软件体系结构

Zhenyan Ji

Beijing Jiaotong University

行为型模式--策略模式

策略模式

为什么需要策略模式?

- 程序Context中封装了许多相关的算法。Client程序可以选择这些算法之一，或者某些情况下，Context可能为Client选择最佳的算法。

策略模式

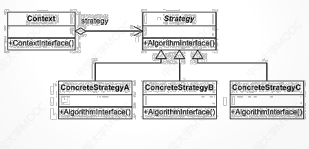
- 保存不同格式的文件。
- 使用不同算法压缩文件。
- 利用不同压缩方案捕捉视频数据
- 使用不同的换行策略来显示文本数据。
- 使用不同方式绘制相同的数据：线图、条形图或饼图。

策略模式

目的

- 定义一系列算法，对每个算法进行封装并使它们可互换。
- Strategy使算法与Client程序解耦。

策略模式



策略模式

Strategy

- 为所支持的算法声明一个公共接口。Context使用此接口调用由ConcreteStrategy定义的算法。

ConcreteStrategy

- 遵循Strategy接口，实现算法。

策略模式

Context

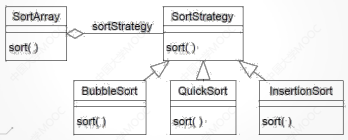
- 配置ConcreteStrategy对象
- 维护对Strategy对象的引用
- 可以定义允许Strategy访问其数据的接口

策略模式例子1

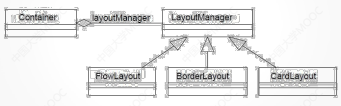
场景：一个类在运行时确定应该用什么算法对数组进行排序。有许多不同的排序算法可用。

解决方案：使用策略模式封装不同的排序算法。

策略模式例子1



策略模式例子2

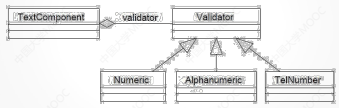


策略模式例子3

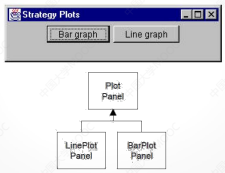
场景：GUI文本组件对象在运行时确定应该用什么策略验证用户输入。许多不同的验证策略都可能：数字字段、字母数字字段、电话号码字段等。

解决方案：使用策略模式封装不同的输入验证策略！

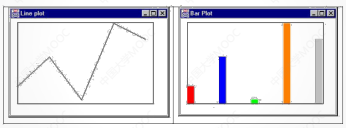
策略模式例子3



策略模式例子4



策略模式例子4



策略模式&状态模式

不同：

- Strategy对象封装算法
- State对象封装与状态相关的行为（及可能的状态转换）。
- Strategy：用户一般选择采用哪个策略，在Context类中一次只有一个策略被实例化并激活。
- State：可能所有的状态都激活，且它们之间可能频繁发生切换。

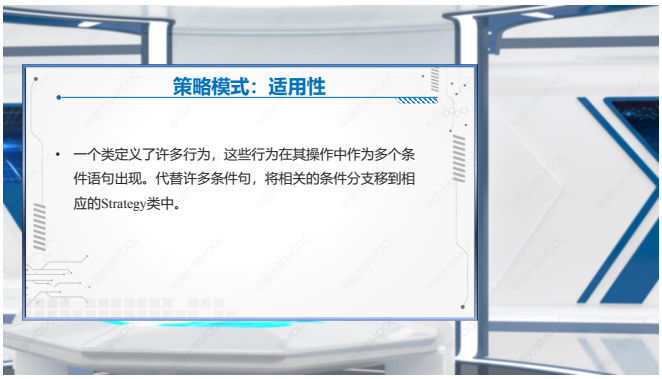
策略模式和状态模式

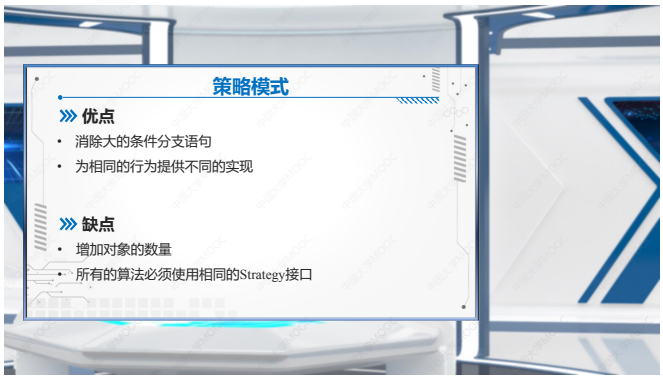
- Strategy封装了几个或多或少做相同事情的算法，而状态封装了不同状态下对象的不同行为。
- 在Strategy模式中，不存在不同状态之间的迁移。

策略模式：适用性

使用策略模式：

- 许多相关的类只在行为上不同。
- 算法有不同的变种。
- 算法使用client不知道的数据。使用策略模式可避免暴露复杂的、特定于算法的数据结构。


策略模式：适用性
一个类定义了许多行为，这些行为在其操作中作为多个条件语句出现。代替许多条件句，将相关的条件分支移到相应的Strategy类中。


策略模式
优点
消除大的条件分支语句
为相同的行为提供不同的实现
缺点
增加对象的数量
所有的算法必须使用相同的Strategy接口