Day23 駝峰轉換器(?) Codewars\_Convert String To Camel Case

今天的題目也是因為看起來很有趣所以就選了

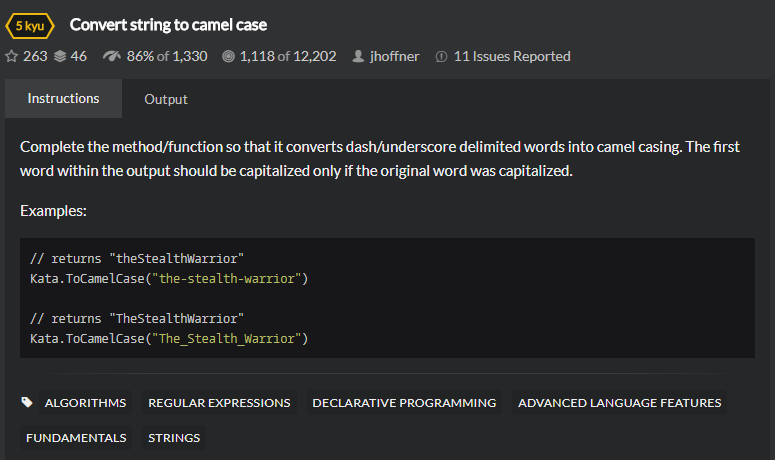
不過他這個題目看起來一點都不5kyu啊…

沒關係啦XDDDDD

反正重點不是難度

重點在於過程!

今天的題目長這樣



今天的題目需求需要將dash跟underscore去除並且將除了第一個單字以外的字串字首大寫化。

現在就來拆一下題目吧。

1. 去除dash和underscore，split方法。
2. 除了第一個單字以外字首大寫化的方法。
3. 將處理完畢的字串處理好，string.Join方法。

一開始我們先來寫去除dash的測試吧!!

```

[TestMethod]

public void Input\_dash\_Should\_Be\_Empty()

{

Assert.AreEqual(string.Empty,Kata.ToCamelCase("-"));

}

```

而Production Code 也就是老樣子會長成這個樣子

```

public static string ToCamelCase(string s)

{

throw new System.**NotImplementedException**();

}  
```

老樣子，跑個測試，沒過很正常，紅燈，commit一下

接下來把Production Code改成這個樣子

```

public static string ToCamelCase(string s)

{

var strs = s.Split('-');

var result = string.Empty;

foreach (var str in strs)

{

result += str;

}

return result;

}  
```

接下來跑個測試，PASS! Commit~

再來加上輸入underscore的測試。

```

[TestMethod]

public void GetHour\_Input\_3600\_Should\_Be\_1()

{

Assert.AreEqual(1, TimeFormat.GetHour(3600));

}

```

而Production Code就會長這個樣子

```

public static string ToCamelCase(string s)

{

string[] strs;

if (s.Contains("-"))

{

strs = s.Split('-');

}

else

{

strs = s.Split('\_');

}

var result = string.Empty;

foreach (var str in strs)

{

result += str;

}

return result;

}

```

現在可以重構一下Production Code。

```

public static string ToCamelCase(string s)

{

var strs = s.Split(s.Contains("-") ? '-' : '\_');

return strs.Aggregate(string.Empty, (current, str) => current + str);

}

```

接下來寫一個只有一個單字的測試，輸入字串a-，很明顯的一定會過

所以再來要寫處理字首大寫的除了字首之外要小寫的測試。

```

[TestMethod]

public void UpperTitle\_Input\_Abc\_Should\_Be\_abc()

{

Assert.AreEqual("Abc", Kata.UpperTitle("abc"));

}

```

這時候讓測試通過的UpperTitle方法的Code會變成

```

public static string UpperTitle(string str)

{

return str.Substring(0, 1).ToUpper() + (str.Length > 1 ? str.Substring(1) : string.Empty);

}

```

接下來要考慮到字首之外的字串要小寫所以寫了這個測試。

```

[TestMethod]

public void UpperTitle\_Input\_aBC\_Should\_Be\_Abc()

{

Assert.AreEqual("Abc", Kata.UpperTitle("aBC"));

}

```

這時候Production Code會變成這個概念

```

public static string UpperTitle(string str)

{

if (string.IsNullOrEmpty(str))

return string.Empty;

return str.Substring(0, 1).ToUpper() + (str.Length > 1 ? str.Substring(1).ToLower() : string.Empty);

}

```

現在已經完成處理字首大寫字首外小寫的問題啦!

所以要回到Production Code的ToCalmelCase加上來運用了。

測試項目就要有兩個字的輸入了。

```

[TestMethod]

public void Input\_adashb\_Should\_Be\_aB()

{

Assert.AreEqual("aB", Kata.ToCamelCase("a-b"));

}

[TestMethod]

public void Input\_aunderb\_Should\_Be\_aB()

{

Assert.AreEqual("aB", Kata.ToCamelCase("a\_b"));

}

[TestMethod]

public void Input\_Aunderb\_Should\_Be\_AB()

{

Assert.AreEqual("AB", Kata.ToCamelCase("A\_b"));

}

[TestMethod]

public void Input\_appledashBIRD\_Should\_Be\_appleBird()

{

Assert.AreEqual("appleBird",Kata.ToCamelCase("apple-BIRD"));

}

```

Production Code 會長這個樣子。

```

public static string ToCamelCase(string s)

{

var strs = s.Split(s.Contains("-") ? '-' : '\_');

for (int i = 1; i < strs.Length; i++)

{

strs[i] = UpperTitle(strs[i]);

}

return strs.Aggregate(string.Empty, (current, str) => current + str);

}

```

但並沒有考慮到第一個單字若全為大寫的時候也需要將除了第一個單字的字首之外的字母小寫化，所以產生了一個新的test case

```

[TestMethod]

public void Input\_ASIApeople\_Should\_Be\_AsiaPeople()

{

Assert.AreEqual("AsiaPeople",Kata.ToCamelCase("ASIApeople"));

}

```

所有的Production Code最後會長成這個樣子。

```

public static string ToCamelCase(string s)

{

var strs = s.Split(s.Contains("-") ? '-' : '\_');

strs[0] = (strs[0].Length > 1) ? strs[0][0] + strs[0].Substring(1).ToLower() : strs[0];

for (int i = 1; i < strs.Length; i++)

{

strs[i] = UpperTitle(strs[i]);

}

return string.Join("", strs);

}

public static string UpperTitle(string str)

{

if (string.IsNullOrEmpty(str))

return string.Empty;

return str.Substring(0, 1).ToUpper() + (str.Length > 1 ? str.Substring(1).ToLower() : string.Empty);

}

```

以下是今天所有的測試案例

```

[TestClass]

public class UnitTest1

{

[TestMethod]

public void Input\_dash\_Should\_Be\_Empty()

{

Assert.AreEqual(string.Empty, Kata.ToCamelCase("-"));

}

[TestMethod]

public void Input\_underscore\_Should\_Be\_Empty()

{

Assert.AreEqual(string.Empty, Kata.ToCamelCase("\_"));

}

[TestMethod]

public void Input\_adash\_Should\_Be\_a()

{

Assert.AreEqual("a", Kata.ToCamelCase("a-"));

}

[TestMethod]

public void UpperTitle\_Input\_Abc\_Should\_Be\_abc()

{

Assert.AreEqual("Abc", Kata.UpperTitle("abc"));

}

[TestMethod]

public void UpperTitle\_Input\_aBC\_Should\_Be\_Abc()

{

Assert.AreEqual("Abc", Kata.UpperTitle("aBC"));

}

[TestMethod]

public void Input\_adashb\_Should\_Be\_aB()

{

Assert.AreEqual("aB", Kata.ToCamelCase("a-b"));

}

[TestMethod]

public void Input\_aunderb\_Should\_Be\_aB()

{

Assert.AreEqual("aB", Kata.ToCamelCase("a\_b"));

}

[TestMethod]

public void Input\_Aunderb\_Should\_Be\_AB()

{

Assert.AreEqual("AB", Kata.ToCamelCase("A\_b"));

}

[TestMethod]

public void Input\_appledashBIRD\_Should\_Be\_appleBird()

{

Assert.AreEqual("appleBird", Kata.ToCamelCase("apple-BIRD"));

}

[TestMethod]

public void Input\_ASIApeople\_Should\_Be\_AsiaPeople()

{

Assert.AreEqual("Asiapeople", Kata.ToCamelCase("ASIApeople"));

}

[TestMethod]

public void Input\_ASIAdashpeople\_Should\_Be\_AsiaPeople()

{

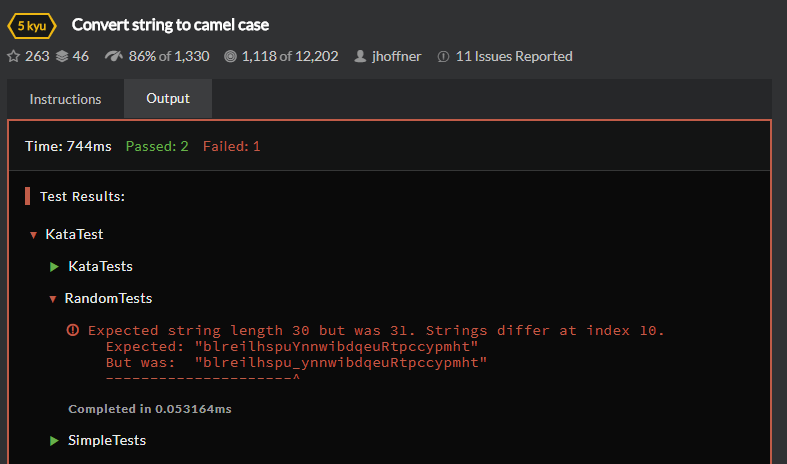
Assert.AreEqual("AsiaPeople", Kata.ToCamelCase("ASIA-people"));

}

}

```

現在就可以很愜意的上傳到Codewars然後就…….Fail啦!



原來他會混合dash跟underscore所以現在就要再加幾個測試啦!

```

[TestMethod]

public void Input\_adashbunderscorea()

{

Assert.AreEqual("aBA",Kata.ToCamelCase("a-b\_a"));

}

```

可以通過混合使用-和\_的Production Code長這樣

```

public static string ToCamelCase(string s)

{

var strs = s.Split('-');

strs[0] = (strs[0].Length > 1) ? strs[0][0] + strs[0].Substring(1).ToLower() : strs[0];

for (int i = 1; i < strs.Length; i++)

strs[i] = UpperTitle(strs[i]);

var strs2 = string.Join("", strs).Split('\_');

for (int i = 1; i < strs2.Length; i++)

strs2[i] = UpperTitle(strs2[i]);

return string.Join("", strs2);

}

```

有沒有聞到一股味道?

是得沒錯

你聞到了Duplicated Code 的味道!!

所以我們來重構一下它吧! 把他抽出來用!

現在的Production Code會長這樣!

```

public static string ToCamelCase(string s)

{

var strs = s.Split('-');

strs[0] = (strs[0].Length > 1) ? strs[0][0] + strs[0].Substring(1).ToLower() : strs[0];

strs = UpperTitleAll(strs);

var strs2 = UpperTitleAll(string.Join("", strs).Split('\_'));

return string.Join("", strs2);

}

```

不過剛才那個交互使用的測試並不完全於是乎我就新增了把-跟\_的順序調換的測試，居然發生了錯誤，所以最後ProductionCode就變成這個樣子

```

public static string ToCamelCase(string s)

{

var c = '-';

var c2 = '\_';

if (s.IndexOf('-')>s.IndexOf('\_'))

{

c = '\_';

c2 = '-';

}

var strs = s.Split(c);

strs[0] = (strs[0].Length > 1) ? strs[0][0] + strs[0].Substring(1).ToLower() : strs[0];

strs = UpperTitleAll(strs);

var strs2 = UpperTitleAll(string.Join("", strs).Split(c2));

return string.Join("", strs2);

}

public static string UpperTitle(string str)

{

if (string.IsNullOrEmpty(str))

return string.Empty;

return str.Substring(0, 1).ToUpper() + (str.Length > 1 ? str.Substring(1).ToLower() : string.Empty);

}

public static string[] UpperTitleAll(string[] strs)

{

for (int i = 1; i < strs.Length; i++)

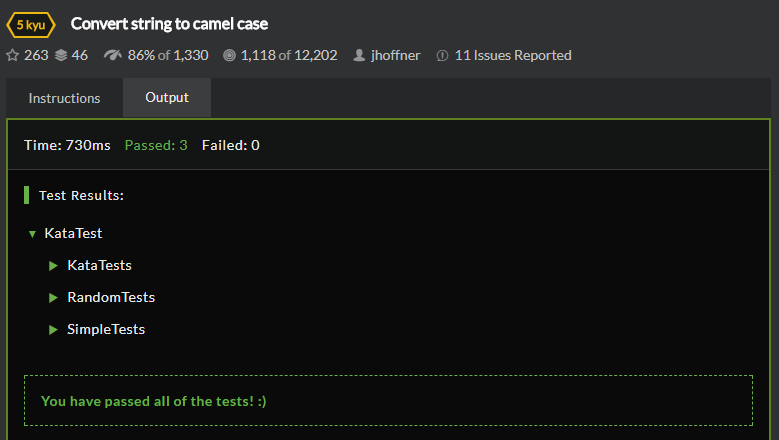
strs[i] = UpperTitle(strs[i]);

return strs;

}

```

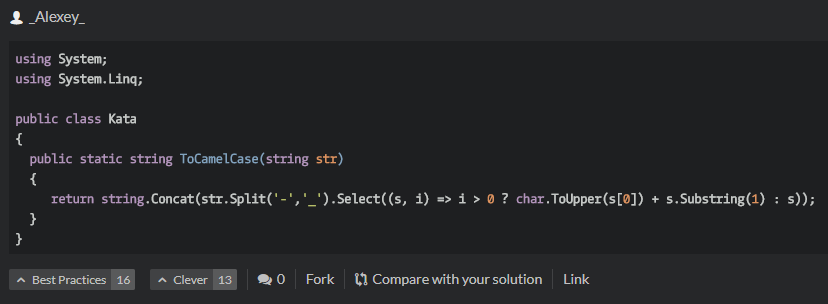
最後終於Pass了 呼!



在Codewars上成功提交了~

來看一下其他人怎麼寫吧!

我寫的真的是又臭又長的…



…我傻眼

原來Split方法可以傳入兩個進行分割啊…..

好喔= =

長知識了 OK

Git url :

<https://github.com/SQZ777/Codewars_ConvertStringToCamelCase>

Codewars Link:

<https://www.codewars.com/kata/convert-string-to-camel-case/train/csharp>

下一題，明天見!