Day29 TDD套路經典!? Tennis Game!

這周周末就要去參加91開的極速開發的課了

想說最後一篇還是寫這一題感覺比較像有結尾的feel

會這麼說的原因是因為有某位學弟給我題目寫一題

但那個題目跟過去28天寫的題目都有點雷同，一樣是拆解題目分析然後解需求，所以我還覺得最後一題還是用這題來做結尾來得好XD

這個套路比較好的原因是因為他比較像真的一般會遇到的事情

Tennis Game的需求就跟一般的規則一樣!

<http://codingdojo.org/kata/Tennis/>

好了，來吧!

首先我先用手寫了這個題目的需求測試案例

一開始我們先來寫Love\_All的測試吧!!

```

[TestMethod]

public void Love\_All()

{

var tennisGame = new TennisGame();

var score = tennisGame.Score();

Assert.AreEqual("Love-All",score);

}

```

而Production Code 也就是老樣子會長成這個樣子

```

public string Score()

{

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 string Score()

{

return "Love-All";

}  
```

接下來跑個測試，PASS! Commit~

Test code Refactor他一下

```

TennisGame tennisGame = new TennisGame();

[TestMethod]

public void Love\_All()

{

ScoreShouldBe("Love-All");

}

private void ScoreShouldBe(string expected)

{

var score = tennisGame.Score();

Assert.AreEqual(expected, score);

}

```

再來新增一個Fifteen\_Love的測試

```

[TestMethod]

public void Fifteen\_Love()

{

tennisGame.SetFirstPlayerScore(1);

ScoreShouldBe("Fifteen-Love");

}

```

通過他的Production Code就需要增加第一個玩家的分數變數了

```

public class TennisGame

{

private int \_firstPlayerScore;

public string Score()

{

if (\_firstPlayerScore == 1)

{

return "Fifteen-Love";

}

return "Love-All";

}

public void SetFirstPlayerScore(int n)

{

\_firstPlayerScore = n;

}

}

```

接下來新增第二個測試 Thirty\_Love

```

[TestMethod]

public void Thirty\_Love()

{

tennisGame.SetFirstPlayerScore(2);

ScoreShouldBe("Thirty-Love");

}

```

加上一個判斷是就可以通過測試了

```

public string Score()

{

if (\_firstPlayerScore == 2)

{

return "Thirty-Love";

}

if (\_firstPlayerScore == 1)

{

return "Fifteen-Love";

}

return "Love-All";

}

```

看到這樣重複的Code就發現其實是可以Refactor他的

這時候看起來是可以用Dictionary!

所以完整的Production Code變成這個樣子

```

public class TennisGame

{

private int \_firstPlayerScore;

private readonly Dictionary<int, string> \_scoreDictionary = new Dictionary<int, string>

{

{1, "Fifteen" },

{2, "Thirty" }

};

public string Score()

{

if (\_firstPlayerScore > 0)

{

return \_scoreDictionary[\_firstPlayerScore] + "-Love";

}

return "Love-All";

}

public void SetFirstPlayerScore(int n)

{

\_firstPlayerScore = n;

}

}

```

現在就來加入Forty\_Love的測試!

```

[TestMethod]

public void Forty\_Love()

{

tennisGame.SetFirstPlayerScore(3);

ScoreShouldBe("Forty-Love");

}

```

因為剛才Refactor的緣故所以只要在Dictionary加上一個key即可通過測試

```

private readonly Dictionary<int, string> \_scoreDictionary = new Dictionary<int, string>

{

{1, "Fifteen" },

{2, "Thirty" },

{3, "Forty" }

};

```

接下來要考慮到第二個玩家的分數囉

所以新增Love\_Fifteen的測試案例

```

[TestMethod]

public void Love\_Fifteen()

{

tennisGame.SetSecondPlayerScore(1);

ScoreShouldBe("Love-Fifteen");

}

```

Love\_Fifteen這個測試案例就要新增Dictionary 1個0的Key是Love

也要將第2個人的分數考慮進去，所以所有Production Code變成這個樣子。

```

public class TennisGame

{

private int \_firstPlayerScore;

private int \_secondPlayerScore;

private readonly Dictionary<int, string> \_scoreDictionary = new Dictionary<int, string>

{

{0, "Love"},

{1, "Fifteen"},

{2, "Thirty"},

{3, "Forty"}

};

public string Score()

{

if (\_firstPlayerScore > 0 || \_secondPlayerScore > 0)

{

return \_scoreDictionary[\_firstPlayerScore] + "-" + \_scoreDictionary[\_secondPlayerScore];

}

return "Love-All";

}

public void SetFirstPlayerScore(int n)

{

\_firstPlayerScore = n;

}

public void SetSecondPlayerScore(int n)

{

\_secondPlayerScore = n;

}

}

```

改成這個樣子時Love\_Thirty跟Love\_Forty的測試也等同於通過了。

```

[TestMethod]

public void Love\_Thirty()

{

tennisGame.SetSecondPlayerScore(2);

ScoreShouldBe("Love-Thirty");

}

[TestMethod]

public void Love\_Forty()

{

tennisGame.SetSecondPlayerScore(3);

ScoreShouldBe("Love-Forty");

}

```

再來寫平手的時候的測試吧!

Fifteen\_All

```

[TestMethod]

public void Fifteen\_All()

{

tennisGame.SetFirstPlayerScore(1);

tennisGame.SetSecondPlayerScore(1);

ScoreShouldBe("Fifteen-All");

}

```

Production Code就會加上一個判斷並回傳，這樣基本上第二個平手測試Thirty\_All也會通過了。

```

public string Score()

{

if (\_firstPlayerScore == \_secondPlayerScore)

{

return \_scoreDictionary[\_firstPlayerScore] + "-All";

}

if (\_firstPlayerScore > 0 || \_secondPlayerScore > 0)

{

return \_scoreDictionary[\_firstPlayerScore] + "-" + \_scoreDictionary[\_secondPlayerScore];

}

return "Love-All";

}

```

這裡會發現最後的Love-All是多餘的

所以其實上述的兩個If是可以合再一起的

來Refactor一下

```

public string Score()

{

if (\_firstPlayerScore == \_secondPlayerScore)

{

return \_scoreDictionary[\_firstPlayerScore] + "-All";

}

return \_scoreDictionary[\_firstPlayerScore] + "-" + \_scoreDictionary[\_secondPlayerScore];

}

```

再來寫Deuce的測試。

```

[TestMethod]

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

{

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

}

```

只要加入一個判斷即可通過測試，這個測試基本上也通過了4:4的測試了

```

public string Score()

{

if (\_firstPlayerScore == \_secondPlayerScore)

{

if (\_firstPlayerScore >= 3)

{

return "Deuce";

}

return \_scoreDictionary[\_firstPlayerScore] + "-All";

}

return \_scoreDictionary[\_firstPlayerScore] + "-" + \_scoreDictionary[\_secondPlayerScore];

}

```

4:4的測試Code我會這麼寫

```

[TestMethod]

public void When\_4\_4\_Then\_Deuce()

{

tennisGame.SetFirstPlayerScore(4);

tennisGame.SetSecondPlayerScore(4);

ScoreShouldBe("Deuce");

}

```

再來加入Deuce之後領先的人的測試了!

因為要考慮到人名了，所以測試Code上方的TennisGame Class要加入人名的建構式了。(這樣就不用再加裡面所有的測試Code了呢

```

TennisGame tennisGame = new TennisGame("Lin","DZ");

[TestMethod]

public void Lin\_Adv()

{

tennisGame.SetFirstPlayerScore(4);

tennisGame.SetSecondPlayerScore(3);

ScoreShouldBe("Lin Adv");

}

```

而Production Code就會長這個樣子

```

public string Score()

{

if (\_firstPlayerScore == \_secondPlayerScore)

{

if (\_firstPlayerScore >= 3)

{

return "Deuce";

}

return \_scoreDictionary[\_firstPlayerScore] + "-All";

}

if (\_firstPlayerScore >= 3 && \_secondPlayerScore >= 3)

{

if (Math.Abs(\_firstPlayerScore - \_secondPlayerScore) == 1)

{

return \_firstPlayerName + " Adv";

}

}

return \_scoreDictionary[\_firstPlayerScore] + "-" + \_scoreDictionary[\_secondPlayerScore];

}

```

再來寫第1個玩家獲勝的測試

```

[TestMethod]

public void Lin\_Win()

{

tennisGame.SetFirstPlayerScore(5);

tennisGame.SetSecondPlayerScore(3);

ScoreShouldBe("Lin Win");

}

```

完成了這個測試之後再新增一個第2個玩家Adv的測試

```

[TestMethod]

public void DZ\_Adv()

{

tennisGame.SetFirstPlayerScore(3);

tennisGame.SetSecondPlayerScore(4);

ScoreShouldBe("DZ Adv");

}

```

通過之後就可以快速地更改Production Code並更改第2個玩家Win的測試

```

[TestMethod]

public void DZ\_Win()

{

tennisGame.SetFirstPlayerScore(3);

tennisGame.SetSecondPlayerScore(5);

ScoreShouldBe("DZ Win");

}

```

Production Code就會變成這個樣子

```

public string Score()

{

if (\_firstPlayerScore == \_secondPlayerScore)

{

if (\_firstPlayerScore >= 3)

{

return "Deuce";

}

return \_scoreDictionary[\_firstPlayerScore] + "-All";

}

if (\_firstPlayerScore > 3 || \_secondPlayerScore > 3)

{

var advPlayer = \_firstPlayerScore > \_secondPlayerScore ? \_firstPlayerName : \_secondPlayerName;

if (Math.Abs(\_firstPlayerScore - \_secondPlayerScore) == 1)

{

return advPlayer + " Adv";

}

return advPlayer + " Win";

}

return \_scoreDictionary[\_firstPlayerScore] + "-" + \_scoreDictionary[\_secondPlayerScore];

}

```

接下來就一口氣Refactor 全部的Code超過癮的

(其實在實作過程中就要Refactor了，只是今天想要這麼做XD)

這是今天的所有測試案例

```

[TestClass]

public class UnitTest1

{

TennisGame tennisGame = new TennisGame("Lin","DZ");

[TestMethod]

public void Love\_All()

{

ScoreShouldBe("Love-All");

}

[TestMethod]

public void Fifteen\_Love()

{

tennisGame.SetFirstPlayerScore(1);

ScoreShouldBe("Fifteen-Love");

}

[TestMethod]

public void Thirty\_Love()

{

tennisGame.SetFirstPlayerScore(2);

ScoreShouldBe("Thirty-Love");

}

[TestMethod]

public void Forty\_Love()

{

tennisGame.SetFirstPlayerScore(3);

ScoreShouldBe("Forty-Love");

}

[TestMethod]

public void Love\_Fifteen()

{

tennisGame.SetSecondPlayerScore(1);

ScoreShouldBe("Love-Fifteen");

}

[TestMethod]

public void Love\_Thirty()

{

tennisGame.SetSecondPlayerScore(2);

ScoreShouldBe("Love-Thirty");

}

[TestMethod]

public void Love\_Forty()

{

tennisGame.SetSecondPlayerScore(3);

ScoreShouldBe("Love-Forty");

}

[TestMethod]

public void Fifteen\_All()

{

tennisGame.SetFirstPlayerScore(1);

tennisGame.SetSecondPlayerScore(1);

ScoreShouldBe("Fifteen-All");

}

[TestMethod]

public void Thirty\_All()

{

tennisGame.SetFirstPlayerScore(2);

tennisGame.SetSecondPlayerScore(2);

ScoreShouldBe("Thirty-All");

}

[TestMethod]

public void Deuce()

{

tennisGame.SetFirstPlayerScore(3);

tennisGame.SetSecondPlayerScore(3);

ScoreShouldBe("Deuce");

}

[TestMethod]

public void When\_4\_4\_Then\_Deuce()

{

tennisGame.SetFirstPlayerScore(4);

tennisGame.SetSecondPlayerScore(4);

ScoreShouldBe("Deuce");

}

[TestMethod]

public void Lin\_Adv()

{

tennisGame.SetFirstPlayerScore(4);

tennisGame.SetSecondPlayerScore(3);

ScoreShouldBe("Lin Adv");

}

[TestMethod]

public void Lin\_Win()

{

tennisGame.SetFirstPlayerScore(5);

tennisGame.SetSecondPlayerScore(3);

ScoreShouldBe("Lin Win");

}

[TestMethod]

public void DZ\_Adv()

{

tennisGame.SetFirstPlayerScore(3);

tennisGame.SetSecondPlayerScore(4);

ScoreShouldBe("DZ Adv");

}

[TestMethod]

public void DZ\_Win()

{

tennisGame.SetFirstPlayerScore(3);

tennisGame.SetSecondPlayerScore(5);

ScoreShouldBe("DZ Win");

}

private void ScoreShouldBe(string expected)

{

var score = tennisGame.Score();

Assert.AreEqual(expected, score);

}

}

```

這是今天所有的Production Code

全部都攤平平的，看起來很過癮XDDD

```

public class TennisGame

{

private int \_firstPlayerScore;

private int \_secondPlayerScore;

private string \_firstPlayerName;

private string \_secondPlayerName;

private readonly Dictionary<int, string> \_scoreDictionary = new Dictionary<int, string>

{

{0, "Love"},

{1, "Fifteen"},

{2, "Thirty"},

{3, "Forty"}

};

public TennisGame(string firstPlayerName, string secondPlayerName)

{

this.\_firstPlayerName = firstPlayerName;

this.\_secondPlayerName = secondPlayerName;

}

public string Score()

{

return IsSameScore()

? (IsDeuce() ? Deuce() : SameScore())

: (ReadyForWin() ? AdvOrWinPlayer() : NormalScore());

}

private bool IsSameScore()

{

return \_firstPlayerScore == \_secondPlayerScore;

}

private static string Deuce()

{

return "Deuce";

}

private bool IsDeuce()

{

return \_firstPlayerScore >= 3;

}

private string SameScore()

{

return \_scoreDictionary[\_firstPlayerScore] + "-All";

}

private string AdvOrWinPlayer()

{

return AdvPlayer() + (IsAdv() ? " Adv" : " Win");

}

private bool ReadyForWin()

{

return \_firstPlayerScore > 3 || \_secondPlayerScore > 3;

}

private string AdvPlayer()

{

return \_firstPlayerScore > \_secondPlayerScore ? \_firstPlayerName : \_secondPlayerName;

}

private bool IsAdv()

{

return Math.Abs(\_firstPlayerScore - \_secondPlayerScore) == 1;

}

private string NormalScore()

{

return \_scoreDictionary[\_firstPlayerScore] + "-" + \_scoreDictionary[\_secondPlayerScore];

}

public void SetFirstPlayerScore(int n)

{

\_firstPlayerScore = n;

}

public void SetSecondPlayerScore(int n)

{

\_secondPlayerScore = n;

}

}

```



今天就是最後一篇TDD的練習啦!

感謝各位收看

今天超趕的，好可怕

因為有重寫這一篇

不過好險有趕上!!!