目录

[设计模式 1](#_Toc500907213)

[单例模式 1](#_Toc500907214)

[工厂模式 1](#_Toc500907215)

# 设计模式

|  |  |
| --- | --- |
| 创建型模式 | 单例模式、抽象工厂模式、建造者模式、工厂模式、原型模式 |
| 结构型模式 | 适配器模式、桥接模式、装饰模式、组合模式、外观模式、享元模式、代理模式 |
| 行为型模式 | 模版方法模式、命令模式、迭代器模式、观察者模式、中介者模式、备忘录模式、解释器模式、状态模式、策略模式、职责链模式(责任链模式)、访问者模式 |

# 单例模式

|  |  |  |
| --- | --- | --- |
| 意图 | 保证一个类仅有一个实例，并提供一个访问它的全局访问点 | |
| 主要解决 | 一个全局使用的类频繁地创建与销毁 | |
| 何时使用 | 当您想控制实例数目，节省系统资源的时候 | |
| 使用场景 | WEB 中的计数器，不用每次刷新都在数据库里加一次，用单例先缓存起来 | |
| 创建的一个对象需要消耗的资源过多，比如 I/O 与数据库的连接等 | |
| 优点 | 在内存只有一个实例，减少了内存的开销，尤其是频繁的创建和销毁实例（比如管理学院首页页面缓存） | |
| 避免对资源的多重占用（比如写文件操作） | |
| 缺点 | 没有接口，不能继承，与单一职责原则冲突，一个类应该只关心内部逻辑，而不关心外面怎么样来实例化 | |
| 注意 | 单例类只能有一个实例 | 私有化该类的构造函数 |
| 单例类必须自己创建自己的唯一实例 | 通过new在本类中创建一个本类对象 |
| 对外提供一个可以获取该对象的方法 | 定义一个公有的方法，将在该类中所创建的对象返回 |
| 双重校验锁：  private volatile static Singleton singleton;  private Singleton() {}  public static Singleton getSingleton() {  if (singleton == null) {  synchronized (Singleton.class) {  if (singleton == null) {  singleton = new Singleton();  }}}  return singleton;} | | 静态内部类：基于classloder机制避免多线程同步问题  public class Singleton {  private static class Holder {  private static final Singleton INSTANCE = new Singleton();  }  public static Singleton getInstance() {  return Holder.INSTANCE;  }} |

# 工厂模式

|  |  |  |
| --- | --- | --- |
| 意图 | 定义一个创建对象的接口，让其子类自己决定实例化哪一个工厂类，工厂模式使其创建过程延迟到子类进行 | |
| 主要解决 | 主要解决接口选择的问题 | |
| 何时使用 | 我们明确地计划不同条件下创建不同实例时 | |
| 使用场景 | 日志记录器：记录可能记录到本地硬盘、系统事件、远程服务器等，用户可以选择记录日志到什么地方 | |
| 数据库访问，当用户不知道最后系统采用哪一类数据库，以及数据库可能有变化时 | |
| 优点 | 一个调用者想创建一个对象，只要知道其名称就可以了 | |
| 扩展性高，如果想增加一个产品，只要扩展一个工厂类就可以 | |
| 屏蔽产品的具体实现，调用者只关心产品的接口 | |
| 缺点 | 每次增加一个产品时，都需要增加一个具体类和对象实现工厂，使得系统中类的个数成倍增加，在一定程度上增加了系统的复杂度，同时也增加了系统具体类的依赖 | |
| 步骤 | 创建一个接口 | public interface Shape{ public void draw() {…} } |
| 创建实现接口的实体类 | public class circle implements Shape {…} … |
| 创建一个工厂，生成基于给定信息的实体类对象 | public class ShapeFactory {  public Shape getShape(String shapeType) {  if (shapeType == null) {  return null;  }  if (shapeType.equalsIgnoreCase("CIRCLE")) {  return new Circle(); … |