参照型

ある程度、スマートポインタの使い分けができるようになったかと思いますが、
ここでややこしくさせるのが、参照型です。

参照型は、ポインタと使い勝手はとても良く似ています。
値渡しと異なり、コピーを行わず、別名(ショートカット)を作ることにより、
実体へ参照を行います。

ポインタと明確に異なる点は、

- Null が許容されていない

- 初期化指定子で初期化を行う必要がある
  (Null が許容されていないため、確実に初期化する必要がある)

- 再代入(参照の再割り当て)が禁止されている

```
int* a = new int(1);
a = new int(1);
```

メモリリークはさておき、上記の例だと同じ変数 a に対して、
ポインタの再割り当てができています。
これが参照型だと、

```
int x = 1;
int y = 2;

int& s = x;          変数 s は x の参照。
s = y;               参照 s が y(2) で更新されたため、 x も 2となる。
                     → 値が更新されるだけで、
                        参照の再割り当てにはなっていない。
```

参照は、ポインタではありませんので、変にポインタ操作を
行われる心配もありませんし、実体を定義しているクラスに全て任せている
ので、メモリの解放なども意識する必要がありません。
使い勝手としては、weak_ptr に近いでしょう。
weak_ptr は shared_ptr を参照していますが、参照型は、実体を参照します。

特に今回のようなシンプルな設計方針だったら、
所有権をそこまで意識しませんので、weak_ptr をそのまま参照型に
置き換えるだけで、ポインタを排除することができます。

試しに、Stage クラスで使用する Player クラスを参照型に変更してみましょう。

Stage.h
```
public:

    // コンストラクタ
    Stage(Player& player);

private:

    Player& player_;
```

Stage.cpp
```
Stage::Stage(Player& player)
    : resMng_(ResourceManager::GetInstance()), player_(player)
{                                                    ↑ 初期化指定子
    activeName_ = NAME::MAIN_PLANET;
    step_ = 0.0f;
}
```

残りのエラー箇所は自分で修正しましょう。

参照型の方が安全に使用できますし、
オーバヘッド等も気にしなくて良いですので、
所有権が複数ある場合を除いて、参照型が好ましいでしょう。
そうなると、スマートポインタの時に解説したまとめが、

使用箇所が１つのクラス内に留まっている
　unique_ptr

複数クラスから、参照される
　shared_ptr

参照する側
　参照型

このようになり、weak_ptr から 参照型に置き換わります。
更に、 参照する側が ポインタ経由でないことから、
複数クラスから参照される側も、shared_ptr を使用する必要が無く、
unique_ptr とすることができます。

```
std::unique_ptr<Player> player_;
```

現在の設計で、GameScene の Player クラスを unique_ptr にすべきか
迷うところですが、

unique_ptr	→	単一の所有者(メモリ解放の責任者)
shared_ptr	→	複数の所有者(メモリ解放の責任者)

と考えるのであれば、Player クラスの解放は、
GameScene のみが管理すれば良い作りになっているので、
違和感もなくなるかと思います。

unique_ptr	→	1箇所で使用する
shared_ptr	→	複数箇所で使用する

と考えたいのであれば、shared_ptr のままにしておくのも手でしょう。

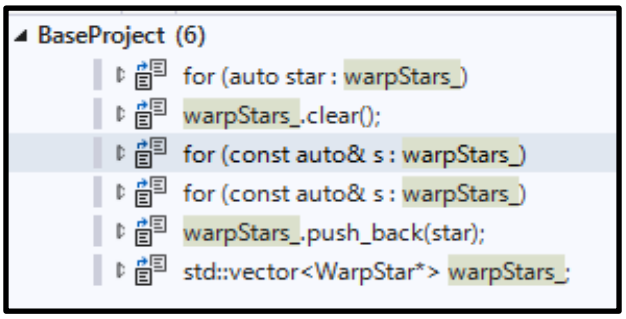
ややこしい所有権管理を行う必要がないのであれば、
参照型を積極的に使っていきたいところですが、使えないケースがあります。
これは、上記で上げたポインタと異なる点そのままで、

- Null が有りうる
- インスタンス生成時に初期化できない
- 再代入(別実体の再割り当て)が有りうる

上記に該当した場合は、参照型が使用できませんので、
shared_ptr と weak_ptr の組み合わせを使用する形になります。

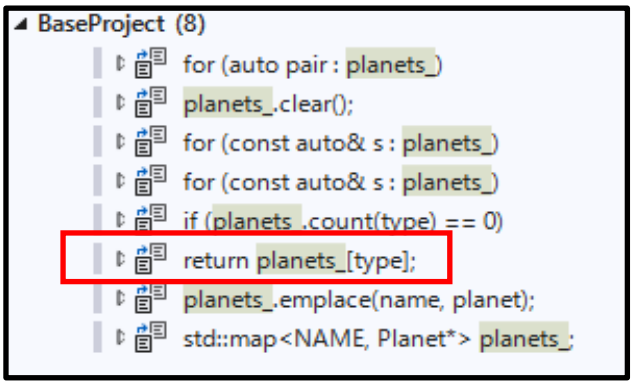
試しに、Stage クラスをリファクタリングしていきます。

最初に warpStars_ 。



他の箇所で使用されていないで、特に気にする必要なく変更していきます。

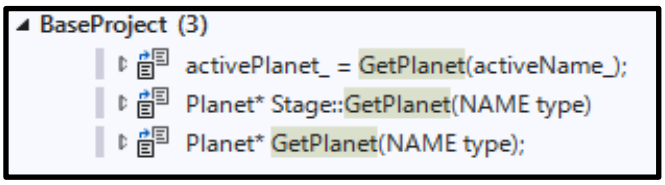
次に planets_ 。




```
Planet* Stage::GetPlanet(NAME type)
{
    if (planets_.count(type) == 0)
    {
        return nullPlanet;
    }

    return planets_[type];
}
```

ゲッターがありますが、
使用先は、同じStageクラス
です。



とはいえ、
他のメンバ変数に代入しており、
ステージが変わるごとに、
アクティブになる惑星を
切り替えています。

アクティブな惑星が保持されている、activePlanet_ は、
初期は Null ですし、ステージが変わるごとに、
惑星Ａ、惑星Ｂという風に再代入もされる形になりますので、
unique_ptr と 参照型の組み合わせが使えません。

この条件に合わせて修正していきましょう。

Stage クラスには、あと何か所か生ポインタがありますので、
スマートポインタや参照型にリファクタリングしていきましょう。