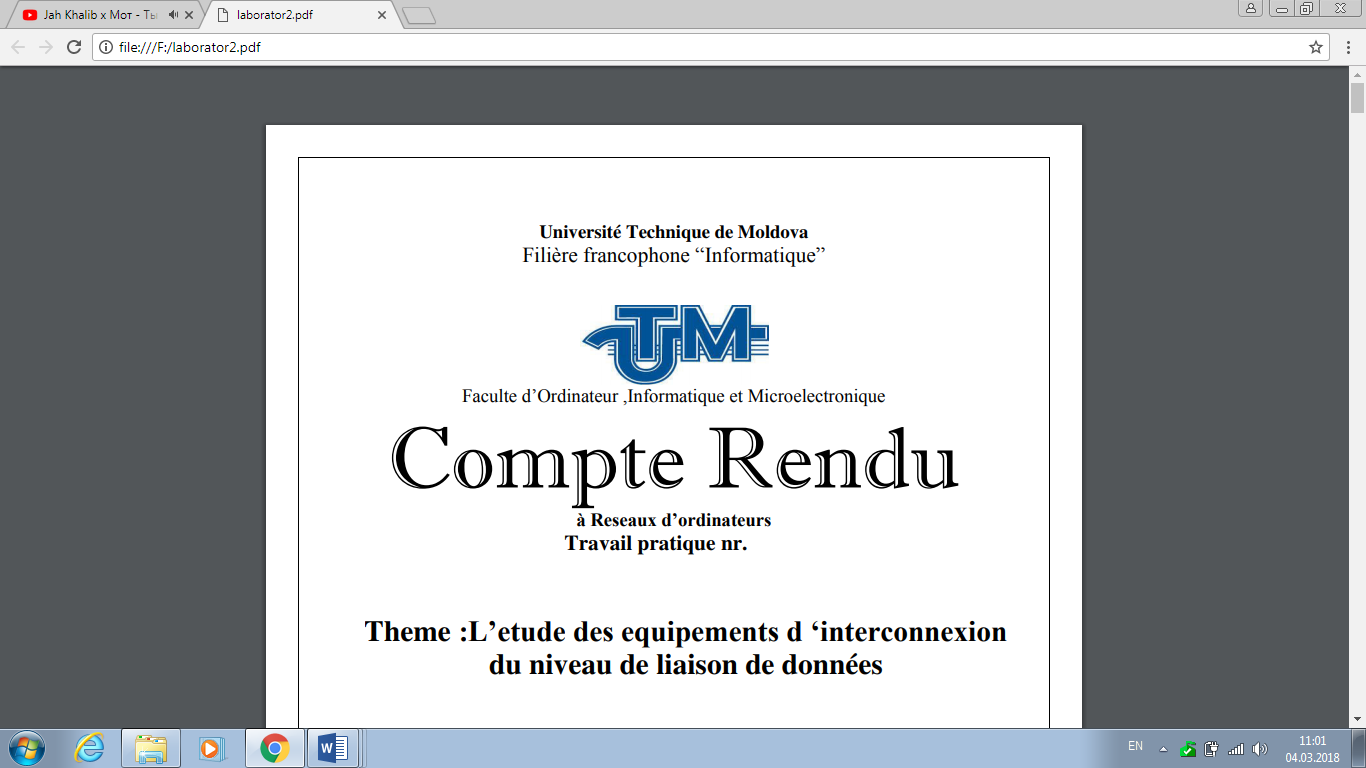
Ministère de l'éducation, de la culture et de la recherche

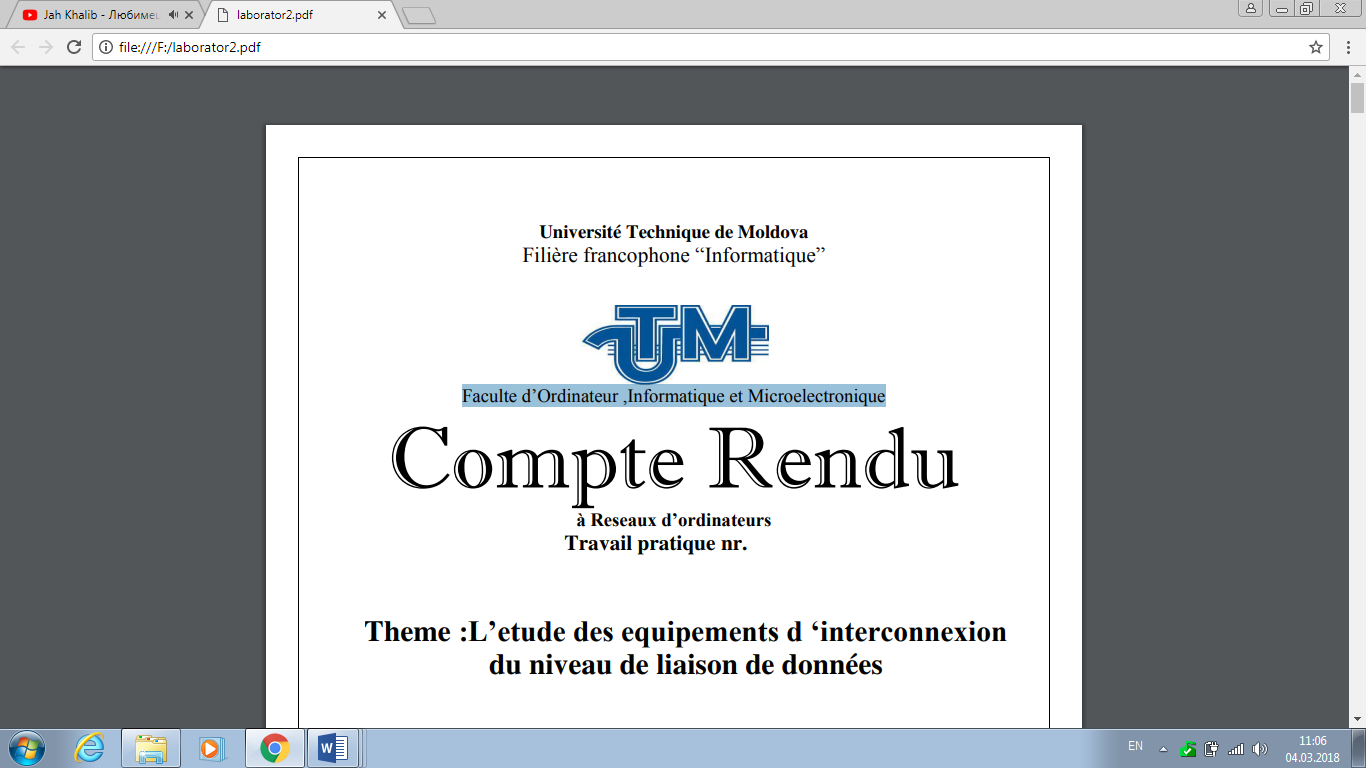
de la République de Moldova

Université technique de Moldavie

Faculté d’Ordinateur, Informatique et Microélectronique

Filière francophone “Informatique”





**Tests de logiciels**

Travail pratique nr.4

**Thème: TEST UNITAIRE**

Effectué par l’étudiant(e) de gr FI-181 : Bonta Alexandr

Vérifié par le professeur : Prisăcaru Andrian

Chișinau 2021

Sarcina 1

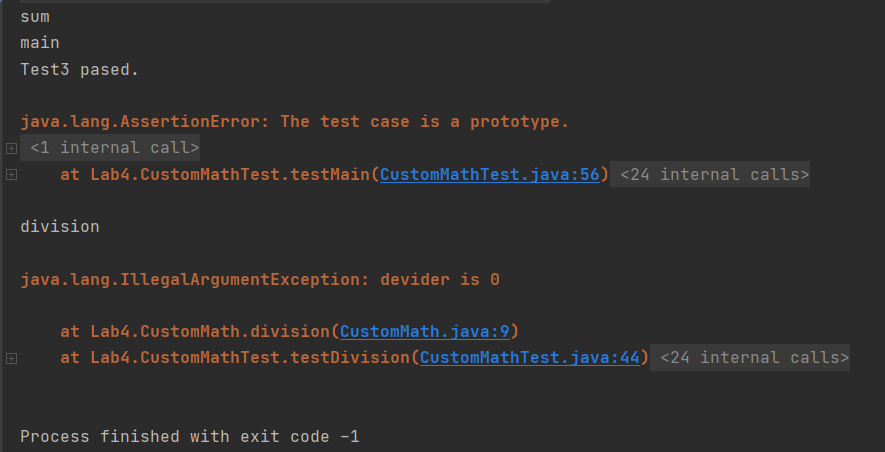
Sa creat un proiect cu clasa CustomMath.

Sa omis din metoda main a clasei CustomMath verificarea funcției sum.

Sa omis din metoda testSum apelul metodei fail. Sa asigurat că testarea funcției sum trece pentru datele de intrare curente.

Codul funcției testSum și rezultatul testării (Figura 1).

**@Test** public void testSum() {  
 System.*out*.println(**"sum"**);  
 int x = 0;  
 int y = 0;  
 int expResult = 0;  
 int result = CustomMath.*sum*(x, y);  
 *assertEquals*(expResult, result);  
 }



(Figura 1)

Sarcina 2

Sa modificat testul testDivision.

Sa omis verificarea metodei division din metoda main a clasei CustomMath.

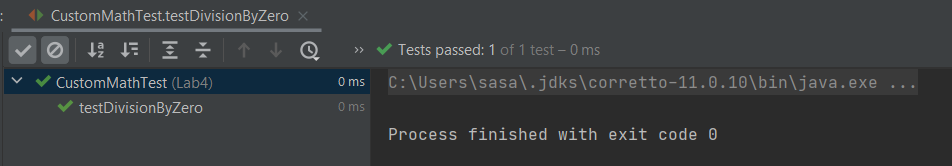
Am pornit testarea pentru y=0 și y!=0 (manual, schimbînd secvențial valorile inițiale a lui y).

Testul testDivisionByZero (test1: y == 0) rezultatele sunt reprezentate in Figura 2.

Testul testDivisionByZero (test2: y != 0) rezultatele sunt reprezentate in Figura 3.

Test1 y == 0

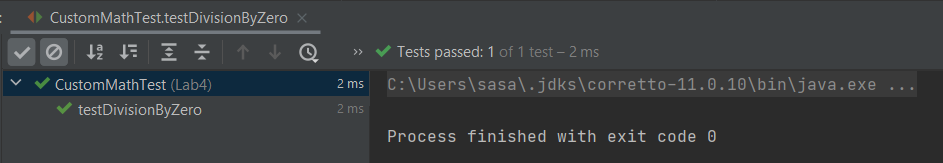
**@Test**public void testDivisionByZero() {  
 int x = 0;  
 int y = 0;  
 int expResult = 0;  
 try {  
 int result=CustomMath.*division*(x, y);  
 *assertEquals*(expResult, result);  
 if(y==0) *fail*(**"Деление на ноли не создает исключителънои ситуации"**);  
 }  
 catch(IllegalArgumentException e){  
 if(y!=0) *fail*(**"Генерация исключения при ненулевом знаменателе"**);  
 }  
}



(Figura 2)

Test2 y != 0

**@Test**public void testDivisionByZero() {  
 int x = 0;  
 int y = 19;  
 int expResult = 0;  
 try {  
 int result=CustomMath.*division*(x, y);  
 *assertEquals*(expResult, result);  
 if(y==0) *fail*(**"Деление на ноли не создает исключителънои ситуации"**);  
 }  
 catch(IllegalArgumentException e){  
 if(y!=0) *fail*(**"Генерация исключения при ненулевом знаменателе"**);  
 }  
}



(Figura 3)

Sarcina 3

Sa modificat metoda de testare testDivisionByZero (), astfel încât funcția să verifice împărțirea la zero, și de asemenea să furnizeze date de intrare corecte.

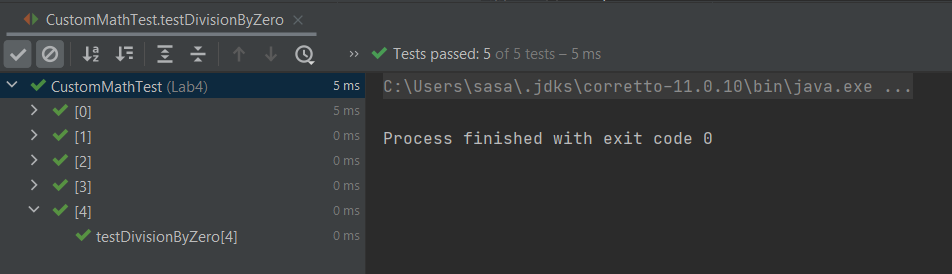
Clasa CustomMath și Clasa CustomMathTest și rezultatele testării (Figura 4).

Clasa CustomMath

public class CustomMath {  
 public static int sum(int x, int y) {  
 return x + y;  
 }  
  
 public static int division(int x, int y) throws IllegalArgumentException {  
 if (y == 0) {  
 throw new IllegalArgumentException(**"devider is 0"**);  
 }  
 return (x / y);  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

Clasa CustomMathTest

**@RunWith**(Parameterized.class)  
public class CustomMathTest {  
  
 **@Parameterized.Parameters** public static Collection divisionValues(){  
 return Arrays.*asList*(new Object[][]{  
 {10,2,5},  
 {-30,-6,5},  
 {0,15,0},  
 {15,0,0},  
 {0,0,0}});  
 }  
 int x,y,divResult;  
  
 public CustomMathTest(int x,int y, int divResult) {  
 this.x=x;  
 this.y=y;  
 this.divResult=divResult;  
 }  
  
 **@BeforeClass** public static void setUpClass() {  
 }  
  
 **@AfterClass** public static void tearDownClass() {  
 }  
 */\*\*  
 \* Test of sum method, of class CustomMath.  
 \*/* **@Test** public void testSum() {  
 System.*out*.println(**"sum"**);  
 int x = 0;  
 int y = 0;  
 int expResult = 0;  
 int result = CustomMath.*sum*(x, y);  
 *assertEquals*(expResult, result);  
// fail("The test case is a prototype.");  
 }  
  
 */\*\*  
 \* Test of division method, of class CustomMath.  
 \*/* **@Test** public void testDivisionByZero() {  
 int x = this.x;  
 int y = this.y;  
 int expResult = divResult;  
 try {  
 int result=CustomMath.*division*(x, y);  
 *assertEquals*(expResult, result);  
 if(y==0) *fail*(**"Деление на ноли не создает исключителънои ситуации"**);  
 }  
 catch(IllegalArgumentException e){  
 if(y!=0) *fail*(**"Генерация исключения при ненулевом знаменателе"**);  
 }  
 }  
  
 */\*\*  
 \* Test of main method, of class CustomMath.  
 \*/* **@Test** public void testMain() {  
 System.*out*.println(**"main"**);  
 String[] args = null;  
 CustomMath.*main*(args);  
 *fail*(**"The test case is a prototype."**);  
 }  
}



(Figura 4)

Sarcina 4

Sa extins clasa de testare, astfel încât să utilizeze metoda assertTrue și / sau assertFalse.

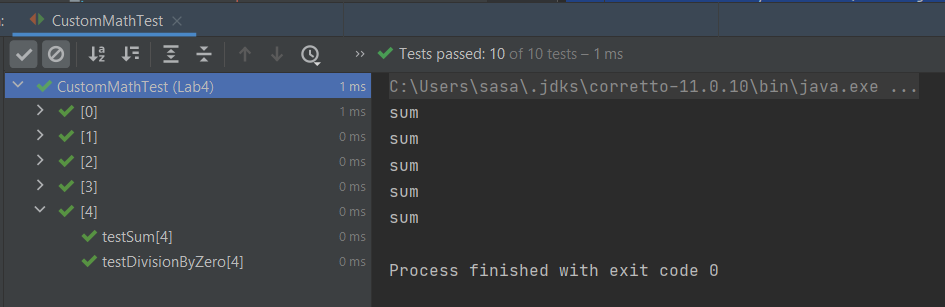
Clasele modificate (CustomMath și CustomMathTest) și rezultatele testării (Figura 5).

Clasa CustomMath

public class CustomMath {  
 public static int sum(int x, int y) {  
 return x + y;  
 }  
  
 public static int division(int x, int y) throws IllegalArgumentException {  
 if (y == 0) {  
 throw new IllegalArgumentException(**"devider is 0"**);  
 }  
 return (x / y);  
 }  
  
 public static void main(String[] args) {  
  
 }  
}

CustomMathTest

**@RunWith**(Parameterized.class)  
public class CustomMathTest {  
  
 **@Parameterized.Parameters** public static Collection divisionValues() {  
 return Arrays.*asList*(new Object[][]{  
 {10, 2, 5},  
 {-30, -6, 5},  
 {0, 15, 0},  
 {15, 0, 0},  
 {0, 0, 0}});  
 }  
  
 int x, y, divResult;  
  
 public CustomMathTest(int x, int y, int divResult) {  
 this.x = x;  
 this.y = y;  
 this.divResult = divResult;  
 }  
  
 **@BeforeClass** public static void setUpClass() {  
 }  
  
 **@AfterClass** public static void tearDownClass() {  
 }  
  
 */\*\*  
 \* Test of sum method, of class CustomMath.  
 \*/* **@Test** public void testSum() {  
 System.*out*.println(**"sum"**);  
 int x = 0;  
 int y = 0;  
 int expResult = 0;  
 int result = CustomMath.*sum*(x, y);  
 *assertTrue*(expResult == result);  
// fail("The test case is a prototype.");  
 }  
  
 */\*\*  
 \* Test of division method, of class CustomMath.  
 \*/* **@Test** public void testDivisionByZero() {  
 int x = this.x;  
 int y = this.y;  
 int expResult = divResult;  
 try {  
 int result = CustomMath.*division*(x, y);  
 *assertFalse*(expResult != result);  
 if (y == 0) *fail*(**"Деление на ноли не создает исключителънои ситуации"**);  
 } catch (IllegalArgumentException e) {  
 if (y != 0) *fail*(**"Генерация исключения при ненулевом знаменателе"**);  
 }  
 }  
}



(Figura 5)

Concluzii:

În lucrarea dată am studiat testarea unitară. Efectuând lucrarea de laborator am învățat cum de folosit clasa JUnit, @BeforeClass, @AfterClass, @Test și metodele assert. La fel am învățat sa introduc parametri pentru testare (@Parameters) și să execut testarea cu clasa Parametrized.

Programul pe github: <https://github.com/sasa-bonta/TestareaSoftware>