Java高级工程师常见面试题（九）-设计模式

<https://blog.csdn.net/u014378181/article/details/93487544>

1. 单例模式：饱汉、饿汉。以及饿汉中的延迟加载,双重检查

**1）饿汉模式：**

**//饿汉模式，很饿很着急，所以类加载时即创建实例对象**

1. public class Singleton1 {
3. private static Singleton1 singleton = new Singleton1();
5. private Singleton1(){
7. }
9. public static Singleton1 getInstance(){
11. return singleton;
13. }
15. }

**2）饱汉模式：**

**//饱汉模式，很饱不着急，延迟加载，啥时候用啥时候创建实例，存在线程安全问题**

1. public class Singleton2 {
3. private static Singleton2 singleton;
5. private Singleton2(){
7. }
9. public static synchronized Singleton2 getInstance(){
11. if(singleton == null)
13. singleton = new Singleton2();
15. return singleton;
17. }
19. }

**3）双重锁模式：**

**//饱汉模式的双重锁模式，提高效率**

1. public class Singleton3 {
3. private static Singleton3 singleton;
5. private Singleton3(){
7. }
9. public static Singleton3 getInstance(){
11. if(singleton == null){
13. synchronized(Singleton3.class){
15. if(singleton == null){
17. singleton = new Singleton3();
19. }
21. }
23. }
25. return singleton;
27. }
29. }

**几种模式的比较：**

1、饿汉模式是线程安全的，因为实例对象在类加载过程中就会被创建，在getInstance()方法中只是直接返回对象引用。之所以被称为“饿汉”，是因为这种模式创建实例对象比较“急”，真的是饿坏了.

好处：简单明了，无需关注线程安全问题。

缺点：如果在一个大环境下使用了过多的饿汉单例，则会生产出过多的实例对象，无论你是否要使用他们。

2、饱汉模式不是线程安全的，因为是在需要的时候才会产生实例对象，生产之前会判断对象引用是否为空，这里，如果多个线程同时进入判断，就会生成多个实例对象，这是不符合单例的思想的。所以饱汉模式为了保证线程安全，就用synchronized关键字标识了方法。之所以被称为“饱汉”，因为它很饱，不急着生产实例，在需要的时候才会生产。

好处：延时加载，用的时候才会生产对象。

缺点：需要保证同步，付出效率的代价。

3、双重锁模式，是饱汉模式的优化，进行双重判断，当已经创建过实例对象后就无需加锁，提高效率。也是一种推荐使用的方式。

2. 工厂模式、装饰者模式、观察者模式。

参考：

[《设计模式（四）-工厂模式（Factory Method Pattern/Factory Pattern）——将实例的生成交给子类》](https://blog.csdn.net/u014378181/article/details/79146806)

[《设计模式（十二）-装饰器模式（Decorator Pattern）——装饰边框与被饰物的一致性》](https://blog.csdn.net/u014378181/article/details/79235177)

[《设计模式（十七）-观察者模式（Observer Pattern）——发送状态变化通知》](https://blog.csdn.net/u014378181/article/details/79260589)

3. 工厂方法模式的优点（低耦合、高内聚，开放封闭原则）

参考:

[《 设计模式（四）-工厂模式（Factory Method Pattern/Factory Pattern）——将实例的生成交给子类》](https://blog.csdn.net/u014378181/article/details/79146806)