







































  

 

































Класс для теста:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MyCalcLib

{

public class MyCalc

{

public int Sum (int x, int y)

{

return x + y;

}

public int Proiz(int x, int y)

{

return x \* y;

}

public int Razn(int x, int y)

{

return x - y;

}

public int Del(int x, int y)

{

return x / y;

}

public int Cel(int x, int y)

{

return x / y;

}

public int Ost(int x, int y)

{

return x % y;

}

}

}

Тестирующая программа

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using MyCalcLib;

namespace MyCalcLibTests

{

[TestClass]

public class MyCalcLibTests

{

[TestMethod]

public void Sum\_10and20\_returned()

{

//arrange

int x = 10, y = 20;

int expected = 30;

//act

MyCalc c = new MyCalc();

int actual = c.Sum(x, y);

//assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void Proiz\_10and20\_returned()

{

//arrange

int x = 10, y = 20;

int expected = 200;

//act

MyCalc c = new MyCalc();

int actual = c.Proiz(x, y);

//assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void Razn\_10and20\_returned()

{

//arrange

int x = 10, y = 20;

int expected = -10;

//act

MyCalc c = new MyCalc();

int actual = c.Razn(x, y);

//assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void Del\_100and20\_returned()

{

//arrange

int x = 100, y = 20;

int expected = 5;

//act

MyCalc c = new MyCalc();

int actual = c.Del(x, y);

//assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void Cel\_100and20\_returned()

{

//arrange

int x = 100, y = 30;

int expected = 3;

//act

MyCalc c = new MyCalc();

int actual = c.Cel(x, y);

//assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void Ost\_100and20\_returned()

{

//arrange

int x = 100, y = 30;

int expected = 10;

//act

MyCalc c = new MyCalc();

int actual = c.Ost(x, y);

//assert

Assert.AreEqual(expected, actual);

}

}

}

Класс регистрации пользователя для тестирования:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Text.RegularExpressions;

namespace UserRegistration

{

public static class PasswordStrengthCheker

{

// Возвращает значение определяющее сложность пароля пользователя.

public static int GetPasswordStrength(string password)

{

if (string.IsNullOrEmpty(password))

{

return 0;

}

int result = 0;

// +1 балл за длину.

if (Math.Max(password.Length, 7) > 7)

{

result++;

}

//+1 балл за наличие символа в нижнем регистре

if (Regex.Match(password, "[a-z]").Success)

{

result++;

}

//+1 балл за наличие символа в верхнем регистре

if (Regex.Match(password, "[A-Z]").Success)

{

result++;

}

// +1 балл за наличие числа.

if (Regex.Match(password, "[0-9]").Success)

{

result++;

}

// +1 балл за наличие специального символа.

if (Regex.Match(password,

"[\\!\\@\\#\\$\\%\\^\\&\\\*\\(\\)\\{\\}\\[\\]\\:\\'\\;\\\"\\/\\?\\.\\>\\,\\<\\~\\`\\-\\\\\_\\=\\+\\|]").Success)

{

result++;

}

return result;

}

}

}

Программа тестирования класса регистрации пользователя:

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using UserRegistration;

namespace UserRegistrationTests

{

[TestClass]

public class PasswordStrengthChekerTests

{

[TestMethod]

public void GetPasswordStrength\_AllCahrs\_5Points()

{

// arrange

string password = "P2ssw0rd#";

int expected = 5;

// act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_UpperCase\_3Points()

{

// Arrange

string password = "Password";

int expected = 3; // верхний регистр 1, за длину строки 1, за нижний регистр 1

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_ConteinsNumber\_0\_4Points()

{

// Arrange

string password = "Passw0rd";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1

int expected = 4;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_ConteinsNumber\_1\_4Points()

{

// Arrange

string password = "Passw1rd";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1

int expected = 4;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

// Tests for special chars

[TestMethod]

public void GetPasswordStrength\_ContainsSpecialChar\_at\_5Points()

{

// Arrange

string password = "Passw0rd@";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1, специальный символ 1

int expected = 5;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_ContainsSpecialChar\_Hash\_5Points()

{

// Arrange

string password = "Passw0rd#";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1, специальный символ 1

int expected = 5;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_ContainsSpecialChar\_Excl\_5Points()

{

// Arrange

string password = "Passw0rd!";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1, специальный символ 1

int expected = 5;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void GetPasswordStrength\_ContainsSpecialChar\_Doll\_5Points()

{

// Arrange

string password = "Passw0rd$";

// верхний регистр 1, за длину строки 1, за нижний регистр 1

// число 1, специальный символ 1

int expected = 5;

// Act

int actual = PasswordStrengthCheker.GetPasswordStrength(password);

// Assert

Assert.AreEqual(expected, actual);

}

}

}