PasswordChecker

Цель:

Научиться пользоваться исключениями

Применять исключения в коде

Получить хорошую оценку

Получить знание о тестах

Задачи:

* Написать код с исключениями для проверки пароля в соответствии с критериями в таблице
* Написать тест для проверки исключений

Теория:

* Знание языка (в данном случае C#)
* Знать, как работают исключения
* Умение проводить тест
  + - использованы оператор if, методы Any, Assert, Intersect, Char.Upper, Char.Lower, Char.Digit

using Microsoft.VisualStudio.TestTools.UnitTesting;

using ClassLibraryPassword;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibraryPassword.Tests

{

[TestClass()]

public class PasswordCheckerTests

{

[TestMethod()]

public void Check\_12Symbols\_ReturnsTrue()

{

string password = "21121994eeee";

bool expected = true;

bool actual = PasswordChecker.validatePassword(password);

Assert.AreEqual(expected, actual);

}

[TestMethod()]

public void Check\_8Symbols\_ReturnsFalse()

{

string password = "2112199";

bool expected = false;

bool actual = PasswordChecker.validatePassword(password);

Assert.IsFalse(actual);

}

[TestMethod()]

public void Check\_20Symbols\_ReturnsFalse()

{

string password = "21121994573164597834f";

bool expected = false;

bool actual = PasswordChecker.validatePassword(password);

Assert.IsFalse(actual);

}

[TestMethod()]

public void Check\_20Symbols\_ReturnsTrue()

{

string password = "123456789qwertyuiopa";

bool expected = true;

bool actual = PasswordChecker.validatePassword(password);

Assert.AreEqual(expected, actual);

}

[TestMethod()]

public void Check\_numbersSymbols\_ReturnsTrue()

{

string password = "123456789";

bool expected = true;

bool actual = PasswordChecker.validatePassword(password);

Assert.AreEqual(expected, actual);

}

[TestMethod()]

public void Check\_numbersSymbols\_ReturnsFalse()

{

string password = "Qwertyuiop";

bool expected = false;

bool actual = PasswordChecker.validatePassword(password);

Assert.IsFalse (actual);

}

[TestMethod()]

public void Check\_SpecSymbols\_ReturnsTrue()

{

string password = "qwertyu@";

bool expected = true;

bool actual = PasswordChecker.validatePassword(password);

Assert.AreEqual(expected,actual);

}

[TestMethod()]

public void Check\_SpecSymbols\_ReturnsFalse()

{

string password = "Hhjsadasdasd";

bool expected = false;

bool actual = PasswordChecker.validatePassword(password);

Assert.IsFalse (actual);

}

[TestMethod()]

public void Check\_CapsSymbols\_ReturnsTrue()

{

string password = "AsfhjfsdkjKjhfs1@";

bool expected = true;

bool actual = PasswordChecker.validatePassword(password);

Assert.AreEqual(expected,actual);

}

[TestMethod()]

public void Check\_CapsSymbols\_ReturnsFalse()

{

string password = "1280848149845»8320:";

bool expected = false;

bool actual = PasswordChecker.validatePassword(password);

Assert.IsFalse(actual);

}

[TestMethod()]

public void Check\_UPSymbols\_ReturnsTrue()

{

string password = "ASA@1987DFE:";

bool expected = true;

bool actual = PasswordChecker.validatePassword(password);

Assert.AreEqual(expected,actual);

}

[TestMethod()]

public void Check\_UPSymbols\_ReturnsFalse()

{

string password = "dejh#93875jsapo";

bool expected = false;

bool actual = PasswordChecker.validatePassword(password);

Assert.IsFalse(actual);

}

public class PasswordChecker

{

public static bool validatePassword (string password)

{

if (password.Length < 8 || password.Length > 20)

return false;

if (!password.Any(char.IsLower))

return false;

if (!password.Any(char.IsUpper))

return false;

if (!password.Any(char.IsDigit))

return false;

if (password.Intersect ("!@#$%^&\*()").Count() == 0)

return false;

return true;

}

}

}

}