生活中的设计模式原则

爱科技勤折腾的苹果二 2024-2-26

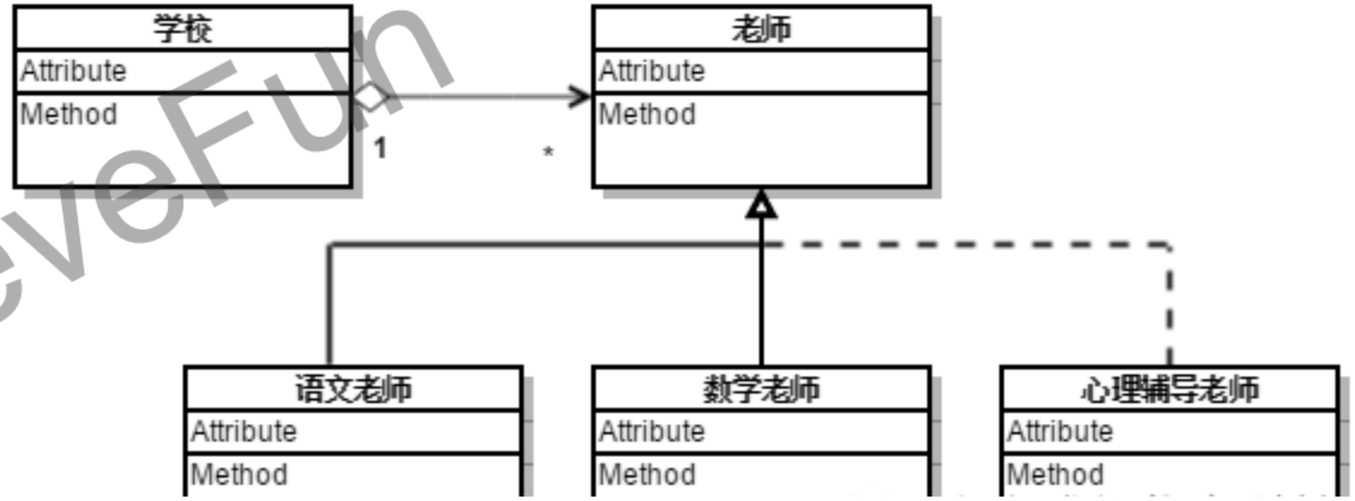
讨论内容

1. 开闭原则 OCP（Open-Close Principle）
2. 依赖倒转原则DIP（Dependence Inversion Principle）
3. 合成聚合复用原则CARP composite /aggregate reuse principle
4. 接口隔离原则 Interface Segregation Principle
5. 里氏代换原则 Liskov Substitution Principle
6. 迪米特法则 Law of Demeter

在软件工程中，设计模式（design pattern）是对软件设计中普遍存在的各种问题提出的解决思路和方案。无数软件开发人员经过长时间的实践总结出来这些经验和基本方法。

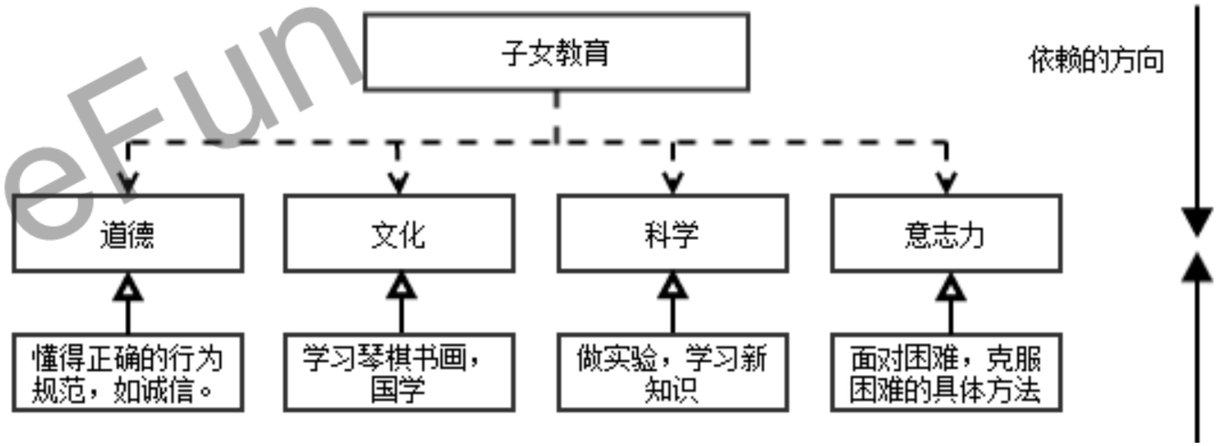
1. 开闭原则 OCP（Open-Close Principle）

- 一个软件实体应当对扩展开放，对修改关闭。
  Software entities should be open or extension, but closed for modification.
- 通过**扩展**已有的软件系统，可以提供新的行为，以满足对软件的新需求，使变化中的软件系统由一定的**适应性和灵活性**。
- 已有的软件模块，特别是最重要的抽象层模块不可被修改，使变化中的软件系统有一定的稳定性和延续性。
- 一个系统不可扩展，就会失去使用的价值，一个系统总是需要修改，就会失去重心。
- 闭：不破坏既有规则也就是重要的抽象层。
- 开：做必要的扩展满足新的功能。



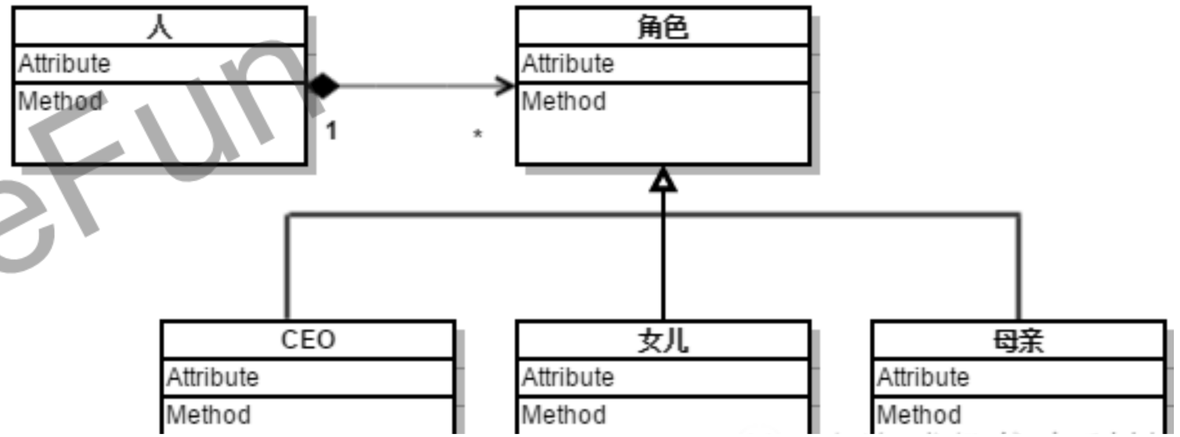
2. 依赖倒转原则DIP（Dependence Inversion Principle

- 抽象不应当依赖于细节，细节应当依赖于抽象。Abstractions should not depend upon details. Details should depend upon abstractions. or Program to an interface, not an implementation.
- 抽象：系统的战略性决定，每个战术不应该偏离基本的战略，或者说是一个原则。抽象不应当依赖于细节，也就是说战略决策（教育理念）决定战术实施（具体的教育方法和方式）。
- 由于面向过程的设计中倾向于**使高层次的模块依赖于低层次的模块**，所以在面向对象的设计中，有些人会基于同样的思想来做设计，就会导致依赖错误。



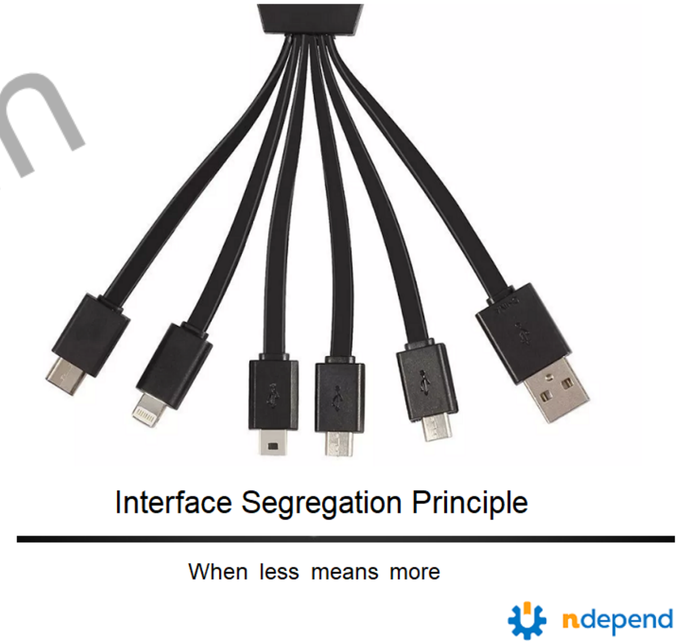
3. 合成聚合复用原则CARP composite /aggregate reuse principle

- 尽量使用**合成/聚合**，尽量**不要使用继承**。
- composite /aggregate 也即Has-A，即某个角色具有某一项责任。比如CEO担负着公司管理和发展的重大责任。
- 继承，Is-A是说明一个类是另一个类的一种。



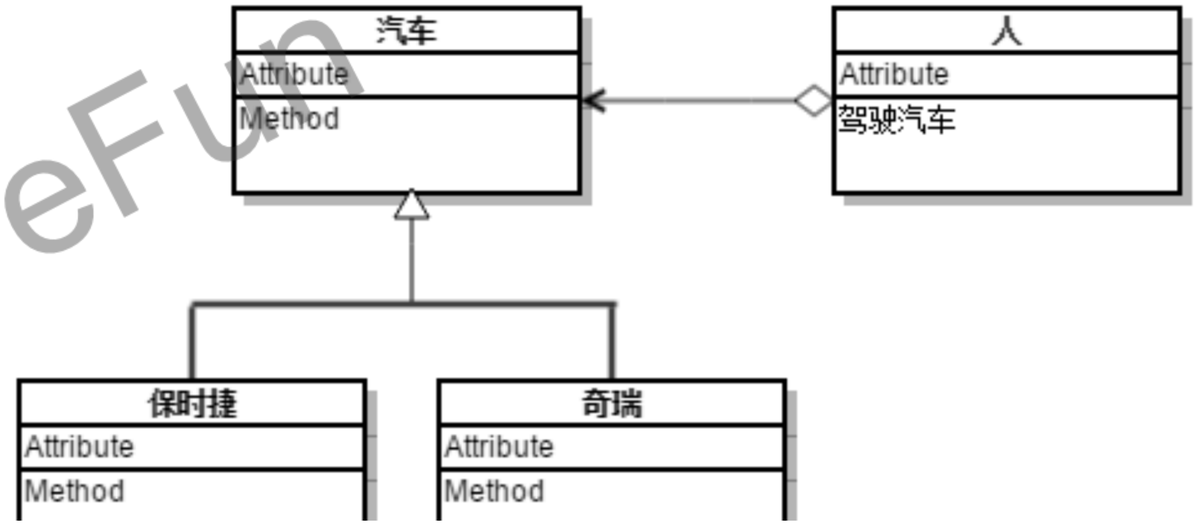
4. 接口隔离原则 Interface Segregation Principle

- 使用**多个专门**的接口比使用**单一**的总接口要好。一个类对另一个类的依赖性应当是建立在最小的接口上。
- 专人做专事，角色和职责清晰。
- 图片来源：https://blog.ndepend.com/solid-design-the-interface-segregation-principle-isp/



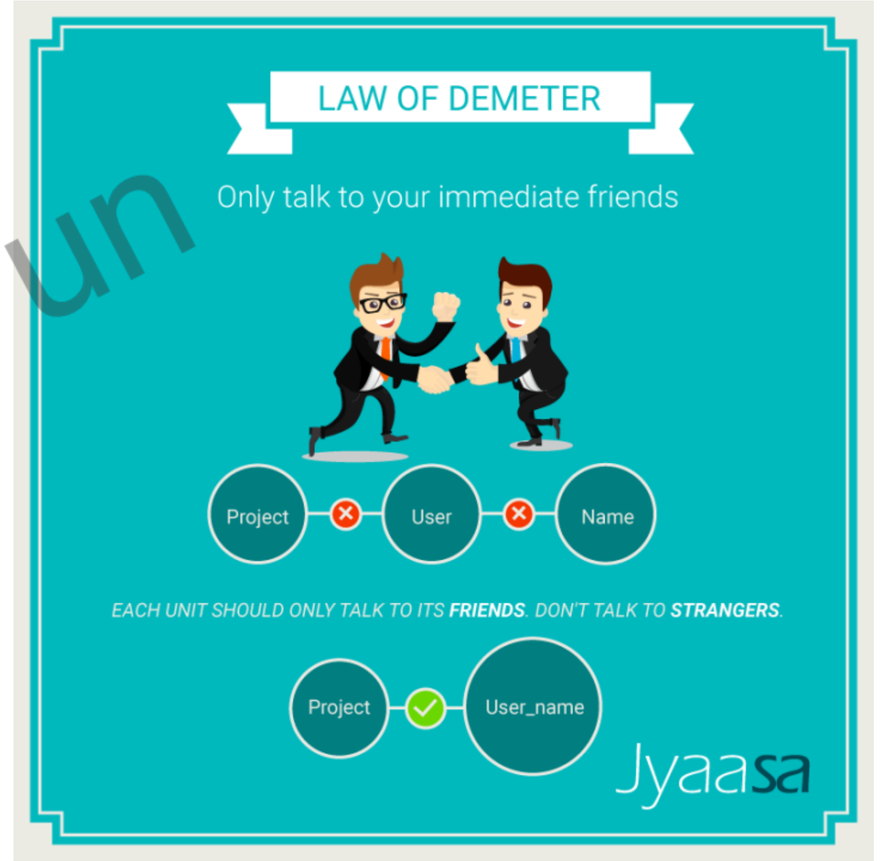
5. 里氏代换原则 Liskov Substitution Principle

- 开闭原则强调了面向对象设计的基本原则，也就是创建抽象化，并且从抽象化导出具体化。
- 利用继承关系和里氏代换原则，就可以实现从抽象化到具体化的过程了。
- **里氏代换**是指衍生类可以替换掉基类，即衍生类具备基类的所有特性。



6. 迪米特法则 Law of Demeter

- 最少知识原则（Least Knowledge Principle），是面向对象设计的一种法则，可以理解为弱耦合。包括以下基本定义
- 1.只与直接的朋友们交流
- 2.不和陌生人说话
- 3.每一个软件模块对其他的模块只有最少的知识
- 迪米特法则控制对象之间的信息流量、流向以及信息所带来的影响，也就是控制信息过载。
- 在系统设计的时候，要平衡弱耦合和结构清晰。



图片来源：https://media.licdn.com/dms/image/C5612AQFtpLD202tBYA/article-cover_image-shrink_720_1280/0/1642246239793

弱耦合

Tight coupling:
1. More Interdependency
2. More coordination
3. More information flow

Loose coupling:
1. Less Interdependency
2. Less coordination
3. Less information flow

图片来源：https://www.geeksforgeeks.org/coupling-in-java/

总结

1. 开闭原则 OCP（Open-Close Principle）
2. 依赖倒转原则DIP（Dependence Inversion Principle）
3. 合成聚合复用原则CARP composite /aggregate reuse principle
4. 接口隔离原则 Interface Segregation Principle
5. 里氏代换原则 Liskov Substitution Principle
6. 迪米特法则 Law of Demeter