Rapport TP#2 GL 03:

212131049271
ben mohamed maria
L3 ISIL A
group tp3

implémentation des classes:

1. Livre :

Livre.java × MethodsUnderTest.java MethodsUnderTestTest.java

```java
package tp2;

public class Livre {
    private String titre , auteur;
    private int nbPages;
    private double prix;

    public Livre ( String titre, String auteur, int nbPages, double prix) {
        this.titre = titre;
        this.auteur = auteur;
        this.nbPages = nbPages;
        this.prix = prix;
    }
}
```

2. MethodsUnderTest :

Livre.java MethodsUnderTest.java × MethodsUnderTestTest.java

```java
package tp2_2;

public class MethodsUnderTest {
    private Livre l= null ;

    public int add(int a, int b){
        int res = a;
        if(b>0){
            while(b--!=0){res++;}
        }
        else {
            if(b<0){
                while(b++!=0){res--;}
            }
        }
        return res;
    }
```

```
    public int mult(int a, int b){
        int res = 0;
        while (b>0) {
            if(b%2!=0){
                res = res+a;
                b--;
            }
        }
        b = b/2;
        a = 2*a;
        return res;
    }

    public double calculatePI(int n ){
        double s = 0 ;
        double PI = 0 ;
        int i;
        if(n<0){
            PI = -1;
        }
        else{
            for(i=0; i<n; i++){s = s+((java.lang.Math.pow(-1, i))/((2 * i)+1));}
            PI = 4*s ;
        }
        return PI;
    }
```

```
    public boolean verifiCode(int T[], int n){
        int res = 0;
        int i = 1;
        while(res<3 && i<n) {
            if(T[i] == T[i-1]+1) {
                res = res+1;
            }
            else {res = 0;}
            i = i+1;
        }
        return (res == 3);
    }

    public int[] leftRotation(int T[], int n){
        int T2[] = new int[n];
        for(int i=0; i<n; i++){T2[((i+4)%5)] = T[i];}
        return T2;
    }

    public Livre addLivre(String titre, String auteur, int nbpages, double prix) {
        if(!titre.equals("") && !auteur.equals("")){l = new Livre(titre, auteur, nbpages, prix);}
        return(l);
    }
```

```
    public Livre addLivre(String titre, String auteur, int nbpages, double prix) {
        if(!titre.equals("") && !auteur.equals("")){l = new Livre(titre, auteur, nbpages, prix);}
        return(l);
    }

    public Livre getLivre() {
        return l;
    }
}
```

construction des tests pour la class “MethodsUnderTest”:
(on utilisant junit version 5)

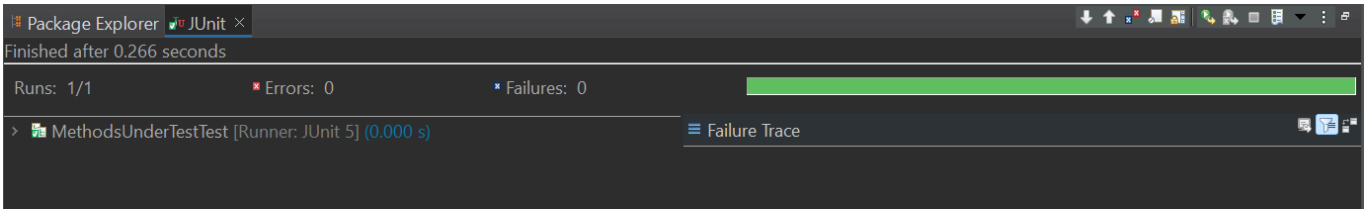
- **pour la méthode add :**

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

class MethodsUnderTestTest {

    @Test
    public void testAdd() {
        MethodsUnderTest test = new MethodsUnderTest();
        assertEquals(7, test.add(3,4), "l'ajout de 2 nombres positifs a échoué !");
        assertEquals(-5, test.add(-2,-3), "l'ajout de 2 nombres négatifs a échoué !");
        assertEquals(-2, test.add(1,-3), "l'ajout de 2 nombres de signes différents a échoué !");
        assertEquals(2, test.add(-3,5), "l'ajout de 2 nombres de signes différents a échoué !");
        assertEquals(3, test.add(3,0), "l'ajout de d' un nombre positif et un zero a échoué !");
        assertEquals(-3, test.add(0,-3), "l'ajout d' un nombre négatif et un zero a échoué !");
        assertEquals(0, test.add(0,0), "l'ajout de 2 zeros a échoué !");
    }
```

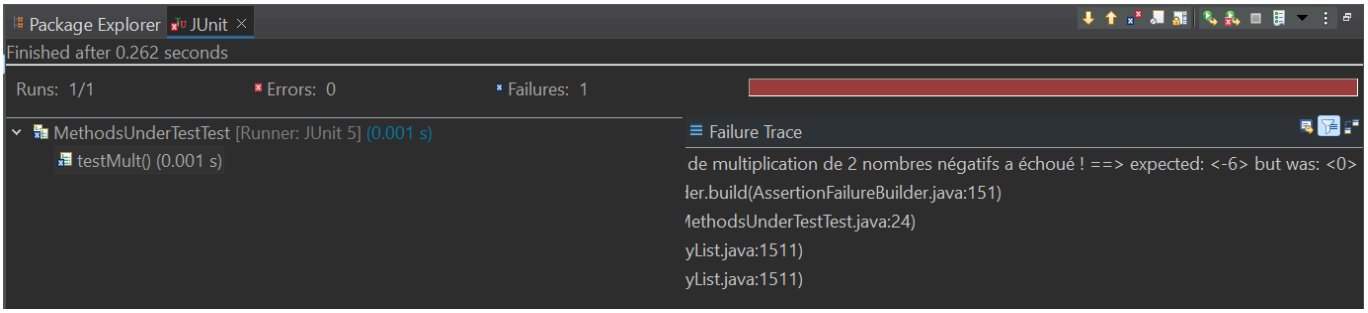
on la section et on click run pour lancer le test:



- **pour la méthode mult :**

```java
    @Test
    public void testMult() {
        MethodsUnderTest test = new MethodsUnderTest();
        assertEquals(6, test.mult(2,3), "le test de multiplication de 2 nombres positifs a échoué !");
        assertEquals(-6, test.mult(-2,-3), "le test de multiplication de 2 nombres négatifs a échoué !");
        assertEquals(-3, test.mult(1,-3), "le test de multiplication de 2 nombres de signes différents a échoué !");
        assertEquals(-15, test.mult(-3,5), "le test de multiplication de 2 nombres de signes différents a échoué !");
        assertEquals(0, test.mult(1,0), "le test de multiplication de 2 nombres de signes différents a échoué !");
        assertEquals(0, test.mult(0,-3), "le test de multiplication de 2 nombres de signes différents a échoué !");
        assertEquals(0, test.mult(0,0), "le test de multiplication de 2 nombres de signes différents a échoué !");
    }
```

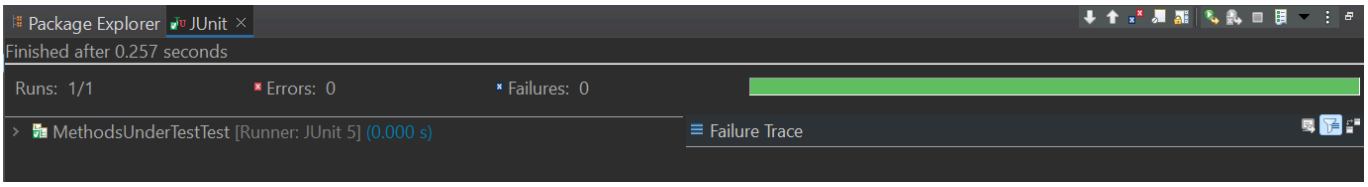
on lancer le test:



on observe que la fonction mult ne fonction pas quand a ou b est negative.

```java
@Test
public void testMult() {
    MethodsUnderTest test = new MethodsUnderTest();
    assertEquals(6, test.mult(2,3), "le test de multiplication de 2 nombres positifs a échoué !");
    assertEquals(0, test.mult(1,0), "le test de multiplication de 2 nombres de signes différents a échoué !");
    assertEquals(0, test.mult(0,0), "le test de multiplication de 2 nombres de signes différents a échoué !");
}
```

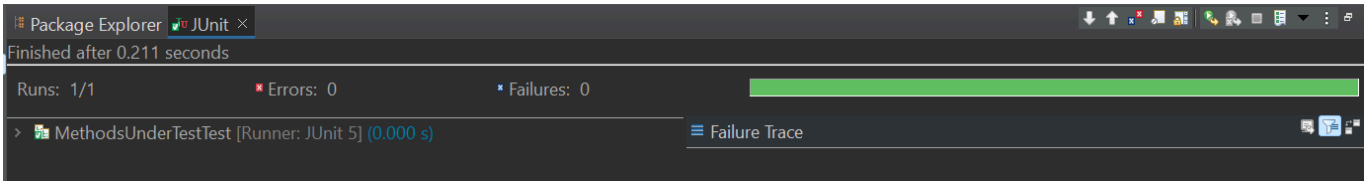
et on lancer le test:



- **pour la méthode calculatePI :**

```java
@Test
public void testCalculatePI() {
    MethodsUnderTest test = new MethodsUnderTest();
    assertEquals((-1), test.calculatePI(-1), 0.2, "probleme dans le if !");
    assertEquals(0, test.calculatePI(0), 0.9, "probleme dans le else (exterieur de la boucle) !");
    assertEquals(3.14, test.calculatePI(1), 0.9, "probleme dans le else !");
    assertEquals(3.14, test.calculatePI(2), 0.9, "probleme dans le else !");
    assertEquals(3.14, test.calculatePI(3), 0.9, "probleme dans le else !");
}
```

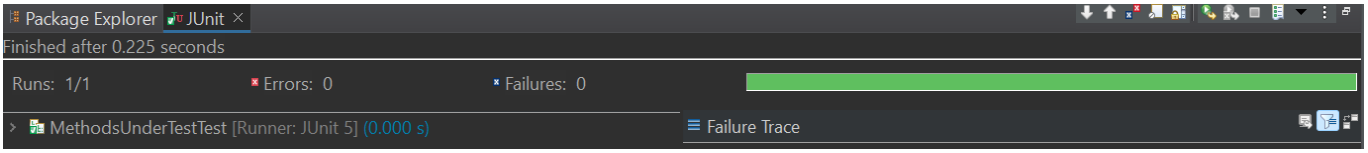
on lancer le test:



- **pour la méthode verifiCode :**

```java
@Test
void testVerifiCode() {
    MethodsUnderTest test = new MethodsUnderTest();
    int[] testSerie1 = {0,2,5,9,5,8,7};
    assertFalse(test.verifiCode(testSerie1, 7), "la methode 'verifiCode' ne foction pas correctement !");
    int[] testSerie2 = {5, 4, 3, 2, 1};
    assertFalse(test.verifiCode(testSerie2, 5), "la methode 'verifiCode' ne foction pas correctement !");
    int[] correctSerie = {3, 4, 5, 6, 7, 8, 9, 10};
    assertTrue(test.verifiCode(correctSerie, 8), "la methode 'verifiCode' ne foction pas correctement !");
}
```

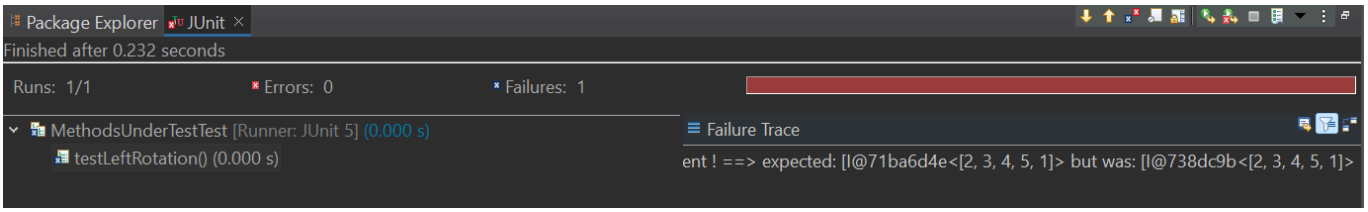
on lancer le test:



- **pour la méthode leftRotation :**

```java
@Test
void testLeftRotation() {
    MethodsUnderTest test = new MethodsUnderTest();
    int[] testSerie = {1, 2, 3, 4, 5};
    int[] expectedSerie = {2, 3, 4, 5, 1};
    assertEquals(expectedSerie, test.leftRotation(testSerie, 5), "la methode 'leftRotation' ne foction pas correctement !");
    int[] testSerie2 = {4, 5, 6, 7, 8};
    int[] expectedSerie2 = {5, 6, 7, 8, 4};
    assertEquals(expectedSerie2, test.leftRotation(testSerie2, 5), "la methode 'leftRotation' ne foction pas correctement !");
}
```

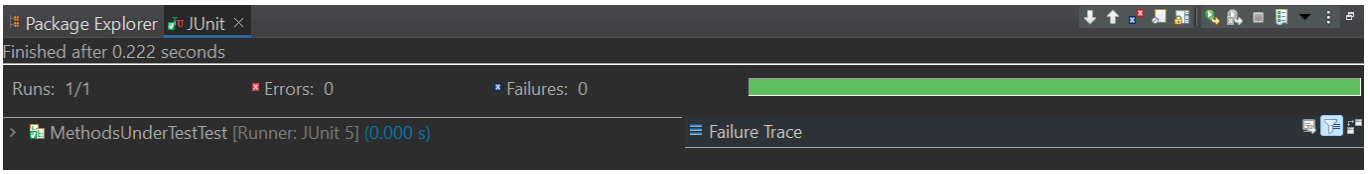
on lancer le test:



on observe que la fonction leftRotation fonction correctement, mais le “assertEquals” ne marche pas sur les objets de type array.
alors on utilise “assertArrayEquals”, come suite :

```
@Test
void testLeftRotation() {
    MethodsUnderTest test = new MethodsUnderTest();
    int[] testSerie = {1, 2, 3, 4, 5};
    int[] expectedSerie = {2, 3, 4, 5, 1};
    assertArrayEquals(expectedSerie, test.leftRotation(testSerie, 5), "la methode 'leftRotation' ne foction pas correctement !");
    int[] testSerie2 = {4, 5, 6, 7, 8};
    int[] expectedSerie2 = {5, 6, 7, 8, 4};
    assertArrayEquals(expectedSerie2, test.leftRotation(testSerie2, 5), "la methode 'leftRotation' ne foction pas correctement !");
}
```

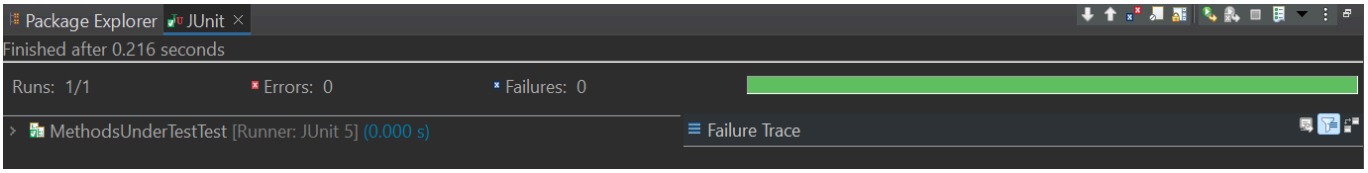
et on lancer le test:



- **pour la méthode addLivre :**

```
@Test
void testAddLivre() {
    MethodsUnderTest test = new MethodsUnderTest();
    assertNull(test.addLivre("","",50,50), "la methode 'addLivre' ne foction pas correctement !");
    assertNull(test.addLivre("","auteur",50,50), "la methode 'addLivre' ne foction pas correctement !");
    assertNull(test.addLivre("titre","",50,50), "la methode 'addLivre' ne foction pas correctement !");
    assertNotNull(test.addLivre("titre","auteur",50,50), "la methode 'addLivre' ne foction pas correctement !");
}
```

on lancer le test:



- **pour la méthode getLivre :**

```java
@Test
void testGetLivre() {
    MethodsUnderTest test = new MethodsUnderTest();
    assertNull(test.getLivre(), "L'instance l de typr Livre a ete instancier !");
    test.addLivre("","",50,50);
    //une methode que a ete tester, est march bien
    assertNull(test.getLivre(), "la methode 'getLivre' ne foction pas correctement !");
    test.addLivre("","auteur",50,50);
    assertNull(test.getLivre(), "la methode 'getLivre' ne foction pas correctement !");
    test.addLivre("titre","",50,50);
    assertNull(test.getLivre(), "la methode 'getLivre' ne foction pas correctement !");
    test.addLivre("titre","auteur",50,50);
    assertNotNull(test.getLivre(), "la methode 'getLivre' ne foction pas correctement !");
}
```

on lancer le test: