# Description of Design Problem

* The ChangeGiver class is implementing giveChange method.
* The method does an explicit for loop to go over each domination of coins.
* There is a tight coupling between the requester and handler.
* It is not open for extensibility due to the tight coupling.

# Candidate Design Patterns Considered

* Chain of Responsibility

# Pattern Chosen

Chain of Responsibility design pattern allows loose coupling between the client, in this case ChangeGiver class, and the handlers processing the requests. Each domination of change will be handled by the respective class.

# Participants

  The classes and objects participating in this pattern are:

* **Handler**
  + “defines the interface for handling requests” (Erich Gamma, 1994)
  + ChangeGiverHandler.java
* **ConcreteHandler**
  + Implements the respective requests
  + Total of five concrete handler classes (1 dollar, 50, 20, 10, 5 cents)
  + CurrencyFiftyCentsChangeGiverHandler.java
  + CurrencyTwentyCentsChangeGiverHandler.java
  + CurrencyTenCentsChangeGiverHandler.java
  + CurrencyFiveCentsChangeGiverHandler.java
  + CurrencyOneDollaeChangeGiverHandler.java
* **Client** 
  + This will initiate the call to handler to start processing the request
  + ChangeGiver.java
  + The successor chains will be defined here

# Diagrams

## Class Diagram

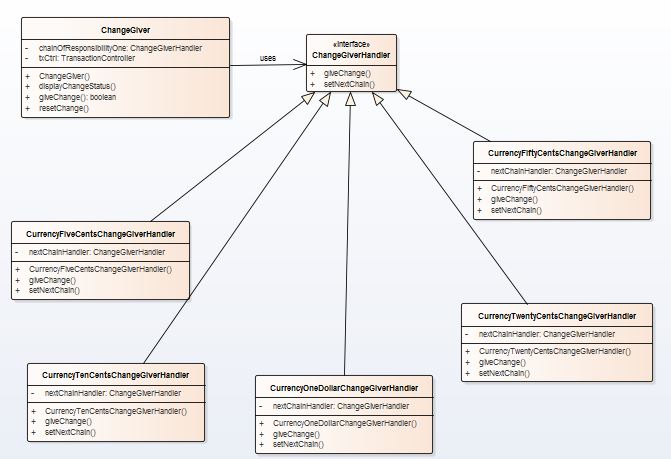


Figure 1: Class Diagram for Chain of Responsibility Design Pattern

## Sequence Diagram

Before



Figure 2: Sequence Diagram for old design

After

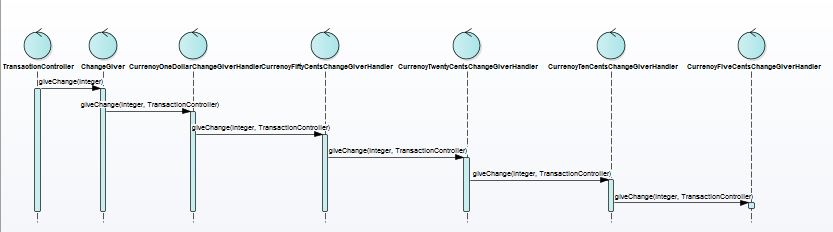


Figure 3: Sequence Diagram for new design (Chain of Responsibility Design Pattern)

# Implementation Issues

1. Implementing the successor chain
   1. The new links can be defined either in the Handler or the ConcreteHandler.
   2. For our implementation, the new links are defined in the ConcreteHandler itself.
   3. The following will list the sequence of calling the different handlers(based on dominations).
      1. One doller
      2. 50 cents
      3. 20 cents
      4. 10 cents
      5. 5 cents
2. Connecting Successors
   1. The handler does not maintain the successor for forwarding the requests.
   2. It is handled solely by the ConcreteHandlers.
3. Representing Requests
   1. Request parameter is taken as request code.
   2. The amount to be returned and the transaction controller are the parameters in this case.
   3. Since the change can be any amount, taking in a parameter will allow more flexibility on how the change is processed.
   4. But we are not using any Request class to bundle the request parameters.

# Reference List

Erich Gamma, 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1 Edition. Addison-Wesley Professional.