java23种设计模式中常用的九种Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022_java23_E7_A7_8D_E8_c104_644786.htm 1 Factory Pattern（工厂模式） 上榜理由：将程序中创建对象的操作，单独出来处理，大大提高了系统扩展的柔性，接口的抽象化处理给相互依赖的对象创建提供了最好的抽象模式。 2 Facade Pattern 上榜理由：将表现层和逻辑层隔离，封装底层的复杂处理，为用户提供简单的接口，这样的例子随处可见。门面模式很多时候更是一种系统架构的设计，在我所做的项目中，就实现了门面模式的接口，为复杂系统的解耦提供了最好的解决方案。 3 Command Pattern 上榜理由：将请求封装为对象，从而将命令的执行和责任分开。通常在队列中等待命令，这和现实多么的相似呀。如果你喜欢发号施令，请考虑你的ICommond吧。 4 Strategy Pattern 上榜理由：策略模式，将易于变化的部分封装为接口，通常Strategy 封装一些运算法则，使之能互换。Bruce Zhang在他的博客中提到策略模式其实是一种“面向接口”的编程方法，真是恰如其分。 5 Iterator Pattern 上榜理由：相信任何的系统中，都会用到数组、集合、链表、队列这样的类型吧，那么你就不得不关心迭代模式的来龙去脉。在遍历算法中，迭代模式提供了遍历的顺序访问容器，GOF给出的定义为：提供一种方法访问一个容器（container）对象中各个元素，而又不需暴露该对象的内部细节。.NET 中就是使用了迭代器来创建用于foreach的集合。 6 Adapter Pattern 上榜理由：在原类型不做任何改变的情况下，扩展了新的接口，灵活且多样的适配一切旧俗。这种打破旧框框，适配新

格局的思想，是面向对象的精髓。以继承方式实现的类的Adapter模式和以聚合方式实现的对象的Adapter模式，各有千秋，各取所长。看来，把它叫做包装器一点也不为过， 7 Observer Pattern 上榜理由：定义对象间的一种一对多的依赖关系,当一个对象的状态发生改变时, 所有依赖于它的对象都得到通知并被自动更新。观察者和被观察者的分开，为模块划分提供了清晰的界限。在.NET中使用委托和事件可以更好的实现观察者模式，事件的注册和撤销不就对应着观察者对其对象的观察吗？ 8 Bridge Pattern 上榜理由：把实现和逻辑分开，对于我们深刻理解面向对象的聚合复用的思想甚有助益。 9 Singleton Pattern(单例模式) 上榜理由：改善全局变量和命名空间的冲突，可以说是一种改良了的全局变量。这种一个类只有一个实例，且提供一个访问全局点的方式，更加灵活的保证了实例的创建和访问约束。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com