Chapter-2 継承

C++の強力な機能の一つに継承というものがあります。この継承という考え方はオブジェクト指向を用いた設計、デザインパターンを学ぶ上で非常に重要な概念になります。このチャプターでは継承について見ていきましょう。

2.1 メンバ変数の継承

まずは、簡単な例で継承を見ていきましょう。例えばレースゲームを作成していることを考えてみて下さい。車には色々な車種があります。ワゴンR、フィット、ヴィッツ、フェラーリ、ポルシェなどなど。これらは当然車種ごとに、ステアリング性能、加速性能、燃費、車体フレームなど異なる点が多数存在します。しかし、どの車種も車であることに違いはありません。そのため共通点がいくつか存在します。タイヤは４つ付いていますし、ハンドルも付いています。アクセル、ブレーキなども付いているはずです。これらの共通部分を抽出して下記のようなクラスを作成します。

```
//車の基底クラス。
class CarBase{
private:
   Tire      tire[4];         //タイヤ
   Handle    handle;      //ハンドル
   BrakePedal  brakePedal;       //ブレーキペダル
   AxelPedal     axelPedal;    //アクセルペダル。
};
```

そして、各車種はCarBaseクラスを継承して実装します。

```
//フィット
class Fit : public CarBase{
Model model;  //フィットの車体モデル。
};
//ワゴンR
class WagonR : public CarBase{
  Model model;  //ワゴンRの車体モデル。
};
//フェラーリ
class Ferrari : public CarBase{
  Model model;  //フェラーリの車体モデル。
};
```

このように記述を行うことで、各車種はCarBaseクラスを継承することができます。そして、各クラスはメンバ変数として、tire[4]、handle、brakePedal、axelPedalを保持することになります。

2.2 メンバ関数の継承

継承はメンバ変数のみではなく、メンバ関数も継承することができます。先ほどの車を例にして見ていきましょう。車には走る処理のRun関数、窓を開けるOpenWindow関数、ドアを開けるOpenDoor関数などなど、いくつも共通の処理が存在するはずです。C++では、このような処理を基底クラスのメンバ関数として記述することで、どの車でも共通の処理として定義することができます。車の基底クラスは次のようになるでしょう。

```
class CarBase{
// 派生クラスでアクセスしたい場合はアクセス指定子をprotectedにする。
protected:
   Tire      tire[4];         //タイヤ
   Handle    handle;      //ハンドル
   BrakePedal  brakePedal;        //ブレーキペダル
   AxellPedal    axellPedal;      //アクセルペダル。
public:
    //走る処理
    void Run();
    //窓開ける。
    void OpenWIndow();
    //ドア開ける
    void OpenDoor();
};
```

この基底クラスを継承した派生クラスはメンバ関数として、Run、OpenWIndow、OpenDoorを保持するようになります。

3.3 継承すべきか委譲すべきか

オブジェクト指向のクラス設計において、車の例のような共通処理のクラス化は継承の他に委譲というテクニックが存在します。では継承と委譲の使い分けはどのようにすればいいのか？ この指針としてよく言われるものに下記のようなものがある。

クラス間の関係がis-aの場合は継承、has-aの場合は委譲。

is-aとは、「フェラーリは車である」のように、派生クラス=基底クラスが成り立つ場合のことを言います。has-aの場合は「フェラーリはブレーキペダルを持っている」という場合になります。 is-aの場合は継承を行うことを検討する、has-aの場合は委譲を行うことを検討してみることが設計の指針になります。