# 概述

参考书籍：《设计模式：可复用面向对象软件的基础》

# 分类

# 设计原则

## 依赖倒置原则DIP

高层模块（稳定）不应该依赖于低层模块（变化），二者都应该依赖于抽象（稳定）。

抽象（稳定）不应该依赖于实现细节（变化），实现细节应该依赖于抽象（稳定）。

## 开放封闭原则OCP

对扩展开放，对更改封闭。

类模块应该是可扩展的，但是不可修改。

## 单一职责原则SRP

一个类应该仅有一个引起它变化的原因。

变化的方向隐含着类的责任。

## Liskov替换原则LSP

子类必须能够替换它们的基本（is-a）。

继承表达类型抽象。

## 接口隔离原则ISP

不应该强迫客户程序依赖它们不用的方法。

接口应该小而完备。

## 优先使用对象组合，而不是类继承

类继承通常为“白箱复用”，对象组合通常为“黑箱复用”。

继承在某种程度上破坏了封装性，子类父类耦合度高。

而对象组合则只要求被组合的对象具有良好定义的接口，耦合度低。

## 封装变化点

使用封装来创建对象之间的分界层，让设计者可以在分界层的一侧进行修改，而不会对另一侧产生不良的影响，从而实现层次间的松耦合。

## 针对接口编程，而不是针对实现编程

不讲变量类型声明为某个特定的具体类，而是声明为某个接口。

客户程序无需获知对象的具体类型，只需要知道对象所具有的接口。

减少系统中各部分的依赖关系，从而实现“高内聚、松耦合”的类型设计方案。