软件体系结构

Zhenyan Ji

Beijing Jiaotong University

结构型设计模式

享元模式

享元模式

目的:

01 通过共享有效地支持大量细粒度对象。

动机:

01 享元模式描述了如何共享对象, 以允许它们以精细的粒度使用, 而不会造成高昂的成本。

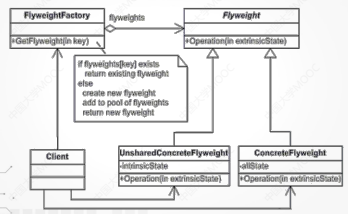
享元模式

每个享元对象分为两部分:

01 状态独立部分(intrinsic).内部状态在Flyweight对象中存储（共享）。

02 状态依赖部分(extrinsic).外部状态由客户端对象存储或计算，并在调用其操作时传递给Flyweight。

享元模式



享元模式

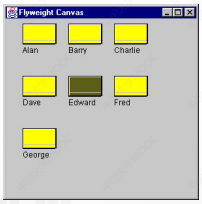
- **Flyweight**
  - 声明一个接口, 通过该接口, 享元可以接收并对外部状态进行操作。
- **ConcreteFlyweight**
  - 实现Flyweight接口, 为内部状态添加存储。ConcreteFlyweight对象必须是共享的。它存储的任何状态都必须是内在的, 也即, 独立于Concreteflyweight对象的上下文。

享元模式

- **UnsharedConcreteFlyweight**

并非所有的 Flyweight子类都需要共享。Flyweight接口支持共享，但它不会强制执行它。对于UnsharedConcreteFlyweight对象，将ConcreteFlyweight对象作为享元对象结构中某个层次的子对象很正常。

Flyweight Pattern



Flyweight Pattern

```
class FolderFactory{
  Folder unSelected, Selected;
  public FolderFactory()
    {
    Color brown = new Color(0x5f5f1c);
    Selected =  new Folder(brown);
    unSelected = new Folder(Color.yellow);
    }
//-------------------------------
  public Folder getFolder(boolean isSelected)
    {
    if (isSelected)
      return Selected;
    else
      return unSelected;
    }
  }
```

- FlyweightFactory用于分发所需的特定flyweight对象。当请求具有特定属性的Flyweight时，它会检查Flyweight对象是否已存在，如果存在，则返回该flyweight。如果请求的flyweight不存在，它将创建请求的flyweight，存储它，返回它。