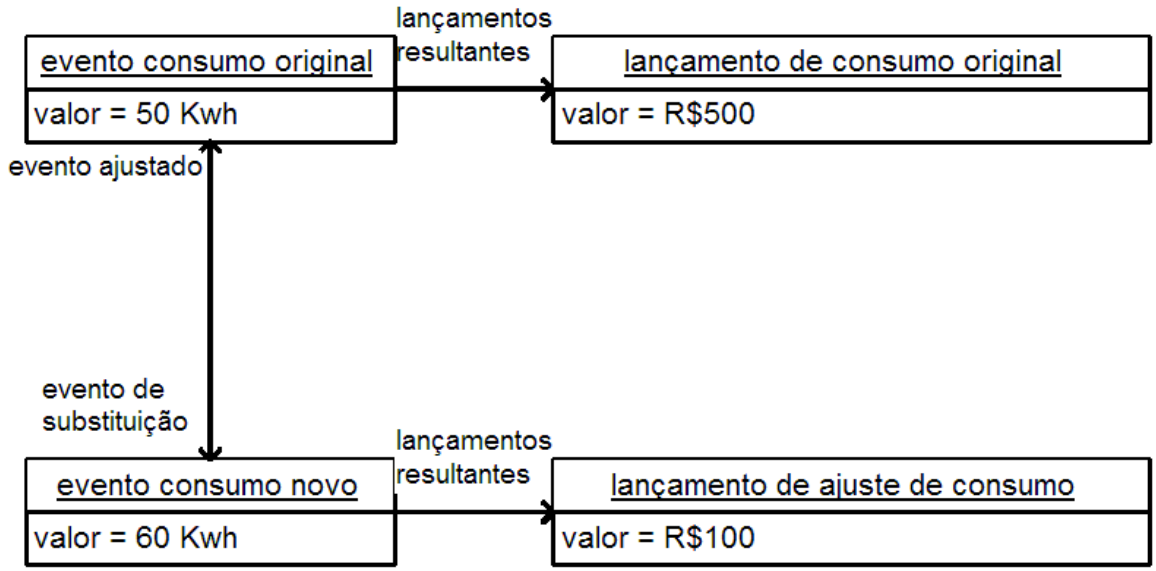
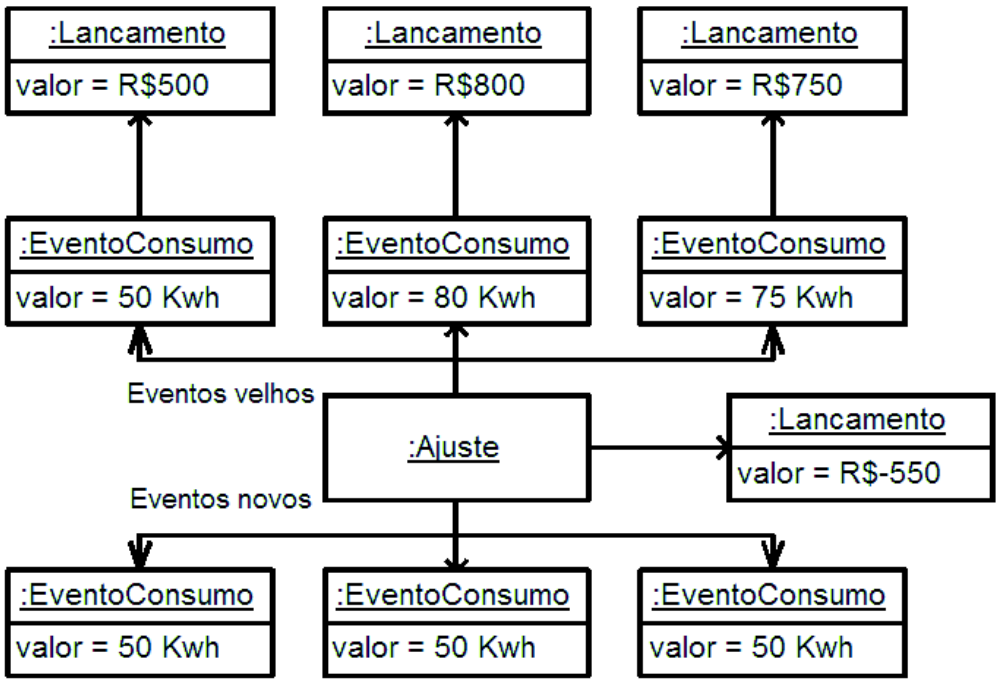
Analysis Pattern: Lançamento Diferencial (Difference Adjustment)

O que é

- Ajuste a um evento errado com lançamentos que refletem a diferença entre o que foi registrado e o que deveria ter sido registrado



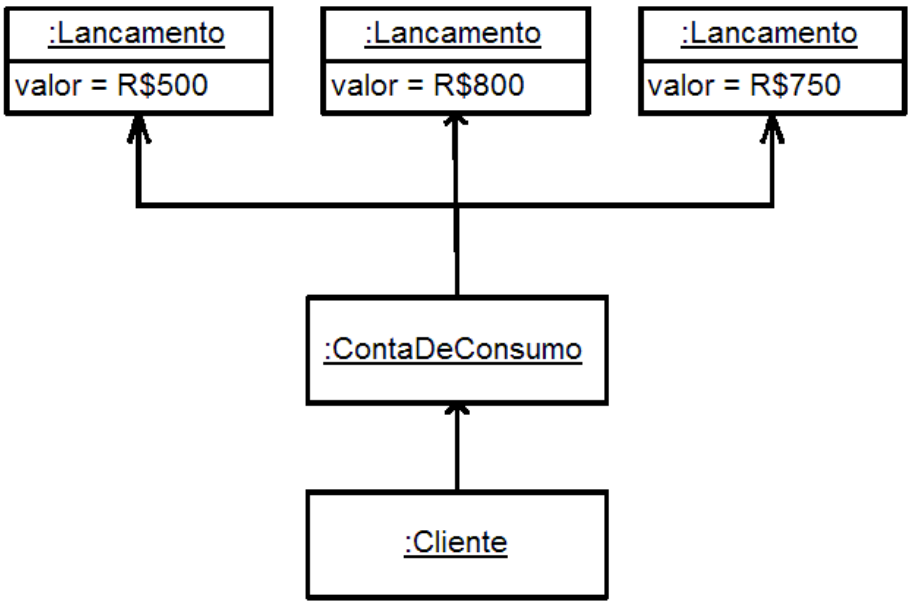
- Resulta em menos lançamentos do que usando Estorno
- Podem-se ajustar vários lançamentos errados com um único lançamento de diferença (veja abaixo)



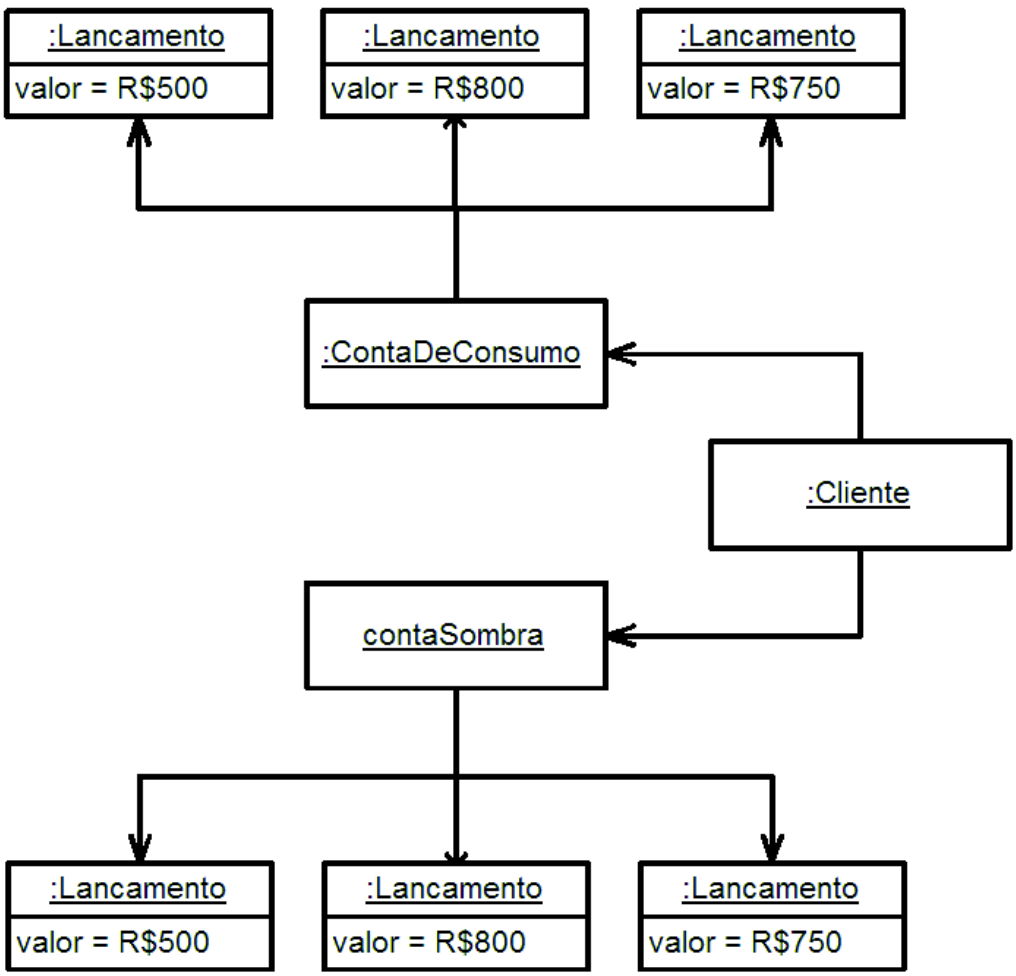
Detalhes de funcionamento

- A complicação surge ao calcular o valor da diferença
- Poderíamos colocar lógica dentro das regras de lançamento para calcular a diferença mas isso não fica muito claro
- Podemos usar uma forma semelhante ao que vimos no caso de Estorno

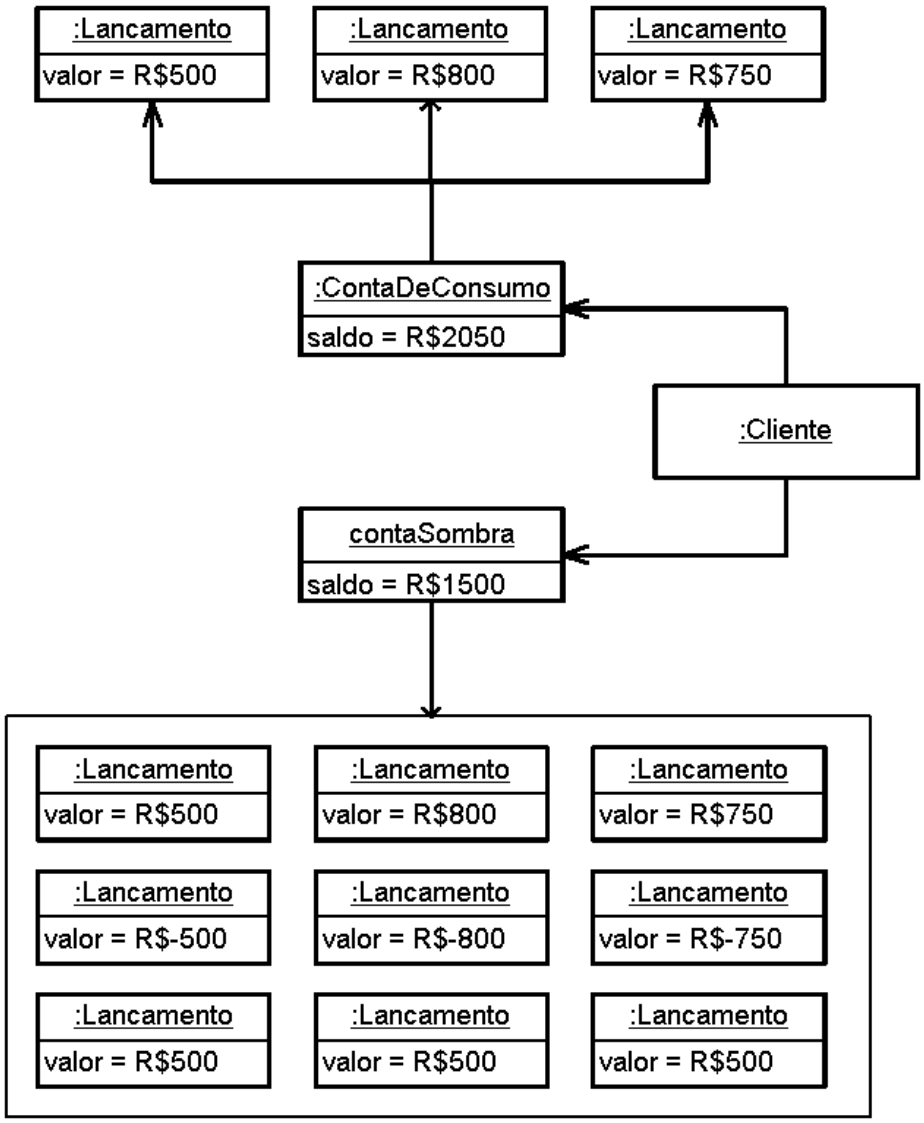
    - Faça reversão e substituição
    - Porém, acumule os lançamentos em *contas-sombra*
- Iniciamos com um cliente que tem uma conta de Consumo com vários lançamentos que sabemos estão errados



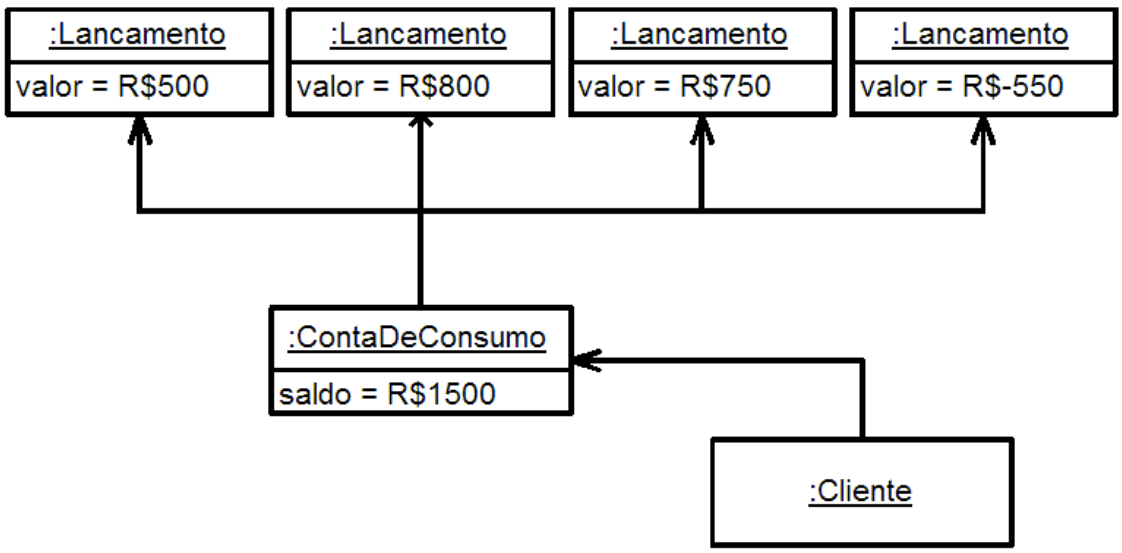
- Criamos um conjunto de contas-sombra
    - Inicialmente, xerocamos a conta Consumo



- Agora, processamos os eventos, revertendo lançamentos errados na contas-sombra



- Agora, calculamos a diferença entre os saldos da conta real e de sua Sombra e lançamos a diferença



- Resumo das etapas:
  - Criar as contas-sombra
  - Reverter os eventos antigos lançando nas contas-sombra
  - Processar os eventos novos lançando nas contas-sombra
  - Lançar nas contas originais a diferença entre as contas originais e sombra

- Para registrar o que está ocorrendo, e para fatorar a lógica disso tudo de forma limpa, é bom criar uma classe de ajuste que implemente as etapas acima
    - A classe de ajuste mantém referências aos eventos antigos e aos novos
    - O ajuste é tratado como evento e processado como qualquer evento

Quando deve ser usado

- Use este padrão se estiver usando o padrão Conta
- Se os Domain Experts querem ver o cancelamento explícito de lançamentos, use Estorno
- Se eles se satisfizerem com um sumário, use Lançamento Diferencial

Código exemplo

- Ao usar uma classe especial de Ajuste, a maior parte do código para tratar de ajuste se concentra nela
- A criação de um ajuste é mostrado abaixo

```
public class Ajuste extends EventoContabil {
        ...
        private List eventosNovos = new ArrayList();
        private List eventosVelhos = new ArrayList();
        public Ajuste(
                Calendar quandoOcorreu,
                Calendar quandoObservado,
                Subject subject) {
                super(null, quandoOcorreu, quandoObservado, subject);
        }
        public void addNovo(EventoContabil arg) {
                eventosNovos.add(arg);
        }
        public void addVelho(EventoContabil arg) {
                if (arg.jaFoiAjustado())
                        throw new IllegalArgumentException(
                                "Nao pode criar " + this +"." + arg + "ja esta ajustado");
                eventosVelhos.add(arg);
                arg.setEventoSubstituto(this);
        }
}
```

- Agora, veja como chamar isso (como sempre, mostramos um teste que exercita a lógica de negócio)

```
class Testador {
        ...
        //eventos originais
        eventoDeConsumo =
                new Consumo(
                        Unit.KWH.valor(50),
                        criaCalendar(2003, 10, 1),
                        criaCalendar(2003, 10, 15),
                        cam);
        listaEventos.add(eventoDeConsumo);
        Consumo consumo2 = new Consumo(...);
        listaEventos.add(consumo2);
        Consumo consumo3 = new Consumo(...);
        listaEventos.add(consumo3);
        listaEventos.processa();
        //eventos substitutos
        Calendar dataAjuste = criaCalendar(2004, 1, 12);
        Consumo novo1 = new Consumo(...);
        Consumo novo2 = new Consumo(...);
        Consumo novo3 = new Consumo(...);
        Ajuste ajuste = new Ajuste(dataAjuste, dataAjuste, cam);
        ajuste.addVelho(eventoDeConsumo);
        ajuste.addVelho(consumo2);
```

```
        ajuste.addVelho(consumo3);
        ajuste.addNovo(novo1);
        ajuste.addNovo(novo2);
        ajuste.addNovo(novo3);
        listaEventos.add(ajuste);
        listaEventos.processa();
}
```

- O comportamento de ajuste entra em ação ao processar o evento:

```
class Ajuste {
        ...
        private java.util.Map contasSalvas;
        public void processa() {
                assert(!foiProcessado); //Nao pode processar um evento duas vezes
                ajustar();
                marcaComoProcessado();
        }
        void ajustar() {
                xerocaContas();
                reverteEventosVelhos();
                processaSubstitutos();
                commit();
                eventosSecundarios = new ArrayList();
                eventosSecundarios.addAll(eventosVelhos);
        }
        public void xerocaContas() {
                contasSalvas = getCliente().getContas();
                getCliente().setContas(copyContas(contasSalvas));
        }
        void reverteEventosVelhos() {
                Iterator it = eventosVelhos.iterator();
                while (it.hasNext()) {
                        EventoContabil each = (EventoContabil) it.next();
                        each.reverse();
                }
        }
        void processaSubstitutos() {
                EventoContabil[] list =
                        (EventoContabil[]) eventosNovos.toArray(new EventoContabil[0]);
                for (int i = 0; i < list.length; i++) {
                        list[i].processa();
                }
        }
        public void commit() {
                TipoConta[] tipos == TipoConta.tipos();
                for (int i = 0; i < tipos.length; i++) {
                        ajustaConta(tipos[i]);
                }
                restauraConta();
        }
        public void ajustaConta(TipoConta tipo) {
                Conta contaCorrigida = getCliente().contaPara(tipo);
                Conta contaOriginal = (Conta) getContasSalvas().get(tipo);
                Money diferenca =
                        contaCorrigida.saldo().subtract(contaOriginal.saldo());
                Lancamento resultado = new Lancamento(diferenca, hoje());
                contaOriginal.addLancamento(resultado);
                lancamentoResultantes.add(resultado);
        }
        public void restauraConta() {
                getCliente().setContas(contasSalvas);
        }
}
```

programa