クラスと継承

目次

１．継承とは

＜概念＞

継承とは、あるクラスの機能を受け継いで新しいクラスを作ることを言います。
今回はPlayerControllerクラスを受け継いで新しいクラスを作ります！

PlayerControllerクラスを直接書き換えるのと何が違うの？となるかもしれませんが、例えば他のプロジェクトで動作確認出来ているスクリプトを使って、機能を追加していくことで効率よく開発出来たりします。
(元々の機能は動作が保証されることになる為)

継承する際に、元になるクラスを基底クラス(スーパークラス)、継承したクラスを派生クラス(サブクラス)と言います。

１．継承とは

＜概念＞

実はUnityでスクリプトを作成した場合、既に継承して作られています！
MonoBehaviourというUnityが作成したクラスを継承することで、
StartやUpdateのメソッドが実行される仕組みになっています。

```
public class PlayerController : MonoBehaviour
```

１．継承とは

＜概念、、ちょっと余談＞

MonoBehaiviourクラスを継承する場合、必ずUnityのヒエラルキー上に
アタッチしないといけない制約がある。
そのため、Unityに直接関わらないような機能を持つクラスの場合は、
無理にMonoBehaviourを継承する必要はない。
　例：サーバーとの通信を受け持つクラスなど…

この場合はnew演算子でクラスを作成し、使用していく。
(C++と同様)

１．継承とは

＜サブクラス作成＞

それでは、実際に作成していきましょう！

新しくスクリプトを作成し、PlayerController_2.csという名称にしてください。

作成したクラス：PlayerController_2.cs

元々作っていたクラス：PlayerController.cs

上記の2つを編集お願いします！

完成したら、Unityちゃんに元々ついているPlayerControllerを削除し、
PlayerController_2をつけてください！

１．継承とは

＜スクリプト解説＞

(1) PlayerController_2.cs

protected override void Start();

protected override void Update();

⇒protectedとはサブクラスのみ公開、という意味を持つ。

(public→protect→private)

overrideとは、スーパークラスのメソッドを引き継いで使用するための宣言。上記の宣言をすることで、EnemyBaseのStartやUpdateメソッドは使用されず、Enemy1のStartやUpdateが使用される。

base.Start(); / base.Update();

⇒スーパークラスのStartやUpdateの呼び出しを実施。

１．継承とは

＜スクリプト解説＞

(2) PlayerController.cs

protected virtual void Start();

protected virtual void Update();

⇒virtualを指定することで、このメソッドはオーバーライドすることが出来ますよ、という目印。

overrideとvirtualについて、C#とC++で同じ意味を持っているけど、使い方が異なるから注意が必要。

C#の場合は必ずoverrideを指定する必要あり。

１．継承とは

＜ダッシュ機能追加＞

PlayerController.csではダッシュ機能は無かったが、今回新たにダッシュ機能を追加する。(SHIFTボタンを押しながらキー入力でダッシュ)

作成したクラス：PlayerController_2.cs

元々作っていたクラス：PlayerController.cs

上記の2つを編集お願いします！

１．継承とは

＜スーパージャンプ機能追加＞

PlayerController.csでは無かったが、今回新たに大きくジャンプする機能を追加する。(Zボタンを押しながらスペースで大ジャンプ)

実装方法については、自分で考えて作成してみてください！

２．雑学的なこと

C++では出来てC＃やJavaでは出来ないこと　⇒　多重継承
この多重継承があるからC++は複雑になりがち。。

多重継承で起きる問題点

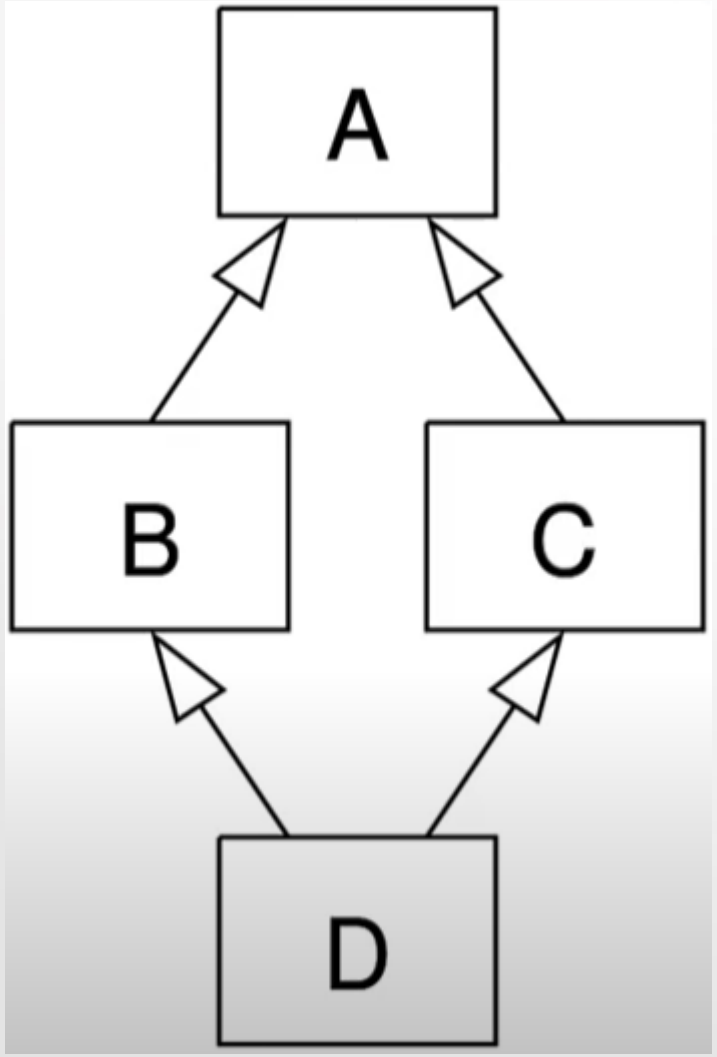
１．名前衝突
２．ダイアモンド継承

ここでは、上記２つの問題がどのようなものか解説する。

２．雑学的なこと(名前衝突)

```
Class A{
Public:
        void Print() { printf(“Aです”); }
};
Class B{
Public:
        void Print() { printf(“Bです”); }
};
Class C : A , B{
        A* a = new A();
        a->Print();
        B* b = new B();
        b->Print();
        C* c = new C();
        c->Print();          ←AかBのどちらかのPrintか分からないからエラー(解決する方法も一応あるが、、、)
};
```

２．雑学的なこと(ダイアモンド継承)



Dで定義しているメソッドを、
BとCでオーバーライドした場合、
Aから見てどちらを使えばよいか分からない。