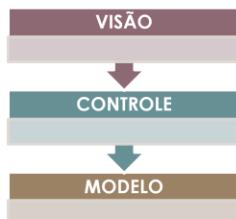
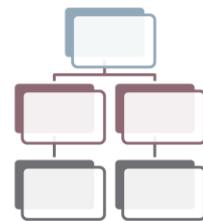


Arquitetura e Desenho de Software

AULA 10A



Profa. Milene Serrano



Agenda



Considerações Iniciais

Padrões GRASP

Considerações Finais

Considerações Iniciais



Considerações Iniciais

Projeto Guiado por Responsabilidades (ou PGR) é uma forma de criar desenhos de software com base em responsabilidades, papéis e colaborações.

Em um projeto de software, os objetos desempenham papéis.

- **Exemplo:** um objeto do tipo **PessoaFísica** pode desempenhar os papéis de **Gerente**, **Cliente** ou **Funcionário** em um projeto de software.

Responsabilidades estão relacionadas com as obrigações de um objeto em termos do papel que ele desempenha. São duas responsabilidades atribuídas a um objeto: fazer e conhecer.

Considerações Iniciais

Responsabilidades “**de fazer**” de um objeto incluem:

- Realizar algo propriamente dito, como por exemplo: executar um cálculo ou criar um objeto;
- Iniciar uma ação em outros objetos, e
- Controlar e coordenar atividades em outros objetos.

Considerações Iniciais

Responsabilidades “**de conhecer**”
de um objeto incluem:

- Conhecer sobre dados privados encapsulados;
- Conhecer objetos relacionados, e
- Conhecer sobre coisas que ele pode derivar ou calcular.

Considerações Iniciais

Atributos e associações apresentadas no modelo de domínio são boas fontes de responsabilidade do tipo "saber" em projetos de objetos, e diminuem o hiato representacional entre análise e projeto de software.

Responsabilidades podem ter **granularidades** definidas:

- Uma grande responsabilidade pode contemplar diversas classes, seus atributos e métodos (ex. responsabilidades atribuídas aos módulos de um software).
- Uma pequena responsabilidade pode contemplar apenas alguns métodos de uma classe (ex. métodos para criar e realizar algumas operações matemáticas).

Conceito: modelo de domínio <<debater sobre>>

Padrões de Projeto



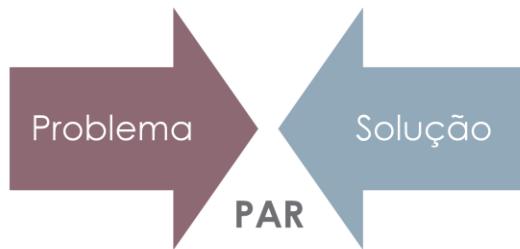
Padrões de Projeto

Padrões são princípios e soluções usados durante a criação do software, codificados em um formato estruturado, descrevendo o problema e a respectiva solução adotada.

Uma característica dos padrões é que podem ser aplicados a novos contextos, trazendo orientações sobre como podem ser utilizados e suas implicações (i.e. como deve ser implementado, análise custo-benefício, dentre outros).

Padrões de Projeto – Formato Típico

Nome do padrão	<i>Descreve sucintamente o propósito do padrão.</i>
Problema	<i>Explicação sobre a situação em que o padrão deve ser utilizado.</i>
Solução	<i>Detalhamento da solução, como implementá-la, suas variações e suas vantagens / desvantagens, etc...</i>



Padrões de Projeto GRASP

Os princípios e o raciocínio utilizado para atribuir responsabilidades (fazer e conhecer) de objetos podem ser descritos de modo metódico, explicável e repetível.

Esses princípios são conhecidos como Padrões GRASP.

São nove princípios:

- Criador - Especialista na Informação - Alta Coesão - Baixo Acoplamento – Controlador - Polimorfismo - Indireção - Fabricação ou Invenção Pura - Variações Protegidas

GRASP

Criador ou de Criação



GRASP – Criador ou de Criação

Problema: Quem deve ser responsável pela criação de uma nova instância de uma classe?

- É necessário ter um princípio geral para a atribuição da responsabilidade de criação de objetos. Caso seja bem atribuída essa responsabilidade, reduz-se o acoplamento e o projeto terá, provavelmente, mais clareza, encapsulamento e reutilização.

GRASP – Criador ou de Criação

Solução: Atribua à classe B a responsabilidade de criar uma instância da classe A se pelo menos uma das seguintes condições for verdadeira:

- B “contém” A ou é uma composição de A.
- B registra A.
- B usa A de maneira muito próxima.
- B tem dados iniciais de A, os quais serão passados para A quando este for criado. B é um “especialista” em relação à criação de A.

GRASP – Criador ou de Criação

Importante!

Caso mais de uma opção seja válida, prefira uma classe B que seja uma composição ou agregação de A.

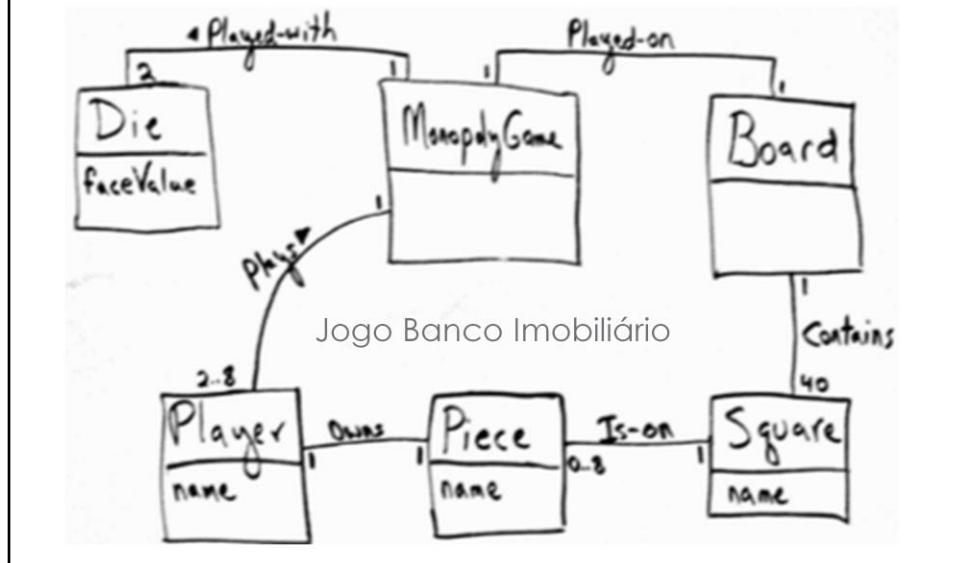
Vantagens:

Acoplamento baixo é apoiado, porque a classe criada provavelmente já é visível para a classe criadora em função das associações existentes entre elas.

Exemplo 01: Jogo imobiliário

Exemplo 02: Sistema de Ponto de Venda (PDV)

GRASP – Criador ou de Criação



GRASP – Criador ou de Criação

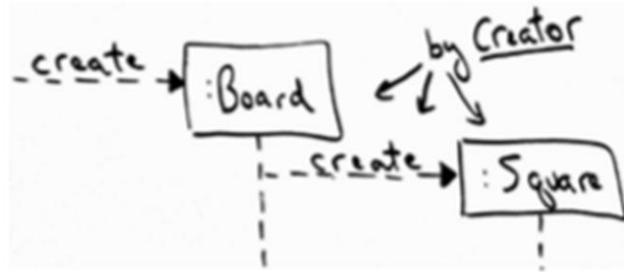
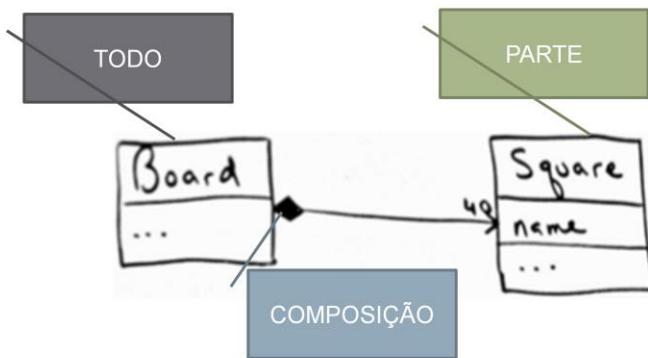


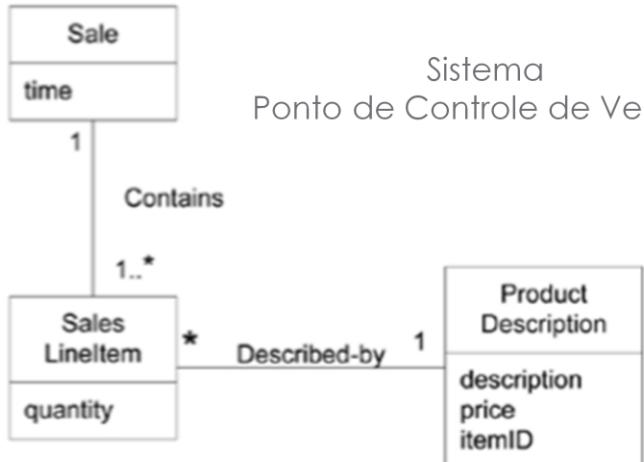
Diagrama de Sequência para a
criação de uma “quadra” (**Square**)
Responsabilidade de Tabuleiro (**Board**)

GRASP – Criador ou de Criação



Relacionamento entre **Board** e **Square**
TODO é responsável pela criação das PARTES

GRASP – Criador ou de Criação

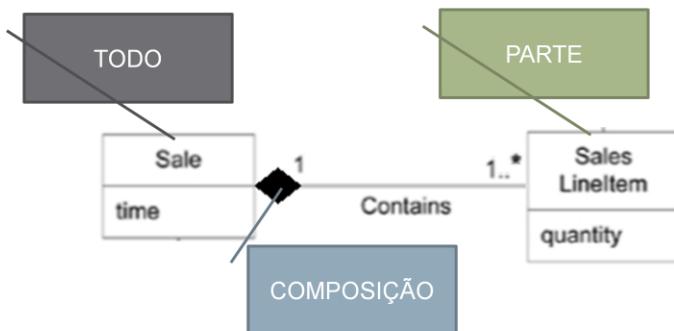


GRASP – Criador ou de Criação



Diagrama de Sequência para a
criação de um item de linha de venda (SalesLineItem)
Responsabilidade de Venda (Sale)

GRASP – Criador ou de Criação



Relacionamento entre **Sale** e **SalesLineItem**
TODO é responsável pela criação das **PARTES**

GRASP
Especialista



GRASP – Especialista

Problema: Qual é um princípio geral de atribuição de responsabilidade a objetos?

- Em outras palavras, se dividirmos bem as responsabilidades de fazer e conhecer entre as classes e objetos de um projeto de software, ele (o projeto) tende a ser mais fácil de entender, manter e estender, além de aumentar as oportunidades de reutilização.

GRASP – Especialista

Solução: Atribua responsabilidades ao especialista – a classe que tem a informação necessária para satisfazer a responsabilidade. Comece a atribuir responsabilidades enunciando claramente essas responsabilidades.

- Mas, o que é mais adequado? Procurar a informação nas classes do modelo de domínio ou no modelo de projeto?
 - **Resposta:** inicialmente, procure classes relevantes no modelo de projeto. Caso não encontre, utilize o modelo de domínio como inspiração para criar as classes de projeto correspondentes.

Modelo de Domínio e Modelo de Projeto <<debater sobre>>

GRASP – Especialista

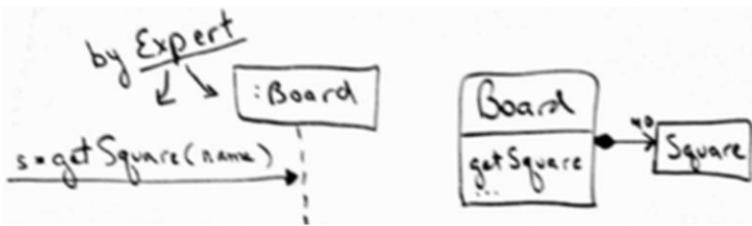
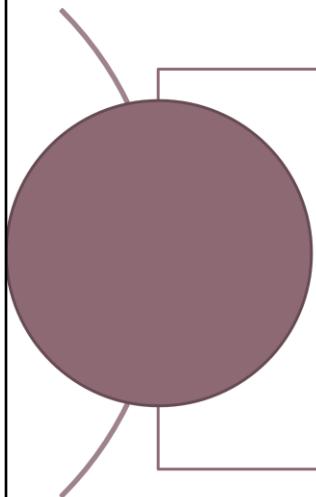


Diagrama de Sequência
Padrão Especialista no Jogo Banco Imobiliário

Tabuleiro (**Board**) contém informações suficientes sobre as “quadras” (**Square**).

Board é a classe que tem a informação necessária para satisfazer a responsabilidade.

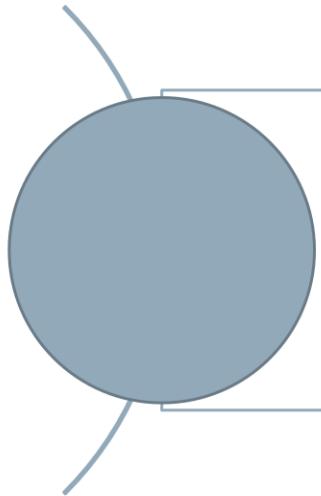
GRASP – Especialista



Importante!

- Para a utilização do padrão (ou princípio) especialista, basta usar o bom senso na hora de distribuir as responsabilidades, percebendo que os objetos fazem coisas relacionadas às informações que possuem.
- Satisfazer uma responsabilidade maior, geralmente, requer informações que estão espalhadas por diversas classes de objetos. Portanto, considere a existência de "especialistas" parciais, os quais compartilham informações por troca de mensagens.

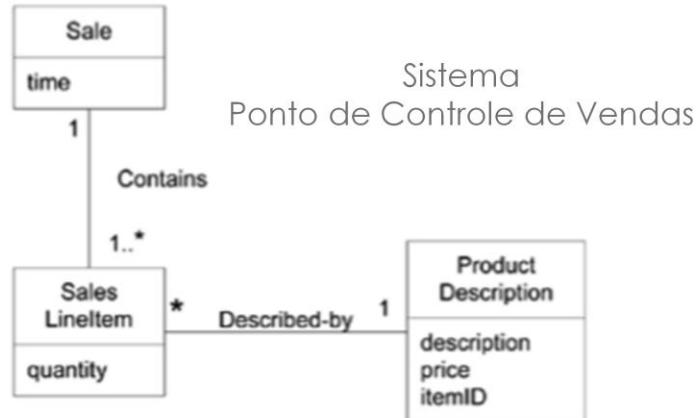
GRASP – Especialista



Vantagens:

- Encapsulamento da informação é mantido para os objetos.
- Acoplamento entre objetos tende a ser baixo – dentro do possível.
- Coesão dos objetos tende a ser alta: comportamento e informações são distribuídos entre as classes, levando a classes leves e mais fáceis de serem entendidas e mantidas.

GRASP – Criador ou de Criação

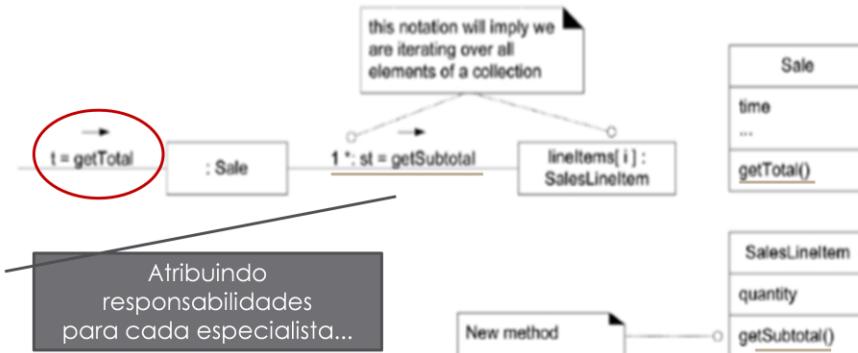


GRASP – Criador ou de Criação

Exemplo de Iteração



GRASP – Criador ou de Criação



Iterações envolvidas para calcular o total de venda

GRASP – Criador ou de Criação



Iterações envolvidas para calcular o total de venda
(Versão Refinada)

GRASP
Alta Coesão



GRASP – Alta Coesão

Problema: Como manter os objetos bem focados, com propósitos claros, gerenciáveis e com acoplamento baixo?

- Um elemento com responsabilidades altamente relacionadas e focalizadas, que não executa um grande volume de trabalho, tem coesão alta.

GRASP – Alta Coesão

Solução: Atribuir responsabilidades de forma que a coesão permaneça alta, sempre avaliando as alternativas.

GRASP – Alta Coesão

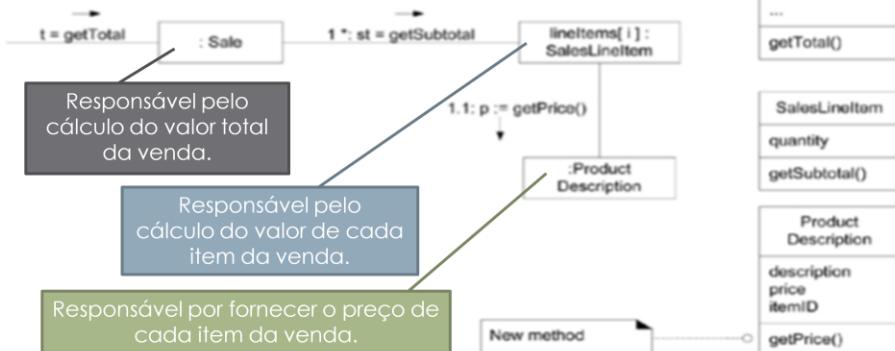
Importante!

Na prática, a coesão não deve ser analisada isoladamente de outros princípios e responsabilidades, sobretudo Especialista e Baixo Acoplamento. Um colabora com o outro para a definição de um bom projeto.

Vantagens:

Clareza, Maior compreensão sobre as classes,
Boa reutilização e Maior manutenibilidade.

GRASP – Alta Coesão



O fato de ter usado o Padrão Especialista e atribuído as responsabilidades aos respectivos especialistas, modularizando adequadamente o cálculo de uma venda, tenho que o resultado é um modelo de classes que valoriza a alta coesão.

GRASP – Alta Coesão

Grados de coesão segundo
Grady Booch, um dos
criadores da UML:

Coesão muito baixa: uma classe é responsável por muitas coisas em áreas funcionais muito diferentes.

PIOR

Coesão baixa: uma classe é responsável por uma tarefa complexa em uma área funcional.

Coesão moderada: uma classe tem responsabilidades moderadas em uma área funcional e colabora com outras classes para realizar tarefas.

Coesão alta: uma classe tem peso leve e responsabilidades exclusivas em algumas áreas logicamente relacionadas ao conceito da classe, mas não uma com as outras.

MELHOR

GRASP
Baixo Acoplamento



GRASP – Baixo Acoplamento

Problema: Como apoiar a baixa dependência, o baixo impacto de modificações e o aumento de reutilização?

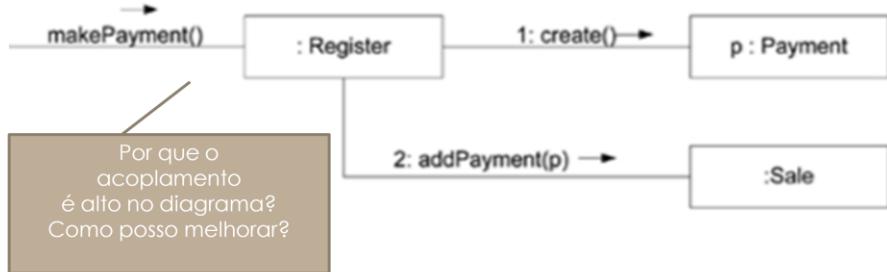
- Lembrando que modificações locais forçadas, decorrentes de modificações em classes relacionadas, são mais difíceis de entender isoladamente bem como de serem reutilizadas.

GRASP – Baixo Acoplamento

Solução: Atribuir responsabilidades de modo que o acoplamento permaneça baixo. Use este princípio para avaliar alternativas.

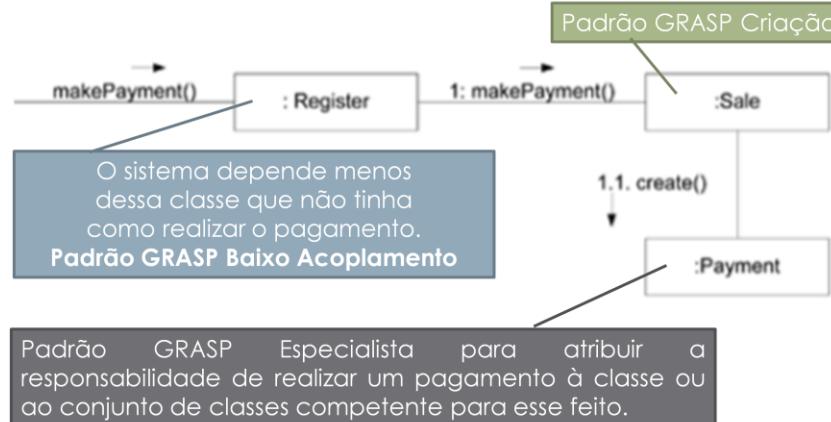
- O acoplamento não deve ser considerado isolado de outros princípios, sendo uma questão a ser considerada no aprimoramento de um projeto.

GRASP – Baixo Acoplamento



Criação de um pagamento,
com alto acoplamento.

GRASP – Baixo Acoplamento



Criação de um pagamento,
com acoplamento menor (mais baixo que o anterior).

GRASP – Baixo Acoplamento

Importante!

- Em linguagens orientadas a objetos, o acoplamento pode ocorrer quando:
 - Um atributo de uma classe X é uma referência a um objeto de outra classe Y.
 - Um objeto de uma classe X invoca métodos de um objeto de outra classe Y.
 - Um método de uma classe X referencia um objeto da classe Y ou a própria classe Y de alguma forma.
 - X é uma subclasse direta ou indireta de uma classe Y.
 - Y é uma interface e X implementa essa interface.

GRASP – Baixo Acoplamento

Importante!

- Subclasses são fortemente acopladas às suas superclasses.
- Portanto, cuidado na utilização de herança.
- O menor grau de acoplamento é quando não há acoplamento entre as classes.
- Isso fere o fato de objetos conectados poderem trocar mensagens entre si.
- Portanto, por mais que desejamos diminuir o acoplamento, um grau mínimo deverá existir.

GRASP – Baixo Acoplamento

Vantagens:

- Acoplamento baixo leva a classes independentes, que minimizam o impacto de modificações, não sendo afetadas, portanto, por mudanças em outros componentes do software.
- Simples de entender isoladamente.
- Conveniente para reutilização.

GRASP
Controlador



GRASP – Controlador

Problema: Qual é o primeiro objeto além da camada de IU que recebe e coordena uma operação do sistema?

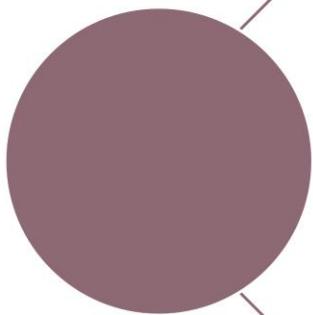
- Operações de sistema são capturadas nos Diagramas de Sequência do Sistema e são os principais eventos de entrada em um software. Um controlador é o primeiro objeto além da camada de Interface do Usuário.
- Ele é responsável por receber e tratar uma mensagem de operação do sistema.

GRASP – Controlador

Solução: Atribua a responsabilidade a uma classe que represente uma das seguintes alternativas:

- Represente o sistema global, um "objeto raiz", um dispositivo dentro do qual o software está sendo processado ou um subsistema importante (semelhante ao padrão FACHADA GoF).
- Represente um cenário do caso de uso dentro do qual ocorre o evento do sistema, geralmente denominado controlador do caso de uso. Neste caso, a classe controladora deve representar todos os eventos do sistema do mesmo cenário do caso de uso.

GRASP – Controlador



Importante!

- Classes de interface com usuário não devem executar tarefas associadas aos eventos de sistema. Apenas delegam os eventos de interfaces às classes controladoras de casos de uso.
- Cabe ao projetista escolher/definir se uma classe controladora representa um subsistema (físico inclusive) ou os eventos de um cenário de caso de uso.
- Durante o projeto, as operações do sistema, identificadas durante a análise, são atribuídas a uma ou mais classes controladoras.

Controladora de fachada...

Controladora por caso de uso...

Múltiplas controladoras...

<<Debater sobre>>

Operações do Sistema? O que é?

Imaginar o sistema como uma caixa preta.

As operações do sistema são aquelas operações de entrada do sistema, que representam sua interface pública.

Portanto, são operações através das quais interagimos com o sistema.

“associação com caixa eletrônico”

GRASP – Controlador

Importante!

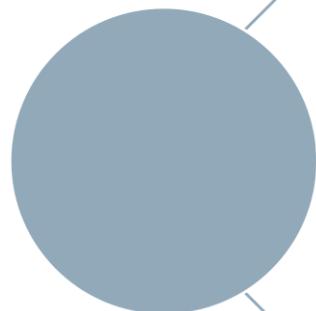
- Estas operações do sistema podem ser agrupadas em uma classe de modo a formar uma espécie de fachada da camada de domínio.
- Classes controladoras possuem estado. O estado de uma controladora pode representar a execução do caso de uso e as sequências de operações realizadas.
- Classes controladoras podem sofrer de baixa coesão, caso a ela sejam atribuídas muitas responsabilidades.

GRASP – Controlador

Diretriz

- Um controlador deve delegar a outros objetos o serviço a ser feito, apenas coordenando ou controlando, e não fazendo tudo por si próprio.
- Controladores fachada são indicados quando não existem muitos eventos de sistema ou quando não é possível que a interface envie mensagens de eventos de sistema a controladores alternativos.
- Controladores de casos de uso são indicados quando é desejável controlar a execução de cada caso de uso; quando um controlador fachada está sobrecarregado, havendo muitos eventos de sistema em diferentes processos.

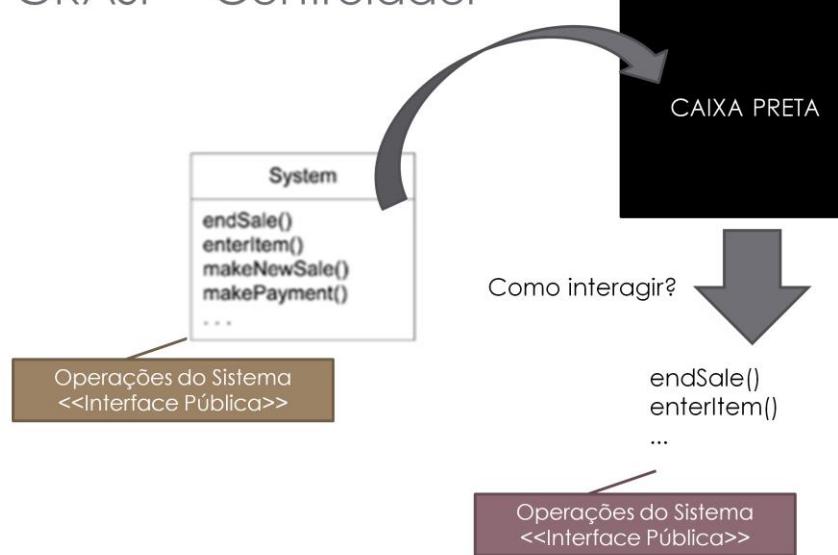
GRASP – Controlador



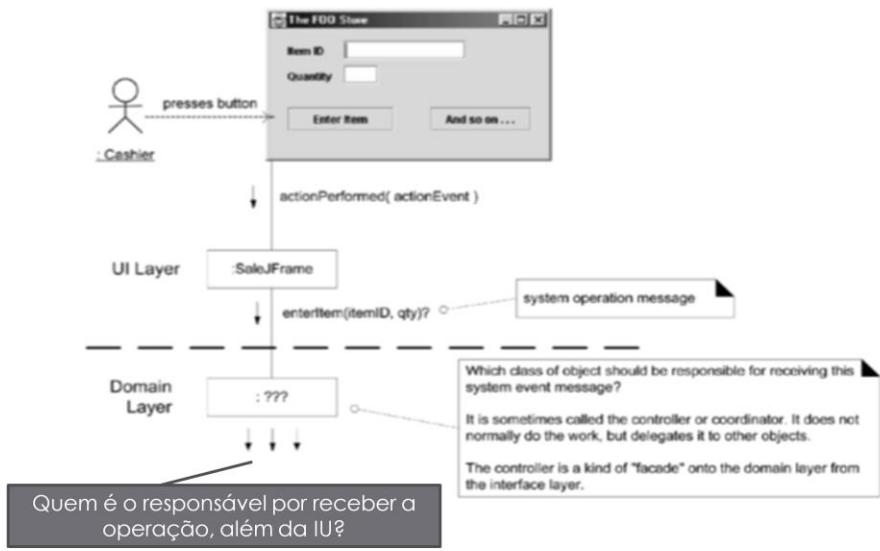
Vantagens:

- Aumento das possibilidades de reutilização e de interfaces "plugáveis" à medida em que a lógica da aplicação não é tratada na camada de interface.
- Ao delegar a responsabilidade de uma operação de sistema a um controlador, fica mais fácil reutilizar a lógica em futuras aplicações, e definir diferentes interfaces para ela.
- Parte da regra de negócio encontra-se nas classes controladoras, facilitando o raciocínio sobre o estado do caso de uso.

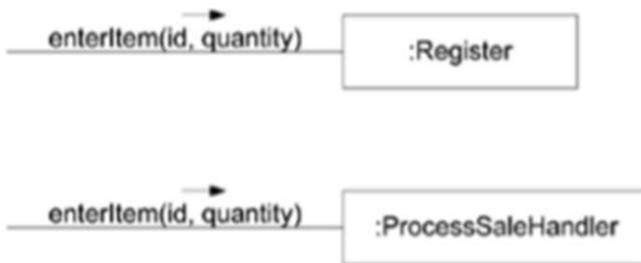
GRASP – Controlador



GRASP – Controlador



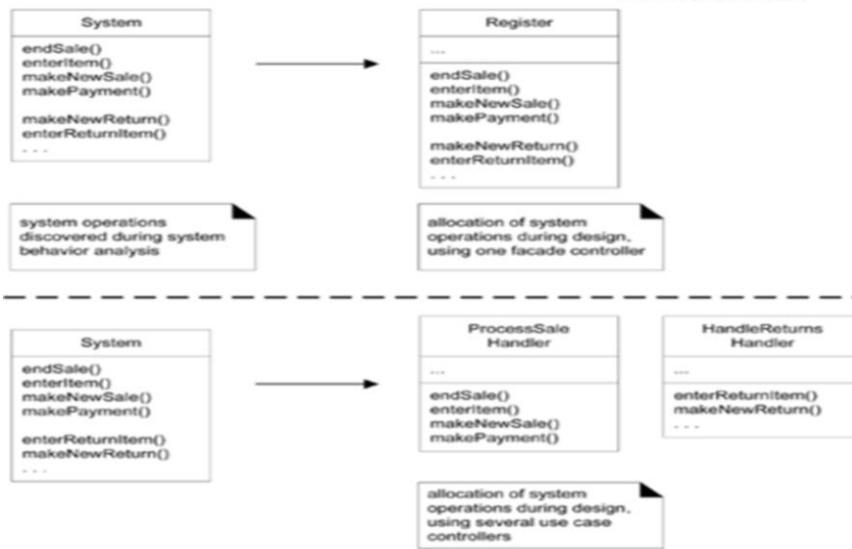
GRASP – Controlador



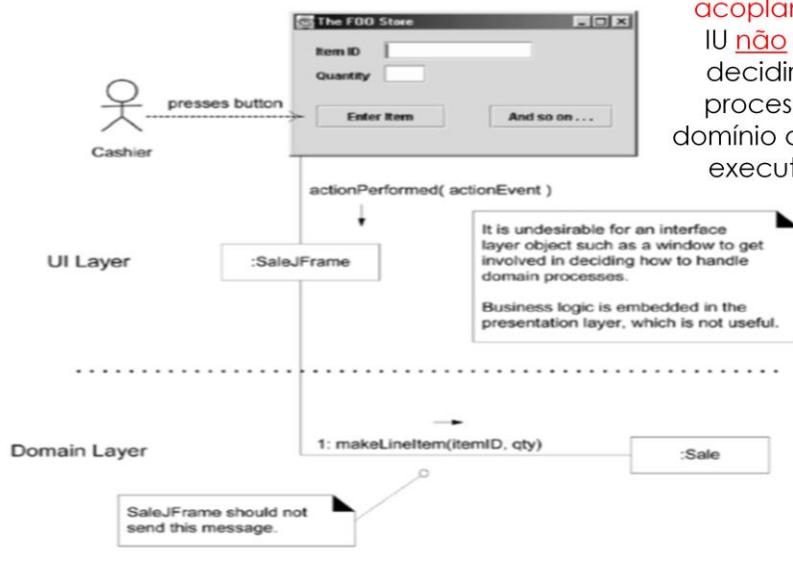
Alternativas para controladora:
fachada ou por caso de uso

GRASP – Controlador

Operações do sistema distribuídas pelas controladoras.

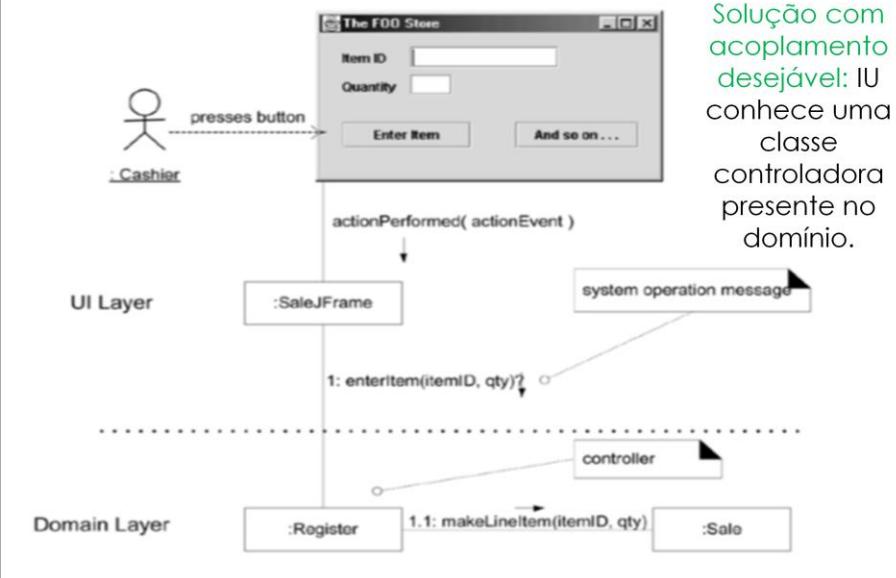


GRASP – Controlador



Solução com alto acoplamento
IU não deve decidir qual processo de domínio deve ser executado.

GRASP – Controlador



GRASP – Controlador

Controladores Sobrecarregados

- Uma única classe controladora recebendo todos os eventos de sistema (e há muitos deles). Passível de acontecer em controladores fachada.
- O próprio controlador executa várias tarefas para tratar um evento de sistema ao invés de delegá-las aos demais objetos.
- O controlador tem muitos atributos e mantém informações significativas sobre o sistema, as quais poderiam ser distribuídas pelos demais objetos.

Extra Classe



Extra Classe

Leitura:

Ler os capítulos 17 e 18 do Larman, especialmente a seção 17.8 (exemplo de projeto de objetos com padrão GRASP).

Vamos lá!:)

Considerações Finais



Considerações Finais

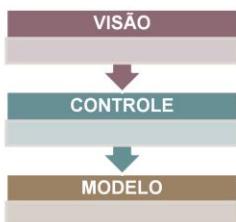
- Nessa aula, conhecemos uma pouco mais sobre boas práticas de projeto de software, mais especificamente introduzimos os padrões GRASP.
- Continuem os estudos! Só se aprende praticando!
- Nas referências, têm vários materiais complementares! :)

Referências



Referências

- LARMAN, Craig. Utilizando UML e Padrões: Uma Introdução a Análise e ao Projeto
- Orientado a Objetos. 3a. edição. Bookman, 2007.
- COCKBURN, Alistair. Escrevendo Casos de Uso Eficazes. Bookman, 2005.
- SILVA, Ricardo Pereira. UML 2 em Modelagem Orientada a Objetos. Visual Books, 2007.
- PRESSMAN, Roger S. Engenharia de Software. 6a. edição . McGraw-Hill, 2006.
- IEEE. SWEBOK-Guide to the Software Engineering Body of Knowledge, 2004.
- SOMMERVILLE, Ian. Engenharia de Software. 8a. edição. Pearson, 2007.



FIM

Dúvidas?

CONTATO:
mileneserrano@unb.br
ou
mileneserrano@gmail.com

Sugestões?

