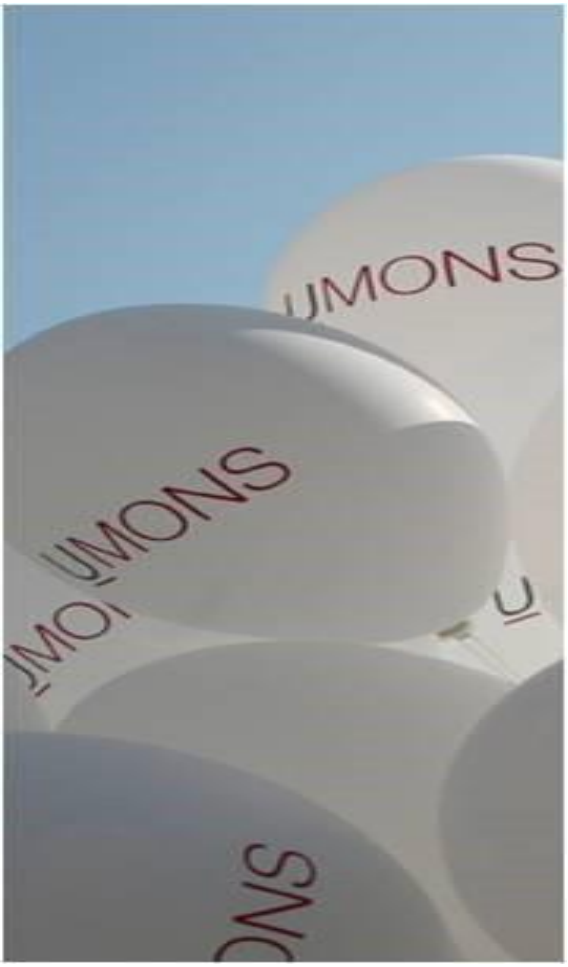
UMONS
Université de Mons

Faculté des Sciences



Développement dirigé par modèles

Séance 2 - Tests unitaires

Rapport
29 novembre 2021 à 21h

Valentin Marchand

Faculté des Sciences

Sous la direction de :
Prof. Stéphane DUPONT
Ass. Bastien VANDERPLAETSE

Année académique 2021-2022

1 Exercice 1

Question 1 : Prédisez le résultat (succeed, failure ou error) pour les trois tests test1(), test2() et test3(). Pour chacun des résultats, expliquez pourquoi le résultat se produit.

Test1 : failure : 2 solutions différentes
Test2 : succeed : 1 seul solution
Test3 : error : Division par zero donc ArithmeticExeption

Question 2 : Écrivez un test unitaire test4() qui vérifie que l'équation 3x2= 0 possède une et une seule solution.

```
public void test4(){
        SolveEquation se = new SolveEquation(3, 0, 0);
        assertEquals(se.getNbSolutions(),1);
    }
```

Question 3 : Écrivez un test unitaire test5() qui vérifie que, si l'équation n'a pas de solution, une exception ArithmeticException est levée lorsqu'on fait appel à getSolutions().

```
public void test5(){
        SolveEquation se = new SolveEquation(0, 0, 0);
        assertThrows(ArithmeticException.class, () -> {se.getSolutions();});
    }
```

Question 4 : Proposez une nouvelle version de test1() qui évite d'appeler la méthode getSolutions() plusieurs fois, en utilisant la méthode Pair.isIdenticalPair().

```
public void test1 () {
        SolveEquation se = new SolveEquation(1.0, -2.0, -3.0);
        assertEquals(se.getNbSolutions(), 2);
        assertTrue(se.getSolutions().isIdenticalPair());
    }
```

2 Exercice 2

Solution dans le code

3 Exercice 3

Question 1 : Prédisez le résultat de chacun des tests (succeed, failure ou error) et expliquez pourquoi ce résultat se produit.

Test1 : failure : On veut 2, on obtient 1
Test2 : error : la méthode n'accepte pas de nombre négatif « IllegalArgumentException »
Test3 : succeed : On veut 3, on obtient 3

Question 2 : Pour les tests qui produisent une erreur ou failure, corrigez-les pour qu'ils réussissent (tout en restant des tests utiles).

Solution dans le code

Question 3 : Ecrivez un quatrième test unitaire test4() vérifiant que la méthode fibo est capable de calculer le 1000ème élément de la suite de Fibonacci, en moins d'une seconde.

```
@Test
    @Timeout(1000)
    public void test4(){
        Fibonacci.fibo(1000);
    }
```

4 Exercice 4

Suite a la suite de test on remarque que l'événement b2 sur l'état SetDate passe à l'état DisplayHour au lieu de DisplayDate

5 Exercice 5

Code disponible