软件体系结构

Zhenyan Ji

Beijing Jiaotong University

行为型模式--命令模式

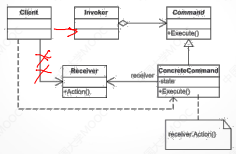
命令模式

目的

- 将服务请求封装成对象，从而可以不同的方式对请求进行操作。

它使client可以调用命令但不需要了解命令的执行细节。并且可以对命令进行修改而无需影响调用命令的client程序。

Structure



结构

Command

- 声明执行操作的接口

ConcreteCommand

- 定义Receiver对象和动作的绑定
- 通过调用Receiver的相应操作实现Execute（）方法

结构

Client

- 创建ConcreteCommand对象从而设置相应的receiver

Invoker

- 要求命令执行用户请求

Receiver

- 负责解析用户的请求并执行相应的操作

命令模式—示例

```
public interface Order {
   public abstract void execute ( );     }// Receiver class.
class StockTrade {
   public void buy() {  System.out.println("You want to buy stocks");   }
   public void sell() {   System.out.println("You want to sell stocks ");  }
}
```

命令模式—示例

```
class Agent {
   private m_ordersQueue = new ArrayList();
   public Agent() {    }
   void placeOrder(Order order) {
      ordersQueue.addLast(order);
      order.execute(ordersQueue.getFirstAndRemove());
   }
}
```

命令模式—示例

```
class BuyStockOrder implements Order {
   private StockTrade stock;
   public BuyStockOrder ( StockTrade st) {
      stock = st;
   }
   public void execute( ) {
      stock . buy( );
   }}.
```

命令模式—示例

```
class SellStockOrder implements Order {
      private StockTrade stock;
      public SellStockOrder ( StockTrade st)
{
            stock = st;
      }
      public void execute( ) {
            stock . sell( );
      }
}
```

命令模式—示例

```
public class Client {
    public static void main(String[] args) {
        StockTrade stock = new StockTrade();
        BuyStockOrder bsc = new BuyStockOrder
(stock);
        SellStockOrder ssc = new SellStockOrder
(stock);
        Agent agent = new Agent();

        agent.placeOrder(bsc); // Buy Shares
        agent.placeOrder(ssc); // Sell Shares
    }
}
```

命令模式总结

- 命令模式实现了调用操作的对象与具体实现操作的对象之间的解耦
- 命令可以像其他对象一样被操作和扩展
- 命令可以被组合为一个复合命令