C#プログラミング講習第五回

1. インターフェース

インターフェース(interface)とは、クラスが実装すべき規約(どういうメソッドににどういうシグネチャを渡すかなど)を定めるものです。すなわち、「そのメソッドが何をできるのか」、「そのメソッドを呼び出すことで何が起こるのか」を決めます。

インターフェースは変数を持つことができず、インターフェースそのもののインスタンスを生成することができません。インターフェースで定義したメソッドは、**実装**したクラス先で中身の処理を定義する必要があります。クラスと違い、インターフェースは複数実装することができます。

```
// このインターフェースを実装したクラスは 「鳴くことができる」 とみなす
interface CanCry {
    // 定義のみ
    void Cry();
}
interface CanRun() {
    // 省略
}
class Dog : CanCry {
    // 実際の処理を記述
    public void Cry {
        Console.WriteLine("わんっ");
    }
}

// クラスと違い、複数を実装可能
class Cat : CanCry, CanRun {
    public void Cry {
        Console.WriteLine("にゃー");
    }
}

public class Program {
    public static void Main(string[] args) {
        CanCry[] animal = { new Dog(), new Cat() };
        for(int i = 0; i < animal.Length; i++)
            animal[i].Cry();
    }
}
```

サンプルとしてわかりやすいように命名しましたが、インターフェースには命名規則があり、普通 I ではじまって***ble で終わります。

また、.NET Framework の標準クラスライブラリでは、以下のようなインターフェースが用意されています。

インターフェース名	説明
System.ICompareble	大小比較が可能なクラスで実装するインターフェース
System.ICloneable	クローン作成が可能なクラスで実装するインターフェース
System.IDisposable	リソースの破棄が必要なクラスで実装するインターフェース

2. 列挙型

列挙型とはいくつかの識別子に値を持たせるための型です。条件分岐をする際、数値ではなく識別子のまま記述することでコードの可読性を高めます。

```
enum Scene {
    Title, // int 型の 0
    Game,    // int 型の 1
    GameOver,    // int 型の 2
}

public class Program {
    public static void Main(string[] args) {
        Scene nowScene = Scene.Title;
        // Scene nowScene = 0;

        if(nowScene == Scene.Title) nowScene = Scene.Game;
        // if(nowScene == 0) nowScene = 1;

        if(nowScene == Scene.GameOver) Enviroment.Exit(0);
        // if(nowScene == 2) Enviroment.Exit(0);
    }
}
```

enum 型はデフォルトでは int 型の数値に置き換えられます。しかし、数値で直接比較するよりも読みやすいことが分かりますよね。

3. 課題

なし