# ClassLibraryPassword

Сделала Буланина А.И. группа 195 дата: 07.09.2021

Требования для вводимого пароля.

1)Количество символов от 8 до 20

2)Наличие цифр

3)Наличие спецсимволов

4)Наличие прописных и строчных букв

1. Цель: ЗАЧЕМ вы это делали? Написать модульный тест для функции проверки корректности пароля
2. Задачи: ЧТО вы должны были сделать? Таблицу в word файле, и сам проект в visual studio.
3. Теория: что надо ЗНАТЬ, чтобы выполнить задачи? Название шаблона и как его найти в visual studio; как создать тестовый проект для нужного метода; команды и их значение

Any(Char.IsDigit)

Проверяет есть ли в строке пароля хотя бы один символ, который является цифрой

IsDigit - метод проверки наличия цифры

Char - проверка одного символа в строке

Any - проверка любого символа (если хотим проверит, чтобы все символы были цифрами, ставим All)

IsLower – показывает, относится ли указанный символ Юникода к категории букв нижнего регистра

IsUpper - Показывает, относится ли указанный символ Юникода к категории букв верхнего регистра

Intersect("@#$%^&\*").Count()

Intersect - Находит пересечение множеств, представленных двумя последовательностями.

Count - Получает число элементов, содержащихся в Intersect

! знак нужен, чтобы условие было неверным

password.Any(Char.IsLower)

предусловие (блок Аrrange) — устанавливает начальные условия для выполнения теста

действие (блок Act) — выполняет сам тест

постусловие (блок Assert) — верифицирует результат теста, и, в данном случае, оформление — повышает читаемость текста и облегчает его использование в качестве документации к тестируемой функциональности

Из чего состоит метод :

переменная для установки пароля из тестовых данных (password)

ожидаемое значение в результате выполнения теста (bool exepted true or false)

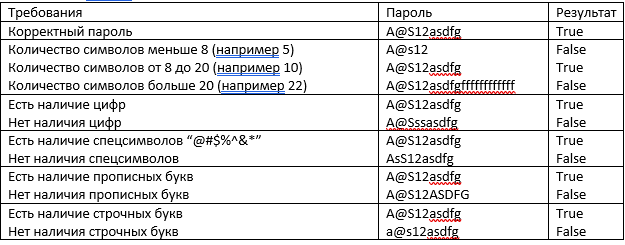
переменная, которая вернет актуальный результат при выполнении метода CheckPassword. В нашем случае ValidatePassword

сравниваем два значения: ожидаемое и реальное, метод AreEquel, и в качестве аргумента — наши данные (exepted, actual)

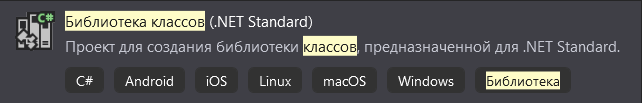
1. Описание: КАК вы решили задачи?

Как делался проект «ClassLibraryPassword»

1 Открываем word и создаём таблицу. Таблица приведена ниже:



2 Открываем Visual Studio, создаём новый проект. В поиске шаблонов вбиваем «Библиотека классов» находим и выбираем (пример ниже – то что выбираем)



3 Выбрав шаблон, называем проект «ClassLibraryPassword». Нажимаем создать.

4 Как только создали сразу переименовываем «Class1.cs» в «PasswordChecker.cs»

5 Так же переименовываем namespase («ClassLibrary1» в «ClassLibraryPassword») и public class (вроде должен переименоваться сам)

6 Затем создаём статичный метод "validatePassword" а в теле метода пока просто возвращаем true

public static bool validatePassword (string password)

{return true;}

7 Создаём тестовый проект для этого метода (правой кнопкой мыши наводим на "validatePassword" и выбираем "создание модульных тестов". После этого переходим в "PasswordCheckerTests.cs", там в public class PasswordCheckerTests создаём метод "public void Check\_5Symbols\_ReturnsFalse()".

Перед методом пишем "[TestMethod()]", внутри метода пишем:

string password = "A@s12";

bool exepted = false;

bool actual = PasswordChecker.validatePassword(password);

Assert.IsFalse(actual);

8 Это был метод False. Теперь создаём метод True. Для примера "public void Check\_10Symbols\_ReturnsTrue()"

Перед методом пишем "[TestMethod()]", внутри метода пишем:

string password = "A@S12asdfg";

bool exepted = true;

bool actual = PasswordChecker.validatePassword(password);

Assert.AreEqual(exepted, actual);

9 Таким образом создаём все строчки таблицы. В итоге получается 11 методов

10 Теперь переходим обратно в PasswordChecker.cs и в "public static bool validatePassword (string password)" создаём условия.

В итоге "public static bool validatePassword (string password)" выглядит так

public static bool validatePassword (string password)

{

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

{

return false;

}

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

{

return false;

}

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

{

return false;

}

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

{

return false;

}

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

{

return false;

}

return true;

}

1. Результат: скриншот

