Ecole National des Sciences Appliqués Oujda
Usine Logiciel: Tests unitaires automatisés
TP-1 : **FrameWork JUnit 3.**

I. Partie 1 (JUnit 3) :

1. Créer un projet Java sous le nom TP1
2. Créer le package : ma.coursEnsao.tp1.dev qui va contenir les classes du TP
3. Créer le package : ma.coursEnsao.tp1.junit qui va contenir les classes de test
4. Créer la classe ***MathTools*** qui contient la méthode ***carre()*** qui prend comme argument un *double* et retourne son carré.
5. Créer la classe ***MathToolsTest*** héritant de la classe ***TestCase*** de Junit
6. Redéfinir les deux méthodes ***setUp()*** *et* ***tearDown()***
7. Préciser le corps de la méthode de test de la méthode **carre()**
8. Exécuter le teste de la méthode **carre().**

II. Partie 2 (JUnit 3) :

Un projet scientifique se compose de deux sous équipes, une de Dév. et autre de teste.

Après le développement de la classe **Calculatrice.java** (par l'équipe de Dév.) , il faut automatiser les tests d'un ensemble des scénarios rédigés par l'équipe de teste.

La classe **Calculatrice.java** permet d'effectuer des opérations simple (+, -, * et /) entre deux entiers saisis par l'utilisateur, elle contient trois méthodes publiques :

- `public void initialize() : pour initialiser la calculatrice.`
- `public String getContenuEcran() : retourne le contenu affiché dans l'écran de la calculatrice.`
- `public void taperTouche(char touche) : méthode appelée lorsque un bouton de la calculatrice est pressé. Elle prend comme argument la valeur du bouton utilisé.`

Par exemple, pour effectuer l'opération 321 + 49, le système doit passer par les appelles suivantes :

```
1.  initialize();        // vider l'écran
2.  taperTouche('3');
3.  getContenuEcran(); // retourne « 3 »
4.  taperTouche('2');
5.  getContenuEcran(); // retourne « 32 »
6.  taperTouche('1');
7.  getContenuEcran(); //  retourne « 321 »
8.  taperTouche('+');
9.  getContenuEcran(); // retourne « + »
10. taperTouche('4');
11. getContenuEcran(); // retourne « 4 »
12. taperTouche('9');
13. getContenuEcran(); // retourne « 49 »
14. taperTouche('=');
15. getContenuEcran(); // retourne « 370 »
```

La liste des scenarios rédigé par l'équipe de teste est :

N°	objectif	description	Résultats attendus
1	tester l'addition	effectuer l'opération suivante 56 + 2011	la valeur affichée dans l'écran est 2067
2	tester la soustraction	effectuer l'opération suivante 3040 - 1020	la valeur affichée dans l'écran est 2020
3	tester la multiplication	effectuer l'opération suivante 90 * 11	la valeur affichée dans l'écran est 990
4	tester la division	effectuer l'opération suivante 144 / 3	la valeur affichée dans l'écran est 48
5	tester la division par zéro	effectuer l'opération suivante 144 / 0	l'écran de la calculatrice affiche le message : "Erreur: division par Zero !!"

1. Copier le projet **TPJunitPartie2** dans votre WorkSpace puis l'importer dans eclipse (File→ Import → Existing projects into workspace).
2. Créer un package sous le nom ma.ensao.junit.calculateur dans ce projet
3. Créer une classer de test « CalculatriceTest .java» dans le package crée précédemment, puis automatiser le teste de la méthode `initialize()`. (utiliser éclipse pour créer cette classe de test [File→New→ Other... → Java→ JUnit→ JUnit Test Case] et dans une étape spécifique cocher seulement la méthode `initialize()`)
4. Pour chaque scénario **i** rédigé par l'équipe de teste, créer dans la classe « **CalculatriceTest .java**», une méthode **testScenarioi()** permettant d'automatiser le teste de ce scénario.
5. Redéfinir les deux méthodes ***setUp()*** *et* ***tearDown()*** afin de réinitialiser la calculatrice avant l'exécution de chaque scénario.
6. Exécuter les cas de teste automatisés puis corriger les anomalies détectées via ces testes.

III. Partie 3 (JUnit 3) :

1. Créer la classe Algorithme qui contient un tableau tab[] des entiers
2. Créer la méthodes ***void initTab(int []aTab)*** pour initialiser le tableau Tab

3. Créer la méthode ***int sommeTab()*** qui retourne la somme des élément du tableau tab
4. Créer la méthode **void *trieTab()*** pour trier le tableau tab de façon croissante
5. Automatiser les tests des méthodes de la classe Algorithme (***sommeTab() et trieTab()***)