Calculator.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace testing\_example

{

public class Calculator

{

public int Add(int a,int b)

{

if (a > 0 && b > 0)

{

return a + b;

}

else

{

return 0;

}

}

public int Divide(int a,int b)

{

if (a > b && a>0 && b>0)

{

return a / b;

}

else

{

return 0;

}

}

public int Multiply(int a,int b)

{

if (a > 0 && b > 0)

{

return a \* b;

}

else

{

return 0;

}

}

public int Subtract(int a,int b)

{

if (a > 0 && b > 0 && a > b)

{

return a - b;

}

else

{

return 0;

}

}

}

}

Calculatortestcases.cs

using NUnit.Framework;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using testing\_example;

namespace UnitTestProject1

{

[TestFixture]

[Author("Divya","ddivyabarathi1999@gmail.com")]

[Category("Math")]

class Caculatortestcases

{

private Calculator \_cal;

[SetUp]

public void SetUp()

{

\_cal = new Calculator();

}

[Test,Order(1)]

[TestCaseSource(nameof(AddCases))]

public void Add\_whencalled\_expectedresult(int a,int b)

{

var result = \_cal.Add(a, b);

if (a == 0 || b == 0)

{

Assert.That(result, Is.EqualTo(0));

}

else

{

Assert.That(result, Is.EqualTo(a + b));

}

}

[Test, Order(2)]

[TestCaseSource(nameof(AddCases))]

[Ignore("Because I wanted to!")]

public void Divide\_whencalled\_expectedresult(int a, int b)

{

var result = \_cal.Divide(a, b);

if (a == 0 || b == 0 || a < b)

{

Assert.That(result, Is.EqualTo(0));

}

else

{

Assert.That(result, Is.EqualTo(a / b));

}

}

[Test, Order(3)]

[TestCaseSource(nameof(AddCases))]

public void Multiply\_whencalled\_expectedresult(int a, int b)

{

var result = \_cal.Multiply(a, b);

if (a == 0 || b == 0)

{

Assert.That(result, Is.EqualTo(0));

}

else

{

Assert.That(result, Is.EqualTo(a \* b));

}

}

[Test, Order(4)]

[TestCaseSource(nameof(AddCases))]

public void Subtract\_whencalled\_expectedresult(int a, int b)

{

var result = \_cal.Subtract(a, b);

if (a == 0 || b == 0 || a < b)

{

Assert.That(result, Is.EqualTo(0));

}

else

{

Assert.That(result, Is.EqualTo(a - b));

}

}

static readonly object[] AddCases =

{

new object[] {1,2},

new object[] {0,0},

new object[] {4,2}

};

}

}















