C#プログラミング講習第三回

1. オーバーロード

オーバーロード(overload)とは、メソッドなどにおいて同じ名前を持ちながら、**シグネチャ**(引数の数や型)の違いを持つメソッドを定義することを指します。オーバーロードされたメソッドを呼び出すときには、シグネチャの一致する適切なメソッドが呼ばれます。これによって名前を書き分けることなく、必要に応じてより必要なパラメータだけを渡したり、余計な型変換なしにパラメータを受け渡しすることができます。

ただし、戻り値のみの異なるシグネチャは区別されないため、そのようなオーバーロードはできません。

```
using System;

public class Program {
    public static void Main(string[] args) {
        Method(10);      // めそっど 1
        Method(10.0);    // めそっど 2
        Method(10, 10.0);          // めそっど 3
    }

    public static void Method(int x) {
        Console.WriteLine("めそっど 1");
    }

    public static void Method(double y) {
        Console.WriteLine("めそっど 2");
    }

    public static void Method(int x, double y) {
        Console.WriteLine("めそっど 3");
    }
}
```

2. 仮想メソッド – virtual 修飾子

前回の資料を見てください。GameObject クラスでは仮に void Update()と void Draw()を用意し、継承した Character クラスや Item クラスでその処理を行えるように設計をしていました。このように、仮想的なメソッドを定義する場合、**virtual 修飾子**を用います。

```
public virtual void Update() { }
```

3. オーバーライド – override 修飾子

継承元の virtual メソッドを元に、そのメソッドの中身の処理を記述する場合、**override 修飾子**を用います。

この、virtual 修飾子と override 修飾子を用いて、前回の課題 4.4 を解いてみましょう。

```
using System;

class GameObject {
    public virtual void Update() { }
}

class Character : GameObject {
    public override void Update() {
        Console.WriteLine("Character");
    }
}

class Item : GameObject {
    public override void Update() {
        Console.WriteLine("Item");
    }
}

public class Program {
    public static void Main(string[] args) {
        // 省略
    }
}
```

4. アクセス修飾子

アクセス修飾子は、クラスやメソッドなどに対してアクセスができる範囲を示す修飾子です。外部から想定してない操作をされると正しい挙動ができない機能を隠蔽して、外部からアクセスしてよい機能だけを公開するためにあります。このようにカプセル化することで、クラスの外部からの正しい使い方を提供することができます。

アクセス修飾子	意味
public	どこからでもアクセスできる
protected	派生クラスからアクセスできる
private	派生クラスや外部クラスからはアクセスできない

```
class GameObject {
    protected int x;
    private int y;

    public setDefalut() {
        Console.WriteLine("デフォルトに設定");
    }

    public virtual void Update() { };
}

class Character : GameObject {
    public override void Update() {
        x = 10; // ○
        y = 15; // ×
        setDefalut();    // ○
    }
}

public class Program {
    public static void Main(string[] args) {
        Character character = new Character();

        character.x = 10;    // ×
        character.y = 15; // ×
        character.setDafalut(); // ○
        character.Update(); // ○
    }
}
```