Projeto Detalhado de Software

GRASP
Polimorfismo, Invenção Pura

Camilo Almendra | camilo@es.ufc.br | https://sites.google.com/a/es.ufc.br/camilo/

Informações

- Referências
  - **Utilizando UML e Padrões**. LARMAN, Craig. 3a Edição. Capítulo 25.
  - Artigo “Herança versus Composição”
    http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/pat/herancavscomposicao.htm
  - Artigo “Como não aprender orientação a objetos (herança)”
    http://blog.caelum.com.br/como-nao-aprender-orientacao-a-objetos-heranca/
  - “Indirection”, Wikipedia http://en.wikipedia.org/wiki/Indirection
  -

Polimorfismo - Motivação

- Como tratar variações condicionais com uma filosofia diferente de **if-the-else,** principalmente se relacionadas com checagem de tipo?
  - É possível em orientação a objetos evitar grandes blocos de tratamento de variação condicional?
- Como trabalhar ao máximo com **componentes intercambiáveis**?
  - É possível desenhar projetos para facilitar o “plug-and-play”?

GRASP

- GRASP (General Responsibility Assignment Software Patterns)
  - Padrões básicos para atribuir responsabilidades
  - Abordagem metódica para projeto OO
  - Princípios amplamente usados
- 09 Padrões

Criador	**Especialista na Informação**	**Acoplamento**
Controlador	**Coesão**	**Polimorfismo**
Indireção	**Invenção Pura**	**Variações Protegidas**

GRASP – Polimorfismo

- **Problema 1**: Como tratar alternativas baseados no tipo?
- **Problema 2**: Como criar componentes de software interconectáveis?
- **Solução**: Atribuir a responsabilidade aos tipos usando operações polimórficas
  - Herança direta ou herança múltipla (interfaces)

GRASP – Polimorfismo

- **Alternativas baseados no tipo**
  - Exemplo de projeto **RUIM**

```
public Class ObjetoGrafico {
   public void desenhar(){
      switch(this.tipoObjeto){
         case QUADRADO: //faz algo
         case CIRCULO: //faz outra coisa
         case TRIANGULO: //...
      }
   }
}
```

GRASP – Polimorfismo

- Alternativas baseados no tipo
  - Exemplo de projeto **BOM**

```
public abstract Class ObjetoGrafico {
   public abstract void desenhar();
}

public class Quadrado extends ObjetoGrafico
{
   public void desenhar(){
      // faz algo
   }
}
```

GRASP – Polimorfismo

- **Críticas ao uso de Herança**
  - Forte acoplamento e fraco encapsulamento entre superclasses e subclasses
    - Exatamente o contrário do que se busca em OO
    - Mudança em superclasse pode afetar todas as subclasses
  - Inflexibilidade na mudança de tipo
    - Objeto de subclasse não pode mudar de tipo em tempo de execução
    - Herança é um relacionamento estático ou invariável

GRASP – Polimorfismo

- **Recomendações no uso de Herança**
  - Objeto da subclasse "**é um tipo especial de**" e não "um papel assumido por"
  - Objeto da subclasse **nunca tem que mudar** para outra subclasse ou hierarquia
  - Subclasse estende a superclasse mas **não faz override ou anulação** de variáveis e/ou métodos
  - Não é uma subclasse de uma classe "**utilitária**"
  - Para classes do domínio do problema, a subclasse **expressa tipos especiais** de papeis, transações ou dispositivos/recursos

GRASP – Polimorfismo

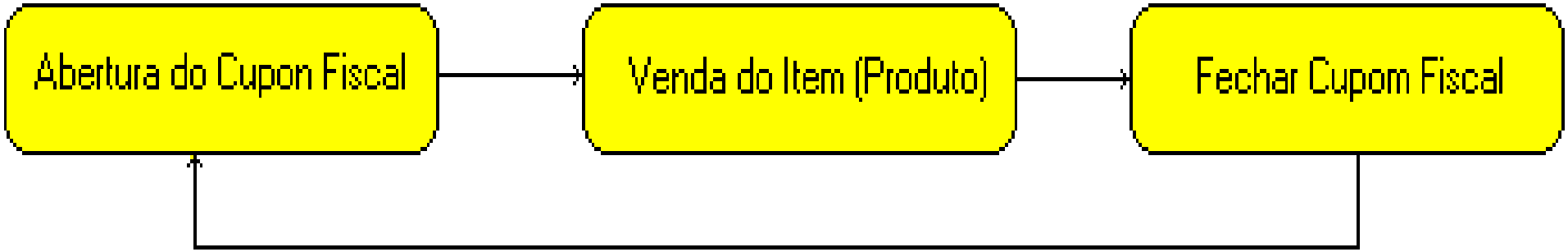
- **Projetando com Interfaces**
  - Use interface para apoiar o Polimorfismo sem se comprometer com uma hierarquia de classes
  - Lembre-se: superclasses carregam implementações, interfaces não
- **A implements B**
  - A se comporta como B
  - A não herda nenhum comportamento prévio de B
- **A extends B**
  - A é um tipo especial de B
  - A herda o comportamento e dados privados (invariáveis) de B

GRASP – Polimorfismo

- **Evitando checagem de tipos**
  - Ao invés de realizar checagem de tipos no meio da lógica de negócios, os tipos são configurados na criação e montagem dos objetos
  - Exemplo do Banco Imobiliário (LARMAN, Cap 25)
    - O objeto tabueiro é configurado com as instâncias corretas dos objetos que implementa a interface "Casa"
    - Os demais objetos atuam sobre as "Casas" sem saber o tipo específico, usando métodos polimórficos

GRASP – Polimorfismo

- **Projetando componentes interconectáveis**
- Exemplo: conexão do PDV com Impressoras Fiscais
  - Existem dezenas de fornecedores de ECF (Emissor de Cupom Fiscal)
  - A legislação é única, o funcionamento é idêntico



  - Protocolos DIFERENTES

Motivação

- Sistemas de software são muito complexos para terem suas responsabilidades distribuídas **apenas nas classes de domínio**
- Após distribuir responsabilidades ligadas a domínio (regras de negócios, funcionalidades), é necessário realizar a distribuir responsabilidades ligadas a: **Persistência, Tratamento de exceções e falhas, Comunicação, Distribuição, ...**
- Não repasse essas outras responsabilidades para classes de domínio. Use sua **Criatividade**! **Invente**!

GRASP

- GRASP (General Responsibility Assignment Software Patterns)
  - Padrões básicos para atribuir responsabilidades
  - Abordagem metódica para projeto OO
  - Princípios amplamente usados
- 09 Padrões

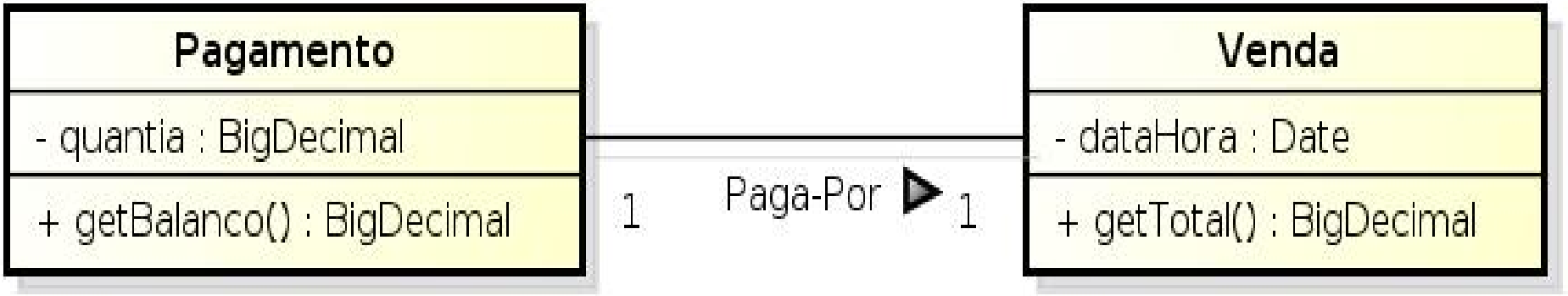
Criador	**Especialista na Informação**	**Acoplamento**
Controlador	**Coesão**	**Polimorfismo**
Indireção	**Invenção Pura**	**Variações Protegidas**

GRASP – Invenção Pura

- **Problema**: Como distribuir responsabilidades, no caso do *Especialista* impactar na Coesão Alta ou Acoplamento Baixo?
  - Projeto OO buscam manter baixo hiato representacional entre software e conceitos do domínio real.
  - Mas não só classes de domínio devem ser projetadas
- **Solução**: Atribuir conjunto coeso de responsabilidades a classes artificiais, que não representam nenhum conceito do domínio real.
  - **Invenção** da sua imaginação!
  - Conjunto de responsabilidades muito coeso, **puro**!

GRASP – Invenção Pura

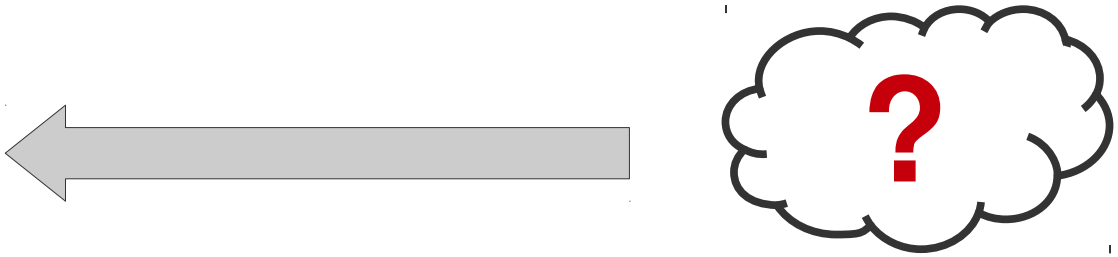
- Exemplo: Sistema de Vendas
- Classes de Domínio

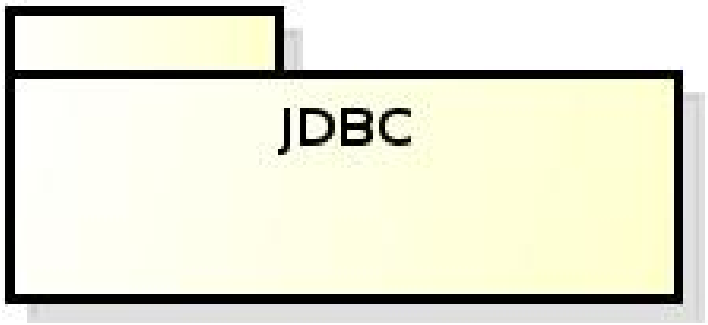


GRASP – Invenção Pura

- Exemplo: Sistema de Vendas
- Como fazer o ***armazenamento persistente*** das Vendas?

- Inserir
- Recuperar
- Atualizar
- Apagar





GRASP – Invenção Pura

Conceitos de Negócio

X

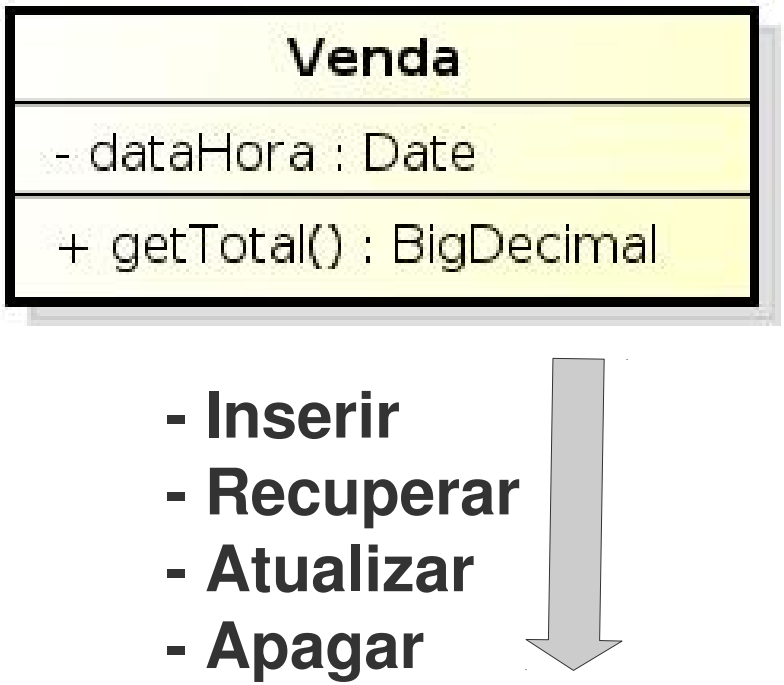
Conceitos de Computação

“Pagamentos”
“Vendas”
“Abrir Venda”
“Alterar Venda”
“Lançar Pagamento”

“Armazenamento”
“Persistência”
“Falha na Transação”
“Erro de Acesso”
“Cache de Dados”

GRASP – Invenção Pura

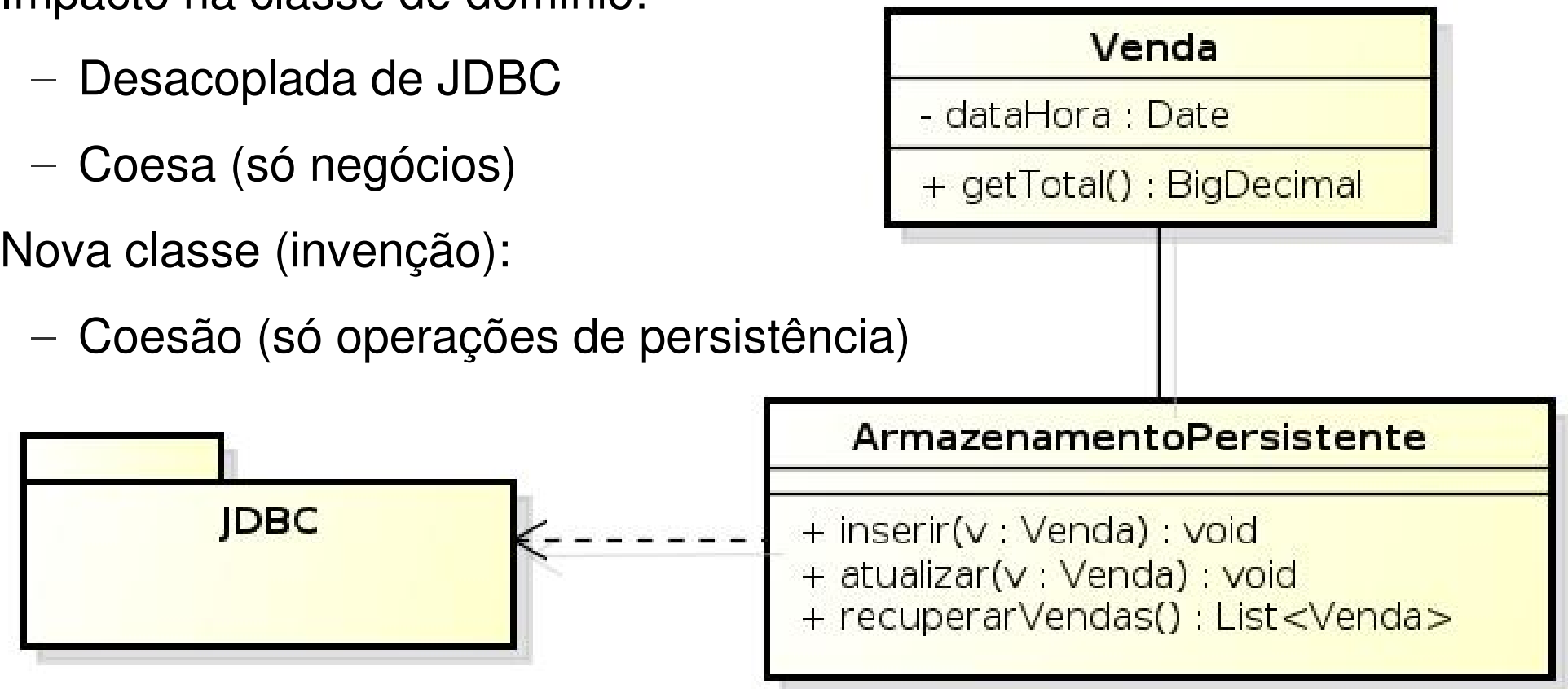
- Exemplo: Sistema de Vendas
- De acordo com o *Especialista*, seria a própria classe Venda.
- Impacto na classe de domínio:
  - Menor coesão
  - Acoplada ao pacote JDBC
- **RUIM**



GRASP – Invenção Pura

- Exemplo: Sistema de Vendas
- INVENTE!
- Impacto na classe de domínio:
  - Desacoplada de JDBC
  - Coesa (só negócios)
- Nova classe (invenção):
  - Coesão (só operações de persistência)

BOM



GRASP – Invenção Pura

- Decomposição **representacional**
  - Criar classes relacionadas com conceitos do domínio
  - Mantendo baixo hiato representacional
  - Ex.: Funcionário
- Decomposição **comportamental**
  - Agrupar comportamentos ou algoritmos
  - Não há necessidade de representar conceitos do domínio
  - Ex.: VerificadorCPF,