جـــامعة هواري بومـــدين للعلـــوم و التكنـــولوجيـــــا
Université des Sciences et de la Technologie Houari Boumediene
Faculté d'Electronique et d'Informatique

RAPPORT TP JUnit

KACI ANIS 191931051628

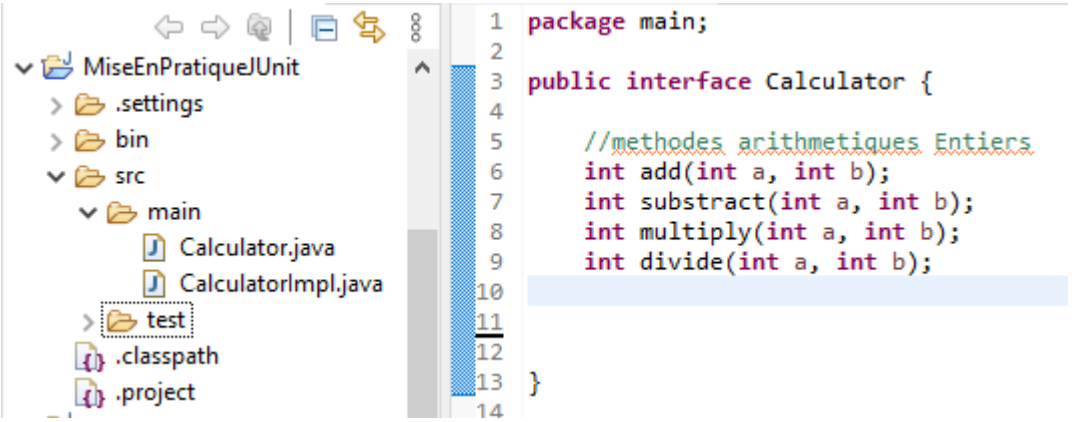
L3 ISIL SECTION A

GROUPE 5

2021/2022

0- Créer un projet java “MiseEnPratiqueJUnit”

1 - Créer l’interface Calculator dans le package main:




```java
package main;

public interface Calculator {

    //methodes arithmetiques Entiers
    int add(int a, int b);
    int substract(int a, int b);
    int multiply(int a, int b);
    int divide(int a, int b);


}
```

2- Créer la classe Calculator Impl qui implémente les méthodes de l’interface Calculator :

```java
package main;

public class CalculatorImpl implements Calculator{
```

2.1 Méthode add (addition de 2 entiers):

```java
@Override
public int add(int a, int b) {
   //initialiser le resultat a a
   int res = a;
   //si b est positif, tq b est positif incrementer res
   if (b > 0) {
   while(b-- != 0) {
   res++;
   }
   }
   //sinon si b est negatif, tq b est negatif decrementer res
   else if (b < 0) {
   while(b++ != 0) {
   res--;
   }
   }
   return res;
   }
```

2.2 Méthode substract(soustraction):

```java
@Override
public int substract(int a, int b) {
      //same thinking avec la fct add juste l'inverse
       int res = a;

       if(b < 0) {
           while(b++ != 0) {
               res++;
           }
       }
       else if(b > 0) {
           while(b-- != 0) {
               res --;
           }
       }

       return res;


   }
```

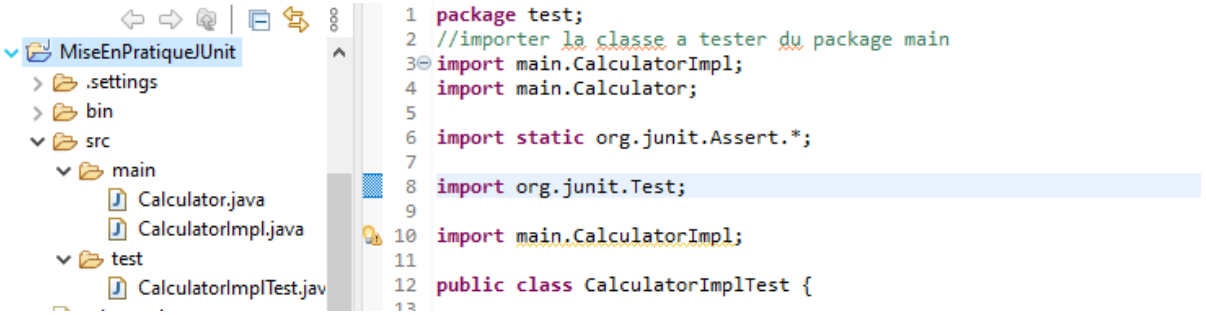
2.3 Methode multiply(multiplication):

```java
@Override
public int multiply(int a, int b) {

    int res=0;
    boolean resneg=false;
    if(a<0) {
        resneg=!resneg;
        a=-a;
    }
    if(b<0) {
        resneg=!resneg;
        b=-b;
    }
    if(b>0) {
        while(b--!=0) {
            res=res+a;
        }
    }
    if(resneg) {
        res=-res;
    }
    return res;

}
```

2.4 Méthode divide(division):

```
public int divide(int a, int b) {

    //cas division par 0
    if(b == 0) {
        throw new ArithmeticException();
    }
    //initialiser le resultat a positif
    boolean resEstNegatif = false;
    int res = 0;

    //traiter le signe de la division
    if(a < 0) {

        resEstNegatif = !resEstNegatif;
        a = -a;
    }

    if( b < 0) {
        resEstNegatif = !resEstNegatif;
        b = -b;

    }

    //a/b = combien de fois on peut soustraire b de a
    while (a > 0) {
        a = substract(a, b);
        res++;
        }
        if (resEstNegatif) {
        res = -res;
        }
        return res;

}
```

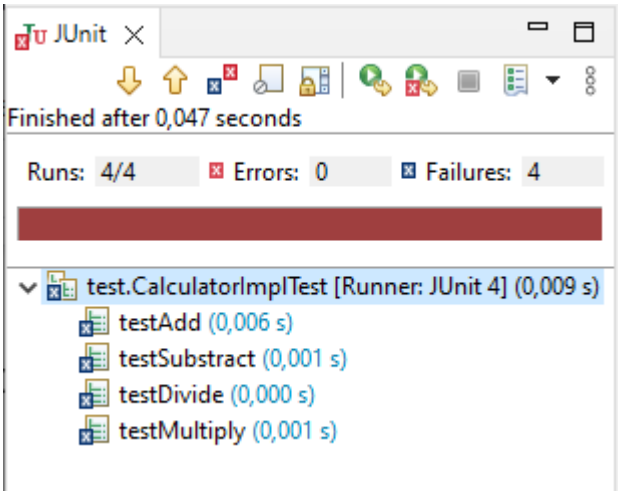
3- Créer le **package test** et la classe CalculatorImplTest pour tester les methodes de la classe CalculatorImpl:



3. par défaut eclipse fait échouer les tests car non encore implémenté!
on remarque que chaque méthode test est précédée par l'annotation **@test**

```
package test;

import static org.junit.Assert.*;

public class CalculatorImpl {

    @Test
    public void testMuliply() {
        fail("Not yet implemented");
    }
    @Test
    public void testDevide() {
        fail("Not yet implemented");
    }
    @Test
    public void testAdd() {
        fail("Not yet implemented");
    }
    @Test
    public void testSubstract() {
        fail("Not yet implemented");
    }

}
```

les tests échouent comme on remarque



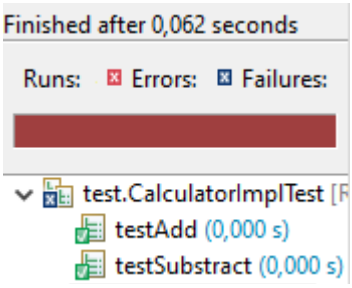
3.1 - Test méthode add:

```java
@Test
public void testAdd() {

//instancier la classe a tester
  Calculator testCalc = new CalculatorImpl();

  //tester la methode add
  int a, b, res;

  //tester tous les cas possibles
  a = 5;
  b = 5;
  res = a + b;
  if (testCalc.add(a, b) != res) {
  fail("a et b positif");
  }
  a = 0;
  b = 5;
  res = a + b;
  if (testCalc.add(a, b) != res) {
  fail("a nul");
  }
  a = 5;
  b = 0;
  res = a + b;
  if (testCalc.add(a, b) != res) {
      fail("b nul");
  }
  a = 0;
  b = 0;
  res = a + b;
  if (testCalc.add(a, b) != res) {
  fail("a et b nuls");
  }
  a = -5;
  b = 5;
  res = a + b;
  if (testCalc.add(a, b) != res) {
  fail("a negatif");
  }
  a = 5;
  b = -5;
  res = a + b;
  if (testCalc.add(a, b) != res) {
  fail("b negatif");
  }

      a = -5;
      b = -5;
      res = a + b;
      if (testCalc.add(a, b) != res) {
      fail("a et b negatif");
      }

    }
```

on remarque une certaine répétition sur les if et fail
on peut gérer cela en utilisant **assert** :

on teste cela sur la test de la méthode substract:

```
@Test
public void testSubstract() {
    //set up
    Calculator calc = new CalculatorImpl();
    int a,b,res;
    //tester tous les cas possibles

    //1
    a = 6;
    b = 5;
    res = a-b;

    assertEquals("a et b positifs",calc.substract(a, b),res);

    a=0;
    b=5;
    res = a-b;

    assertEquals("a nul",calc.substract(a, b),res);


    a = 5;
    b = 0;
    res = a-b;

    assertEquals("b nul",calc.substract(a, b),res);


    a = -5;
    b = 1;

    res =  a-b;

    assertEquals("a negatif",calc.substract(a, b),res);

    a = 1;
    b = -5;

    res = a-b;

    assertEquals("b negatif",calc.substract(a, b),res);


    a = -5;
    b = -5;

    res = a-b;

     assertEquals("a et b negatifs",calc.substract(a, b),res);


}
```



le test marche parfaitement

4- Gestion des exceptions:

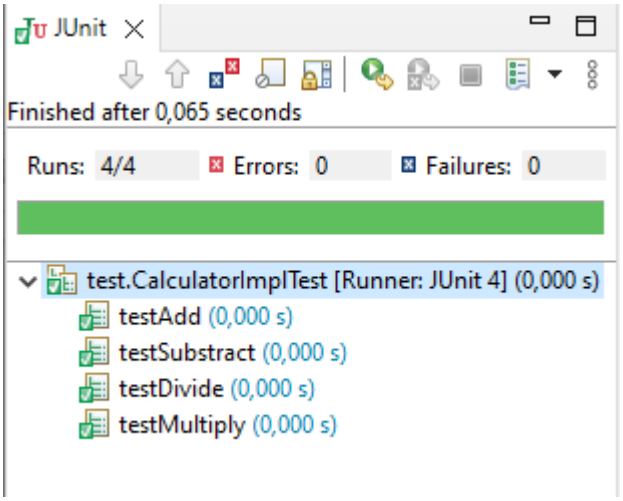
on gère l'exception arithmétique qui doit être jetée dans lors de l'exécution de la méthode divide:

```
public int divide(int a, int b) {

    //cas division par 0
    if(b == 0) {
        throw new ArithmeticException();
    }
```

test:

```
    //on attend qu'une exception soit jetée
    //si ce n'est pas le cas , le test echoue
    @Test (expected = ArithmeticException.class)
    public final void testDivide() {
        Calculator calc = new CalculatorImpl();
        int a, b, res;
        a = 5;
        b = 5;
        res = a / b;
        assertEquals("a et b positifs",calc.divide(a, b),res);
        a = 0;
        b = 5;
        res = a / b;
        assertEquals("a nul",calc.divide(a, b),res);
        a = -5;
        b = 5;
        res = a / b;
        assertEquals("a negatif",calc.divide(a, b),res);

        a = 5;
        b = -5;
        res = a / b;
        assertEquals("b negatif",calc.divide(a, b),res);

        a = -5;
        b = -5;
        res = a / b;
        assertEquals("a et b negatifs",calc.divide(a, b),res);

        a=5;
        b=0;
        res = a/b;
        assertEquals("b nul",calc.divide(a, b),res);

        }


    }
```

on teste et tous nos tests passent:



Fin;