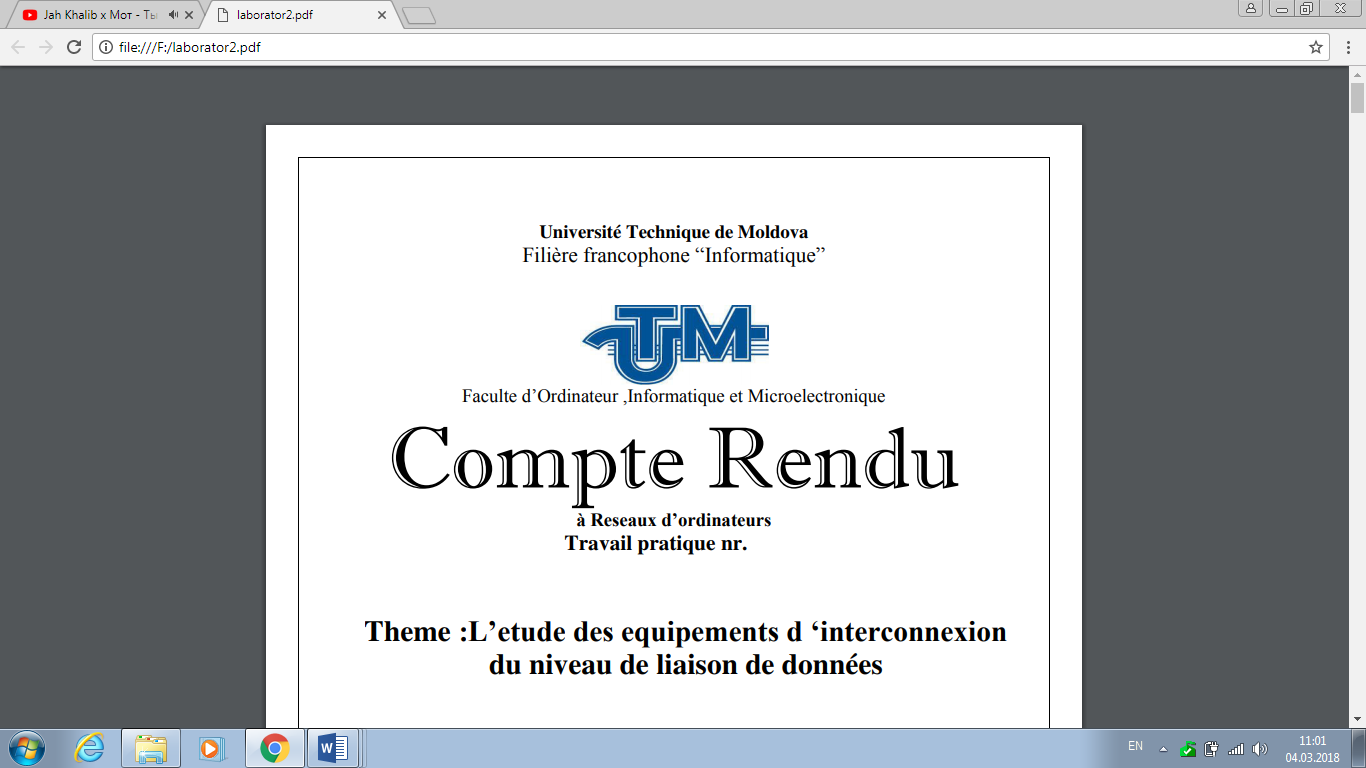
Ministère de l'éducation, de la culture et de la recherche

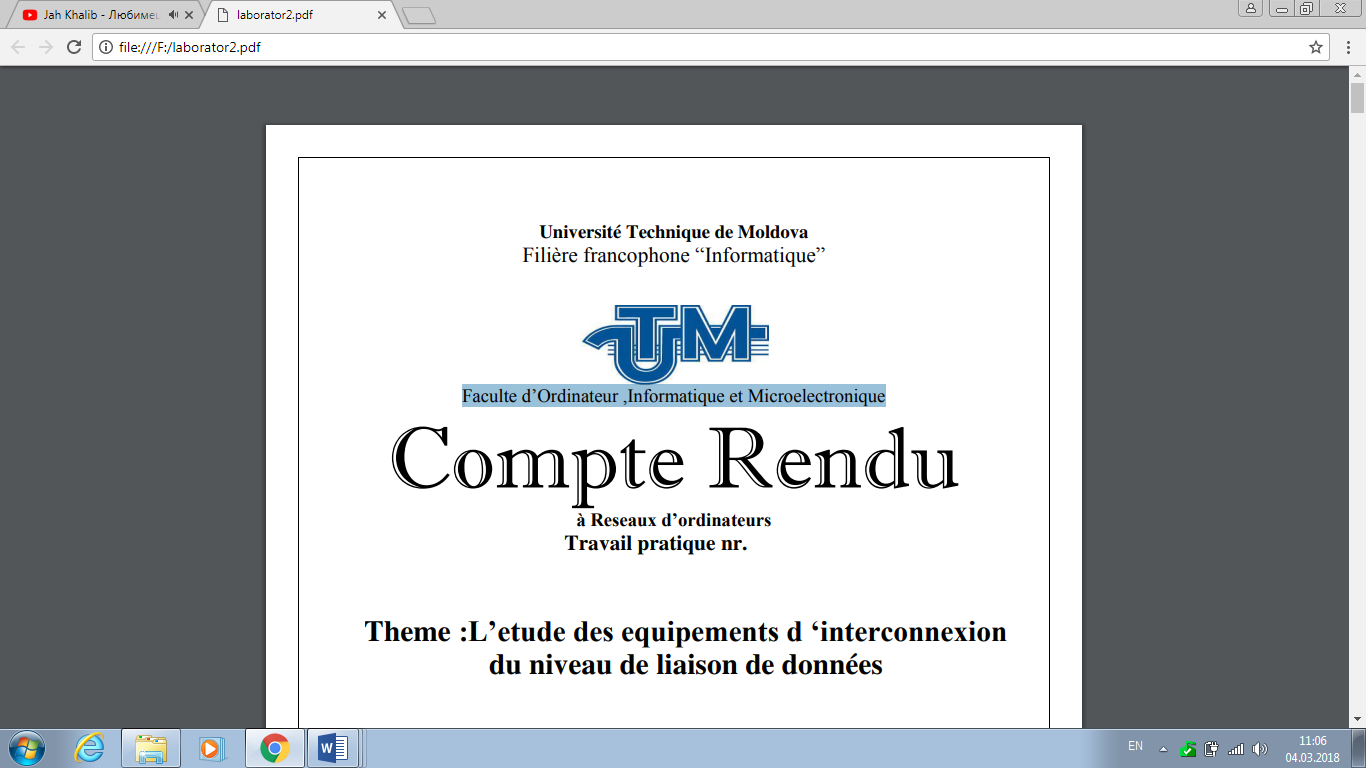
de la République de Moldova

Université technique de Moldavie

Faculté d’Ordinateur, Informatique et Microélectronique

Filière francophone “Informatique”





**TIDPP**

Travail pratique nr.3

**Thème:** **Unit testing, Dependency Injection, Mockups**

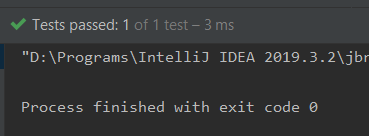
Effectué par l’étudiant de gr FI-181 : Damean Alexandra

Vérifié par le professeur : Rusu Viorel

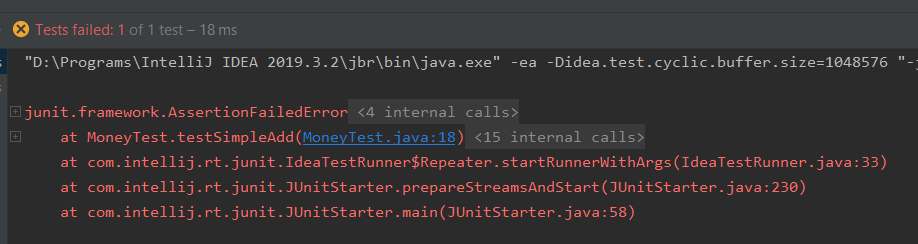
Chișinau 2020

**But:** passer les étapes de test et de déploiement dans le cycle de développement d'un programme et en utilisant les outils Java spécifiques à ces étapes.

**a)**

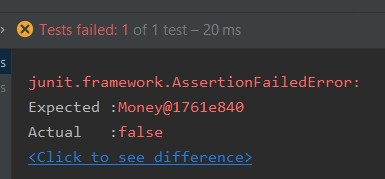


**b)**

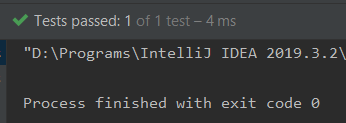


**c)**

public void testSimpleequals() {  
 Money m12CHF = new Money(12, "CHF");  
 Money m14CHF = new Money(14, "CHF");  
  
 Money expected = new Money(26, "CHF");  
 Boolean result = m12CHF.equals(m14CHF);  
  
 *assertEquals*(expected, result);  
}



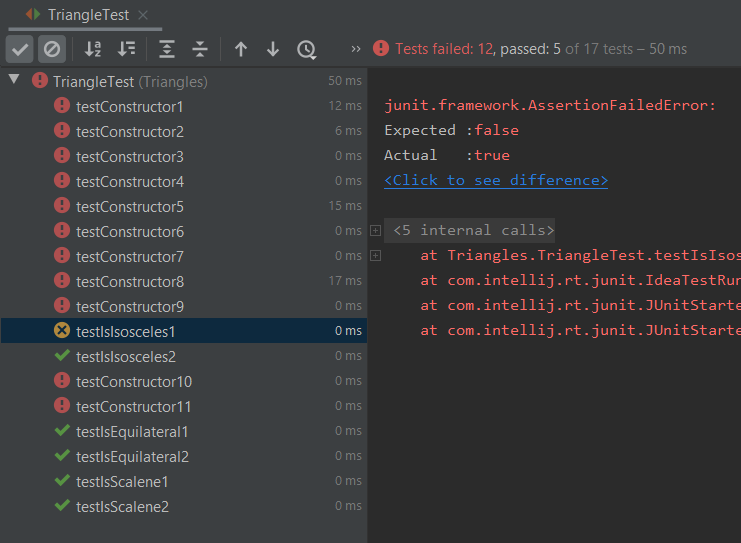
public void testSimpleequals() {  
 Money m12CHF = new Money(12, "CHF");  
 Money m14CHF = new Money(14, "CHF");  
  
 Boolean expected = false;  
 Boolean result = m12CHF.equals(m14CHF);  
  
 *assertEquals*(expected, result);  
}





public class TriangleTest extends TestCase {  
  
 public TriangleTest(String name) {  
 super(name);  
 }  
  
 // On verifie le constructeur  
  
 public void testConstructor1() throws InvalidTriangleException {  
 Triangle tr = new Triangle(5, 1, 2);  
 }  
 public void testConstructor2() throws InvalidTriangleException {  
 Triangle tr = new Triangle(1, 5, 2);  
 }  
 public void testConstructor3() throws InvalidTriangleException {  
 Triangle tr = new Triangle(2, 5, 1);  
 }  
 public void testConstructor4() throws InvalidTriangleException {  
 Triangle tr = new Triangle(5, 0, 2);  
 }  
 public void testConstructor5() throws InvalidTriangleException {  
 Triangle tr = new Triangle(0, 5, 2);  
 }  
 public void testConstructor6() throws InvalidTriangleException {  
 Triangle tr = new Triangle(2, 5, 0);  
 }  
 public void testConstructor7() throws InvalidTriangleException {  
 Triangle tr = new Triangle(0, 0, 0);  
 }  
 public void testConstructor8() throws InvalidTriangleException {  
 Triangle tr = new Triangle(5, -5, 2);  
 }  
 public void testConstructor9() throws InvalidTriangleException {  
 Triangle tr = new Triangle(-5, 5, 2);  
 }  
 public void testConstructor10() throws InvalidTriangleException {  
 Triangle tr = new Triangle(2, 5, -5);  
 }  
 public void testConstructor11() throws InvalidTriangleException {  
 Triangle tr = new Triangle(-4,-5,-3);  
 }  
  
 // On verifie la methode isScalene  
  
 public void testIsScalene1() throws InvalidTriangleException {  
 Boolean expected = false;  
 Triangle tr = new Triangle(3,4,3);  
 Boolean ourTr = tr.isScalene();  
 *assertEquals*(expected,ourTr);  
 }  
  
 public void testIsScalene2() throws InvalidTriangleException {  
 Boolean expected = false;  
 Triangle tr = new Triangle(3,3,3);  
 Boolean ourTr = tr.isScalene();  
 *assertEquals*(expected,ourTr);  
 }  
  
 // On verifie la methode isIsosceles  
  
 public void testIsIsosceles1() throws InvalidTriangleException {  
 Boolean expected = false;  
 Triangle tr = new Triangle(3,3,3);  
 Boolean ourTr = tr.isIsosceles();  
 *assertEquals*(expected,ourTr);  
 }  
 public void testIsIsosceles2() throws InvalidTriangleException {  
 Boolean expected = false;  
 Triangle tr = new Triangle(5,4,3);  
 Boolean ourTr = tr.isIsosceles();  
 *assertEquals*(expected,ourTr);  
 }  
  
 // On verifie la methode isEquilateral  
  
 public void testIsEquilateral1() throws InvalidTriangleException {  
 Boolean expected = false;  
 Triangle tr = new Triangle(5,4,3);  
 Boolean ourTr = tr.isEquilateral();  
 *assertEquals*(expected,ourTr);  
 }  
 public void testIsEquilateral2() throws InvalidTriangleException {  
 Boolean expected = false;  
 Triangle tr = new Triangle(5,4,5);  
 Boolean ourTr = tr.isEquilateral();  
 *assertEquals*(expected,ourTr);  
 }  
  
}

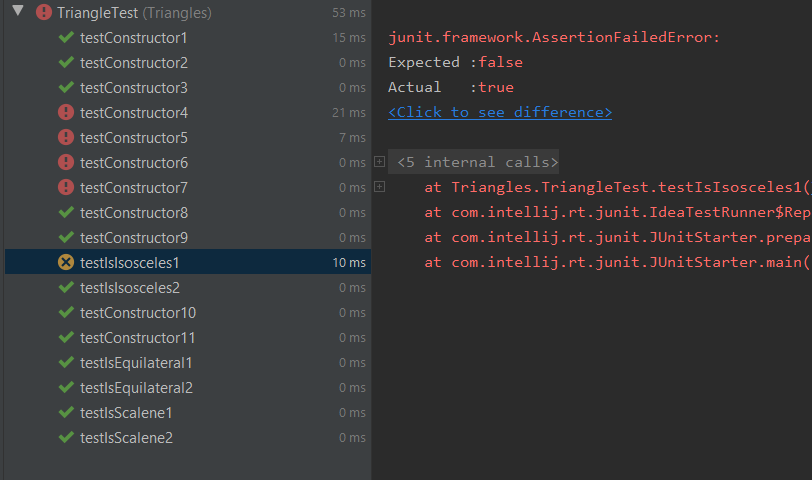
1. **Error1Triangle**



*Le test montre que la méthode isIsoscele est fausse.*

public boolean isIsosceles() {  
 return this.a == this.b || this.a == this.c || this.b == this.c;  
}

1. **Error2Triangle**



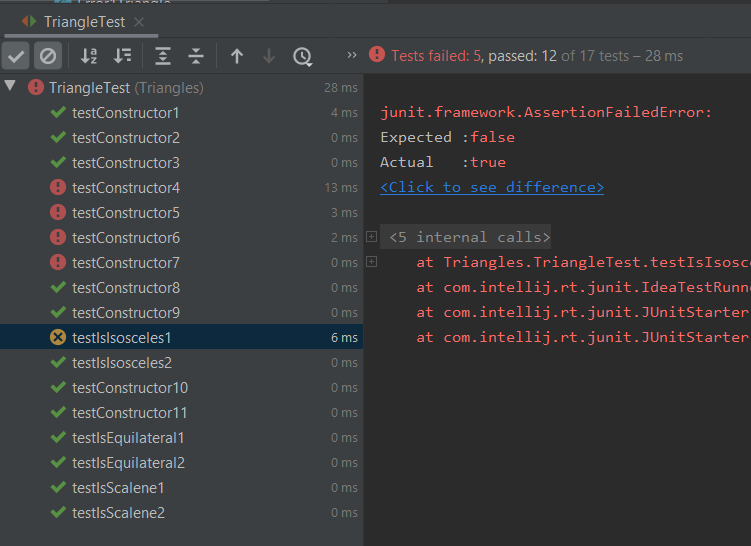
public boolean isIsosceles() {  
 return this.a == this.b || this.a == this.c || this.b == this.c;  
}

*Il accepte les valeurs a + b < c et a + c < b et c + b < a.*

*Il accepte des valeurs de longueur de côté négatives.*

public Triangle(int var1, int var2, int var3) throws InvalidTriangleException {  
 this.a = var1;  
 this.b = var2;  
 this.c = var3;  
 int var4 = (var1 + var2 + var3) / 2;  
 boolean var5 = var1 == 0 || var2 == 0 || var3 == 0;  
 if (var5) {  
 throw new InvalidTriangleException();  
 }  
}

1. **Error3Triangle**



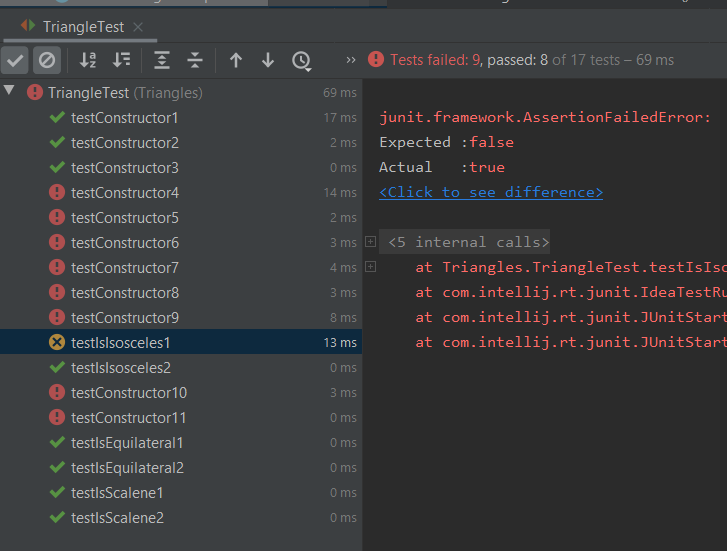
public boolean isIsosceles() {  
 return this.a == this.b || this.a == this.c || this.b == this.c;  
}

*Il accepte les valeurs a + b < c et a + c < b et c + b < a.*

*Il accepte des valeurs de longueur de côté négatives*

public Triangle(int var1, int var2, int var3) throws InvalidTriangleException {  
 this.a = var1;  
 this.b = var2;  
 this.c = var3;  
 int var4 = (var1 + var2 + var3) / 2;  
 boolean var5 = var1 == 0 || var2 == 0 || var3 == 0;  
 if (var5) {  
 throw new InvalidTriangleException();  
 }  
}

1. **Error4Triangle**



public boolean isIsosceles() {  
 return this.a == this.b || this.a == this.c || this.b == this.c;  
}

*Le test montre que le constructeur est fausse.*

*Il accepte les valeurs a + b < c et a + c < b et c + b < a.*

public Triangle(int var1, int var2, int var3) throws InvalidTriangleException {  
 this.a = var1;  
 this.b = var2;  
 this.c = var3;  
 int var4 = (var1 + var2 + var3) / 2;  
 boolean var5 = var1 == 0 || var2 == 0 || var3 == 0 || var1 < 0 || var2 < 0 || var3 < 0;  
 if (var5) {  
 throw new InvalidTriangleException();  
 }  
}

public class Locator {  
 Door door;  
 Window window;  
 Roof roof;  
  
 public Locator(Door door, Window window, Roof roof){  
 this.door = door;  
 this.window = window;  
 this.roof = roof;  
 }  
 protected Door getDoor(){  
 return door;  
 }  
 protected Roof getRoof(){  
 return roof;  
 }  
 protected Window getWindow(){  
 return window;  
 }  
}

import junit.framework.TestCase;  
  
public class HouseTest extends TestCase {  
 public void testDoor() {  
 Locator locator = new Locator(new Door(), new Window(), new Roof());  
 House h = new House(locator);  
 Door d = h.getDoor();  
 }  
 public void testRoof() {  
 Locator locator = new Locator(new Door(), new Window(), new Roof());  
 House h = new House(locator);  
 Roof r = h.getRoof();  
 }  
 public void testWindow() {  
 Locator locator = new Locator(new Door(), new Window(), new Roof());  
 House h = new House(locator);  
 Window w = h.getWindow();  
 }  
}

**Conclusion :**

Le but de ce travail pratique était passer les étapes de test et de déploiement dans le cycle de développement d'un programme et en utilisant les outils Java spécifiques à ces étapes. Je me suis familiarisé avec l'outil JUnit. On a etudie la methode Black-box testing. Je me suis familiarisé avec l'outil Dependency Injection.

Au bout de tout on peut dire que l’etape de controle est tres importante, parce que cela peut éviter un risque coûteux à l'avenir.