Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică şi Microelectronică

Departamentul Ingineria Software și Automatică

**RAPORT**

Lucrarea de laborator nr. 5

# La disciplina „Testarea Produselor Program”

A efectuat: st. gr. SI-211 S. Chirita

A verificat: dr. conf.univ. A. Prisacaru

Chișinău – 2024

**LUCRAREA DE LABORATOR NR.5**

**Elaborarea sarcinii, rezultate:**

**Sarcina 1**

În secvența de cod prezentată mai jos se observă crearea unui proiect cu clasa CustomMath. Are loc omiterea din metoda main a clasei CustomMath care anterior verifica funcțiea sum. De asemenea, are loc omiterea din metoda testSum apelul metodei fail. Funcția sum trece pentru datele de intrare curente.

Este adăugată în raport codul funcției testSum și rezultatul testării (rezultatele testelor PrtSc al ferestrei).

using System;

public static class CustomMath{

public static int Sum(int x, int y){

return x + y;

}

public static int Division(int x, int y) {

if (y == 0) {

throw new ArgumentException("devider is 0");

}

return (x / y);

}

public static void Main(string[] args){}

}

public void TestSum(int x, int y, int expected) {

int result = CustomMath.Sum(x, y);

Assert.AreEqual(expected, result);

}

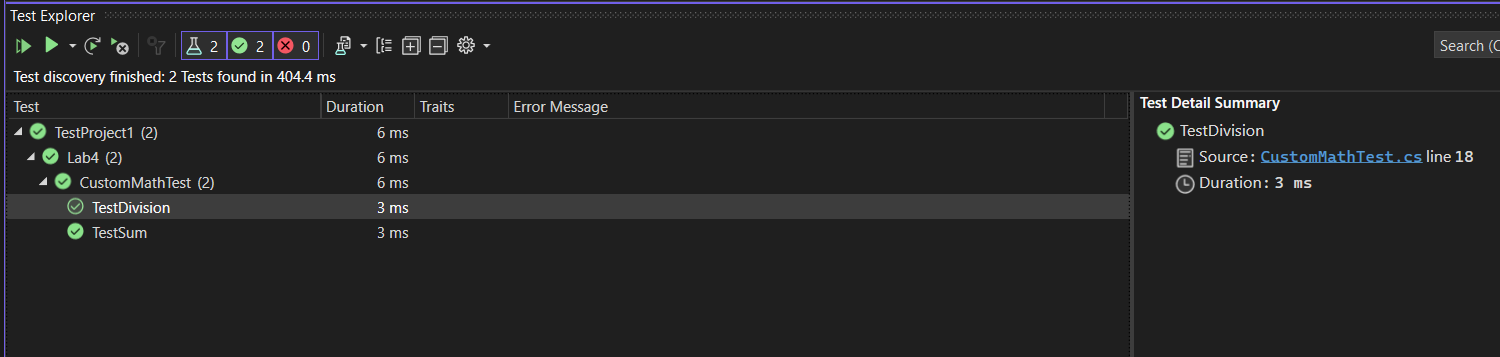


Figura 1 - Rezultatul testării pentru testul TestSum

**Sarcina 2**

Omiterea verificarii metodei division din metoda main a clasei CustomMath.

public static void Main(string[] args){}

Modificați testul testDivision pentru exemplul propus în sarcină.

namespace Lab4

{

[TestClass]

public class CustomMathTest{

[TestMethod]

public void TestSum() {

int x = 2;

int y = 3;

int expResult = 5;

int result = CustomMath.Sum(x, y);

Assert.AreEqual(expResult, result);

}

[TestMethod]

public void TestDivision(){

int x = 6;

int y = 3;

int expResult = 2;

int result = CustomMath.Division(x, y);

Assert.AreEqual(expResult, result);

}

}

}

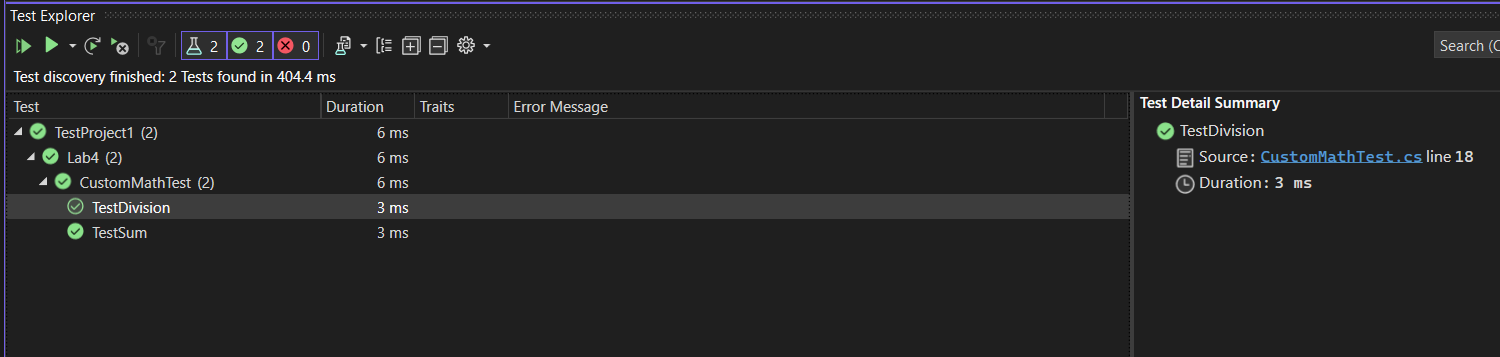


Figura 2 - Rezultatul testării pentru testul TestSum și TestDevision

**Sarcina 3**

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

using System;

public static class CustomMath{

public static int Sum(int x, int y){

return x + y;

}

public static int Division(int x, int y) {

if (y == 0) {

throw new ArgumentException("devider is 0");

}

return (x / y);

}

public static void Main(string[] args) {}

}

namespace Lab4{

[TestClass]

public class CustomMathTest{

[TestMethod]

[DataRow(1, 1, 2)]

[DataRow(-1, 1, 0)]

[DataRow(10, 15, 25)]

public void TestSum(int x, int y, int expected) {

int result = CustomMath.Sum(x, y);

Assert.AreEqual(expected, result);

}

[TestMethod]

public void TestDivision() {

int x = 6;

int y = 3;

int expResult = 2;

int result = CustomMath.Division(x, y);

Assert.AreEqual(expResult, result);

}

[TestMethod]

public void TestDivisionByZero() {

int x = 1;

int y = 0;

int expResult = 0;

try {

int result = CustomMath.Division(x, y);

Assert.AreEqual(expResult, result);

if (y == 0) Assert.Fail("Division by zero should not throw an exception.");

}

catch (ArgumentException e) {

if (y != 0) Assert.Fail("Exception thrown for non-zero denominator.");

}

}

}

}

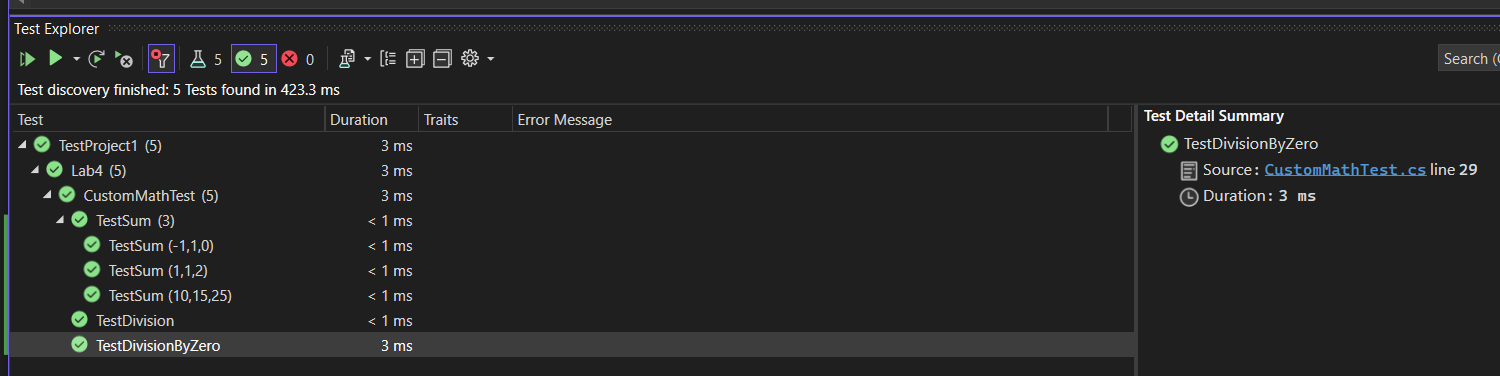


Figura 3 - Rezultatul testărilor pentru testDivisionByZero modificată

**Sarcina 4**

Extinderea clasei de testare, astfel încât să utilizeze metoda assertTrue și / sau assertFalse.

try {

int result = CustomMath.Division(x, y);

Assert.AreEqual(expResult, result);

Assert.IsTrue(y != 0, "Division by zero should not throw an exception.");

}

catch (ArgumentException e) {

Assert.IsFalse(y == 0, "Exception thrown for non-zero denominator.");

}

Assert.IsTrue este folosit pentru cazul în care o condiție particulară este adevărată, iar Assert.IsFalse o condiție particulară este falsă.

**Concluzie**

În cea de-a patra lucrare de laborator, în cadrul disciplinei "Testarea Software" am avut oportunitatea de a experimenta testarea unitară prin intermediul unui IDE la alegre. Am realizat sarcinile propuse în îndrumarul de laborator în Visual Studio în limbajul de programeare C#. Pentru fiecare sarcină propusă am realizat, pas cu pas, cerințele pentru ficare test unitar aparte TestSum, TestDivision, TestDivisionByZero. Ca urmare pot conclude că am îndeplinit cu succes sarcina acestui laborator.