软件体系结构

Zhenyan Ji

Beijing Jiaotong University

行为型模式--中介者模式

中介者模式

为什么需要中介者模式?

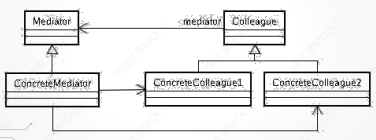
- 程序中出现越来越多的彼此分离的类，这些类之间的通讯变得越来越复杂。
- 程序变得越来越难以阅读和维护
- 中介者模式通过降低这些类之间的耦合度来解决这一问题。

中介者模式

目的

- 将对象间的交互封装在一个中介者对象中。中介者使各对象不需要显示地相互引用，从而使其耦合松散，而且可以实现独立改变它们之间的交互。

中介者模式



中介者模式

Mediator

- 定义Colleague类之间交互的接口。

Concrete Mediator

- 实现Colleague对象之间的交互协作。
- 维护其Colleague类。

中介者模式

Colleague

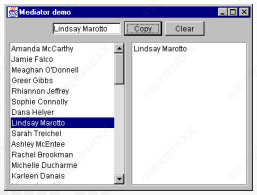
- 定义和Mediator类之间通讯的接口

ConcreteColleague

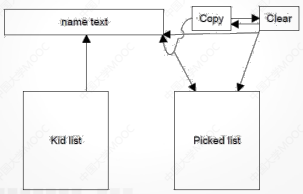
- 每个Colleague类都知道其中介者
- 每个Colleague类要和另一个Colleague类进行通讯时，都要通过中介者。

中介者模式例子:

中介者模式例子:

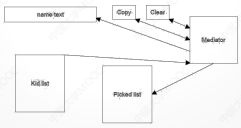


中介者模式例子:



中介者模式：示例

- 中介者模式简化了系统，因为中介者类是知道系统中其他类的唯一的类。



中介者模式：例子

- 首先创建一个中介者实例然后将中介者实例通过构造方法传给每个类。

```
Mediator med = new Mediator();
kidList = new KidList( med);
tx = new KTextField(med);
Move = new MoveButton(this, med);
Clear = new ClearButton(this, med);
med.init();
```

中介者模式

优点:

- 中介者降低了程序中类之间的耦合度。
- 可以通过改变中介者类或实现其子类来改变程序的行为。
- 中介者模式允许系统添加新的Colleague而无需改变程序的其他部分。
- 中介者模式解决了每个对象要充分了解其他通信对象才能进行通讯的问题

中介者模式总结:

- **便于理解**– 中介者封装了Colleague之间的交互逻辑。因为所有交互逻辑都封装在一个类中，便于理解。
- **解耦Colleague类 – Colleague**类之间完全解耦，添一个新的**Colleague**类变得更加容易。

中介者模式总结:

- **简化了对象协议 – colleague**对象只需和中介者对象交互，中介者模式有效地将多对多协议简化为一对多或多对一。
- **限制子类数量** – 所有的交互逻辑都封装在中介者类中。当交互逻辑需要扩展时，只需扩展中介者类。

中介者模式

缺点:

- 中介者类会变得越来越复杂，难以改变和维护。
- 很难在不同的项目中重用中介者类的代码
- 每个类都需要知道中介者的存在