**Modulli yoki iteratsion testlash ma’lumotlar to’plamini yaratish**

Modulli testlash- bu dastur manba kodining alohida modullarining to'g'riligini tekshirishga imkon beradigan dasturlash jarayoni.

Sinovlar har qanday ahamiyatsiz bo'lmagan funktsiya yoki usul uchun yoziladi. Bu sizga kodning o'zgarishi dasturning allaqachon sinovdan o'tgan qismlarida xatolar paydo bo'lishiga olib kelganligini tezda tekshirishga imkon beradi, shuningdek, bunday xatolarni aniqlash va yo'q qilishni osonlashtiradi.

Modulli testlash dasturning alohida qismlarini ajratib turadi va alohida qismlarning ishlashini ko'rsatadi.

Modulli testlash qayta ishlash uchun foydalidir va uning hali ham to'g'ri ishlashini ta'minlaydi (regressiya sinovi). Sinovga pastdan yuqoriga qarab yondashish uchun modulli sinovidan foydalaniladi: avval dasturning alohida qismlari, so'ngra butun dastur.

Modulli testlashdan foydalanilmaydi:

- Kombinatorial muammo hal qilinmoqda. Masalan, mantiqiy o'zgaruvchining har bir mumkin bo'lgan qiymati uchun ikkita test kerak bo'ladi: biri TRUE uchun, ikkinchisi FALSE uchun. Natijada, har bir kod manbai uchun 3-5 qator sinov kodi talab qilinadi.

- Natija faqat taxminan ma'lum. Masalan, matematik modellashtirishda ko'p hollarda modellashtirish sifati "ko'z bilan" aniqlanadi va oxirgi natija "ma'lumotnoma" sifatida qayd etiladi. Agar nomuvofiqlik aniqlansa, yangisi natija qo'lda tekshiriladi va qaysi biri yaxshiroq ekanligi aniqlanadi: eski yoki yangi.

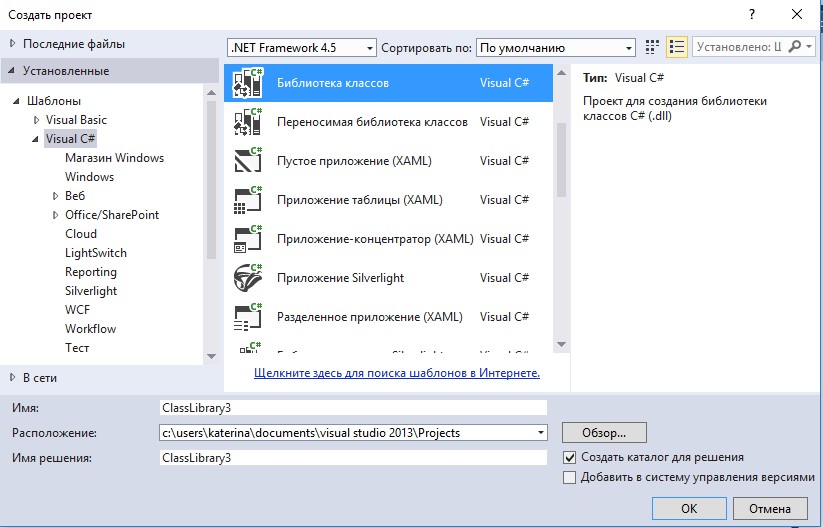
Modulli testlashlarini o'tkazishda modullarning har biri alohida sinovdan o'tkaziladi. Bu shuni anglatadiki, integratsiya, tizim darajasi, bir nechta modullarda bajarilgan funktsiyalar xatolari aniqlanmaydi. Ushbu texnologiya ishlash sinovlari uchun ham foydasiz. Modulli testlashboshqa sinov usullari bilan birgalikda ishlatilganda samaraliroq bo'ladi.

Modulli testlashdan foyda olish uchun dasturiy ta'minotni ishlab chiqish jarayonida sinov texnologiyasiga qat'iy rioya qilish kerak. Faqatgina barcha o'tkazilgan testlarning yozuvlarini emas, balki barcha modullarda manba kodidagi barcha o'zgarishlarni ham saqlash kerak. Shu maqsadda dasturiy ta'minot versiyasini boshqarish tizimidan foydalanish kerak. Agar dasturiy ta'minotning keyingi versiyasi avval muvaffaqiyatli o'tgan testdan o'ta olmasa, manba kodining variantlarini tekshirish va xatoni tuzatish oson bo'ladi. Shuningdek, muvaffaqiyatsiz testlarni kuzatish va tahlil qilish kerak. Ushbu talabni e'tiborsiz qoldirish testning muvaffaqiyatsiz natijalarini ko'payishiga olib keladi.

Ekstremal dasturlash postulatlardan biri sifatida avtomatlashtirilgan modulli sinov vositalaridan foydalanishni o'z ichiga oladi. Ekstremal dasturlash sinovdan o'tkaziladigan rivojlanish uchun modulli sinovlaridan foydalanadi. Buning uchun dasturchi kod yozishdan oldin modulga qo'yiladigan talablarni aks ettiruvchi test yozadi. Shubhasiz, kodni yozishdan oldin test ishlamasligi kerak. Jarayonning qolgan qismi ushbu testni qondiradigan eng qisqa kodni yozishga qisqartiriladi. Shundan so'ng, ishlab chiquvchi keyingi testni, kodni va boshqalarni bir necha bor yozadi.

# Visual Studio bilan modulli testini yaratish

Modulli testlarini yozishda testning asosiy qismini yozishning izchil uslubiga rioya qiling. AAA (tartibga solish, harakat qilish, tasdiqlash) yondashuvi o'zini yaxshi isbotladi. Raqamlar yig'indisini hisoblashning to'g'riligini tekshiradigan usul uchun test yozamiz. Buning uchun Visual Studio-da yangi Visual C # loyihasini yarating va "Sinf kutubxonasi" ni tanlang (1-rasm).



1-rasm. Yangi loyiha oynasi

Keling, buni MySalculation deb ataymiz. "Class1" ning nomini "MyCalc" ga o'zgartiring.

namespace MyCalc

{

public class MyCalc

{

public int sum(int x, int y)

{

return x + y;

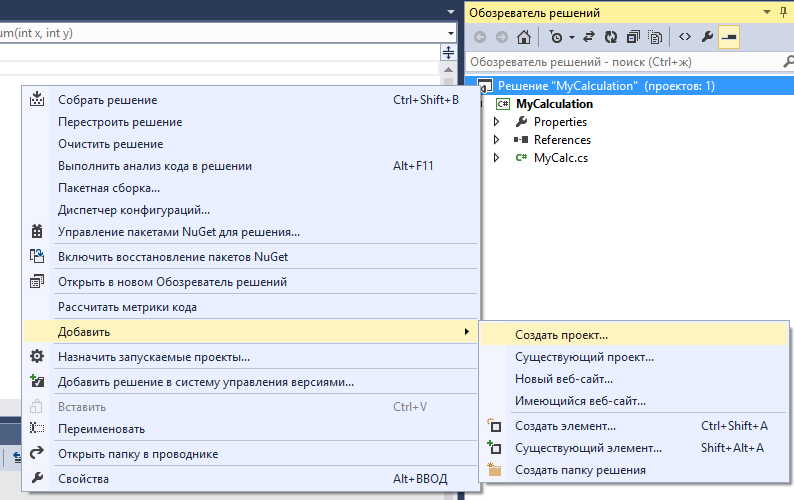
}

}

}

Har bir sinf uchun modulli testini yaratishda,

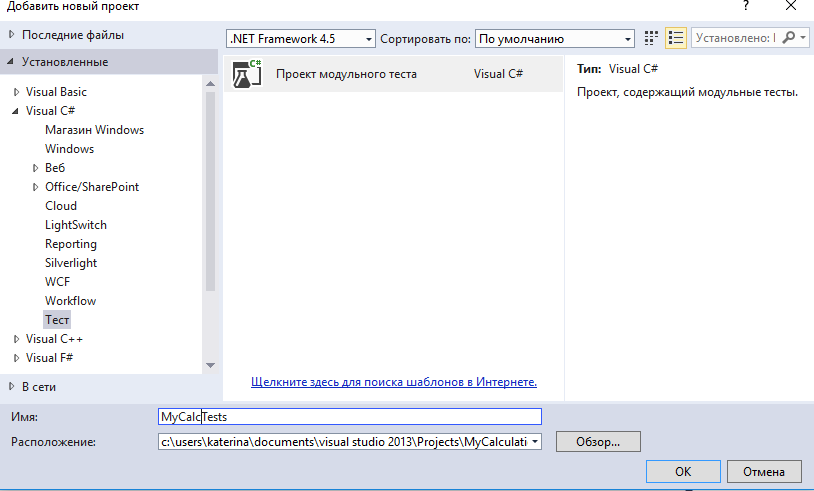
sinov ostida, alohida modulli sinov fayli yaratiladi. Har bir modulli sinov fayli har bir tekshirilayotgan usul uchun sinov usulini o'z ichiga oladi. Buning uchun xuddi shu echim doirasida tegishli turdagi boshqa loyihani yaratish kerak. Yechimni o'ng tugmasini bosing, "Qo'shish" -ni tanlang va keyin "Loyihani yarating ..." (2-rasm).



2-rasm. "Loyihani echimga qo'shish"

Ochilgan oynada Visual C # guruhida "Sinov" tugmachasini bosing va keyin "Modulli sinov loyihasi" ni tanlang 3-rasm. Modulli testlarini yaratishda o'zgaruvchan nomlash konventsiyalari qo'llaniladi, modulli test faylining nomi quyidagicha olinadi

tekshirilayotgan kodni o'z ichiga olgan fayl nomi bilan "Sinov" so'zini qo'shish. Bizning misolimizda bu "MyCalcTest.cs". "MyCalcTests" loyiha nomini kiriting va "OK" tugmasini bosing. Shu tarzda loyiha yaratiladi.



3.Rasm "Modulli sinov loyihasini qo'shish" oynasi

Siz quyidagi kodni ko'rasiz:

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace MyCalcTests

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethod1()

{

}

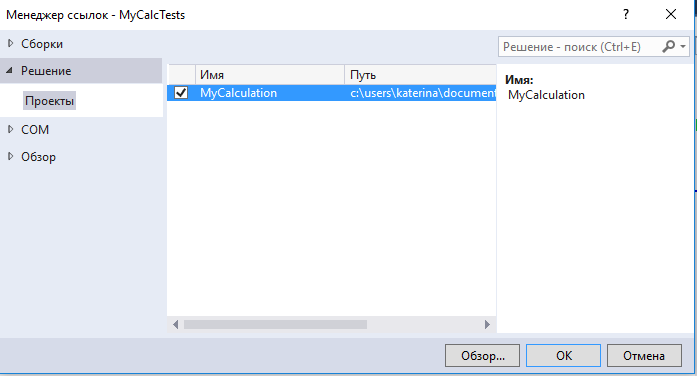
}

}

[TestMethod] direktivasi keyingi usul ekanligini ko'rsatadi,

modulli sinovini o'z ichiga olgan. Va [TestClass] ning aytishicha, navbatdagi modulli testlari mavjud bo'lgan usullarni o'z ichiga olgan sinf keladi.

Keling, "UnitTest1" sinfini "MyCalcTests" deb o'zgartiring. Keyin loyiha havolalarida siz loyihaga havolani qo'shishingiz kerak, uning kodini biz tekshiramiz. Adabiyotlarni o'ng tugmasini bosing va "Havola qo'shish ..." ni tanlang. Ko'rsatilgan oynada "Qaroringiz" guruhini oching, "Loyihalar" ni tanlang va MySalculation loyihasi oldiga belgi qo'ying..



4-rasm. Link Manager oynasi

Shuningdek, kodda quyidagi nom maydonini foydalanish direktivasi yordamida ulashingiz kerak: MySalculation yordamida.

Endi test yozamiz. Dastur 10 va 20-sonli yig'indilarni hisoblab chiqadimi-yo'qligini tekshirib ko'ramiz. Kutilgan natija (to'g'ri)

yechim) bu holda u 30-songa teng. TestMethod1 () usulini sum\_10and20\_30returned () ga o'zgartiring. Yangi usul nomi nima tekshirilishini va qanday qiymatlar (10 va 20) va to'g'ri natija sifatida nimani kutishini tushuntiradi (30).

Sinov usuli odatda uchta zarur komponentni o'z ichiga oladi:

- dastlabki ma'lumotlar: kirish qiymatlari va kutilgan natija (tartibga solish);

- tekshirilayotgan usul yordamida qiymatni hisoblaydigan kod

(harakat);

- kutilgan natijani olingan bilan solishtiradigan kod

(tasdiq).

Sinov kodi shunday bo'ladi:

namespace MyCalcTests

{

[TestClass]

public classMyCalcTests

{

[TestMethod]

public void sum\_10and20\_30returned()

{

//arrange int x = 10; int y = 20;

int expected = 30;

//ast

MyCalc c = new MyCalc(); int actual = c.sum(x, y);

//assert

Assert.AreEqual(expected, actual);

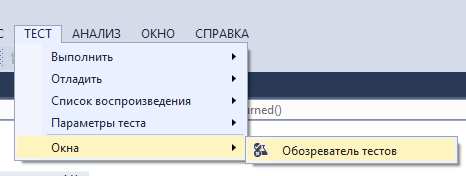
}

}

}

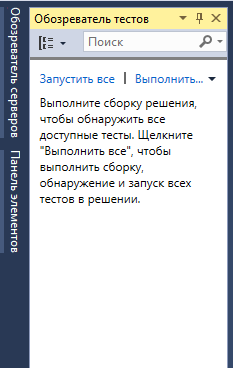
Kutilayotgan natijani olingan natijaga solishtirish uchun "Assert" sinfining AreEqual () usuli qo'llaniladi. Ushbu sinf har doim Visual Studio-da birlik testlarini yozishda foydalaniladi.

Amalga oshiriladigan barcha testlarni ko'rish uchun Test Explorer oynasini ochishingiz kerak. Buning uchun Visual Studio menyusida "TEST" tugmachasini bosing, "Windows" -ni tanlang va keyin "Test Explorer" bandini bosing (5-rasm).



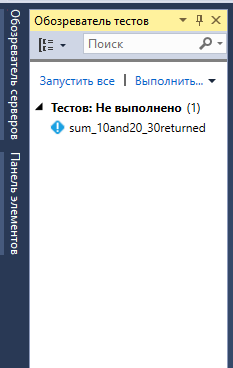
5-rasm. Test Explorer oynasini qo'shish

Studiyada quyidagi oyna paydo bo'ladi (6-rasm):



6-rasm. Sinov Explorer oynasi

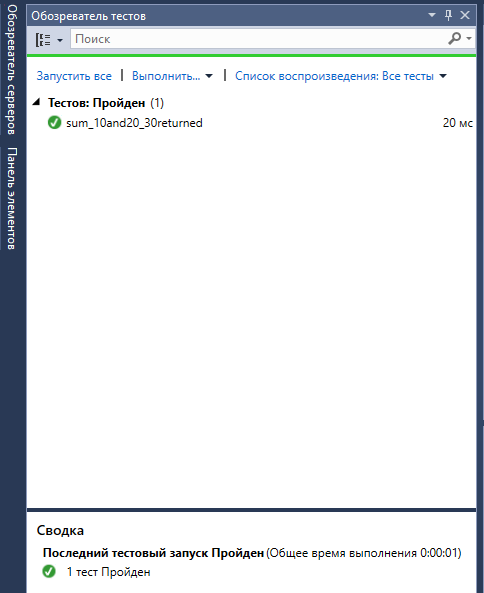
Sinovlar ro'yxati hozirda bo'sh, chunki bu yechim hali hech qachon tuzilmagan. Keling, yig'ilishni Ctrl + Shift + B tugmachalarini bosib bajaraylik, u tugagandan so'ng biz "Test Explorer" da o'zimizning testimizni ko'ramiz (7-rasm)..



7-rasm. Qurilish tugagandan so'ng Test Explorer oynasi

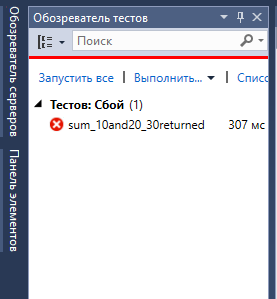
Moviy undov belgisi belgilangan sinov hech qachon bajarilmaganligini bildiradi. Keling, buni amalga oshiraylik. Buning uchun uning nomini o'ng tugmasini bosing va "Tanlangan testlarni ishga tushirish" -ni tanlang.

Belgilangan belgisi bo'lgan yashil doira birlik sinovidan muvaffaqiyatli o'tganligini anglatadi: kutilgan va olingan natijalar teng (8-rasm).



8-rasm. Tanlangan testlar muvaffaqiyatli bajarilgandan so'ng Test Explorer oynasi

Raqamlarning yig'indisini hisoblaydigan sum\_10and20\_30returned () usuli uchun kodni o'zgartiramiz, test natija bermaganligini simulyatsiya qilish va Visual Studio o'zini qanday tutishini ko'rish uchun. Qaytgan qiymatga 10 qo'shing, testni o'tkazing (9-rasm).



9-rasm. Sinov muvaffaqiyatsiz tugaganidan so'ng Test Explorer oynasi

Shunday qilib, biz Visual Studio-da C # -da eng oddiy modulli testini qanday yaratishni bilib oldik.

# Ism maydoni Microsoft.VisualStudio.TestTools.UnitTesting

Microsoft.VisualStudio.TestTools.UnitTesting nom maydonida modullik sinovlarini o'tkazish uchun sinflar mavjud. Keling, ushbu sinflarni batafsil ko'rib chiqaylik [6].

Assert klassi testlarda ishlatilishi mumkin bo'lgan statik usullar to'plamini belgilaydi. "Assert" klassi eng ko'p ishlatiladigan sinflardan biridir. Usullar 4.1-jadvalda tasvirlangan.

Assert sinfidagi har bir statik usul birlik testining ba'zi jihatlarini sinab ko'rishga imkon beradi va agar test muvaffaqiyatsiz tugasa, bu usullar istisno qiladi. Birlik sinovidan o'tish uchun barcha tasdiqlar muvaffaqiyatli bo'lishi kerak.

"Assert" sinfidagi har bir usulda string parametrini qabul qiladigan ortiqcha yuklangan versiyasi mavjud. Agar tasdiqlash bajarilmasa, ushbu satr istisno ob'ekti ichidagi xabar elementiga joylashtiriladi.

AreEqual () va AreNotEqual () usullari ma'lum turlarni taqqoslash uchun bir nechta ortiqcha yuklarga ega. Masalan, satrlarni kichik harflar bilan taqqoslashga imkon beruvchi versiya mavjud.

Таблица 4.1

"Assert" klassi metodlarining tavsifi

|  |  |
| --- | --- |
| Usuli | Tavsifi |
| AreEqual<T>(T, T); AreEqual<T>(T, T, string) | T tipidagi ikkita ob'ekt bir xil qiymatga ega ekanligini tasdiqlaydi |
| AreNotEqual<T>(T, T);  AreNotEqual<T>(T, T, string) | T tipidagi ikkita ob'ekt bitta narsaga ega emasligini tasdiqlaydi  va bir xil qiymat |
| AreSame<T>(T, T);  AreSame<T>(T, T, string) | Ikki o'zgaruvchiga tegishli ekanligini tasdiqlaydi  xuddi shu ob'ekt |
| AreNotSame<T> (T, T); AreNotSame<T>(T, T, string) | Ikki o'zgaruvchining turli xil ob'ektlarga murojaat qilishini tasdiqlaydi |
| Fail();  Fail(string) | Bayonotning salbiy natijasi - yo'q  shartlar tasdiqlanmagan |
| Inconclusive(); Inconclusive(string) | Modulli testlash natijasini aniq belgilash mumkin emasligini bildiradi |
| IsTrue(bool); IsTrue(bool, string) | Mantiqiy qiymat to'g'ri ekanligini tasdiqlaydi -  mantiqiy natijani beradigan ifodani baholash uchun eng ko'p ishlatiladi |
| IsFalse(bool); IsFalse(bool, string) | Mantiqiy qiymat yolg'on ekanligini tasdiqlaydi |
| IsNull(object);  IsNull(object, string) | O'zgaruvchining tayinlanmaganligini tasdiqlaydi  ob'ektga mos yozuvlar |
| IsNotNull(object);  IsNotNull(object, string) | Ob'ektga o'zgaruvchining tayinlanganligini tasdiqlaydi  havola |
| IsInstanceOfType(object, Type);  IsInstanceOfType(object, Type, string) | Ob'ekt ko'rsatilgan narsaga tegishli ekanligini tasdiqlaydi  turi yoki belgilangan turdan kelib chiqqan |
| IsNotInstanceOfType(object,Type);  IsNotInstanceOfType(object, Type, string) | Ob'ekt belgilangan turdagi emasligini tasdiqlaydi |

Keling, berilgan sonning kvadrat ildizini hisoblaydigan usul uchun test yozamiz.

namespace Class Ass

{

public class MyClass

{

public static double GetSgrt(double value)

{

return Math.Sqrt(value);

}

}

}

Ushbu usulni sinovdan o'tkazishda ikkita variantni ko'rib chiqish kerak

echimlar, kvadrat ildizni hisoblashda biz butun sonni olamiz - IsSqrtTest () usuli va tenglik uchun qiymatlarni tekshirishda xatolik hisobga olingan holda bajariladi, agar eritmada kasr qismi bo'lsa, DeltaTest () usuli.

namespace ClassAssTest

{

[TestClass]

public class MyClassTest

{

[TestMethod] public void IsSqrtTest()

{

// arrange

const double input = 4; const double expected = 2;

//act

double actual = ClassAss.MyClass.GetSgrt(input);

//assert

Assert.AreEqual(expected, actual, "Sqrt of {0} should have been {1}!", input, expected);

}

[TestMethod] public voidDeltaTest()

{

// arrange

const double expected = 3.1; const double delta = 0.07;

//act

double actual = ClassAss.MyClass.GetSgrt(10);

//assert

//проверка значений на равенство с учетом погрешности Assert.AreEqual(expected, actual, delta, "fail mtssege!");

}

}

}

TestContext sinfi ma'lumotni saqlash uchun ishlatiladi

birlik sinovlari uchun topshirildi. Keling, ushbu sinf ishini misol yordamida ko'rib chiqamiz.

namespace ContexUnitTest

{

[TestClass]

public classUnitTest1

{

public TestContextTestContext { get; set; } // свойство

[TestMethod]

public void TestMethod1()

{

TestContext.WriteLine("TestContext.TestRunDirectory{0}",TestContext.TestRunDir ectory);

TestContext.WriteLine("TestName{0}", TestContext.TestName); TestContext.WriteLine("CurrentTestOutcomey{0}", TestContext.CurrentTestOutcome);// результатработытеста

}

[TestCleanup]

public void TestCleanup()

{

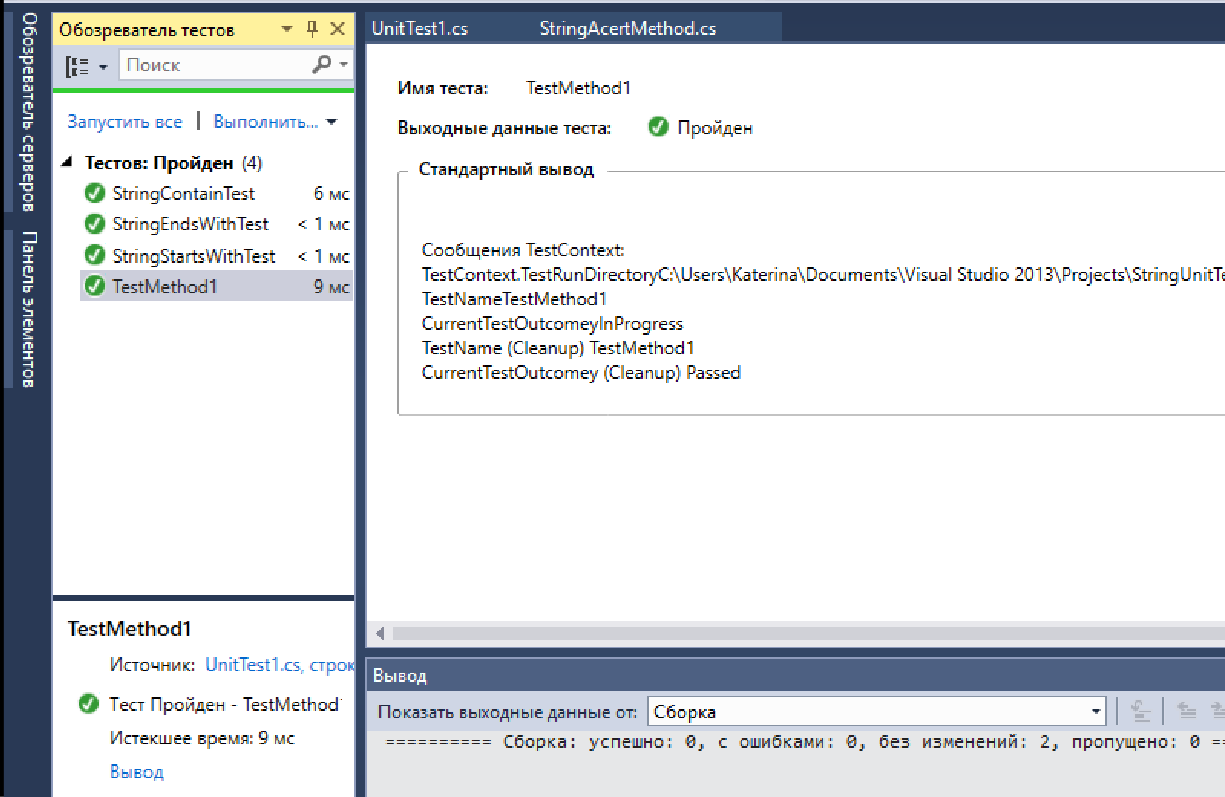
TestContext.WriteLine("TestName (Cleanup) {0}", TestContext.TestName);

TestContext.WriteLine("CurrentTestOutcomey (Cleanup) {0}", TestContext.CurrentTestOutcome);

}

}

}

 Sinov natijasini 9-rasmda ko'rish mumkin..

9-rasm. "TestContext" sinfidan foydalanish natijasi

Qo'shimcha test atributlari:

[ClassInitialize ()] sinfdagi birinchi testni bajarishdan oldin kodni bajarish uchun ishlatiladi.

[ClassCleanUp ()] sinfdagi barcha testlar bajarilgandan so'ng kodni bajarish uchun ishlatiladi.

[TestInitialize ()] har bir test bajarilishidan oldin kodni bajarish uchun ishlatiladi.

[TestCleanUp ()] har bir test bajarilishi tugagandan so'ng kodni bajarish uchun ishlatiladi.

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MinAp

{

public class Program

{

public static int Min(int a, int b, int c) {

int min = a;

if (min > b) min = b;

if (min > c) min = c;

return min;

}

static void Main(string[] args)

{

Console.WriteLine("C# da modulli testlash");

}

}

}

Unittest1.cs

using System;

using System.Text;

using System.Collections.Generic;

using System.Linq;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using MinAp;

namespace UnitTest1

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMin()

{

int min;

min = Program.Min (2, 3, 4);

Assert.AreEqual(2, min);

}

}

}

**Topshiriq**

Dasturni testlashni amaliy ko’rsatib berish.