C# Console Rock-Paper-Scissors

Web Clip

接下來請用英文回答，請寫一個 C# 猜拳遊戲，使用 Console application project，針對二位玩家隨機取得"剪刀，石頭，布"，並顯示每一個玩家的值，以及誰勝利； 程式先宣告一個遊戲的核心class, 它具備生成二個玩家的值，判斷誰勝的method, 最後在 console application 叫用此 class

Here is a sample C# code for a Rock-Paper-Scissors game using a Console application project:

csharp

```
using System;

namespace RockPaperScissors
{
    class Game
    {
        private string player1;
        private string player2;

        public void Start()
        {
            Console.WriteLine("Welcome to Rock-Paper-Scissors!");

            Random rnd = new Random();
            int choice1 = rnd.Next(1, 4);
            int choice2 = rnd.Next(1, 4);

            switch (choice1)
            {
                case 1:
                    player1 = "Rock";
                    break;
                case 2:
                    player1 = "Paper";
                    break;
                case 3:
                    player1 = "Scissors";
                    break;
            }

            switch (choice2)
            {
                case 1:
                    player2 = "Rock";
                    break;
                case 2:
                    player2 = "Paper";
                    break;
                case 3:
                    player2 = "Scissors";
                    break;
            }

            Console.WriteLine("Player 1 chose " + player1);
            Console.WriteLine("Player 2 chose " + player2);

            DetermineWinner();
        }

        private void DetermineWinner()
        {
            if (player1 == player2)
            {
                Console.WriteLine("It's a tie!");
            }
            else if (player1 == "Rock" && player2 == "Scissors")
            {
                Console.WriteLine("Player 1 wins!");
            }
            else if (player1 == "Paper" && player2 == "Rock")
            {
                Console.WriteLine("Player 1 wins!");
            }
            else if (player1 == "Scissors" && player2 == "Paper")
            {
                Console.WriteLine("Player 1 wins!");
            }
            else
            {
                Console.WriteLine("Player 2 wins!");
            }
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Game game = new Game();
            game.Start();
            Console.ReadKey();
        }
    }
}
```

請修改程式，因為我希望也能在 windows forms application 呼叫 Game class, 所以不要在 Game class 裡撰寫 Console.WriteLine, 應該在 method 裡將資訊傳回比較好； 另外，class 'Game' 名稱太過空泛，請修改它的命名，讓工程師一看就知道它是什麼遊戲； 在每一個 method 都加入台灣繁體中文的註解，說明method 的功用

Here's the updated code for the Rock-Paper-Scissors game, where the `Game` class has been renamed to `RockPaperScissorsGame` and it no longer uses `Console.WriteLine`:

csharp

```csharp
using System;

namespace RockPaperScissors
{
    class RockPaperScissorsGame
    {
        private string player1Choice;
        private string player2Choice;

        // 啟動遊戲
        public void Start()
        {
            Random rnd = new Random();
            int choice1 = rnd.Next(1, 4);
            int choice2 = rnd.Next(1, 4);

            // 取得玩家1的選擇
            switch (choice1)
            {
                case 1:
                    player1Choice = "Rock";
                    break;
                case 2:
                    player1Choice = "Paper";
                    break;
                case 3:
                    player1Choice = "Scissors";
                    break;
            }

            // 取得玩家2的選擇
            switch (choice2)
            {
                case 1:
                    player2Choice = "Rock";
                    break;
                case 2:
                    player2Choice = "Paper";
                    break;
                case 3:
                    player2Choice = "Scissors";
                    break;
            }

            // 傳回玩家1和玩家2的選擇
            string[] choices = { player1Choice, player2Choice };
            return choices;
        }

        // 判斷誰勝利
        public string DetermineWinner()
        {
            if (player1Choice == player2Choice)
            {
                return "It's a tie!";
            }
            else if (player1Choice == "Rock" && player2Choice == "Sci
            {
                return "Player 1 wins!";
            }
            else if (player1Choice == "Paper" && player2Choice == "Ro
            {
                return "Player 1 wins!";
            }
            else if (player1Choice == "Scissors" && player2Choice ==
            {
                return "Player 1 wins!";
            }
```

```
                else
                {
                    return "Player 2 wins!";
                }
            }
        }
    }
```

In a Windows Forms application, you can call the `RockPaperScissorsGame` class and display the results in a form, such as a `MessageBox`, like this:

typescript

```
private void btnStart_Click(object sender, EventArgs e)
{
    RockPaperScissors.RockPaperScissorsGame game = new RockPaperSciss
    string[] choices = game.Start();
    string winner = game.DetermineWinner();
    MessageBox.Show("Player 1 chose " + choices[0] + "nPlayer 2 chose
}
```

請修改程式，由於 Start method 裡面呼叫了 Random, 這樣會造成這支 method 不容易進行 unit testing, 我想宣告一個 interface IRandomProvider, 它包含一個 int Next(int minValue, int maxValue) method, 然後再自訂一個 class 實作此 interface, 在建立 game 的 intance 時將它傳入，如此一來，我在進行單元測試時，就可以控制每一個玩家要生成什麼值； 請你連同單元測試也寫出來

---

csharp

```
using System;

public interface IRandomProvider
{
    int Next(int minValue, int maxValue);
}

public class RandomProvider : IRandomProvider
{
    private readonly Random _random;
```

```
        public RandomProvider()
        {
            _random = new Random();
        }

        public int Next(int minValue, int maxValue)
        {
            return _random.Next(minValue, maxValue);
        }
    }

    public class RockPaperScissorsGame
    {
        private readonly IRandomProvider _randomProvider;
        private readonly int _player1Choice;
        private readonly int _player2Choice;
        private readonly string[] _choices = { "剪刀", "石頭", "布" };

        // 建構子，用來生成兩個玩家的選擇
        public RockPaperScissorsGame(IRandomProvider randomProvider)
        {
            _randomProvider = randomProvider;
            _player1Choice = _randomProvider.Next(0, 3);
            _player2Choice = _randomProvider.Next(0, 3);
        }

        // 取得玩家1的選擇
        public string GetPlayer1Choice()
        {
            return _choices[_player1Choice];
        }

        // 取得玩家2的選擇
        public string GetPlayer2Choice()
        {
            return _choices[_player2Choice];
        }

        // 判斷誰贏
        public string GetWinner()
        {
            if (_player1Choice == _player2Choice)
            {
                return "平手";
            }

            if (_player1Choice == 0 && _player2Choice == 2 ||
                _player1Choice == 1 && _player2Choice == 0 ||
                _player1Choice == 2 && _player2Choice == 1)
            {
                return "玩家1勝";
            }

            return "玩家2勝";
        }
    }
```

csharp

```
using NUnit.Framework;

[TestFixture]
public class RockPaperScissorsGameTests
{
    [Test]
    public void GetPlayer1Choice_ShouldReturnCorrectValue()
    {
        var randomProvider = new StubRandomProvider(0, 1);
        var game = new RockPaperScissorsGame(randomProvider);

        Assert.AreEqual("剪刀", game.GetPlayer1Choice());
    }

    [Test]
    public void GetPlayer2Choice_ShouldReturnCorrectValue()
    {
        var randomProvider = new StubRandomProvider(0, 2);
        var game = new RockPaperScissorsGame(randomProvider);

        Assert.AreEqual("布", game.GetPlayer2Choice());
    }

    [Test]
    public void GetWinner_ShouldReturnCorrectValue()
    {
        var randomProvider = new StubRandomProvider(0, 2);
        var game = new RockPaperScissorsGame(randomProvider);

        Assert.AreEqual("玩家2勝", game
```

more

請修改程式，由於在顯示結果時，我們使用了 switch case, 這樣無法很好地顯示所有的結果； 我希望能將所有的結果放在一個清單裡面，因此我們可以利用清單存放所有的結果，再利用清單裡面的元素去顯示所有結果； 同時，我希望可以在程式碼中增加一個 Result object, 它具備 player1 的結果，player2 的結果，以及判斷勝負的結果，這樣可以更好地組織程式碼

請繼續完成剛才中斷的單元測試程式碼，使用 NUnit

以下是完整的代码，包括 RPSGame、IRandomProvider、RandomProvider、Result 以及 RPSGameTests 五个类。

csharp

```csharp
using System;
using System.Collections.Generic;

namespace RPSGame
{
    public class RPSGame
    {
        private IRandomProvider _randomProvider;
        private List<string> _results;

        public RPSGame(IRandomProvider randomProvider)
        {
            _randomProvider = randomProvider;
            _results = new List<string> { "剪刀", "石頭", "布" };
        }

        public Result Start()
        {
            int player1Value = _randomProvider.Next(0, 3);
            int player2Value = _randomProvider.Next(0, 3);
            string player1Result = _results[player1Value];
            string player2Result = _results[player2Value];
            string winner = GetWinner(player1Value, player2Value);
            return new Result
            {
                Player1Result = player1Result,
                Player2Result = player2Result,
                Winner = winner
            };
        }
        private string GetWinner(int player1Value, int player2Value)
        {
            if (player1Value == player2Value)
                return "平手";
            else if ((player1Value + 1) % 3 == player2Value)
                return "Player 2 贏";
            else
                return "Player 1 贏";
        }
    }

    public interface IRandomProvider
    {
        int Next(int minValue, int maxValue);
    }

    public class RandomProvider : IRandomProvider
    {
        private readonly Random _random;

        public RandomProvider()
        {
            _random = new Random();
        }

        public int Next(int minValue, int maxValue)
        {
            return _random.Next(minValue, maxValue);
        }
    }

    public class Result
    {
        public string Player1Result { get; set; }
        public string Player2Result { get; set; }
        public string Winner { get; set; }
    }
}

using NUnit.Framework;
using RPSGame;

namespace RPSGameTests
{
    [TestFixture]
    public class RPSGameTests
    {
        [Test]
        public void Start_ShouldReturnCorrectResult()
        {
            // Arrange
            IRandomProvider randomProvider = new FakeRandomProvider()
            RPSGame game = new RPSGame(randomProvider);

            // Act
            Result result = game.Start();

            // Assert
            Assert.AreEqual("剪刀", result.Player1Result);
            Assert.AreEqual("石頭", result.Player2Result);
            Assert.AreEqual("Player 2 贏", result.Winner);
        }
    }

    public class FakeRandomProvider : IRandomProvider
    {
        private int _count = 0;

        public int Next(int minValue, int maxValue)
        {
            if (_count == 0)
```

請繼續，完成FakeRandomProvider class 的程式碼； 另外，在 GetWinner method中，你寫到了 (player1Value + 1) % 3 == player2Value，它用來判斷誰是 winner ，我覺得很精簡，但請你解釋一下它的意思

Sure! Here's the code for the `FakeRandomProvider` class:

csharp

```
public class FakeRandomProvider : IRandomProvider
{
    private readonly int[] _values;
    private int _index;

    public FakeRandomProvider(int[] values)
    {
        _values = values;
        _index = 0;
    }
    public int Next(int minValue, int maxValue)
    {
        if (_index >= _values.Length)
        {
            _index = 0;
        }
        return _values[_index++];
    }
}
```

As for the expression `` `(player1Value + 1) % 3 == player2Value` `` in the `` `GetWinner` `` method:

- `` `player1Value` `` and `` `player2Value` `` are the values representing "scissors", "rock", and "paper".
- `` `(player1Value + 1) % 3` `` will give the next value in the sequence (0, 1, 2), which is effectively the value for the next item in the game ("scissors", "rock", "paper").
- So, if `` `(player1Value + 1) % 3 == player2Value` ``, it means that player1 lost because player2 chose the winning item.

請用台灣繁體中文重新解釋 (player1Value + 1) % 3 == player2Value 程式碼的意思

"(player1Value + 1) % 3 == player2Value" 這段程式碼是用來判斷兩位玩家誰贏了。

這裡使用了模數運算，將 player1Value 加 1 並除以 3，再取餘數，如果餘數等於 player2Value，則表示 player1 贏了；反之，則 player2 贏了。

為什麼要這樣判斷呢？因為猜拳遊戲中有三種手勢：剪刀、石頭、布，如果 player1 的值為剪刀，player2 的值為布，那麼 player1 就贏了；如果 player1 的值為石頭，player2 的值為剪刀，那麼 player1 就贏了；如果 player1 的值為布，player2 的值為石頭，那麼 player1 就贏了。這樣的關係可以用一個環的方式來表示：剪刀 -> 布 -> 石頭 -> 剪刀 ......

所以，"(player1Value + 1) % 3 == player2Value" 就是透過這個環的關係來

判斷誰贏了。