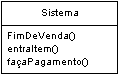
**Murilo Silva Andrade Souza**

**Tarefa do Módulo 03 (4,0 pontos)**

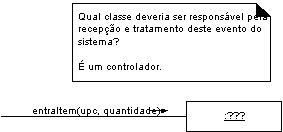
1) Escolha um padrão GRASP que não tenha sido detalhado em nossa aula 3 e descreva as características abaixo.

**Padrão GRASP Controller**

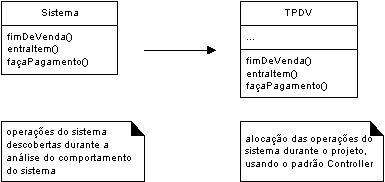
* **Problema**  
  Quem deveria receber a responsabilidade de tratar eventos do sistema?
* Um evento do sistema é um evento de alto nível gerado por um ator externo
* Estão associados a operações do sistema que já vimos nos Diagramas de Sequência do Sistema
* Exemplo do estudo de caso: Caixa pressiona "Fim de venda"
* **Solução**  
  Use um controlador
  + Um controlador é um objeto que não é de interface GUI responsável pelo tratamento de eventos do sistema
  + Um controlador define métodos para as operações do sistema
  + Atribuir a responsabilidade pelo tratamento de eventos do sistema a uma classe de acordo com uma das alternativas abaixo:
  + Representa o "sistema" como um todo (facade controller)
  + Representa o negócio ou organização como um todo (facade controller)
  + Representa algo no mundo real que é ativo (por exemplo, o papel de uma pessoa) que poderia estar envolvido na tarefa (role controller)
  + Representa um handler artificial de todos os eventos do sistema para um Use Case particular, normalmente chamado "<NomeDoUseCase>Handler" (use case controller)
* **Exemplo**No estudo de caso, há várias operações de sistema:



* Quem deveria ser o controlador para os eventos do sistema?

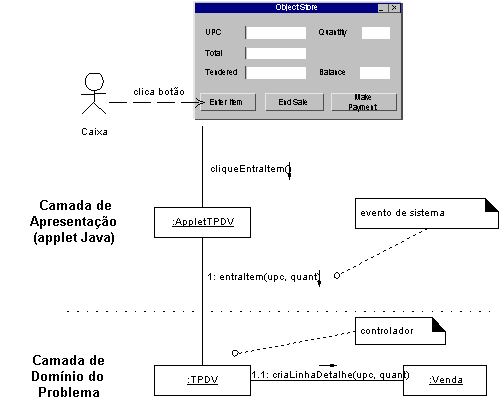


* Pelo padrão Controller, temos as seguintes alternativas:
  + Representa o "sistema": TPDV
  + Representa o negócio ou organização: Loja
  + Representa algo no mundo real ...: Caixa
  + Representa um handler artificial ...: CompraItemHandler
  + A escolha particular depende de fatores discutidos na seção Discussão
  + Por exemplo, se fosse TPDV, teríamos:



**Benefícios**

* De forma geral, o mesmo controlador deve ser usado para todas as operações de um mesmo Use Case de forma a manter a informação de estado do Use Case
  + A informação de estado pode ser útil para detectar sequências erradas de eventos de sistema
  + Exemplo: façaPagamento() antes de fimDeVenda()
  + Não coloque toda a inteligência no controlador
  + Delegue para outros objetos,para manter coesão
  + Use um Handler artificial quando as outras alternativas exibem acoplamento alto ou coesão baixa
  + Quando está surgindo um "God class"
  + Usado em sistemas mais complexos
  + Observe que classes "Window", "Applet", "Application", "View", "Document" não devem ser controladores
  + Tais classes podem receber o evento e delegá-lo ao controlador
  + Não se deve colocar business logic num objeto de interface com o usuário
  + Um design correto seria:



* + O Role Controller pode levar a um mau projeto
    - O fato de algo ser feito por uma pessoa no mundo real não necessariamente significa que isso é uma boa alternativa em software
    - É mais comum "dar vida aos objetos" (não animados)

**Contra Indicações.**

* Maior possibilidade de reuso, já que o business logic não está nos objetos de interface
  + Exemplo: embutir o business logic num objeto de interface não permitiria fazer EAI (Enterprise Application Integration)
* Ajuda a verificar o sequenciamento das operações do sistema, através do estado do controlador.