tests automatiques ne valident pas simplement l'exécution sans erreur des différentes branches du code (concept de couverture des tests) mais incorporent au maximum les réels critères métier d'acceptation du logiciel.

Une définition moins rigoureuse, mais plus largement appliquée au développement logiciel, consiste en l'automatisation et en l'intégration des tests d'acceptation (logique métier), de qualité et de non régression dans les cycle de développement des composants, à l'aide d'outils comme Junit, Cobertura, Sonar, Jenkins (pour les développements java).

Dans ce concours, nous vous proposons d'appliquer le Test-Driven Development à un cas concret (et volontairement assez simple).



Description

L'exercice à réaliser consiste à développer selon les principes TDD le moteur d'un jeu de Puissance4© (cf. [wiki](#)) capable de stocker l'état de la partie, de valider un coup, et de déterminer si le jeu est terminé et qui est le vainqueur.

Pour ce faire, vous devrez écrire une classe **Puissance4Impl** qui implémente l'interface **Puissance4** ci-dessous, qui vous est fournie :

```
package fr.ippon.contest.puissance4;
/**
 * Moteur pour le jeu puissance 4.
 */
public interface Puissance4 {

    public enum EtatJeu {
        EN_COURS, ROUGE_GAGNE, JAUNE_GAGNE, MATCH_NUL;
    }

    /**
     * Vide la grille de jeu, et tire au sort le joueur qui commence.
     */
    public void nouveauJeu();

    /**
     * Charge une partie en cours.
     * @param grille une grille de puissance 4 (6 lignes x 7 colonnes).
     * Une case vide est représentée par le caractère '-',
     * Une case occupée par un jeton rouge, par le caractère 'R'
     * Une case occupée par un jeton jaune, par le caractère 'J'.
     * @param tour le joueur dont c'est le tour (J ou R)
     * @throws IllegalArgumentException si la grille est invalide,
     * ou si tour ne vaut ni J ni R.
     */
    public void chargerJeu(final char[][] grille, final char tour);

    /**
     * @return l'état dans lequel est le jeu :
     * EN_COURS, ROUGE_GAGNE, JAUNE_GAGNE, MATCH_NUL
     */
    public EtatJeu getEtatJeu();

    /**
     * @return le joueur dont c'est le tour : 'R' pour rouge, 'J' pour jaune.
     */
    public char getTour();

    /**
     * @param ligne de 0 à 5
     * @param colonne de 0 à 6
     * @return la couleur - R(ouge) ou J(aune) - du jeton occupant la case
     * aux coordonnées saisies en paramètre (si vide, '-')
     * @throws IllegalArgumentException si les coordonnées sont invalides.
     */
}
```

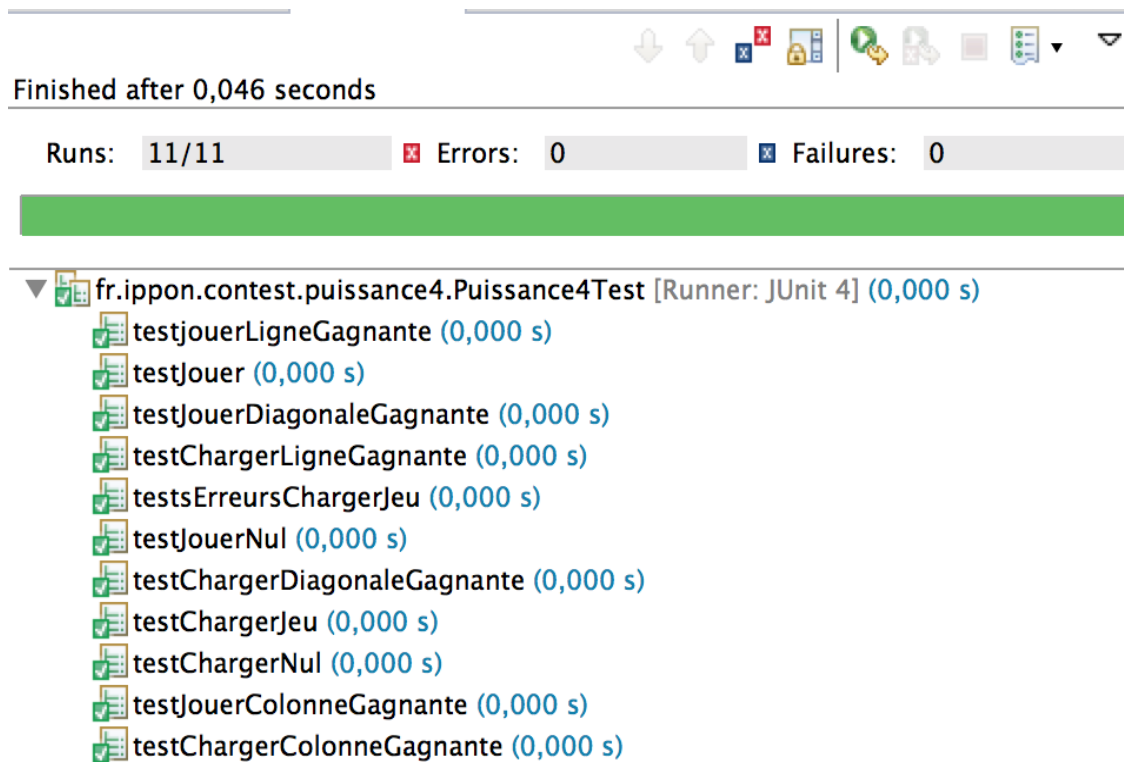
```

    public char getOccupant(int ligne, int colonne);

    /**
     * Un "coup" de puissance 4.
     * @param colonne numéro de colonne où le joueur courant fait glisser son jeton
     * (compris entre 0 et 6)
     * @throws IllegalStateException
     * si le jeu est déjà fini, ou si la colonne est pleine
     * @throws IllegalArgumentException si l'entier en paramètre est > 6 ou < 0.
     */
    public void jouer(int colonne);
}

```

Conformément aux principes TDD, votre classe doit permettre de passer l'ensemble des tests unitaires définis par l'équipe fonctionnelle dans la classe **Puissance4Test** - qui vous est également fournie :



The screenshot shows a JUnit test runner interface. At the top, there is a toolbar with various icons. Below the toolbar, it says "Finished after 0,046 seconds". A summary bar shows "Runs: 11/11", "Errors: 0", and "Failures: 0". Below this is a green progress bar. The test results are listed under the heading "fr.ippon.contest.puissance4.Puissance4Test [Runner: JUnit 4] (0,000 s)". There are 11 test cases, all of which passed, each with a green checkmark icon and a duration of (0,000 s):

- testjouerLigneGagnante (0,000 s)
- testjouer (0,000 s)
- testjouerDiagonaleGagnante (0,000 s)
- testChargerLigneGagnante (0,000 s)
- testsErreursChargerJeu (0,000 s)
- testJouerNul (0,000 s)
- testChargerDiagonaleGagnante (0,000 s)
- testChargerJeu (0,000 s)
- testChargerNul (0,000 s)
- testJouerColonneGagnante (0,000 s)
- testChargerColonneGagnante (0,000 s)

Vous serez évalués sur la cohérence de votre applicatif avec ces spécifications (il doit passer un maximum des tests unitaires fournis – tous étant le niveau recherché), sur la lisibilité du code et sa rapidité d'exécution (le plus petit nombre d'instructions pour évaluer le jeu étant le meilleur).

Mode d'emploi :

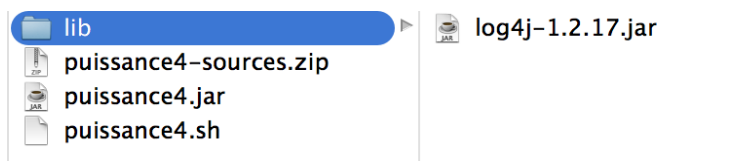
Vous pourrez récupérer le squelette du programme à développer, comprenant l'interface Puissance4 et le jeu de tests Puissance4Test, sur le GitHub de Ippon Technologies :

<https://github.com/ippontech/devoxx-tdd-puissance4>

Pour plus d'informations sur Git, vous pouvez consulter la formation Git proposée par Ippon et disponible sur SlideShare :

http://fr.slideshare.net/ippontech/formation-git-gratuite-par-ippon-2014?qid=51bdeaa2-c0b9-48d5-86b2-6712a9754c9f&v=qf1&b=&from_search=1

Pour construire le logiciel, nous avons mis à votre disposition un artefact Maven qui construit un zip comprenant votre exécutable au format jar, toutes ses dépendances – dans un sous-répertoire « lib » - , un script permettant de lancer votre programme dans une console Unix, ainsi qu'un zip contenant toutes les sources :



Vous trouverez sur internet une vaste documentation sur Maven – la commande à exécuter (sur le dossier où se trouve le fichier pom.xml) pour construire le composant est :

```
mvn clean package
```

Pour soumettre votre réponse, et gagner une place pour le Devoxx, vous devrez nous envoyer ce zip par mail à jmcaillaud@ippon.fr