بررسی الگوهای طراحی–
Mediator

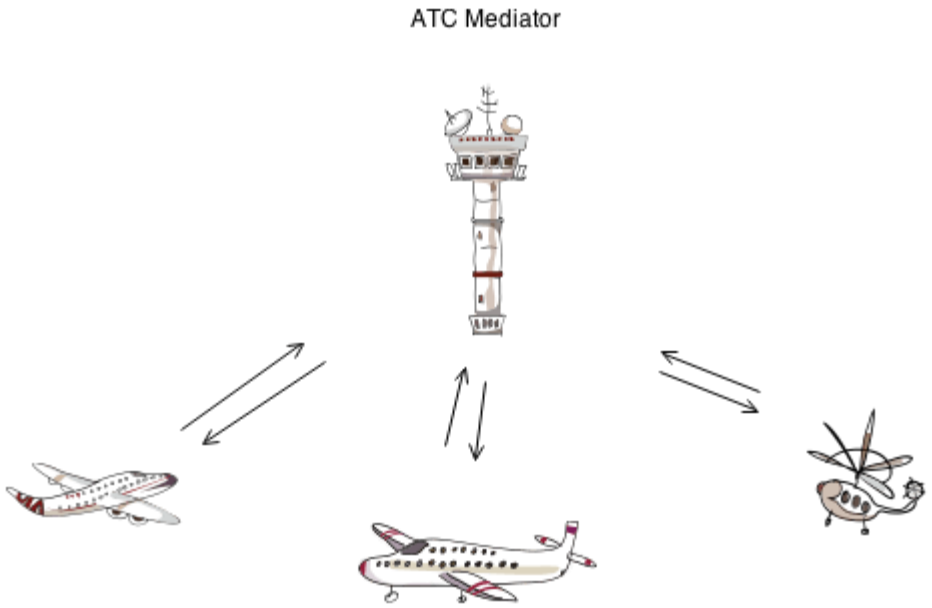
توسط : محمد حسین زارع
@zare88r: twitter
telegram
Join Us : @JavaLandCH

مفاهیم

- Loose Coupling
- استفاده مجدد از component ها
- بازی کردن نقش Hub / Router

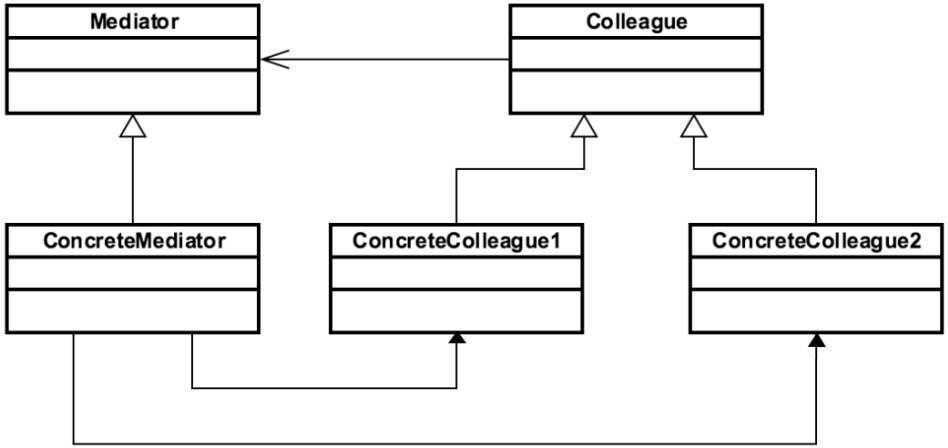
- نمونه ها
- java.util.Timer

مفاهيم



طراحی

- Colleagueها که طی یک سلسله مراتب تعریف میشوند
- Mediator که میتواند interface هم داشته باشد



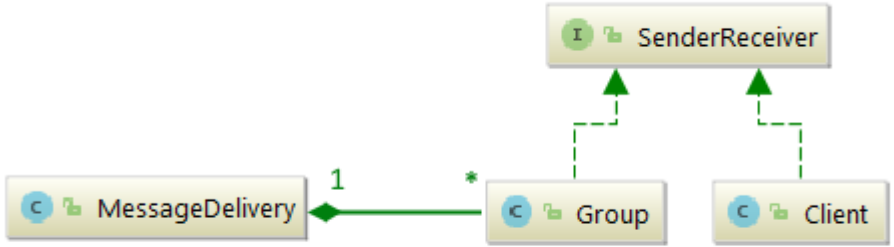
مثال 1

```
Timer timer = new Timer();

timer.schedule(new TimerTask() {
    @Override
    public void run() {
        System.out.println("TIME 1");
        }
}, TimeUnit.SECONDS.toMillis(10));

timer.schedule(new TimerTask() {
    @Override
    public void run() {
        System.out.println("TIME 2");
    }
}, TimeUnit.SECONDS.toMillis(15));
```

یک نمونه



یک نمونه

```
MessageDelivery messageDelivery = new MessageDelivery();

Client hoseinZare = new Client(messageDelivery, "@h.zare");
Client mohamadTaghizade = new Client(messageDelivery, "@m.taghizade");

hoseinZare.sendMessage("@m.taghizade", "Salam Chetori?");
mohamadTaghizade.sendMessage("@h.zare", "Mamnoun Khoobam.");

Group javaLandGroup = new Group(messageDelivery, "@javaLandGroup");
javaLandGroup.addClient(hoseinZare);
javaLandGroup.addClient(mohamadTaghizade);

javaLandGroup.sendMessage("#News Here there is some news " +
        "about java. Join Us @JavaLandCH");
```

یک نمونه

```
public Client(MessageDelivery messageDelivery, String name) {
    this.messageDelivery = messageDelivery;
    this.name = name;
    messageDelivery.registerClient(this);
}

@Override
public void receiveMessage(SenderReceiver sender, String message) {
    System.out.println("------------------------------");
    System.out.println(getName());
    System.out.println(String.format("<%s><%s> : %s",
            sender.getName(), new Date().toString(), message));
}

public void sendMessage(String clientName, String message) {
    messageDelivery.broadcast(this, clientName, message);
}
```

یک نمونه

```
public class Group implements SenderReceiver {

    private List<Client> clientList = new ArrayList<>();
    private MessageDelivery messageDelivery;
    private String name;
    public void sendMessage(String message) {
        clientList.forEach(client ->
                messageDelivery.broadcast(this, client.getName(), message));
    }
```

یک نمونه

```
//Mediator
public class MessageDelivery {
    private Map<String, SenderReceiver> senderReceiverMap = new HashMap<>();

    public void registerClient(SenderReceiver receiver) {
        senderReceiverMap.put(receiver.getName(), receiver);
    }

    public void broadcast(SenderReceiver sender, String clientName, String message) {
        if (senderReceiverMap.containsKey(clientName)) {
            SenderReceiver receiver = senderReceiverMap.get(clientName);
            receiver.receiveMessage(sender, message);
        }
    }
}
```

جمع بندی

- مسئول ارتباط بین گروهی از اشیا میباشد
- Loose coupling
- ارتباطات را ساده میکند
- در پیاده سازی میتوان از command نیز استفاده کرد
- ممکن است کلاس های mediator تبدیل به کلاس های پیچیده ای شوند