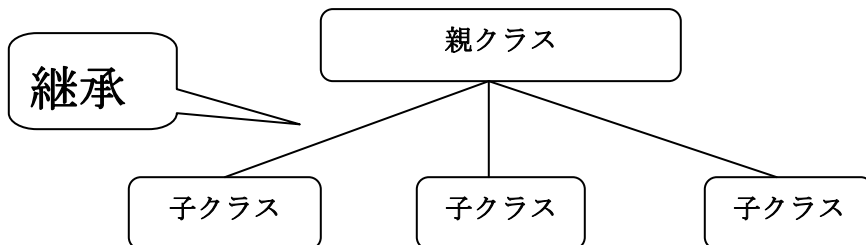


## クラス (class) について ～継承～

基本ではクラスの宣言とそのメソッドの製作、オブジェクトの使用方法を説明してきました。今度はそのクラスの継承について説明していきたいとおもいます。

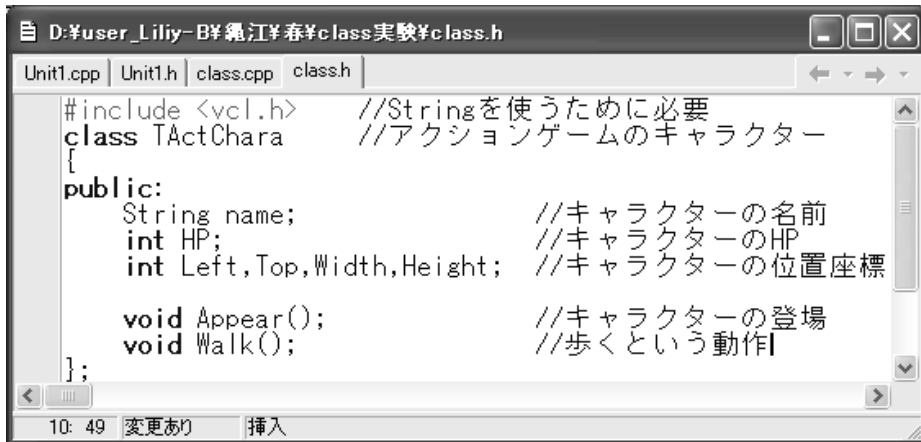
継承とはクラスの機能拡張をするもので、これを使うことにより一つのクラスから枝分かれ式に、機能、構造を共有し、さらに独自の機能を持ったクラスを作ることができます。

このようにして新たに機能を継承して作られたクラスを「子クラス、(サブクラス)」とよび、その子クラスが継承した、大元のクラスを「親クラス、(スーパークラス)」と呼びます。



この継承を使うことにより、新たなクラスを作るときまた一から作り直さなくとも、親クラスを継承することで、効率よくクラスを作ることができます。

これも実際にやってみましょう。これからやるプログラミングは「クラス (class) について ～基本～」の続きでやるのでそのセクションの「class.cpp」、「class.h」をあらかじめ作っておいてください。



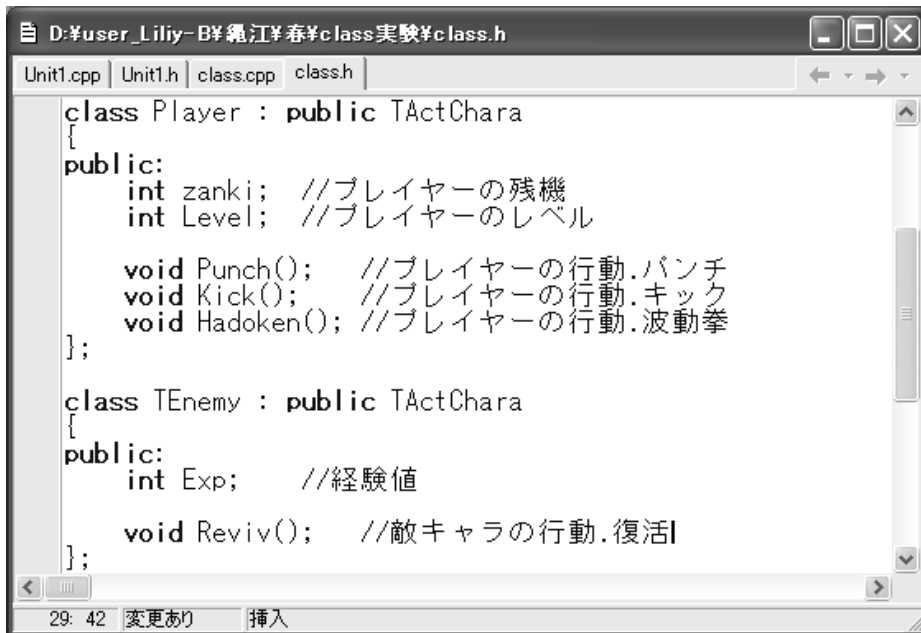
```
D:\user_Liliy-Bye江春¥class実験¥class.h
Unit1.cpp | Unit1.h | class.cpp | class.h
#include <vcl.h> //Stringを使うために必要
class TActChara //アクションゲームのキャラクター
{
public:
    String name; //キャラクターの名前
    int HP; //キャラクターのHP
    int Left,Top,Width,Height; //キャラクターの位置座標

    void Appear(); //キャラクターの登場
    void Walk(); //歩くという動作
};
10: 49 変更あり 挿入
```

上記のプログラムにはすでに TActChara というアクションゲームのキャラクターのクラスがあります。このクラスを親クラスにして、主人公を定義する子クラス、Player と敵キャラを定義する子クラス TEnemy を作ってみましょう。

継承するにはつぎのようにします。

<例>



```
D:\user_Liliy-Bye江春¥class実験¥class.h
Unit1.cpp | Unit1.h | class.cpp | class.h
class Player : public TActChara
{
public:
    int zanki; //プレイヤーの残機
    int Level; //プレイヤーのレベル

    void Punch(); //プレイヤーの行動.パンチ
    void Kick(); //プレイヤーの行動.キック
    void Hadoken(); //プレイヤーの行動.波動拳
};

class TEnemy : public TActChara
{
public:
    int Exp; //経験値

    void Reviv(); //敵キャラの行動.復活
};
29: 42 変更あり 挿入
```

```
class 子クラスの名前 : public 親クラスの名前
{
public:
    子クラスの新たなメンバ、メソッド
};
```

クラスの継承は上記のようにして行います。例のようにすると親クラス TActChare の変数 name, HP などのメンバ、関数 Apper () などのメソッドを受け継ぎ、また独自のメンバ、メソッドを持った子クラス Player と TEnemy ができあがります。

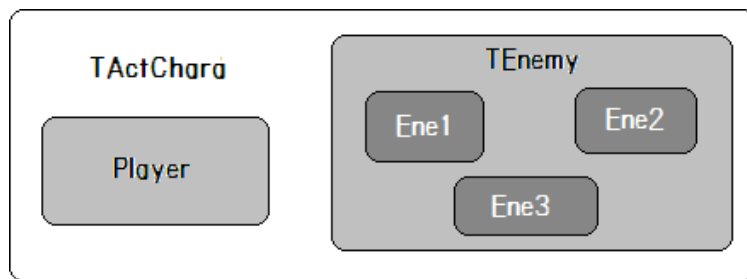
Player には HP 以外にも残機をあらわす変数、関数が必要だったりします。そのために TActChare とは別のクラスを一から作るのは面倒です。しかし、このように継承を使えば、TActChara のメンバ、メソッドをひきつぐので、Player に新たに必要なメンバ、メソッドを書き加えるだけでいいのです。

TEnemy も同じことができるのですが、ひと言で敵キャラといっても飛んでるものや、落ちてくるものなどさまざまです。TActChara の機能、構造だけでは、物足らなくなってきます。

そういう時は TEnemy を親クラスとして継承する子クラスをさらに作ってしまいます。TEnemy 自体 TActChara の子クラスですが、また別のクラスの親クラスになることが可能なのです。

こうして芋づる式にクラスとオブジェクトを生成します。

### <イメージ>



親クラスの継承、子クラスの生成についてはこれで以上です。使い方に関しては、親クラスも子クラスも同じ使い方が可能です。

```
//-----  
fastcall TForm1::TForm1(T  
: TForm(Owner)  
{  
    PL = new Player;  
    Slime = new Ene1;  
    Flingball = new Ene2;  
    Yogy = new Ene3;  
  
    PL->name = "棒人間Z";  
    PL->HP = 300;  
    PL->zanki = 5;  
    PL->Level = 1;  
    PL->Appear();  
  
    Slime->name = "ほりん";  
    Slime->HP = 100;  
    Slime->Exp = 15;  
  
    Flingball->name = "がんきゅう";  
    Flingball->HP = 80;  
    Flingball->Exp = 80;  
  
    Yogy->name = "ヨギー";  
    Yogy->HP = 1000;  
    Yogy->Exp = 20010;  
}  
}-----  
//-----  
fastcall TForm1::~TForm1()  
{  
    delete PL, Slime, Flingball, Yogy;  
}  
//-----  
void __fastcall TForm1::Timer1Timer(TO  
{  
    PL->Pun ();  
    PL->Kick ();  
    PL->Hadd ();  
  
    Slime->R ();  
    Flingball->v ();  
}  
}-----
```

忘れないように!!!

以上で説明は終わりです。是非ためしてみてください。

