TP 2

JUint 5

1. Les annotations

Le package **org.junit.jupiter.api** de **JUnit Jupiter (JUnit)** contient plusieurs annotations pour la définition et la configuration des tests. Le tableau suivant illustre les principales annotations.

Annotation	Rôle
@Test	La méthode annotée est un cas de test. Elle ne retourne pas de valeur et elle ne possède pas d'attributs
@BeforeEach	La méthode annotée sera invoquée avant l'exécution de chaque test.
@AfterEach	La méthode annotée sera invoquée après l'exécution de chaque test
@BeforeAll	La méthode annotée sera invoquée une fois avant l'exécution de l'ensemble des tests
@AfterAll	La méthode annotée sera invoquée une fois après l'exécution de l'ensemble des tests
@Disabled	Désactiver les tests de la classe ou la méthode annotée.
@EnabledOnOs	Elle est utilisée pour signaler que la classe de test annotée ou la méthode de test n'est activée que sur un ou plusieurs systèmes d'exploitation spécifiés.
@DisplayName	Elle est utilisée pour fournir un nom personnalisé pour la classe de test et les méthodes de test.
@EnabledOnJre	Elle est utilisée pour signaler que la classe ou la méthode de test annotée n'est activée que sur une ou plusieurs versions spécifiées de Java Runtime Environment (JRE)
@EnabledForJreRange	Elle est utilisée pour signaler que la classe ou la méthode de test annotée n'est activée que pour une plage spécifique de versions de Java Runtime Environment (JRE) de min() à max().

Exemple1 :

- Créez un nouveau projet JAVA
- Ecrivez la classe JAVA suivante :

```java
public class MyClass{
  public int calculer(int a, int b) {
    int res = a + b;
    if (a == 0){
      res = b * 2;
    }
    if (b == 0) {
      res = a * a;
    }
    return res;
  }
}
```

- Ecrivez la classe de **Junit test** suivante :

```java
class MyTestClass1 {
      private MyClass myClass;
      @BeforeAll
      public static void beforAll () {
            System.out.println("Etablir la connexion");
      }
      @AfterAll
      public static void afterAll () {
            System.out.println("Fermer la connexion");
      }
      @BeforeEach
      public void init () {
            myClass = new MyClass;()
      }
      @DisplayName("Somme de deux nombres positifs")
      @Test
      public void test1 () {
            System.out.println("Cas de test 1");
            assertEquals(2,myClass.calculer(1,1));
      }
      @DisplayName("Somme de deux nombres négatifs")
      @Test
      public void test2 () {
            System.out.println("Cas de test 2");
            assertEquals(-2,myClass.calculer(-1,-1));
      }
      @DisplayName("Somme d'un nombres négatif et nombre positif")
      @Test
      public void test3 () {
            System.out.println("Cas de test 3");
            assertEquals(0,myClass.calculer(-1,1));
      }
}
```

Exemple2 :

- Ecrivez une nouvelle classe de **Junit test** de la spécification suivante :

```java
public class TestCondition {
	@Test
	@EnabledOnOs(OS.LINUX)
	 public void enabledConditionOsLinux() {
		System.out.println("Exécuter le test sous linux");
	  }
	@Test
	@EnabledOnOs(OS.WINDOWS)
	public void enabledConditionOsWindows() {
		System.out.println("Exécuter le test sous Windows");
	   }
	@Test
	@EnabledOnJre(JRE.JAVA_17)
	public void enabledConditionOnJre17() {
		System.out.println("Exécuter le test avec JAVA 17");
	  }
	@Test
	@EnabledForJreRange(min = JRE.JAVA_8, max=JRE.JAVA_18)
	public void enabledConditionOnJreBetween8and18() {
		System.out.println("Exécuter le test entre JAVA 8 et 18");
	  }
  }
```

Exercice :

Ecrivez en JAVA une classe qui s'appelle PCGD.

Ecrivez une méthode qui s'appelle calculePCGD pour calculer le PCGD entre deux nombres entiers. L'algorithme Calcule PCGD est le suivant :

```
entier a, b
écrire "Introduisez le 1er nombre" :
lire a
écrire "Introduisez le 2ème nombre" :
lire b
tant que (a-b ≠0 ) faire
    si a>b alors
        a ← a-b
    sinon
        b ← b-a
    fin si
fin tant
si a=0 alors
    écrire "PGCD = ", b
sinon
    écrire "PGCD = ", a
fin si
```

- Ecrivez un test Junit5 pour tester la méthode calculePCGD en utilisant les entrées suivantes : (a=0,b=0),(a=0 ,b=5),(a=7,b=0),(a=14,b=32),(a=21,b=13), (a=30,b=5), (a=12,b=15), (a=12, b=7).
- Utilisez les annotations suivantes : @BeforeEach, @AfterEach, @BeforeAll , @AfterAll et @DisplayName
- Exécutez l'ensemble des cas de test et évaluez les résultats de votre test
- Exécutez à nouveau l'ensemble des cas de test en désactivant tous les tests contenant des entrées nulles pour a ou b.