继承 复合 委托

复合：Composition

概念理解：在一个类里面声明了另外一种类的对象

复合函数下的构造和析构函数

- 构造由内到外
- 析构由外至内
- 两个类的生命周期是相同的

```
使用实列
利用链表实现的队列
```

委托：Delegation（composition by reference）

概念理解：在一个类里面有另外一个类的指针（两个类之间用指针相连）pimpl

```
class StringRep;
class String {
public:
String();
String(const char* s);
String(const String& s);
String &operator=(const String& s);
~String();
private:
StringRep* rep; // pimpl

class StringRep {
friend class String;
StringRep(const char* s);
~StringRep();
int count;
char* rep;
};
**********************************************************************
String是接口，所有的实现都在 StringRep 中完成
这种机制可以实现
```

- 两个类的生命周期不同

继承：Inheritance

- 语法：

```
class 派生类名:[继承方式] 基类名{
    派生类新增加的成员
};
```

- 继承的构造函数和析构函数

1. 构造是由内向外
2. 析构是由外向内

基类的析构函数必须是virtual,否则会出现underfined behavior 这样才会实现由外向内

继承需要搭配虚函数

语法：在类中成员函数之前写上virtual；

纯虚函数：一定要被子类重新定义 pure virtual;

子类的对象可以调用父类的函数