Padrão para atribuir responsabilidades: Alta Coesão

Problema

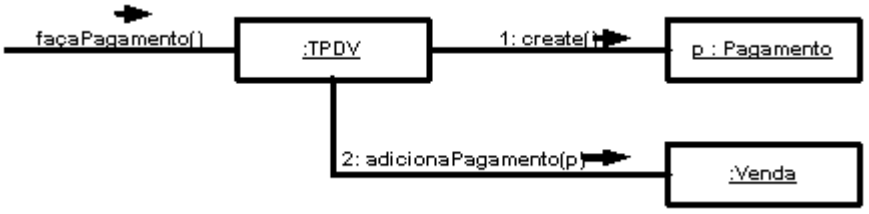
- Como gerenciar a complexidade?
- A coesão mede quão relacionadas ou focadas estão as responsabilidades da classe
  - Também chamada de "coesão funcional" (ver à frente)
- Uma classe com baixa coesão faz muitas coisas não relacionadas e leva aos seguintes problemas:
  - Difícil de entender
  - Difícil de reusar
  - Difícil de manter
  - "Delicada": constantemente sendo afetada por outras mudanças
- Uma classe com baixa coesão assumiu responsabilidades que pertencem a outras classes

Solução

- Atribuir responsabilidades que mantenham alta coesão

Exemplo

- Mesmo exemplo usado para Low Coupling
- Na primeira alternativa, TPDV assumiu uma responsabilidade de efetuar um pagamento (método façaPagamento())

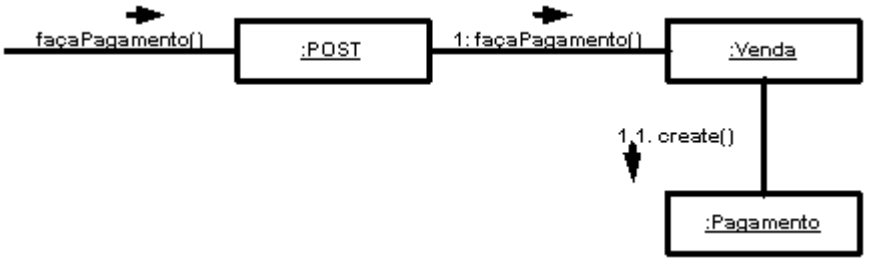


- Até agora, não há problema
- Mas suponha que o mesmo ocorra com várias outras

operações de sistema

- TPDV vai acumular um monte de métodos não muito focados
- Resultado: baixa coesão

- A segunda alternativa delega façaPagamento() para a classe Venda
  - Mantém maior coesão em TPDV



Discussão

- Alta coesão é outro princípio de ouro que deve ser sempre mantido em mente durante o projeto
- Tipos de coesão entre módulos
  - Coincidente (pior)
  - Lógico
  - Temporal
  - Procedural
  - De comunicação
  - Sequencial
  - Funcional (melhor)

Coesão coincidental

- Há nenhuma (ou pouca) relação construtiva entre os elementos de um módulo
- No linguajar OO:
  - Um objeto não representa nenhum conceito OO
  - Uma coleção de código comumente usado e herdado através de herança (provavelmente múltipla)

```
class Angu {
  public static int acharPadrão(String texto, String padrão)
    // ...
```

```
  }
  public static int média(Vector números) {
    // ...
  }
  public static outputStream abreArquivo(string nomeArquivo)
    // ...
  }
}

class Xpto extends Angu { // quer aproveitar código de Angu
  ...
}
```

Coesão lógica

- Um módulo faz um conjunto de funções relacionadas, uma das quais é escolhida através de um parâmetro ao chamar o módulo
- Semelhante a acoplamento de controle
- Cura: quebrar em métodos diferentes

```
public void faça(int flag) {
  switch(flag) {
  case ON:
    // coisas para tratar de ON
    break;
  case OFF:
    // coisas para tratar de OFF
    break;
  case FECHAR:
    // coisas para tratar de FECHAR
    break;
  case COR:
    // coisas para tratar de COR
    break;
  }
}
```

Coesão temporal

- Elementos estão agrupados no mesmo módulo porque são processados no mesmo intervalo de tempo
- Exemplos comuns:
  - Método de inicialização que provê valores defaults para um monte de coisas diferentes

- Método de finalização que limpa as coisas antes de terminar

```
procedure inicializaDados() {
  font = "times";
  windowSize = "200,400";
  xpto.nome = "desligado";
  xpto.tamanho = 12;
  xpto.localização = "/usr/local/lib/java";
}
```

- Cura: usar construtores e destrutores

```
class Xpto {
  public Xpto() {
    this.nome = "desligado";
    this.tamanho = 12;
    this.localização = "/usr/local/lib/java";
  }
}
```

- Outro exemplo: arquivo de configuração típico

```
[Macintosh]
EquationWindow=146,171,406,661
SpacingWindow=0,0,0,0

[Spacing]
LineSpacing=150%
MatrixRowSpacing=150%
MatrixColSpacing=100%
SuperscriptHeight=45%
SubscriptDepth=25%
LimHeight=25%
LimDepth=100%
LimLineSpacing=100%
NumerHeight=35%
DenomDepth=100%
FractBarOver=1pt
FractBarThick=0.5pt
SubFractBarThick=0.25pt
FenceOver=1pt
SpacingFactor=100%
MinGap=8%
RadicalGap=2pt
EmbellGap=1.5pt
```

```
PrimeHeight=45%

[General]
Zoom=200
CustomZoom=150
ShowAll=0
Version=2.01
OptimalPrinter=1
MinRect=0
ForceOpen=0
ToolbarDocked=1
ToolbarShown=1
ToolbarDockPos=1

[Fonts]
Text=Times
Function=Times
Variable=Times,I
LCGreek=Symbol,I
UCGreek=Symbol
Symbol=Symbol
Vector=Times,B
Number=Times

[Sizes]
Full=12pt
Script=7pt
ScriptScript=5pt
Symbol=18pt
SubSymbol=12pt
```

Coesão procedural

- Associa elementos de acordo com seus relacionamentos procedurais ou algorítmicos
- Um módulo procedural depende muito da aplicação sendo tratada
  - Junto com a aplicação, o módulo parece razoável
  - Sem este contexto, o módulo parece estranho e muito difícil de entender
    - "O que está acontecendo aqui!!!????!!"
- Não pode entender o módulo sem entender o programa e as condições que existem quando o módulo é chamado

- Cura: reprojete o sistema

Coesão de comunicação

- Todas as operações de um módulo operam no mesmo conjunto de dados e/ou produzem o mesmo tipo de dado de saída
- Cura: isole cada elemento num módulo separado
- "Não deveria" ocorrer em sistemas OO usando polimorfismo (classes diferentes para fazer tratamentos diferentes nos dados)

Coesão sequencial

- A saída de um elemento de um módulo serve de entrada para o próximo elemento
- Cura: decompor em módulos menores

Coesão funcional (a melhor)

- Um módulo tem coesão funcional se as operações do módulo puderem ser descritas numa única frase de forma coerente
- Num sistema OO:
  - Cada operação na interface pública do objeto deve ser funcionalmente coesa
  - Cada objeto deve representar um único conceito coeso
- Exemplo: um objeto que esconde algum conceito ou estrutura de dados ou recurso e onde todos os métodos são relacionados por um conceito ou estrutura de dados ou recurso
  - Meyer chama isso de "information-strength module"

Consequências

- Melhor claridade e facilidade de compreensão do projeto
- Simplificação da manutenção
- Frequentemente vai mão na mão com acoplamento fraco

- Com granularidade baixa e funcionalidade bem focada, aumenta o reuso

Exercício para casa

- Discuta a coesão (ou falta de coesão) do método extrato deste exemplo

programa