**设计模式选择/填空【可以转变成选择题】【结合文件A中的“适用场景/应用实例”】**

**（1）原则：**

1、在面向对象设计原则中，【里氏代换原则】是指子类应该可以替换父类并出现在父类能够出现的任何地方。

2、为了防止界面组件之间产生复杂的引用关系，可以提供一个中央控制器来专门负责控制界面组件之间的相互引用，这是使用【迪米特法则】的重构实例。

3、所谓高内聚，是指一个软件模块内各个元素彼此结合的紧密程度要高，即一个软件模块是由相关性很强的代码组成，只负责一项任务，也就是常说的【单一职责】原则。

4、对扩展开放，对修改关闭，是【开闭】原则。

5、面向对象系统中功能复用的两种最常用技术是类继承和对象组合。类继承允许你根据其他类的实现来定义一个类的实现。这种通过生成子类的复用通常被称为【白箱复用】。

6、在继承方式中，父类的内部细节对子类可见。对象组合是类继承之外的另一种复用选择。新的更复杂的功能可以通过组装或组合对象来获得。对象组合要求被组合的对象具有良好定义的接口。这种复用风格被称为【黑箱复用】，因为对象的内部细节是不可见的。

**（2）具体模式：**

1. 当一个系统要由多个产品系列中的一个来配置时，我们选择【抽象工厂】模式；
2. 当创建复杂对象的算法应该独立于该对象的组成部分以及它们的装配方式时，我们选择【建造者】模式；
3. 当一个类希望由它的子类来指定它所创建的对象的时候，我们选择【工厂方法】模式；
4. 当一个系统应该独立于它的产品创建、构成和表示时，要使用【抽象工厂】模式；
5. 当类只能有一个实例而且客户可以从一个众所周知的访问点访问它时，要使用【单例】模式；
6. 当你想使用一个已经存在的类，而它的接口不符合你的需求，要使用【适配器】模式；
7. 当你想表示对象的部分-整体层次结构，要使用【组合】模式；
8. 当不能采用生成子类的方法进行扩充时，要使用【装饰】模式；
9. 当你要为一个复杂子系统提供一个简单接口时，要使用【外观】模式
10. 一个应用程序使用了大量的对象，要使用【享元】模式；
11. 在需要用比较通用和复杂的对象指针代替简单的指针的时候，使用【代理】模式；
12. 你想在不明确指定接收者的情况下向多个对象中的一个提交一个请求时，要使用【职责链】模式；
13. 访问一个聚合对象的内容而无需暴露它的内部表示，要使用【迭代器】模式；
14. 一组对象以定义良好但是复杂的方式进行通信。产生的相互依赖关系结构混乱且难以理解，要使用【中介者】模式；
15. 当对一个对象的改变需要同时改变其它对象，而不知道具体有多少对象有待改变，要使用【观察者】模式；
16. 一个对象的行为取决于它的状态，并且它必须在运行时刻根据状态改变它的行为，要使用【状态】模式；
17. 【简单工厂】模式可以根据参数的不同返回不同类的实例。
18. 当需要创建的对象具有复杂的内部结构时，为了逐步构造一个完整的对象，并使得对象的创建更具弹性，可以使用【建造者】模式。
19. 已知某子系统为外界提供功能服务，但该子系统中存在很多粒度十分小的类，不便被外界系统直接使用，采用【外观】模式可以定义一个高层接口，这个接口使得这一子系统更加容易使用。
20. 当应用程序由于使用大量的对象造成很大的存储开销时，可以通过【享元】模式运用共享技术来有效地支持大量细粒度对象的重用。
21. **在某电子商务系统中，站内检索功能的基本实现过程如下：先搜索商品表查询相关信息，再搜索商品类型表查询相关信息，然后搜索新闻表查询相关信息。该搜索次序可以灵活地调整并且可能会加入新的待查询的数据表。对于该站内检索功能，可采用【职责链】模式进行设计。**
22. 某软件公司计划开发一套简单的数据库同步指令，通过这套指令可以对数据库中的数据和结构进行快速备份，例如输入指令“MOVE VIEW v\_FemaleEmployee FROM srcDB to desDB”表示将数据库srcDB中的视图 v\_FemaleEmployee移动至数据库desDB中。针对以上需求，可以使用【解释器】模式来设计该数据库同步指令系统。
23. 【状态】模式允许一个对象在其内部状态改变时改变它的行为。
24. 如果需要对一个集合对象中不同类型的元素施加不同的操作，可以采用【访问者】模式。
25. 支持转换行为 【适配器】
26. 支持转移行为 【命令】
27. 支持简化行为 【外观】
28. 支持控制访问行为 【代理】
29. 支持发布/订阅 【观察者】
30. 可依据对象结构控制对象的产生 【建造者】
31. 可依据对象的数据决定对象的行为 【状态】
32. 能够在运行时改变一个对象的行为的算法 【策略】
33. 可简化多个不同种类的对象之间复杂的关联或依赖关系 【中介者】
34. 能够在不清楚具体步骤实现细节的情况下，给出一个过程的完整的流程定义 【模板方法】
35. **判断**

1、一个类承担的职责越多，越容易复用，被复用的可能性越大。【错】

2、工厂方法模式对应唯一一个产品等级结构，而抽象工厂模式则需要面对多个产品等级结构。【对】

3、命令模式将一个请求封装为一个对象，从而使用户可用不同的请求对客户进行参数化。【对】

4、在某酒店客房预订系统中，房间具有空闲、已预订、已入住等多个不同的状态，且在不同的状态下用户对于房间具有不同的操作行为，例如空闲的房间不支持退房操作，已入住的房间不支持再次入住操作等。此时可使用状态模式来设计该系统，状态模式可以封装对象状态的转换过程，增加新的状态无须修改已有代码，完全符合开闭原则。【错，状态模式并不完全符合开闭原则】

5、在某系统中经常需要重复创建一些相同或者相似的对象，可以考虑采用模板方法模式。【错，“创建”】

6、控制对一个对象的访问，给不同的用户提供不同级别的使用权限时可以考虑使用虚拟代理。【错，保护代理】

7、在某电子商务系统中站内检索功能的基本实现过程如下：先搜索商品表查询相关信息，再搜索商品类型表查询相关信息，然后搜索新闻表查询相关信息。该搜索次序可以灵活地调整并且可能会加入新的待查询的数据表。对于该站内检索功能，可采用职责链模式进行设计。【对】

8.、Windows操作系统中的应用程序桌面快捷方式体现了代理模式。【对】

9、建造者模式允许用户可以只通过指定复杂对象的类型就可以创建它们，而不需要知道内部的具体构建细节。【对】

10、 Java I/O库的设计应用了组合模式，其中 OutputStream类和 InputStream类充当抽象构件角色。【错，Java I/O库的设计应用的是装饰模式】

11、合成复用将已有对象纳入新对象中，使之成为新对象的一部分，新对象可以调用已有对象的方法，从而实现行为的复用。【对】

12、一个软件实体应当尽可能少地与其他软件实体发生相互作用，这样当一个模块修改时就会尽量少地影响其他模块，扩展会相对容易。【对】

13、接口应该尽量细化，同时接口中的方法应该尽可能少，理想情况是在每个接口中只定义一个方法，该接口使用起来最为方便。【错，不是越细化越好】

14、在某多功能文本编辑器中允许用户插入图片、动画和视频等多媒体素材，为了节约系统资源，可使用享元模式来处理相同的素材。【对】

15、在某财务系统中需要将阿拉伯数字(例如1、2、3 等)转换成中文大写数字(例如壹、贰、叁等),并且系统需要支持中文大写数字的基本数学运算，例如“壹拾贰加贰拾捌”可计算得到“肆拾”，可以使用解释器模式来设计和实现该数字转换和计算功能。【对】

16、某数据处理软件需要提供一个数据恢复功能，用户在操作过程中如果发生异常操作，可以将数据恢复到某一个历史状态。针对该需求可以采用备忘录模式来设计该数据恢复功能。【对】

17、访问者模式让用户可以在不改变各元素对应的类的前提下定义作用于这些元素的新操作，在访问者模式中增加新的操作和元素都很方便，完全符合开闭原则。【错】

18、采用模板方法模式可以定义一个操作中算法的骨架，而将一些步骤延迟到子类中实现。模板方法模式是一种代码复用技术，可以让系统更加符合合成复用原则。【错，模板方法采用的是继承，而不是组合】

19、中介者模式通过引入一个中介对象来封装一系列其他对象之间的交互，降低对象之间的耦合度，使得系统更加符合迪米特法则。【对】

20、电视机遥控器的设计中蕴含了迭代器模式和命令模式的思想。【对】