n	n!
0	1
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40 320
9	362 880

JUnit 5 vs. JUnit

TP Factorielle

Le framework JUnit intégré à Eclipse

thèmes: développement des tests unitaires avec le Framework JUnit

Pour ceux qui auraient fini en avance, consultez les différentes documentations et exemples sur le framework JUnit5 (Junit4, JUnit3). Lisez la page Wikipédia et le site de M. Doudoux.
Vous développez un « cas de test » pour la classe Math avec plusieurs @test unitaires pour les méthodes factorielle et factorielleFor.
Voir aussi la page : https://koor.fr/Java/Tutorial/java_junit5.wp

1 Tests unitaires JUnit

Sur Eclipse, dans votre projet, créez un « source folder » **test**.
~~Dans ce source folder, créez les packages en correspondance de ceux qui sont sous src.~~ Avec les facilités d'Eclipse c.a.d. clic droit sur la classe Math <new <JUnit Test case , sans oublié de changer le source folder en **test,** vous placez votre « cas de test » MathTest contenant un @test (test unitaire) par méthode. Répliquez ce test en changeant le nom de la méthode.
Vous testez les valeurs « normales », les valeurs particulières et aux limites : 0 , 1 , 3, 20 , 21, -1, 45
@Test testFactorielleZero() ,@Test testFactorielle1() ,…,
@Test testFactorielleMoinsUn().,… , etc
Les recommandations pour JUnit: **une assertion** par @test et présence du message explicite (visualisable en cas d'échec du @test).

2 Le cas particulier des tests unitaires avec levée d'une exception

Vous testez les valeurs hors limites qui provoquent une anomalie: 21, -1, 45, …
Il est normal de lever une exception **`IllegalArgumentException`**.
Dans ce cas, en JUnit5, utilisez l'écriture suivante (lambda expression):

```
@Test public void testFactorielle45() {
assertThrows(IllegalArgumentException.class,
     ()->{DEMO2_CalculFactorielle.factorielle(45);},
          "La factorielle de 45 ne peut pas être calculée" );
}
```

Rq : en JUnit 4, vous utilisiez l'annotation suivante :

```
@Test(expected=IllegalArgumentException.class)
public void testFactorielle45() {
      DEMO2_CalculFactorielle.factorielle(45); }
```

3 Test paramétré (@ParameteredTest)

Pour ceux qui avancent vite et en autonomie, il y a la possibilité de faire un test unitaire paramétré qui peut remplacer plusieurs de vos tests précédents. Faites une recherche sur le site de Jean-Michel Doudoux :
https://www.jmdoudoux.fr/java/dej/chap-junit5.htm#junit5

Créez un test paramétré pour la factorielle.

4 Une suite de cas de tests (in english : test suite)

En autonomie, incluez le « cas de test » MathTest dans une « suite de tests ».
En cas d'absence d'aide avec Eclipse, voir le site de JM. Doudoux et codez à la main le « TestSuit »
https://www.jmdoudoux.fr/java/dej/chap-junit5.htm#junit5-16

5 Veille technologique sur le « code coverage »

Pour ceux qui avancent vite et en autonomie, faites une recherche la page suivante :
https://koor.fr/Java/Tutorial/java_junit_code_coverage.wp