1. 单例模式

该类负责创建自己的对象，同时确保只有单个对象被创建。这个类提供了一种访问其唯一的对象的方式，可以直接访问，不需要实例化该类的对象。

主要解决：一个全局使用的类频繁地创建与销毁。

主要思想：构造函数是私有的

getInstance() 方法中需要使用同步锁 synchronized (Singleton.class) 防止多线程同时进入造成 instance 被多次实例化。

getInstance方法是static的

几种实现方式： 多线程安全需要同步

1. 懒汉式 (+线程安全，这种方式效率低)
2. 饿汉式：类加载时就new初始化，浪费内存，但是不用同步
3. 双重检验锁

public static Singleton getSingleton() {

if (singleton == null) {

synchronized (Singleton.class) {

if (singleton == null) {

singleton = new Singleton();

}

}

}

1. 策略模式

将算法封装成类，利用多态

解决问题：在有多种算法相似的情况下，使用 if...else 所带来的复杂和难以维护

1. 工厂模式
2. 装饰者模式
3. 观察者模式

静态static方法不能被重写的原因：static方法与一般方法存储的位置一样，一般虚函数会通过对象里面的虚函数表中的虚函数指针指向对应的虚函数。