设计模式

C++实现23种设计模式

单例模式

```
class Singleton{
public:
    static Singleton * getInstance(){
        return instance;
    }
    static void delInstance(){
        if(instance != nullptr){
            delete instance;
            instance = nullptr
        }
    }
private:
    static Singleton *instance;
}
Singleton* Singleton::instance = new Singleton();
```

```
class Singleton{
public:
    static Singleton *getInstance(){
        if(instance == nullptr){
            std::lock_guard<std::mutex> lck(m_mutex);
            if(instance == nullptr){
                intance = new Singleton();
            }
        }
    }
    static void delInstance(){
        if(instance != nullptr){
            delete instance;
            instance = nullptr
        }
    }
private:
    static Singleton *instance;
    static std::mutex m_mutex;
}
Singleton* Singleton::instance = nullptr;

```

工厂模式

简单工厂

- 抽象商品类

- 商品类
- 工厂类（根据名字进行生产）

工厂方法

- 抽象商品类
- 商品类
- 抽象工厂
- 工厂类
- 使用两个基类指针，进行不同工厂的选择，然后调用工厂的方法生产商品

抽象工厂

- 一个工厂生产一系列商品
- 抽象商品类
- 商品类
- 抽象工厂
- 工厂类
- 使用两个基类指针，进行不同工厂的选择，然后调用工厂的方法生产一些列商品

观察者模式

- Subject：提供attach()，remove()，notify()
- Observer：提供update()

```
1
```

适配器模式

- 目标类
- 源类
- 适配器类

```
class Target{
        void Convert_110v();
};
class PowerPort220VAdaptee{
        void Output_220v(){}
};
class Adapter220V: public Target, private PowerPort220VAdaptee{
    void Convert_110v(){
        Output_220v();
    }
}
```