设计模式基本思想 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/167/2021_2022__E8_AE_BE_E8_AE_A1_E6_A8_A1_E5_c104_167295.htm 好的系统设计追求如下特性：l 可扩展性（ Extensibility ）：新的功能或特性很容易加入到系统中来；l 灵活性（ Flexibility ）：可以允许代码修改平稳发生，对一处的修改不会波及到很多其他模块；l 可插入性（ Pluggability ）：可以很容易地将一个类或组件抽出去，同时将另一个有相同接口的类 / 接口加入进来。 具有如上特性的系统才有真正的可维护性和可复用性。而可维护性和可复用性对一个持 续接入新需求，现有功能逐步完善，新功能不断丰富，版本不会终止的大型软件产品来说至关重要。 传统的复用包括：代码的 copy 复用，算法的复用，数据结构的复用。 在面向对象领域，数据的抽象化、封装、继承和多态性是几项最重要的语言特性，这些特性使得一个系统可以在更高的层次上提供可复用性。数据的抽象化和继承关系使得概念和定义可以复用；多态性使得实现和应用可以复用；而抽象化和封装可以保持和促进系统的可维护性。这样，复用的焦点不再集中在函数和算法等具体实现细节上，而是集中在最重要的宏观的业务逻辑的抽象层次上。复用焦点的倒转不是因为实现细节的复用不再重要，而是因为这些细节上的复用往往已经做的很好（例如，很容易找到并应用成熟的数据结构类库等），而真正冲击系统的是其要实现业务的千变万化。 本质上说，如果说一个软件的需求是永不变更或发展的，该软件也就不需要任何设计，怎么编码实现都行，只要需求满足，性能达标。但事实上，软件的本性就是不

断增强，不断拓展的，不断变化的。我们可以控制指尖流淌出的每行代码，但控制不了奉为上帝的用户的需求。编码结束，测试全部通过，用户在试用过程中才发现原来的需求有问题，需要变更或提出新需求，怎么办？向用户抗议：需求总在变，没法做！？平抑心中的抱怨，加班加点大量的修改代码，疯狂的测试，依然是时间紧迫，心中没底？抑或了然于胸：这个变更或小需求合理，系统很方便纳入；于是坦然地和用户协商下一个交付时间点？ 要使系统最大程度的适应需求的变更或新增，就必须在其要实现的宏观业务逻辑的抽象复用上下功夫。而设计模式就是综合运用面向对象技术和特性来提高业务逻辑可复用性的常用方法和经验的提取和汇总。 掌握 23 种设计模式的关键是理解它们的共通目的：使所设计的软件系统在一般或特定（系统将来在特定点上扩展的可能性大）场景下，尽可能的对扩展开放，对修改关闭。即面对新需求或需求变更时，容易开发独立于既有代码的新代码接入到现有系统或对现有代码做可控的少量修改，而不是在现有代码基础上做大量的增、删、改。为了这一目的， 23 种设计模式贯穿了面向对象编程的基本原则：l 面向接口或抽象编程，而不是面向实现编程。l 一个对象应当对其他对象有尽可能少的了解，即松耦合。l 纯粹的功能复用时，尽量不要使用继承，而是使用组合。使已有的对象成为新对象的一部分；新的对象通过向这些对象的委派达到复用已有功能的目的。 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