

Alessandro Almeida | [www.alessandroalmeida.com](http://www.alessandroalmeida.com)

# **Engenharia de Software II**

# **Atividade – Diagramas da UML**

# Importante

- Os próximos slides fazem parte de uma atividade realizada pelos alunos da turma SIN-NA6 (6º semestre de Sistemas de Informação – 2º semestre de 2012)

# ORIENTAÇÃO A OBJETOS

---

# INTEGRANTES

- Ana Paula de Sousa - 10101629
- Caroline Soares - 10100815
- Elias Nascimento - 10101626
- Guilherme Oliveira - 10101599
- Rafael V Cordeiro - 10103318

# HISTÓRICO DA ORIENTAÇÃO A OBJETOS

- **Evolução da Programação Estruturada**

A Programação Orientada a Objetos é uma evolução da Programação Estruturada

- **Linguagem Estruturada**

Funções e Dados podem ser acessados por qualquer função.

- **Linguagem Orientada à Objetos**

Funções agregadas aos dados em uma unidade chamada ‘objeto’.

# HISTÓRICO DA ORIENTAÇÃO A OBJETOS

- **Anos 60:**

A Linguagem de programação chamada de **SIMULA** (projetadas para apoiar a simulação de eventos discretos), já utilizavam o uso de conceitos de O.O.
- **Anos 70:**

Surge a primeira linguagem totalmente voltada a O.O., que é **SMALLTALK**, Incorporou ideias da linguagem de programação **SIMULA**.
- **Anos 80:**

Quase todas as linguagens de programação já apresentavam conceitos de O.O., como o **Delphi**, **Java**, **C++**, **PASCAL**, **LISP**, **C++**, **ADA**, **EIFFEL**, **OBJECT PASCAL** entre outras.
- **Anos 90:**

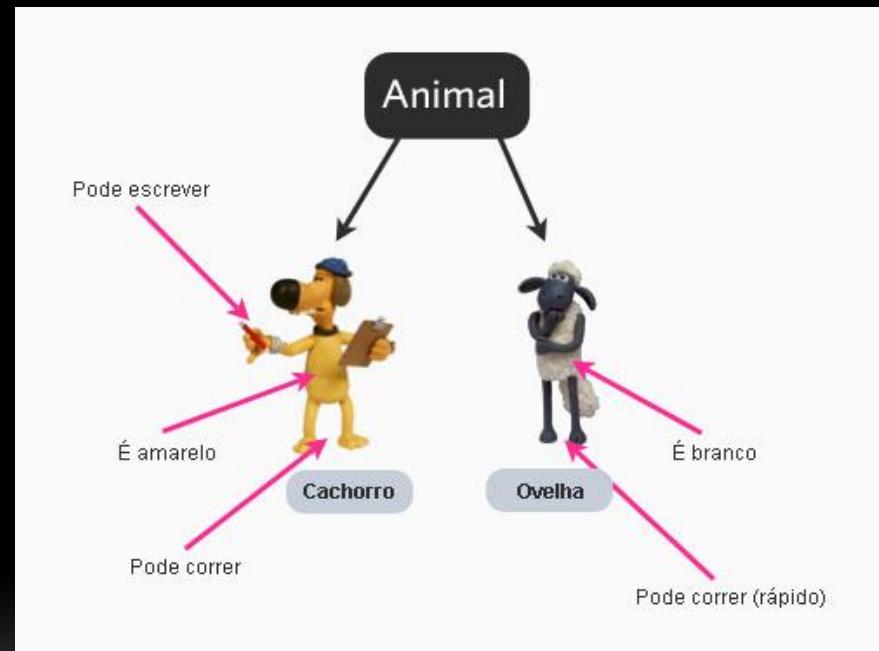
**JAVA** é ao mesmo tempo um ambiente e uma linguagem de programação O.O., e foi projetado para resolver os problemas da área de programação **cliente/servidor**.

# EVOLUÇÃO DA ORIENTAÇÃO A OBJETOS

- **Larry Constantine** – (1960) Foi quem primeiro lançou a ideia de que softwares poderiam ser projetados antes que fossem programados
- **Ole-Johan Dahl** e **Kristen Nygaard** - (1966) - Foi quem primeiro lançou a ideia de Classes introduzida na linguagem **SIMULA**
- **Alan Curtis Kay** - (1970) Iniciou o conceito de Mensagem e Herança, usados na linguagem **SmallTalk**, e **Adele Goldberg**.

# CONCEITOS DE ORIENTAÇÃO A OBJETOS

- O que é a Orientação a Objetos?
- E o que colaboram para construir?
- O que é um paradigma?
- O paradigma pode auxiliar em que?



# VANTAGENS DA TECNOLOGIA DE OBJETOS

## Código:

- Eliminação de redundância
- Fácil manutenção
- Reaproveitamento de código

Se houver mudanças nos requisitos?

## Pensamento...



Eu vou subir e ver o que eles querem e o resto de vocês comecem a codificar

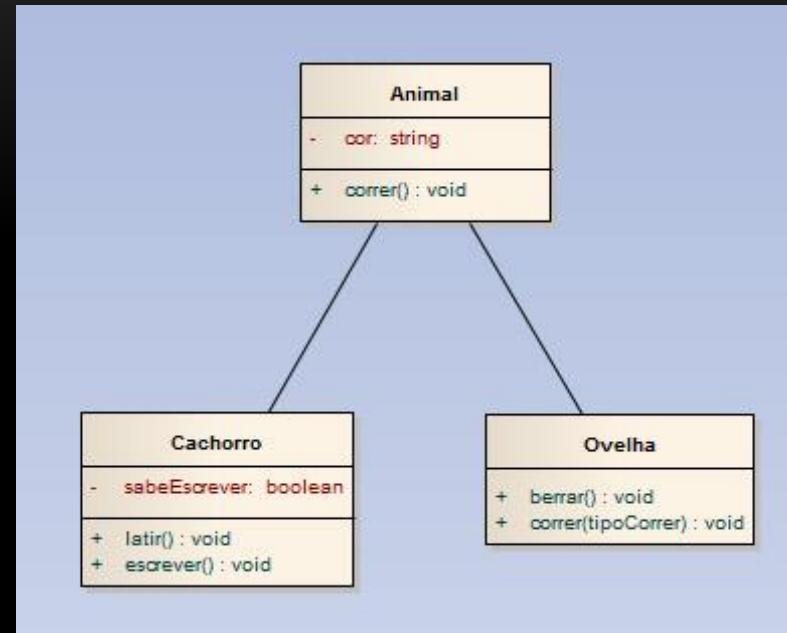
# CLASSE

- O que é Classe?

- Atributos e Métodos.

Atributos = características

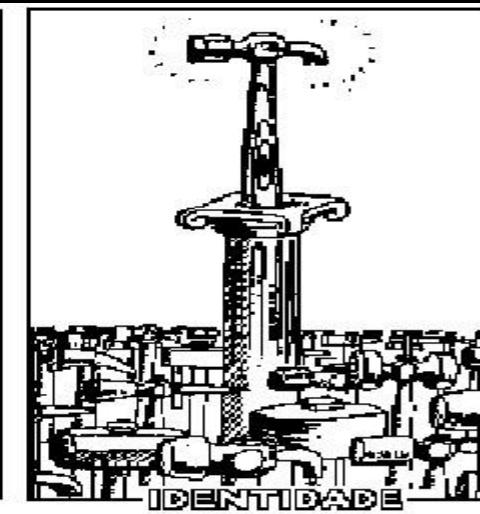
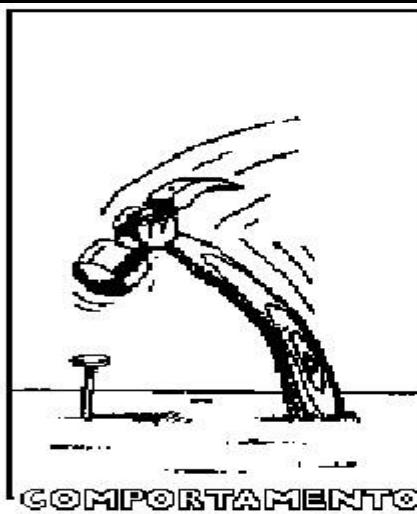
Métodos = ações



Operações caracterizam o comportamento de um objeto, e são o único meio de acessar, manipular e modificar os atributos de um objeto.

# OBJETOS

- Um objeto possui um estado, exibe um comportamento bem-definido e possui uma identidade única.



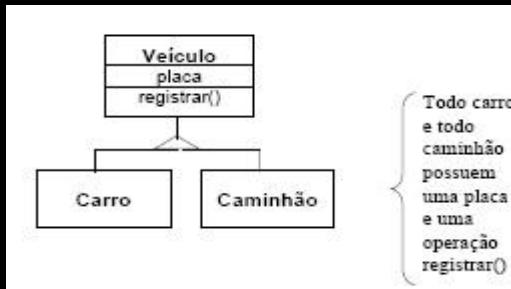
Conjunto de Propriedades

Reação (M ;P)

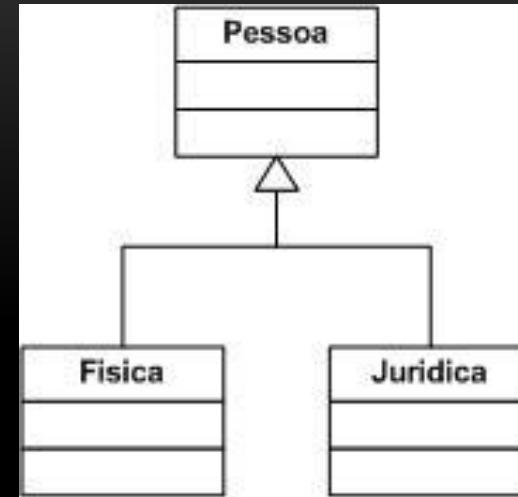
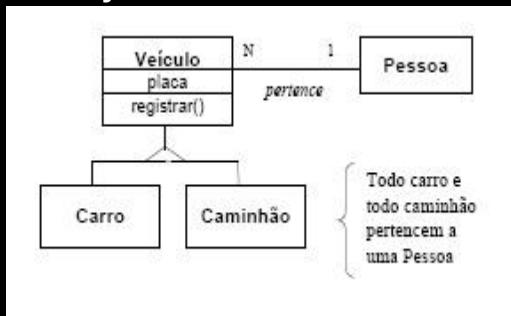
Propriedade do Objeto

# HERANÇA

- O que nos permite a herança?
- Também conhecida como?
- Herança de Atributos e de Métodos



- Herança de Relacionamentos



Através da herança é possível representar a relação de generalização/especialização entre duas classes:  
" a superclasse é uma generalização da(s) subclasse(s), e  
" a subclasse é uma especialização da(s) superclasse(s).

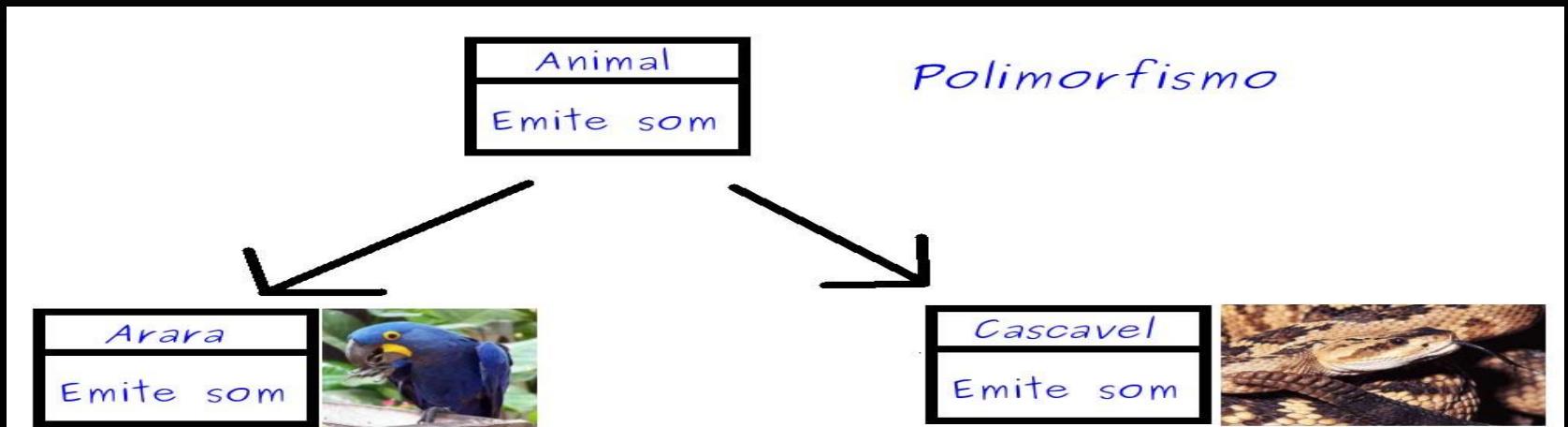
Temos os tipos de herança:

Herança estrita: as subclasses podem redefinir ou excluir propriedades herdadas da superclasse.

Herança não estrita: as mudanças citada acima não são permitidas.

# POLIMORFISMO

*Polimorfismo* é um conceito abstrato que, de forma sucinta, diz que algo pode responder de formas diferentes para uma mesma situação.



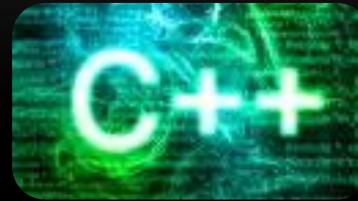
A implementação de Polimorfismo se faz através de Classes e métodos abstratos:

-Métodos abstratos (mostrar programa)

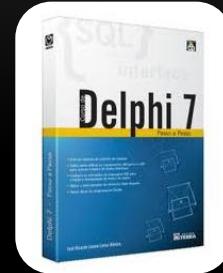
# LINGUAGENS DE PROGRAMAÇÃO ORIENTADAS A OBJETOS



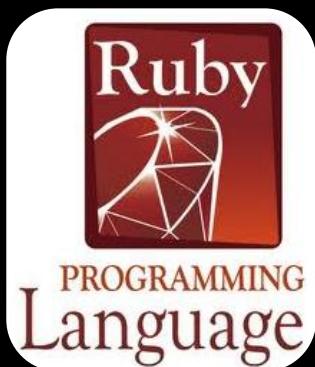
Java



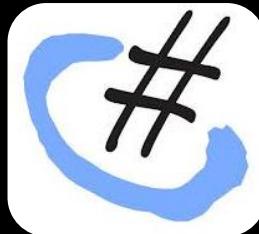
C++



Delphi



Ruby



C#

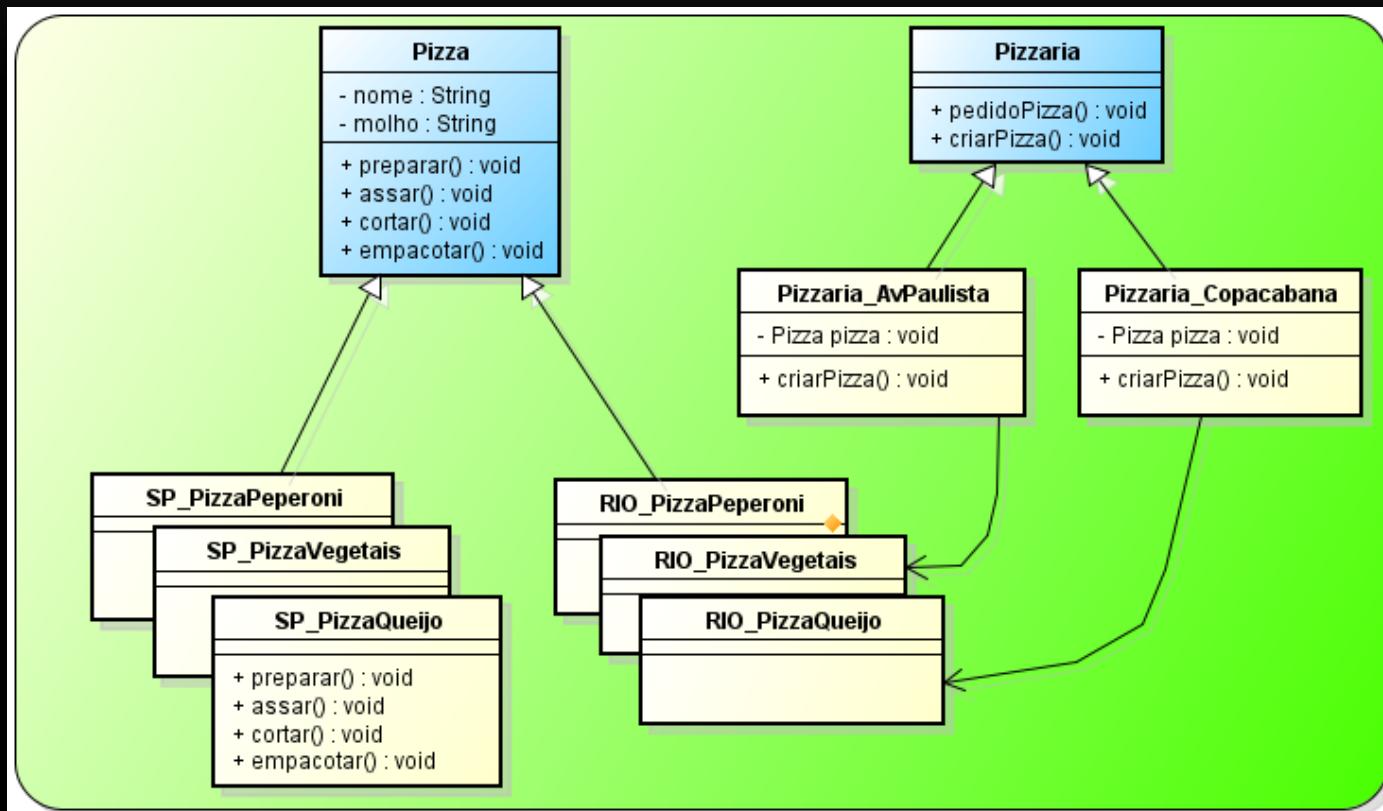


Python



Lisp

# APLICAÇÃO NA VIDA REAL



# BIBLIOGRAFIA

- <http://www.webgoal.com.br/origem-da-orientacao-a-objetos/> (08/09/2012 19:45hs)
- <http://www.hardware.com.br/artigos/programacao-orientada-objetos/> (08/09/2012 – 19:40hs)

## Conceitos de O.O

- Blair, G. et al. (Editors) *Object-Oriented Languages, Systems and Applications*, Pitman Publishing, 1991.
- Buzato, L. E., Rubira, C. M. F. *Construção de Sistemas Orientados a Objetos Confiáveis*, Décima Primeira Escola de Computação, Rio de Janeiro, Julho de 1998.
- Cardelli, L, and Wegner, P. On understanding types, data abstraction and Oliva, Alexandre. *Programação em Java. II Simpósio Brasileiro de Linguagens de Programação*, Campinas, setembro de 1997.
- Takahashi, T. Programação Orientada a Objetos, Escola de Computação, São Paulo, 1990.
- Rubira, C. M. F. Tópicos Especiais em Engenharia de Software II, Universidade Estadual de Campinas, notas de aula, 1996.
- <http://www.ufpa.br/cdesouza/teaching/es/3-OO-concepts.pdf>



# Diagrama de Caso de Uso

# Grupo

- ▶ Aline Mayara Coqueto – 10102152
- ▶ Felipe Gustavo Marques Pires – 10103101
- ▶ Gustavo Henrique da Cunha – 10101635
- ▶ Stéfanie Caroline Rodrigues Martins –  
10101621
- ▶ Thais Cirqueira da Silva – 10103368

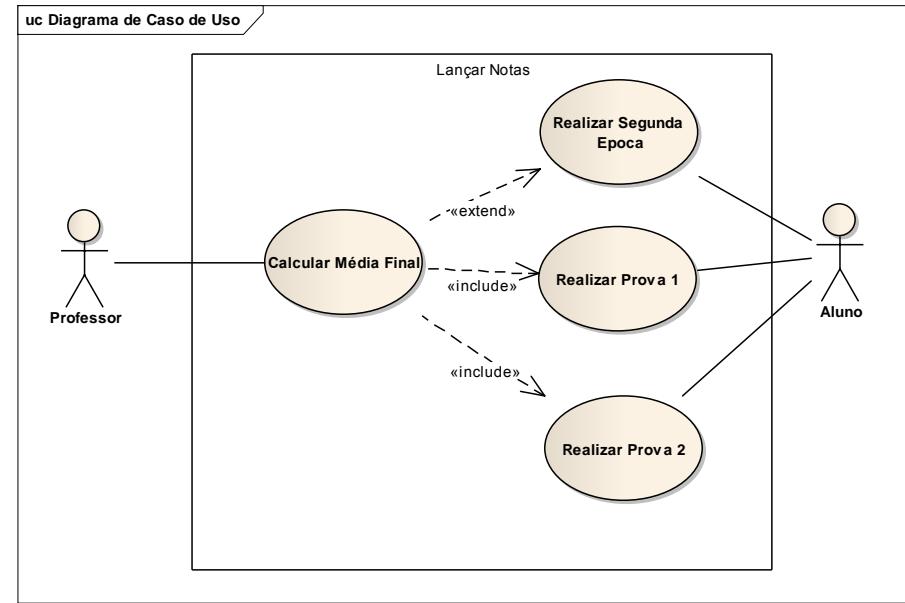
# REQUISITOS

- ▶ Requisitos são as necessidades do meu cliente. O que meu sistema irá fazer.
- ▶ Servem para ajudar a entender e delimitar o que deve ser implementado em um software.



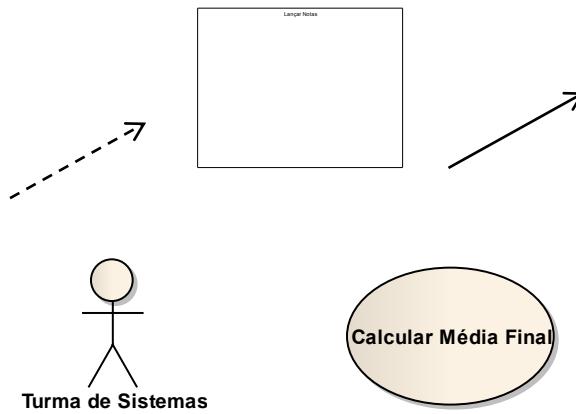
# DIAGRAMA DE CASO DE USO

- ▶ Responsável por exibir de forma prática as interações das funcionalidades do sistema uma com as outras e do sistema com o usuário.



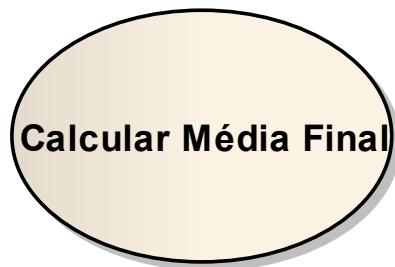
# ELEMENTOS DO DIAGRAMA

1. Caso de Uso;
2. Ator;
3. Relacionamento;
4. Fronteira do Sistema;
5. Cenário.



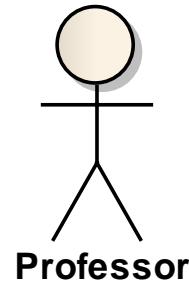
# 1. Caso de Uso

- ▶ Representa uma função do sistema.



## 2. Ator

- ▶ Papel desempenhado por algo ou alguma coisa externa ao sistema.

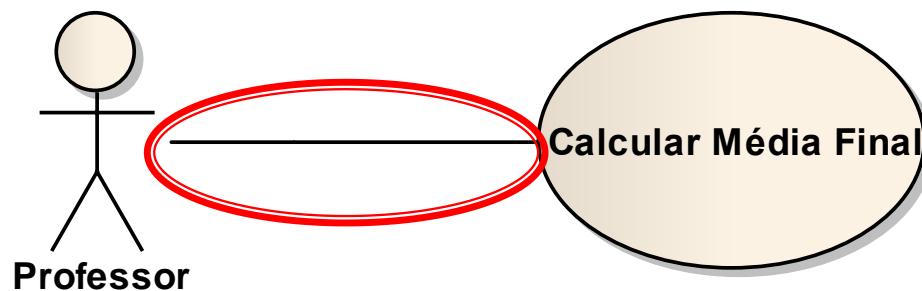


### 3. Relacionamento

- ▶ São as conexões feitas entre os elementos .
- ▶ Tipos de Relacionamento:
  - Associação ;
  - Generalização ;
  - Include ;
  - Extend.

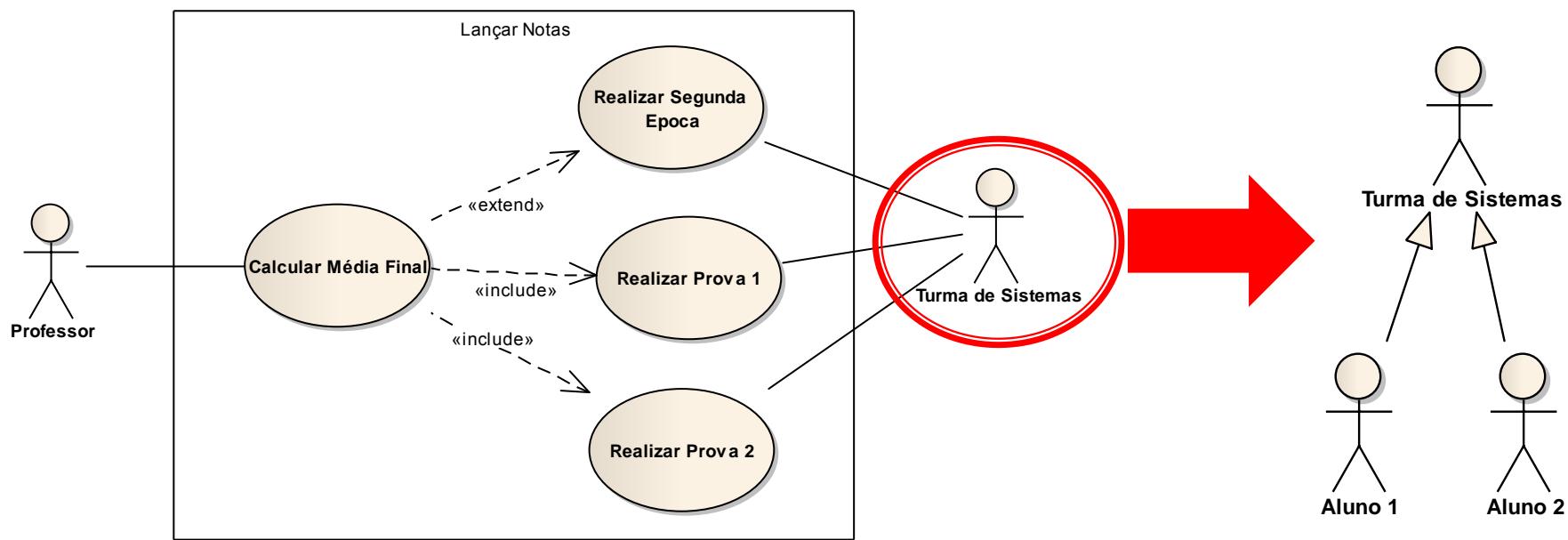
# Associação

- Relacionamento entre um ator e o caso de uso.



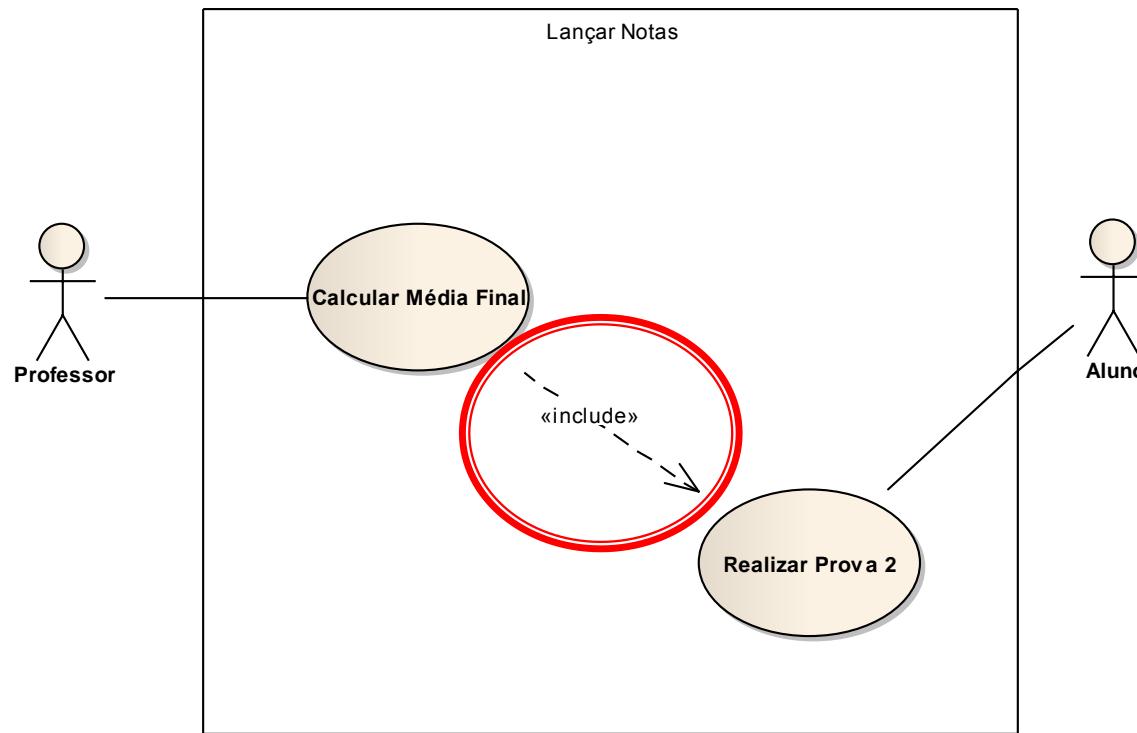
# Generalização

- ▶ Demonstra quando há algo em comum entre os papéis do sistema.



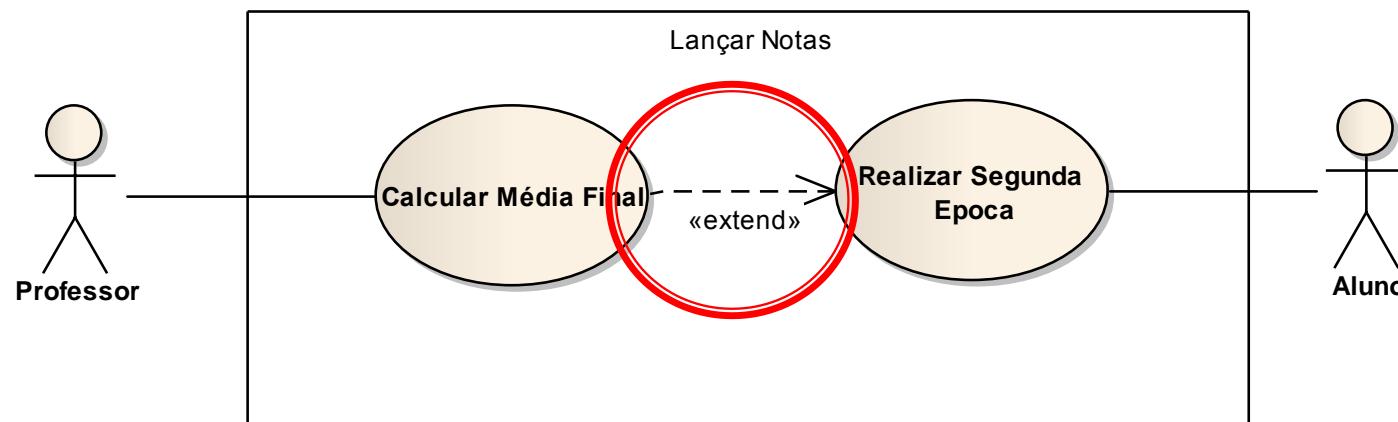
# Include

- ▶ Demonstra a dependência entre dois casos de uso.



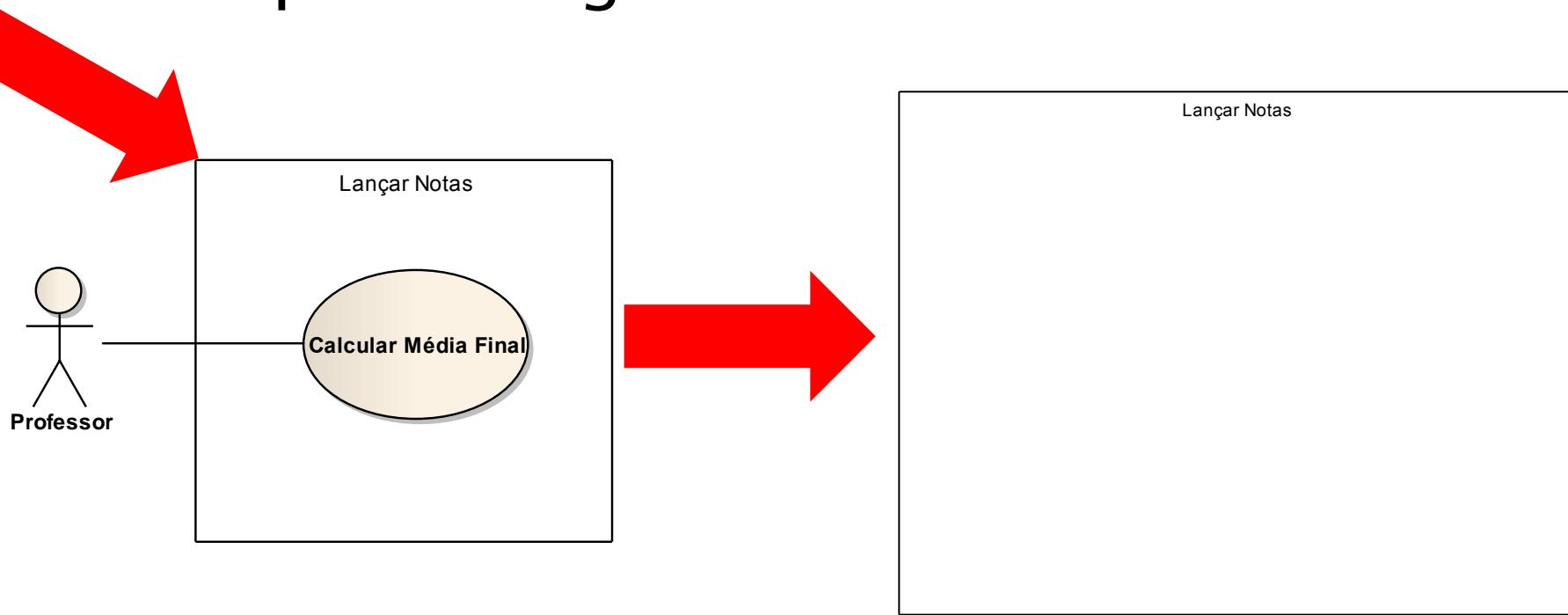
# Extend

- ▶ Demonstra que o caso de uso base pode ser complementado por outro caso de uso.



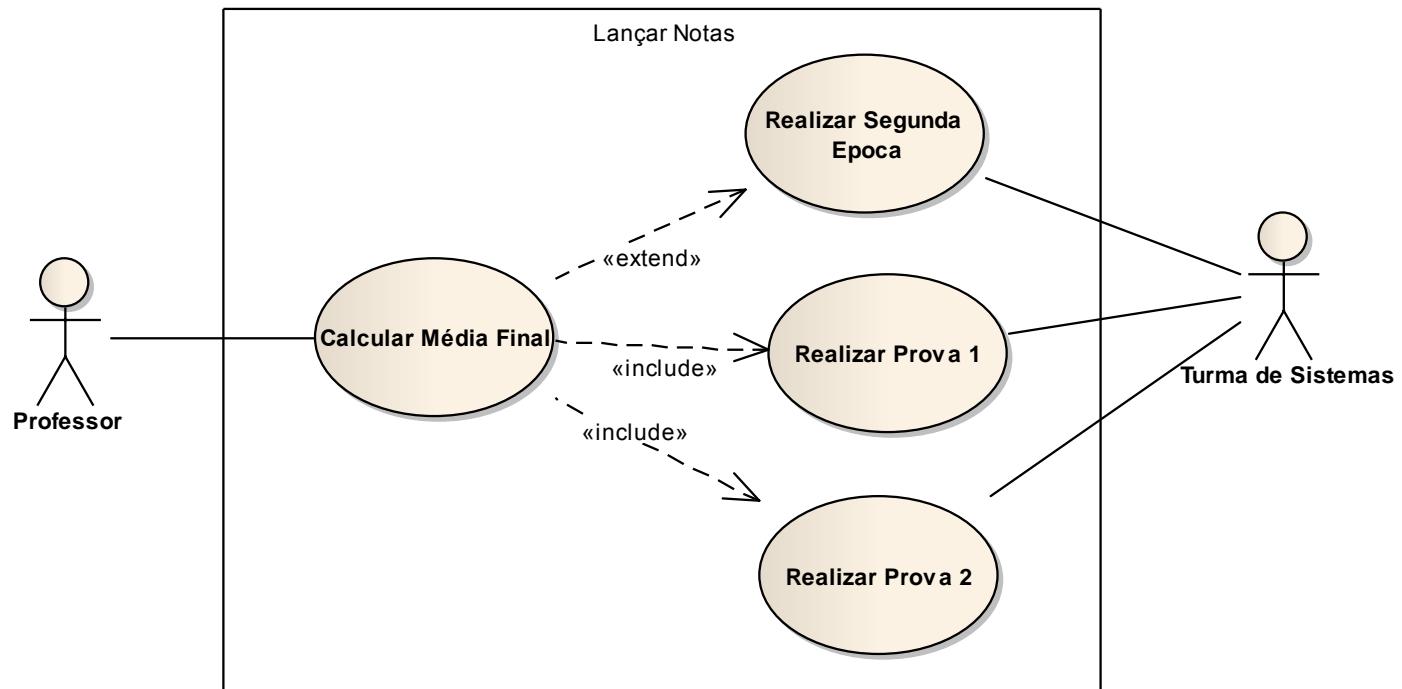
# 5. Fronteira do Sistema

- ▶ Delimita os casos de uso que irão compor o diagrama.



# 6. Cenário

- ▶ Sequência de eventos gerados a partir de uma ação do usuário.



# DESCRIÇÃO DE CASO DE USO

- ▶ Documento que descreve o passo a passo a respeito das ações geradas pelo sistema através de determinadas ações do ator.
- ▶ Composto por:
  - Fluxo Principal;
  - Fluxo Alternativo;
  - Ator.

# IMPORTÂNCIA DO CASO DE USO

- ▶ Escopo bem definido;
- ▶ Organização e Divisão de Trabalho;
- ▶ Estimativa do Tamanho do Projeto;
- ▶ Direcionado de Testes.



# NA PRÁTICA



Documento do  
Microsoft Office Word 9



Documento do  
Microsoft Office Word 9



G:\Exemplos\  
Caso de Uso.png



H:\Exemplos\  
Estrutura EA.png



H:\Exemplos\  
Caso de Uso EA.png

# Diagrama de Classes

# UML

Nome: Fellipe Ricardo  
Fellipe Callegari  
Jordana Müller  
Luana Soares

Prontuário: 10100813  
10100831  
10103659  
10100819

Professor: Alessandro

Turma: SIN-NA6

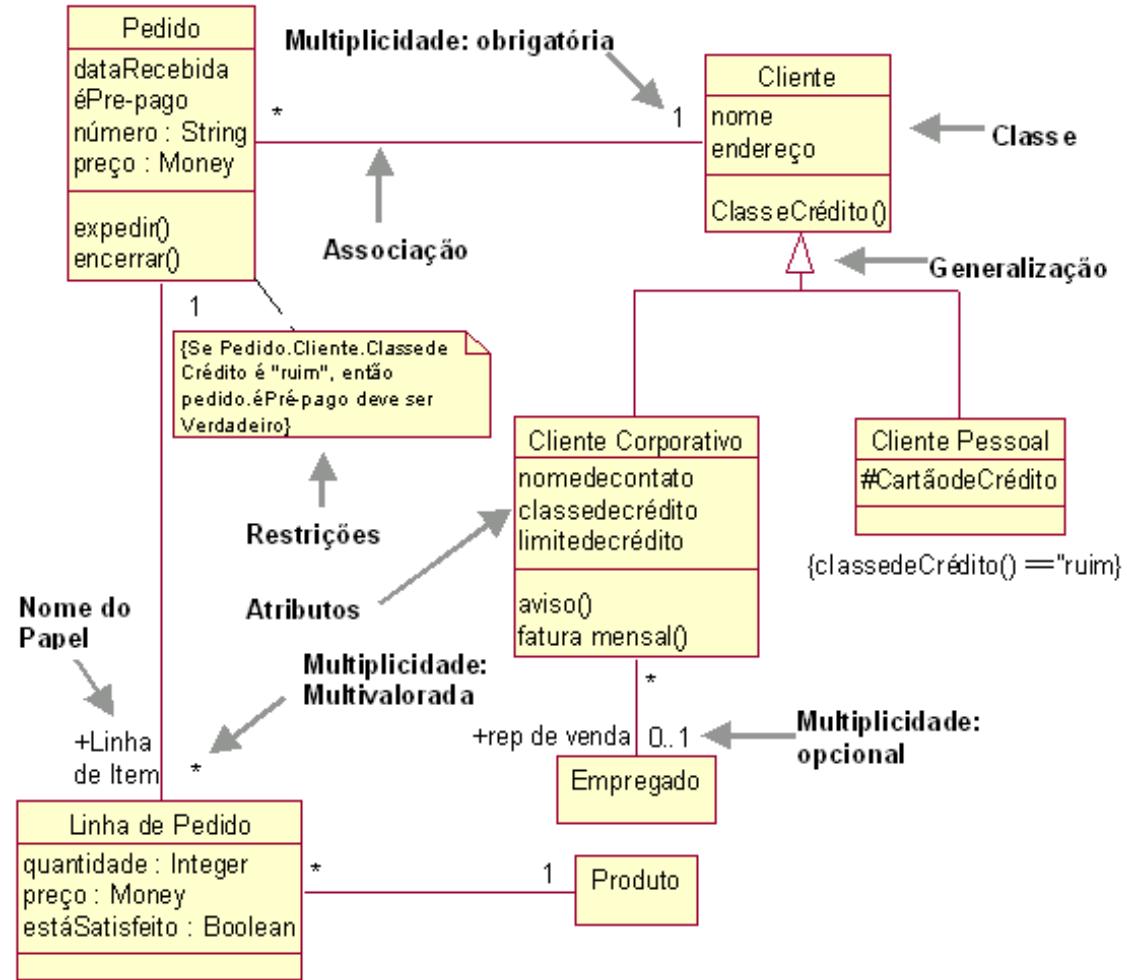
# Agenda

- Introdução
- Exemplo
- Utilização
- Composição
- Relacionamento
- Níveis de abstração
- Criação de uma boa estrutura

# Introdução

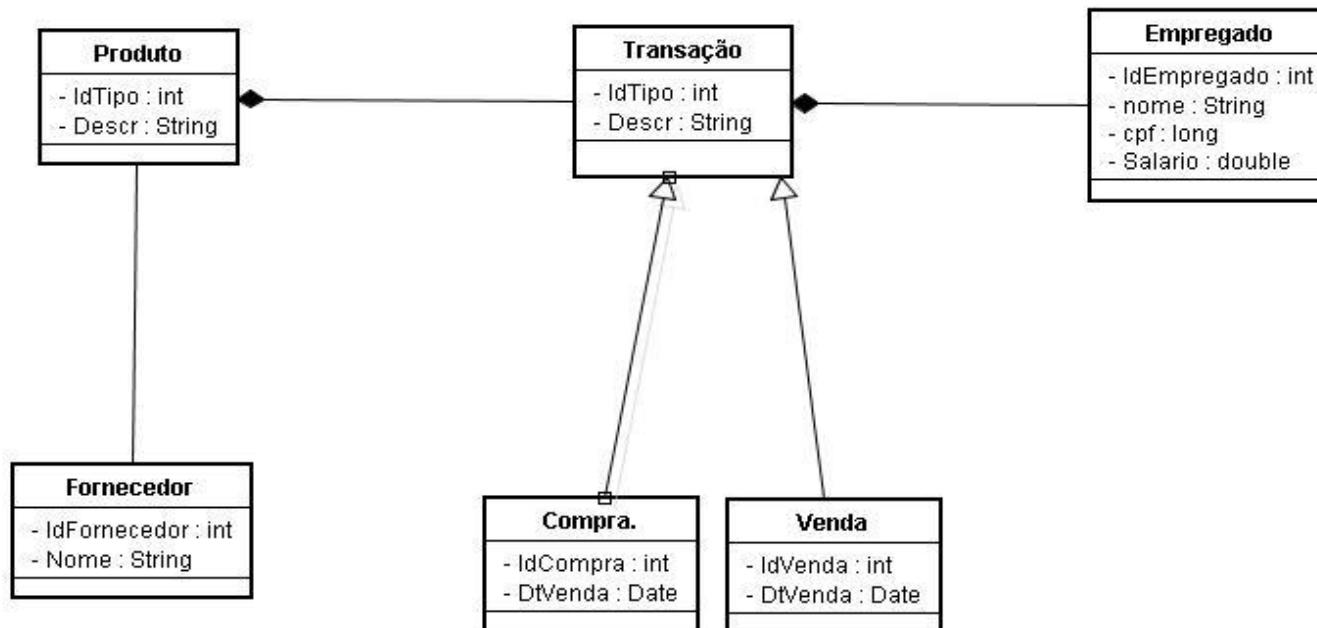
- Um diagrama de classes mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos.
- Os Diagramas de Classe são a base para: Diagramas de Componentes e os Diagramas de Implantação.
- Os diagramas de classe são importantes para a construção de sistemas executáveis por intermédio de engenharia do produção e reversa.

# Exemplo

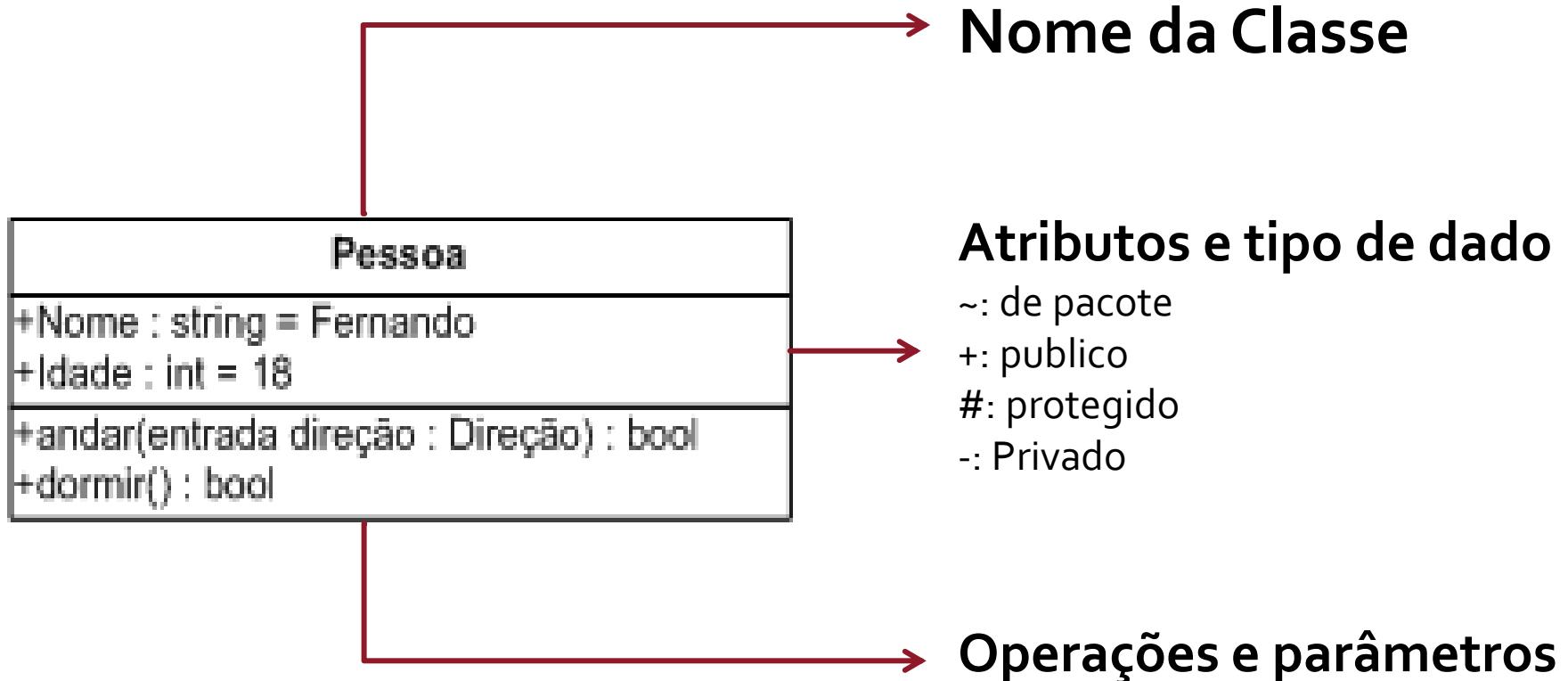


# Utilização

- Use os diagramas de classe para fazer a modelagem da visão estática do projeto de um sistema.



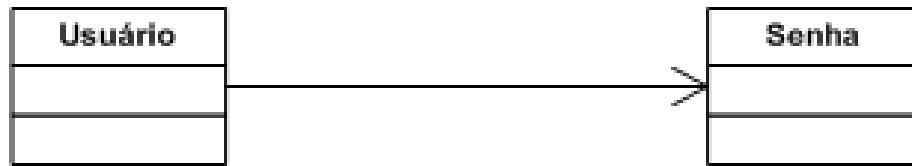
# Composição



# Relacionamento

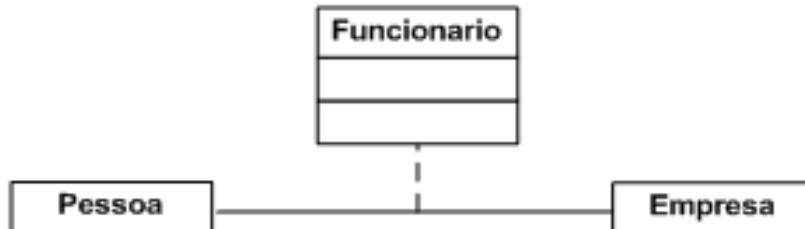
## Associações

- Define relacionamento entre as classes



## Classe de Associações

- Uma classe de associação contem informações de relacionamento entre outras classes



# Relacionamento

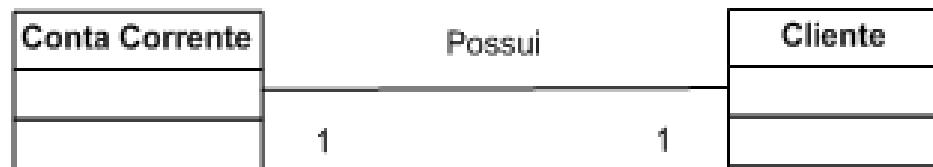
## Dependências

- Ocorre quando não é possível existir a classe A sem que a classe B já exista



## Multiplicidade

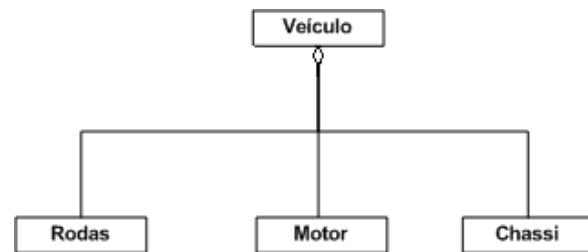
- Utiliza notações ("0..1", "0..\*", "\*", "1", "1..\*")



# Relacionamento

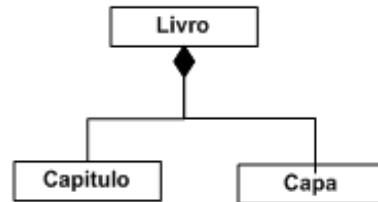
## Agregações

- Relacionamento onde uma classe é formada por diversas outras classes



## Composição

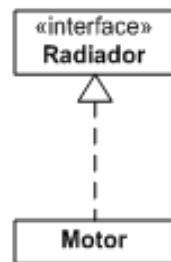
- É outro tipo de agregação, em que a relação é mais forte



# Relacionamento

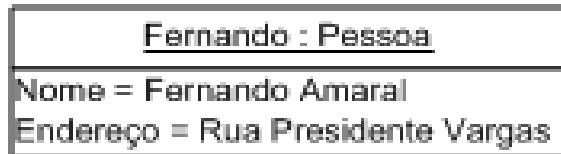
## Interfaces

- As interfaces são apenas modelos de comportamentos, não podendo ser instanciadas



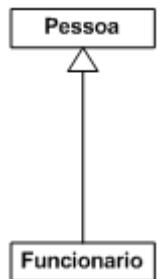
## Diagrama de Objetos

- Diagrama de objeto representa uma instancia de uma classe específica

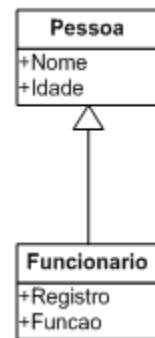


# Níveis de abstração

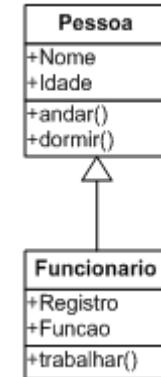
Nível de domínio



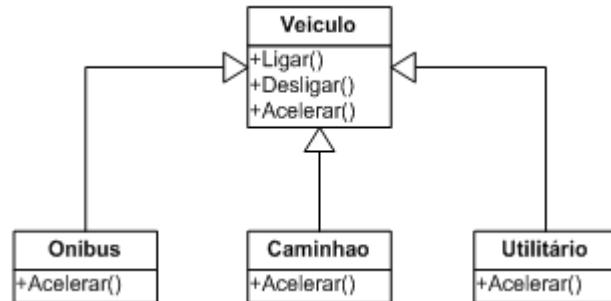
Nível de análise



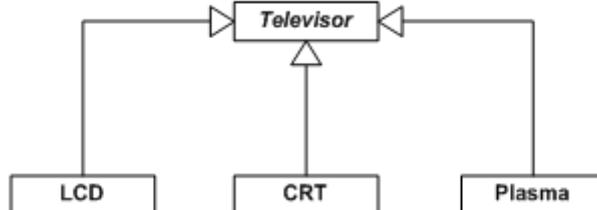
Nível de design



Generalização



Classes abstratas



# Criação de uma boa estrutura

- Atribua-lhe um nome que comunique seu propósito
- Distribua organizadamente seus elementos de modo a minimizar o cruzamento de linhas
- Comunicação de um único aspecto da visão estática do projeto do sistema
- Apresentação de elementos essenciais à compreensão desse aspecto
- Exibir somente os detalhes essenciais à compreensão
- Use notas e cores para características importantes do diagrama

# Diagrama de Classes UML

# Fontes

- LIVRO: UML Guia do Usuário EDITORA: Campus AUTOR: Grady Booch
- LIVRO: Utilizando UML e Padrões EDITORA: Bookman AUTOR: Craig Larman
- <http://www.fernandoamaral.com.br/Default.aspx?Artigo=40>
- [http://www.macoratti.net/net\\_uml1.htm](http://www.macoratti.net/net_uml1.htm)
- [http://pt.wikipedia.org/wiki/Diagrama\\_de\\_classes](http://pt.wikipedia.org/wiki/Diagrama_de_classes)
- <http://www.slideshare.net/suissepg/diagrama-de-classe-5802269>
- <http://gilmarborba.com.br/?p=184>
- <http://www.dsc.ufcg.edu.br/~jacques/cursos/map//html/uml/diagramas/classes/classes3.htm>
- <http://javafree.uol.com.br/topic-876366-Diagrama-de-Classes-e-Objetos.html>

# Diagrama de Classes

# Integrantes

- Jéssica Soares Barbosa
- Marcelo Yassuo Cecilio Furuko
- Rafael de Lima Abreu
- Rafael Rodrigues Alves
- Tatiana Ribeiro de Oliveira

# Diagrama de Classes

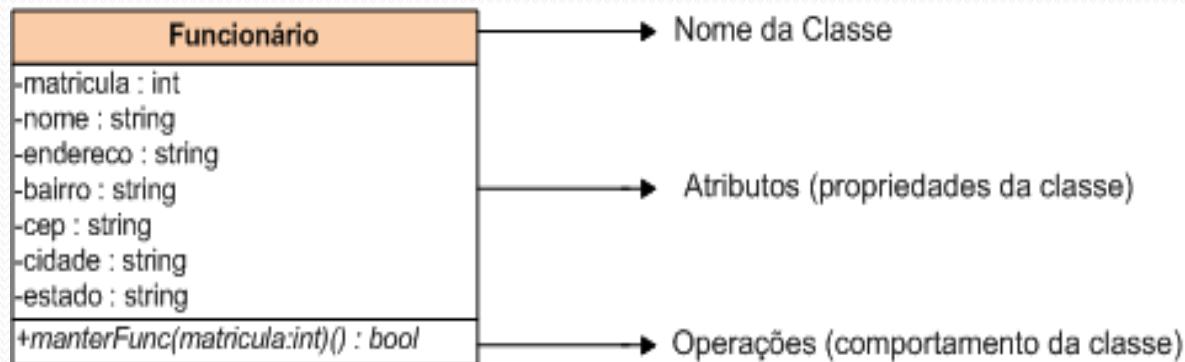
- Pode-se dizer que é um dos mais importantes diagramas da UML. A partir dele que os demais diagramas são elaborados;
- Um diagrama de classes descreve os tipos de objetos no sistema e os vários tipos de relacionamentos estáticos que existem entre eles.
- Em programação podemos dizer que diagrama de classes é uma representação da estrutura e relações das classes que servem de modelo para objetos

# Seus Elementos

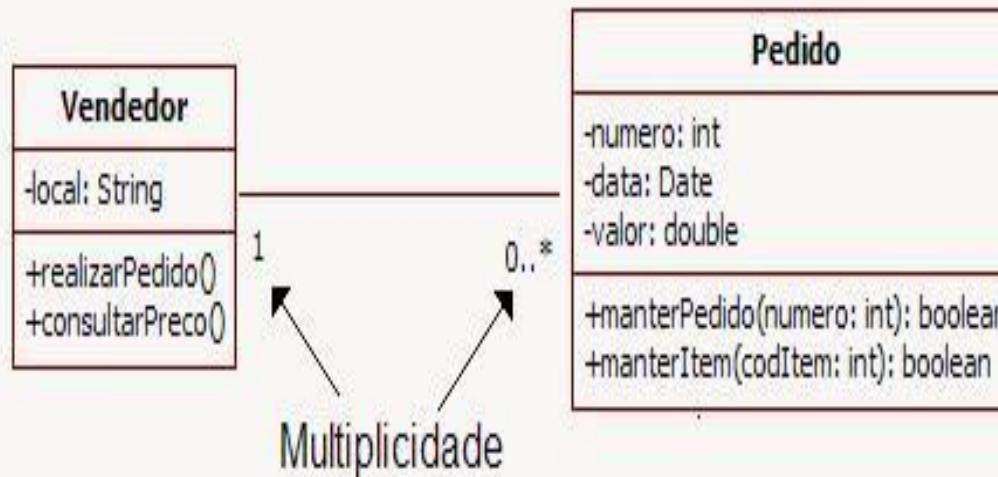
- Atributos
- Multiplicidade
- Associação
- Relacionamento
  - Generalização ou Herança
  - Agregação
  - Composição

# Atributo

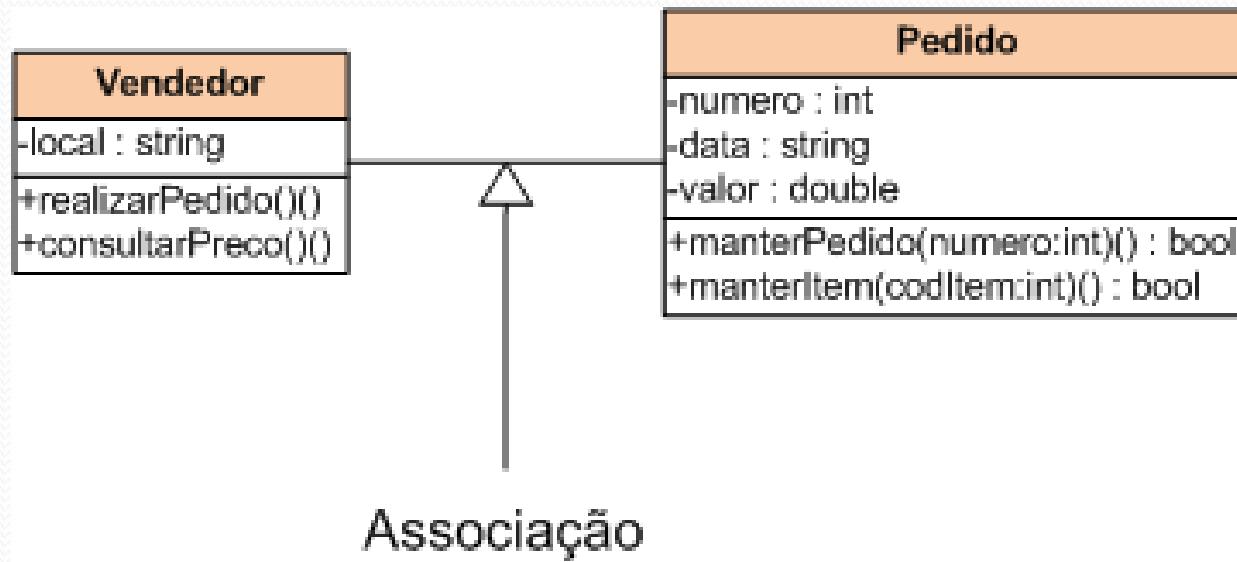
- Define as características da classe como um todo.
  - Nome
  - Tipo de dados
  - Valor inicial (depende da linguagem de programação, e é opcional)
  - Propriedade (opcional, depende da característica do elemento)



# Multiplicidade

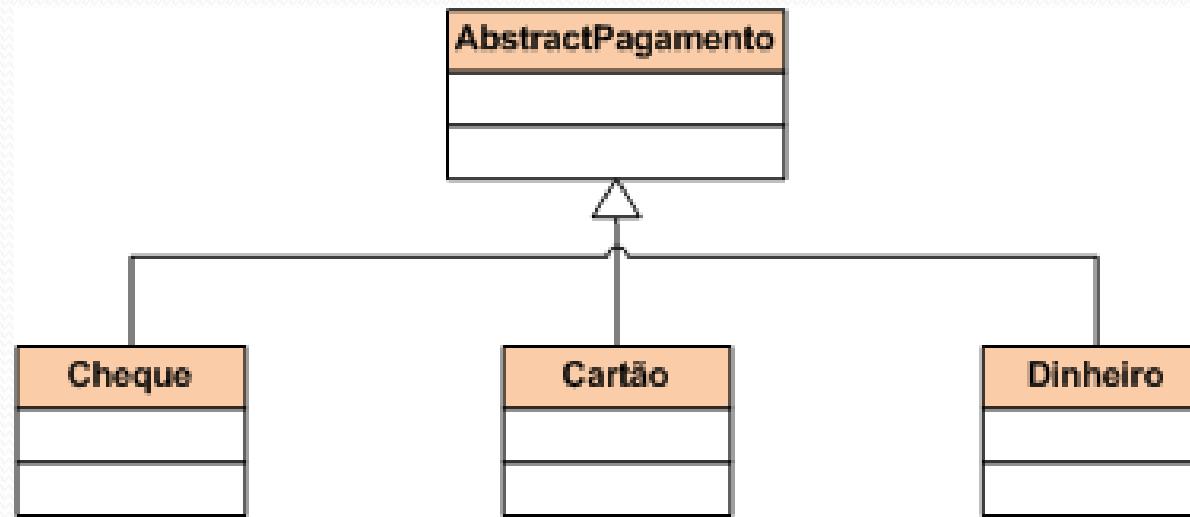


# Associação



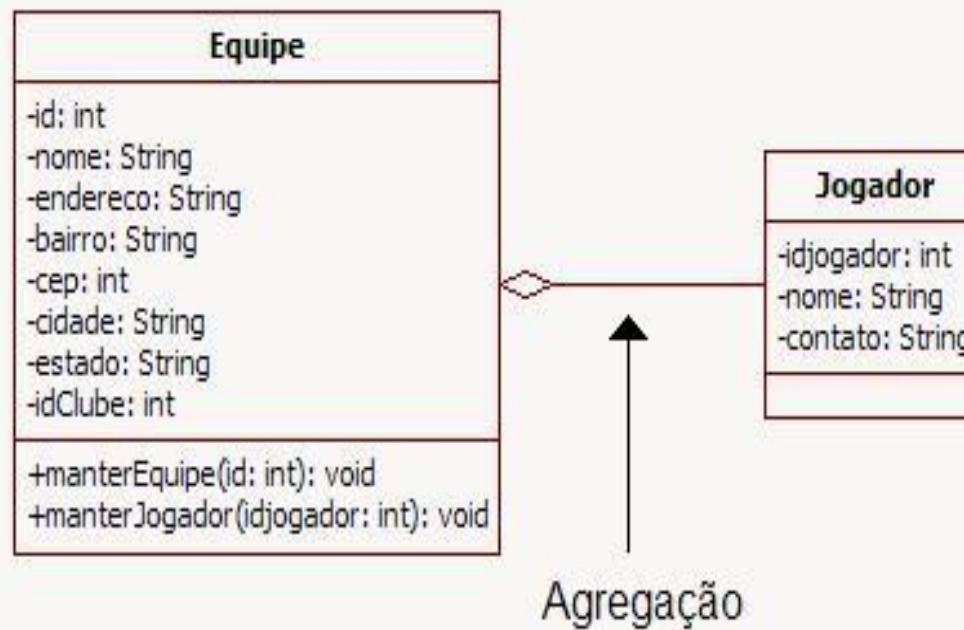
- As associações em um diagrama de classe definem os tipos de ligações que os objetos participam

# Relacionamento – Generalização



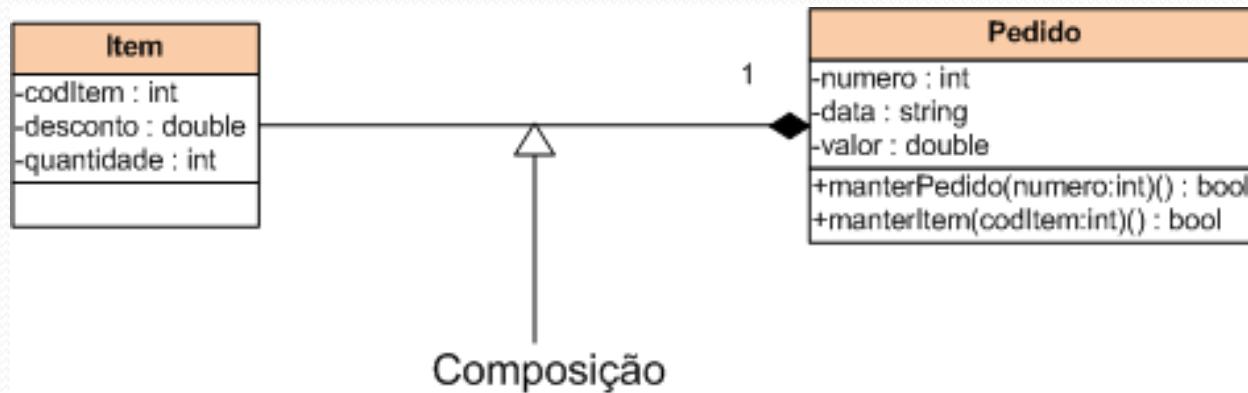
- Na implementação física corresponde a um processo de herança.

# Relacionamento - Agregação



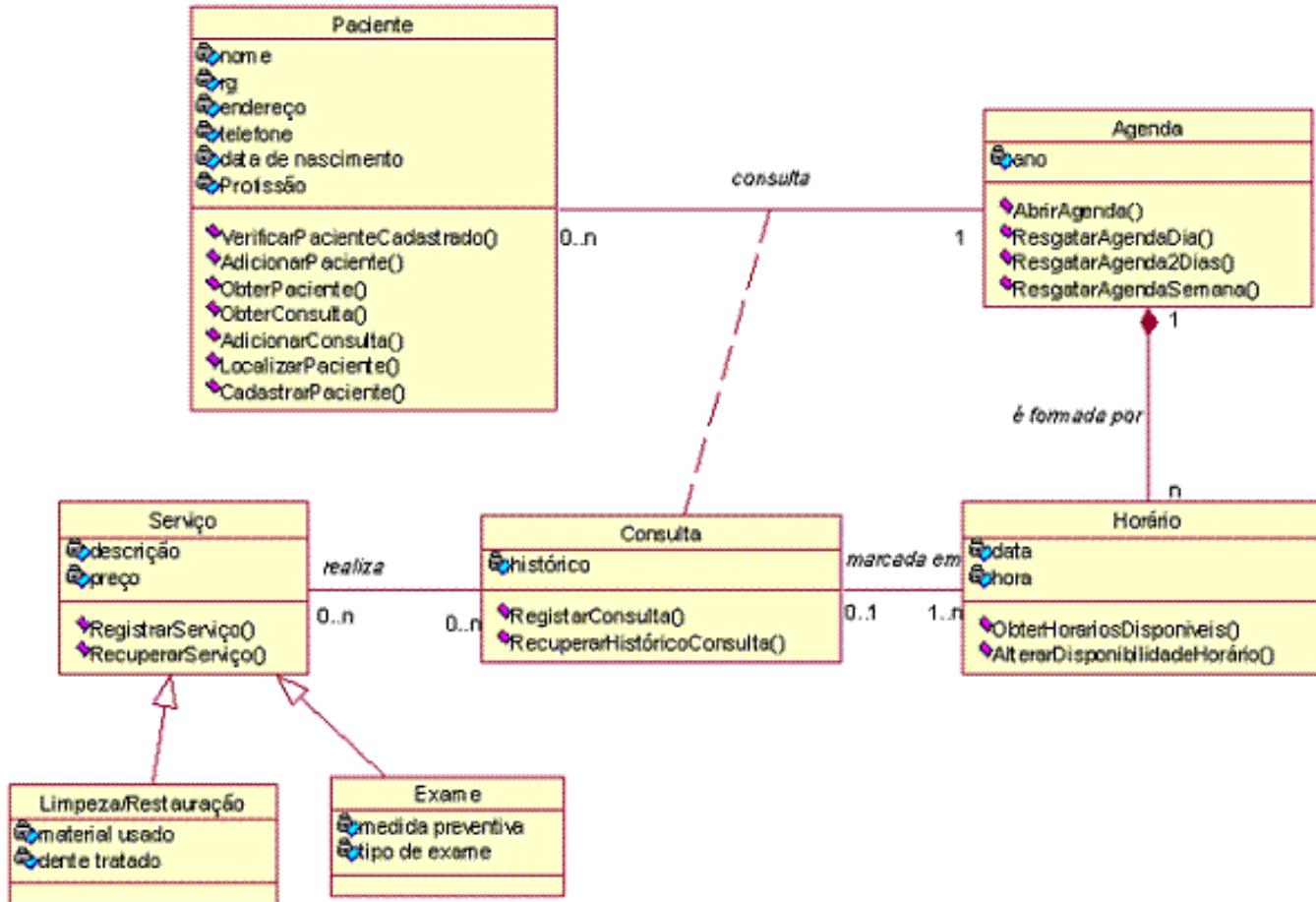
- Uma agregação representa um todo que é composto de várias partes;

# Relacionamento - Composição



- A composição, é diferente da agregação, pois é um relacionamento de contenção. Um objeto CONTÉM outros objetos(elementos). Esses elementos que estão contidos dentro de outro objeto depende dele para existir.

# Exemplo



# Especificação de instância

- O que é a especificação da instância?
- Qual o conceito utilizado para especificar?
- Diagrama de Objetos utilizado na especificação de instâncias.

# Especificação de instância

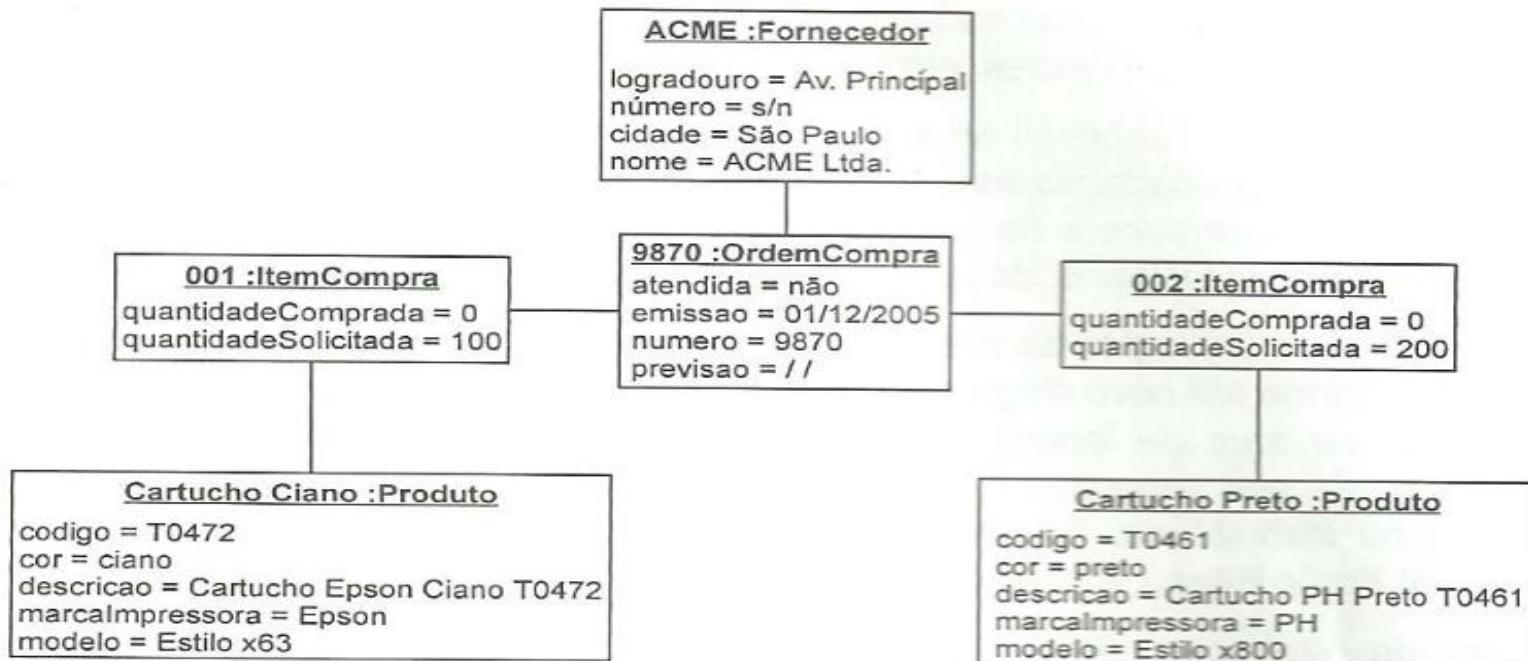
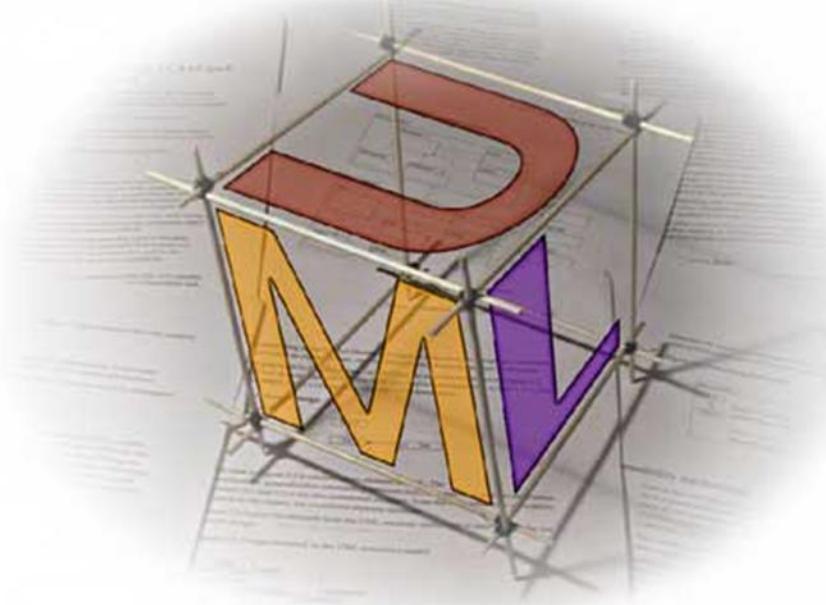


Figura 9.25. Diagrama de especificação de instância.

# Engenharia de código



# Engenharia Reversa

- Importação de código para geração de interfaces UML
- Auxilia na recuperação de dados
- Verificação de código original para identificar possíveis alterações
- Permite modelar melhores proteções para um software já existente

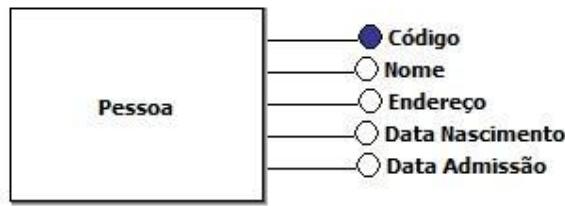
# Geração de Código



# Classes de Persistências

- Visa preservar de maneira permanente os objetos de uma classe – “gravar em disco”
- Nem toda classe é/precisa ser persistente
- Necessário explicitamente definir através de um estereótipo/restrição
- Diagrama de classe → esquema lógico BD

# Mapeamento de Atributos

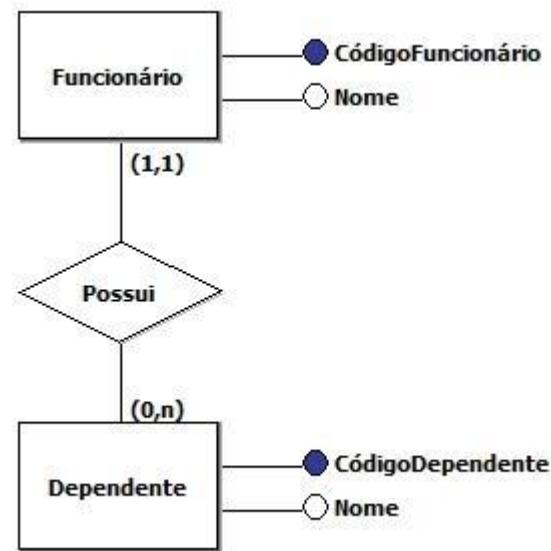


Pessoa
Código: Inteiro
Data Admissão: Data
Nome: Texto(50)
Data Nascimento: Data
Endereço: Texto(50)



Produto
Código: Inteiro
Descrição: Texto(50)

# Mapeamento de Relacionamentos



# Conclusão

- Visão geral do diagrama de classes
- Vantagens da utilização em projetos
- O que agrega para o curso?



# Diagrama de Objetos

SIN – NA6 - Engenharia de Software II

Alexandre Alisson

Leonardo Botelho

Lucas Palma

Luciana Sassaki

## • Função do Diagrama

- O diagrama de objetos é uma poderosa ferramenta da UML para a modelagem de exemplos.
- Esse diagrama é também usado para testar o comportamento de uma determinada “sociedade de classes”.

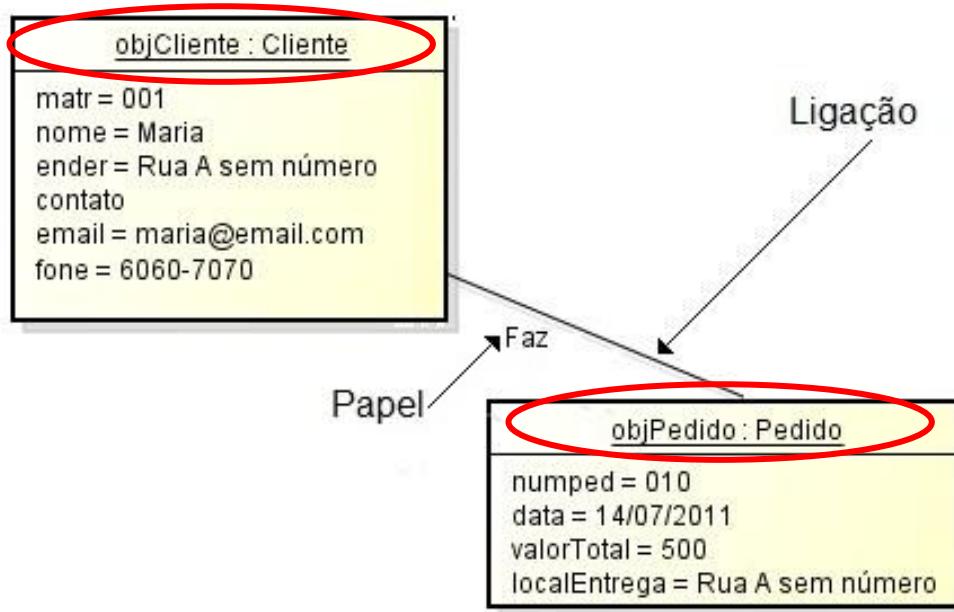


## • Função do Diagrama

- A função do Diagrama de Objetos é detalhar a classe, ele mostra os objetos da classe.
- Testar se a classe foi especificada corretamente.
- São importantes para visualizar, especificar e documentar os modelos estruturais.

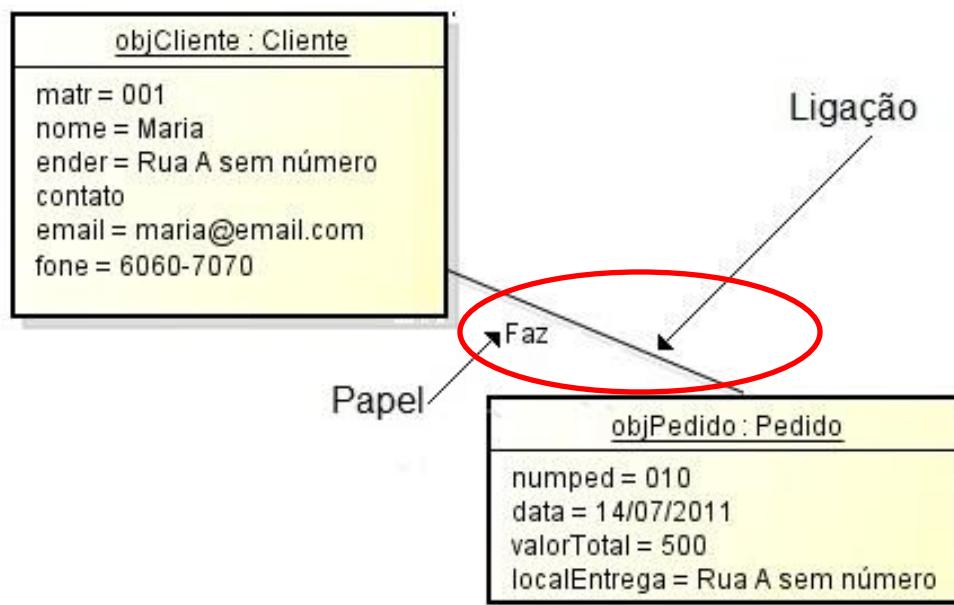
# • Estrutura do Diagrama

- Objeto: É apresentado a partir de duas partes: O nome do objeto e (com a separação de dois pontos) o nome da classe correspondente.



# • Estrutura do Diagrama

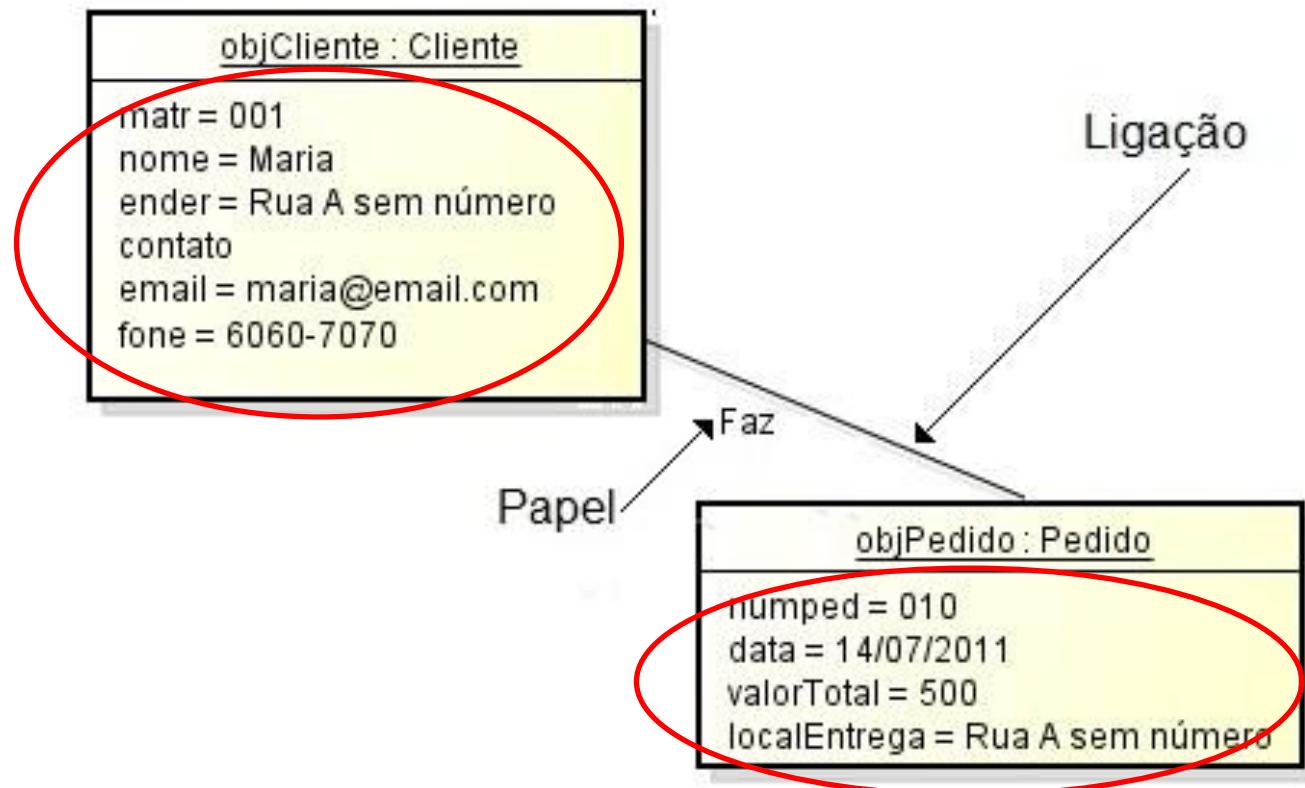
- Ligações: Os objetos além de representar/exemplificar os dados, também mostram os enlaces com outros objetos. Esses enlaces são denominados **ligações**.



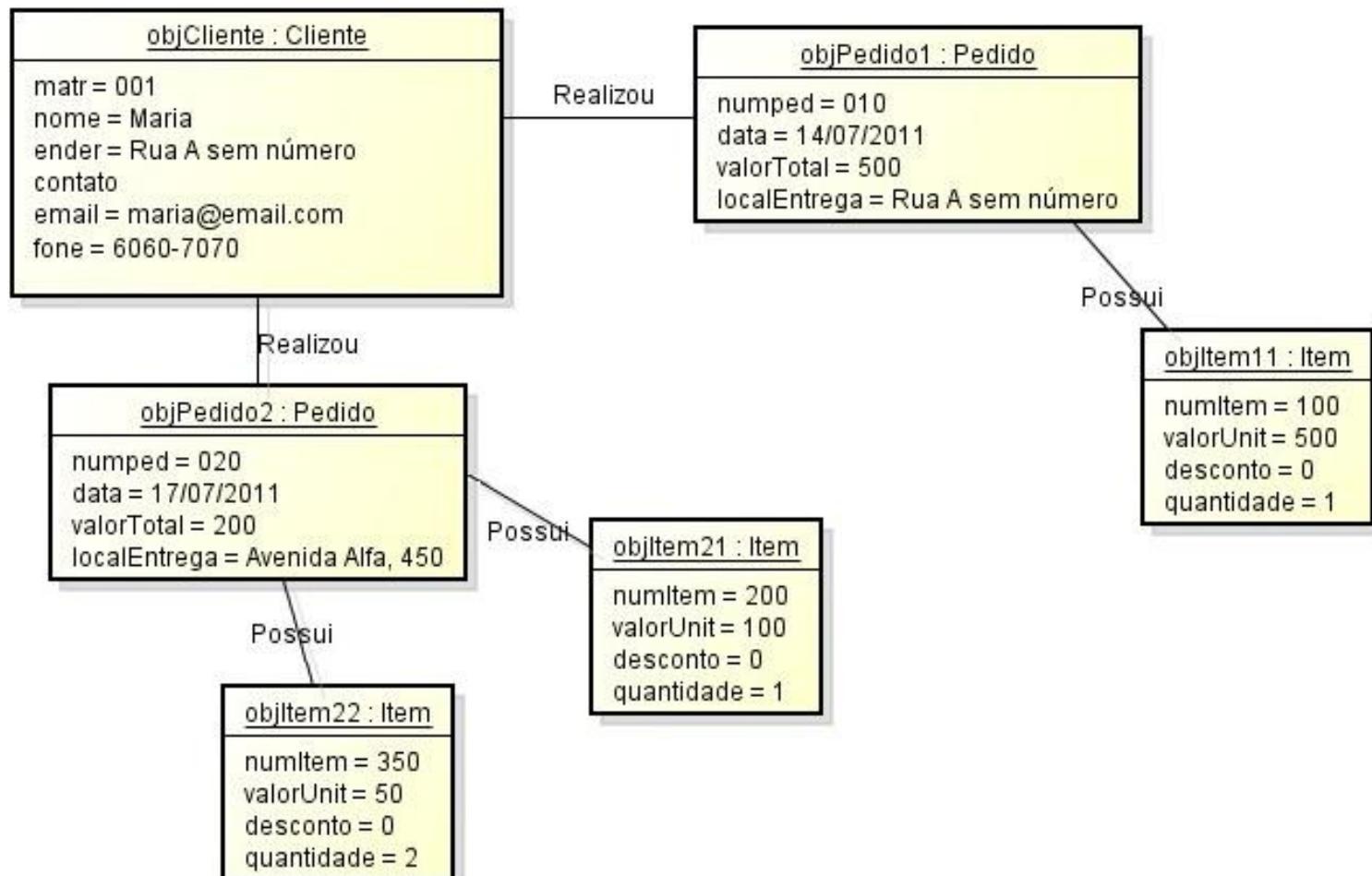
## • Estrutura do Diagrama

- Nome e Valores de Atributos: O diagrama de objetos NÃO MOSTRA a definição dos atributos (tipo, visibilidade e assim por diante). Ele mostra nomes e valores de atributos em uma instrução de atribuição, como nome = “João”; CEP=”30000-000” etc. Este diagrama simplesmente apresenta os nomes e os valores dos atributos no segundo compartimento da caixa representativa do objeto no diagrama.

# • Estrutura do Diagrama



# • Estrutura do Diagrama

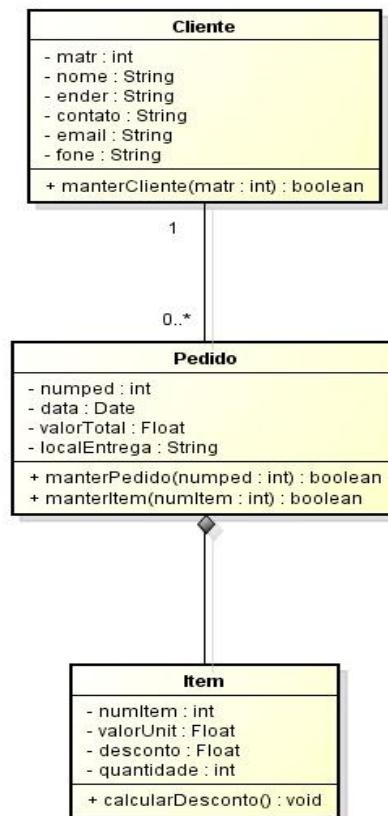


# • Exemplo de Utilização

- Diagrama de Objeto



- Diagrama de Classe



## • Como deve ser usado

- Esse diagrama é muito útil à medida que exemplifica para o usuário final, ou até mesmo o analista de sistemas. A finalidade da classe ou de determinados atributos, que em alguns casos poderiam indicar dúvidas. Além disso esse digrama é muito importante para testar se a classe foi definida corretamente, principalmente no que diz respeito às multiplicidades, ou a quantidade de ocorrências de um objeto com relação a outro.

## • Conclusão

- Vantagens e Desvantagens.
- O grupo concluiu que na teoria é de grande ajuda utilizar o diagrama de objetos. Entretanto, na prática, levando em consideração a agilidade e urgência da entrega dos projetos fica complicado a aplicação do diagrama.

# • Bibliografia

- <http://www.unesp.br/gs/treinamento/graduacao/CursoUML-Diagramas.pdf>
- <http://www.fag.edu.br/professores/elielder/materias/apoo/09.pdf>
- <http://techblog.desenvolvedores.net/2011/05/28/diagrama-de-objeto-uml>
- [http://tadeujnr.sites.uol.com.br/pcc/txt\\_uml.html](http://tadeujnr.sites.uol.com.br/pcc/txt_uml.html)
- <http://gilmarborba.com.br/?p=706>

# DIAGRAMA DE SEQUÊNCIA

O diagrama que enfatiza interações entre objetos.

# Grupo

- Francine - 10102189
- Kelly - 09106691
- Lyncon - 09211083
- Marina - 09107265
- Rodrigo - 10104385

# Introdução

- O que é?
- Função
- Estrutura
- Exemplos
- Conclusão

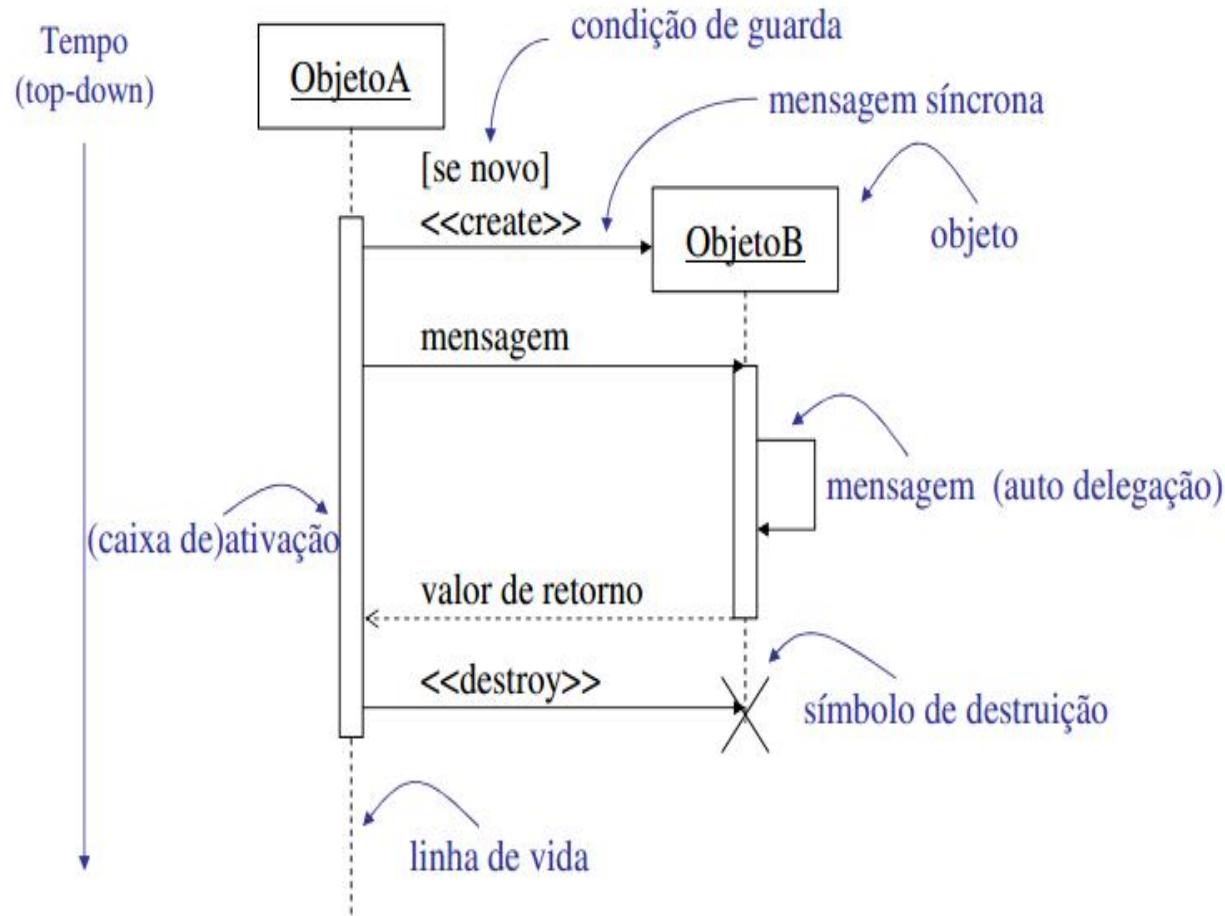
# O que é?

- O Diagrama de Sequencia é um modelo suportado pela UML, ele se localiza dentro do grupo de diagramas dinâmicos, pois exerce colaboração dinâmica entre os vários objetos de um sistema.

# Função

- Mostrar a sequência de mensagens enviadas entre os objetos;
- Mostrar o que ocorre em pontos específicos da execução do sistema;
- Apresentar as interações entre atores e sistema.

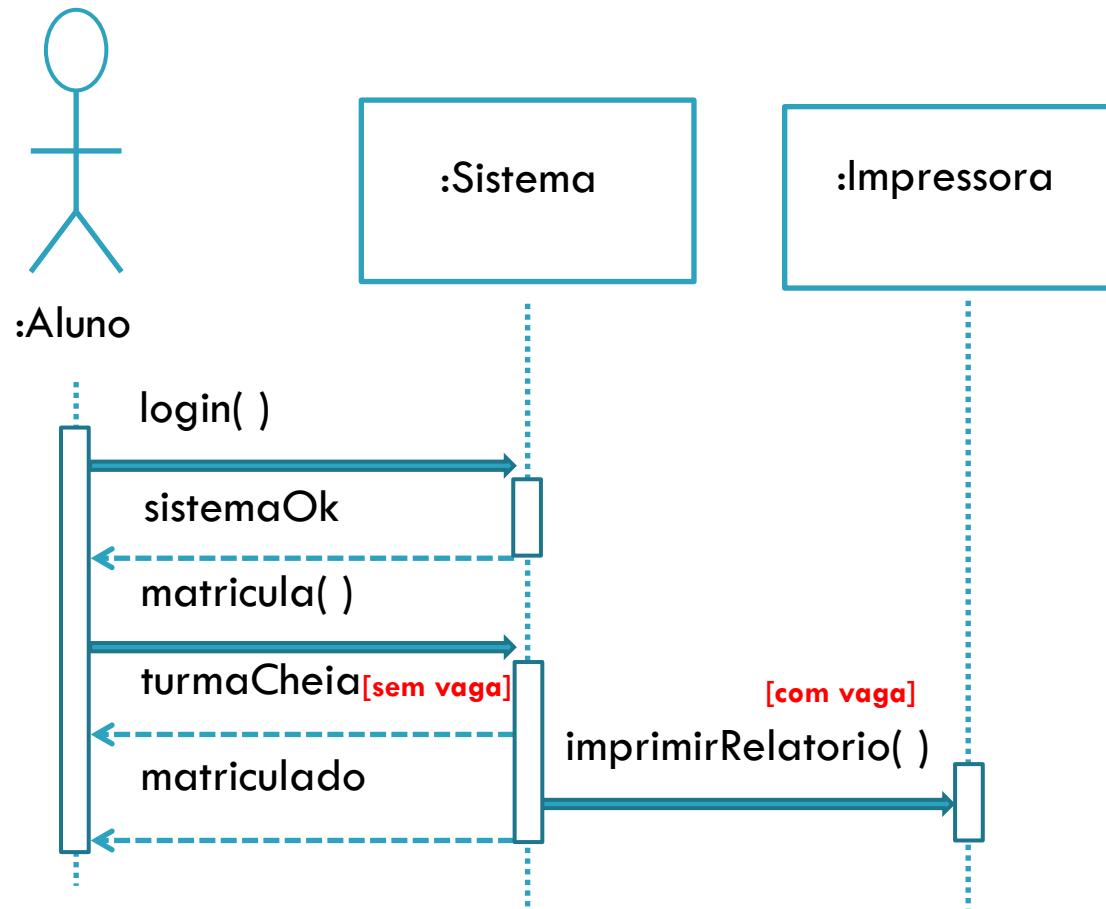
# Estrutura



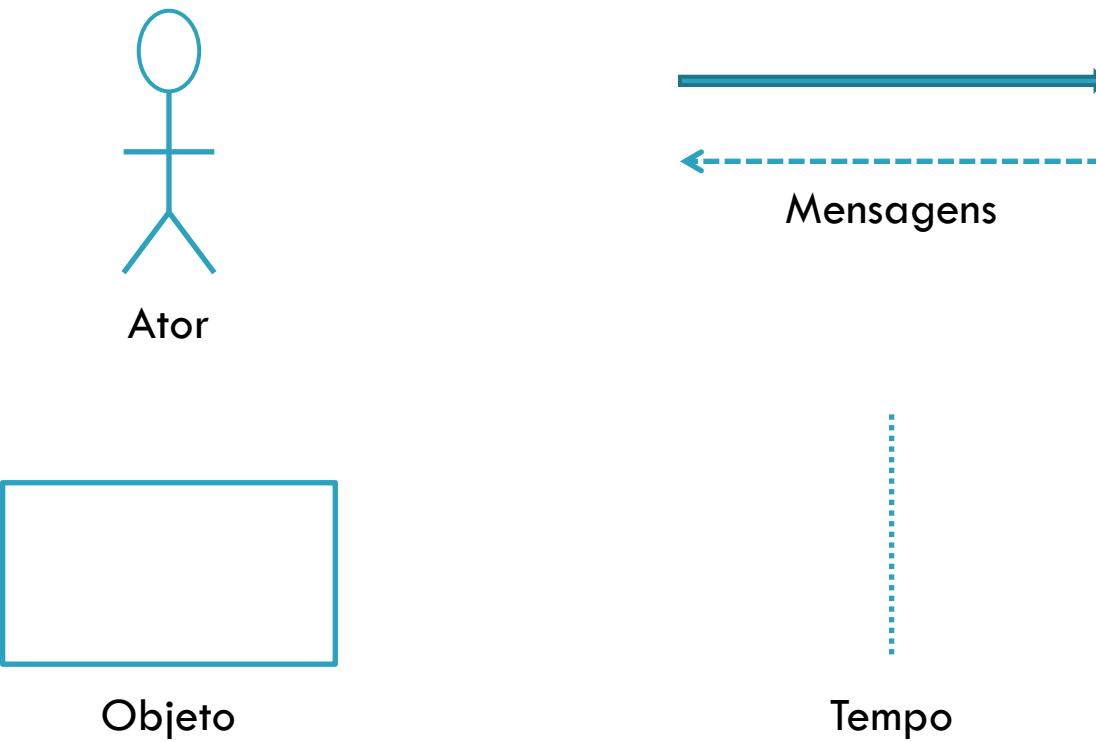
# Estrutura → Mensagens → Tipos

Símbolo	Significado
→	Mensagem síncrona
→	Mensagem assíncrona
←.....	Mensagem de retorno (opcional)

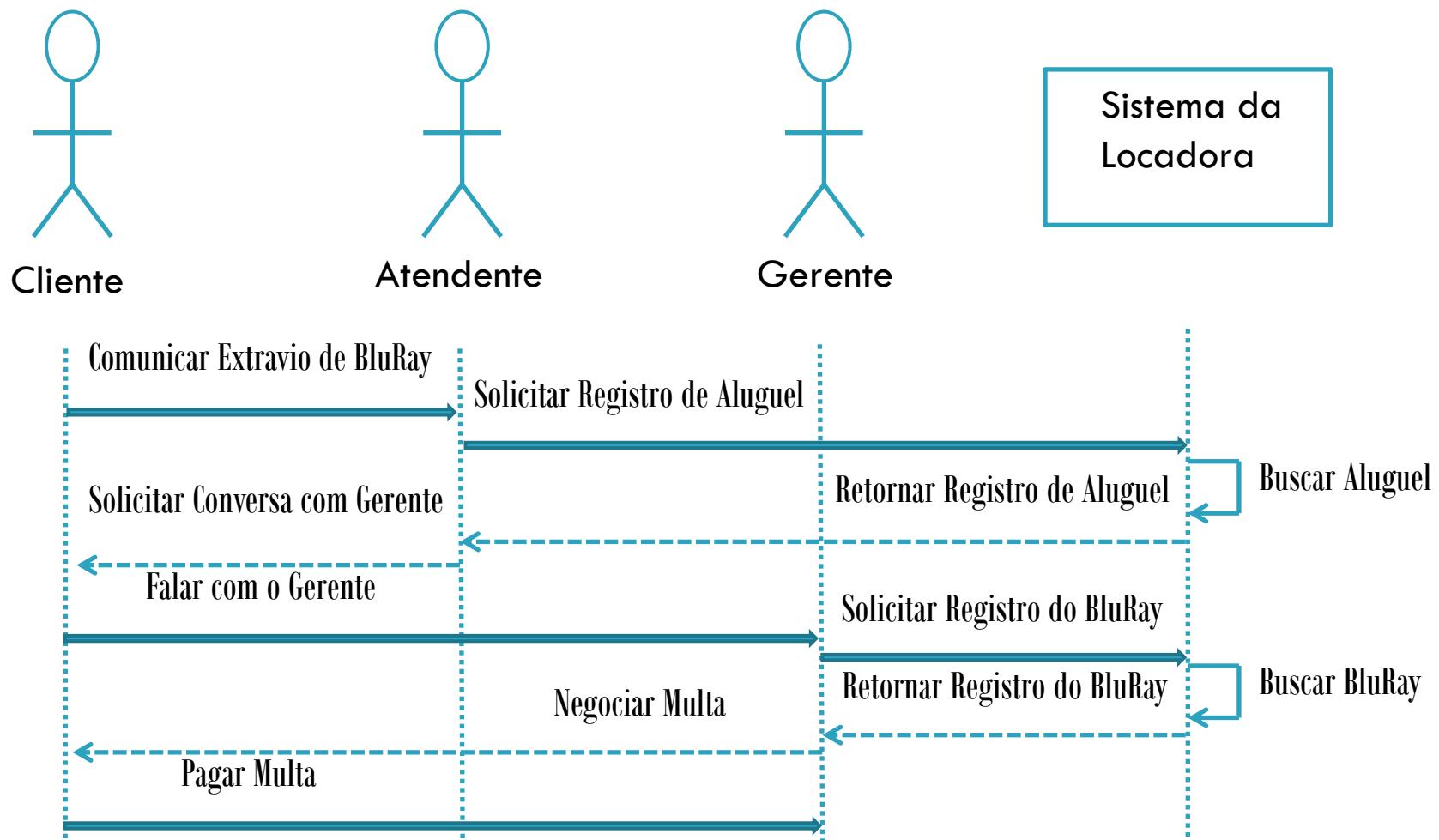
# Estrutura → Mensagens → Condições de guarda



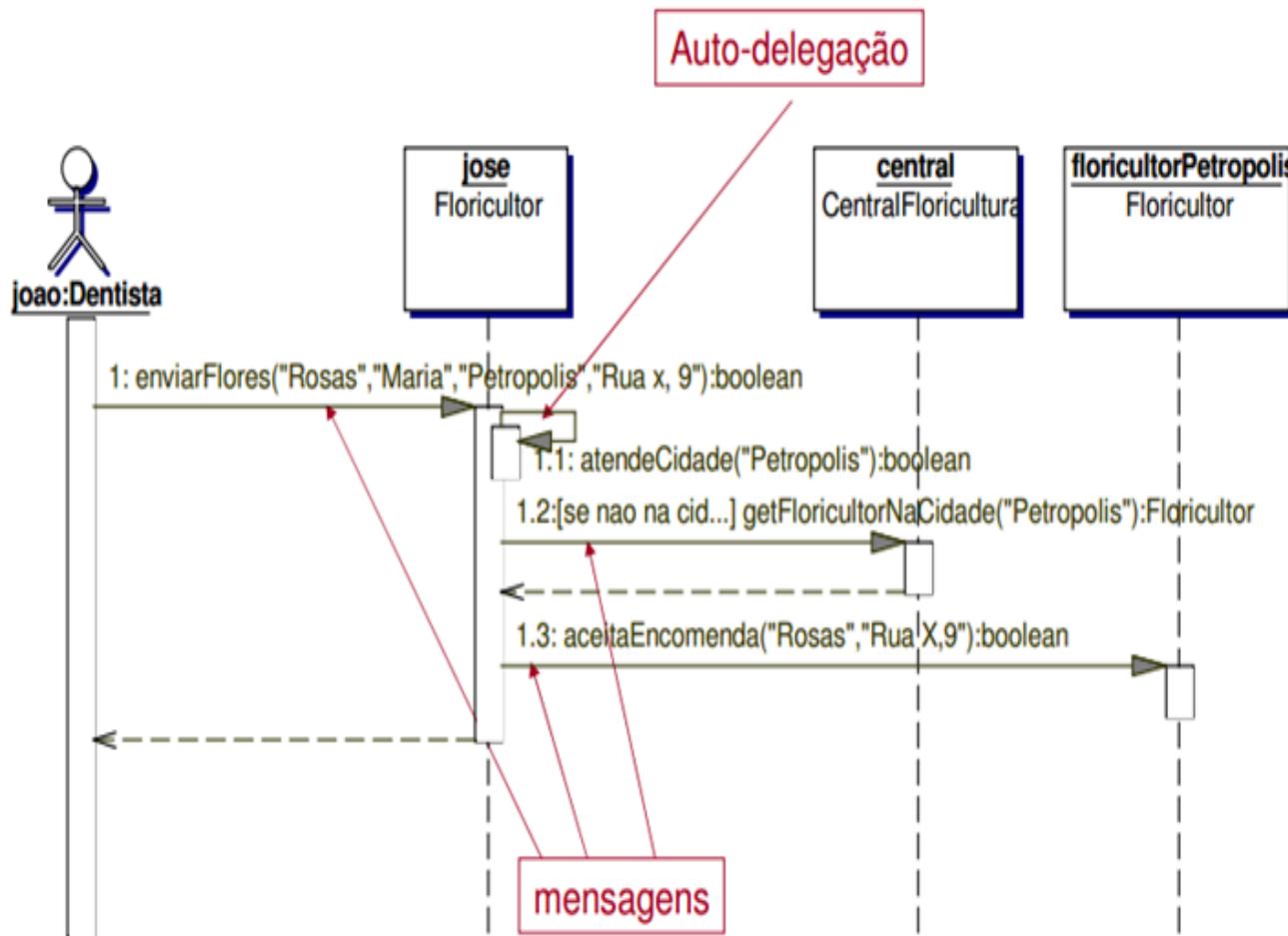
# Estrutura → Objetos Principais



# Exemplo 1 → Diagrama de Sequência



# Exemplo 2 → Diagrama de Sequência



# Obrigado!

- Francine - 10102189
- Kelly - 09106691
- Lyncon - 09211083
- Marina - 09107265
- Rodrigo - 10104385





# **DIAGRAMA DE ATIVIDADES**

**Engenharia de Software II**

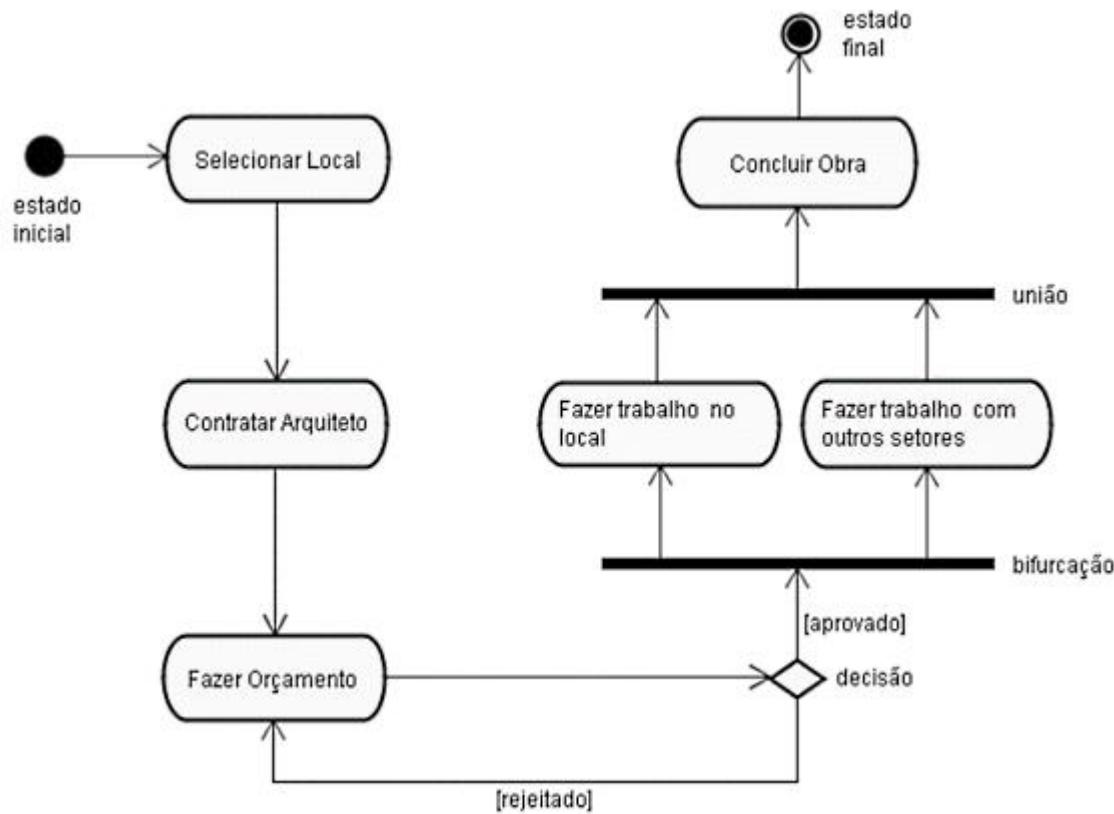
Átila Romão	10102407
Fabio Henrique	10103664
José Carlos	10100807
Lucas Akeda	10103693
Murillo Carvalho	10101630

## FUNÇÃO DO DIAGRAMA

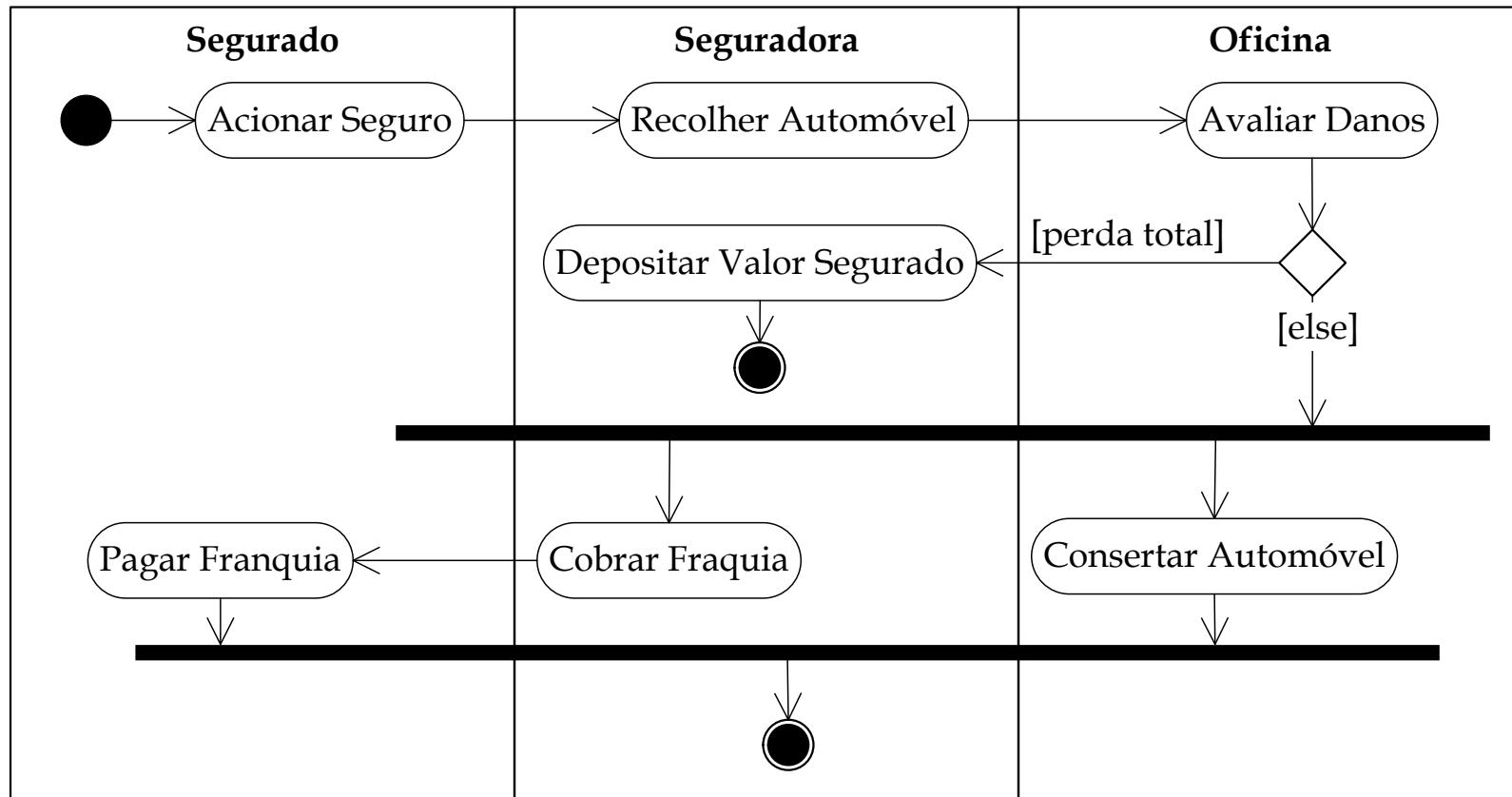
É utilizado para modelar o aspecto comportamental de processos.

Uma atividade é modelada como uma sequência estruturada de ações, controladas potencialmente por nós de decisão e sincronismo.

# EXEMPLO DIAGRAMA DE ATIVIDADES



# RAIA



## COMO DEVE SER USADO

- Para capturar as ações que serão executadas.
- Para mostrar como um grupo de ações relacionadas pode ser executado.
- Para mostrar como uma instância pode ser executada.
- Para mostrar como um negócio funciona.

## EXEMPLO PARA UTILIZAÇÃO

- Pode ser aplicado em qualquer processo dentro de uma empresa.
- Exemplo:
- Impressão de extrato bancário;
- E-commerce.

## CONCLUSÃO

- Fácil entendimento
- Esboça bem a comunicação e dependências das tarefas.
- Muito usada no mercado.



- Carlos Renan
- Diogo Carvalho
- Felipe Polizelo
- Joilson Lazaro
- Rafael Vecchi

# DIAGRAMA DE VISÃO GERAL DA INTERAÇÃO

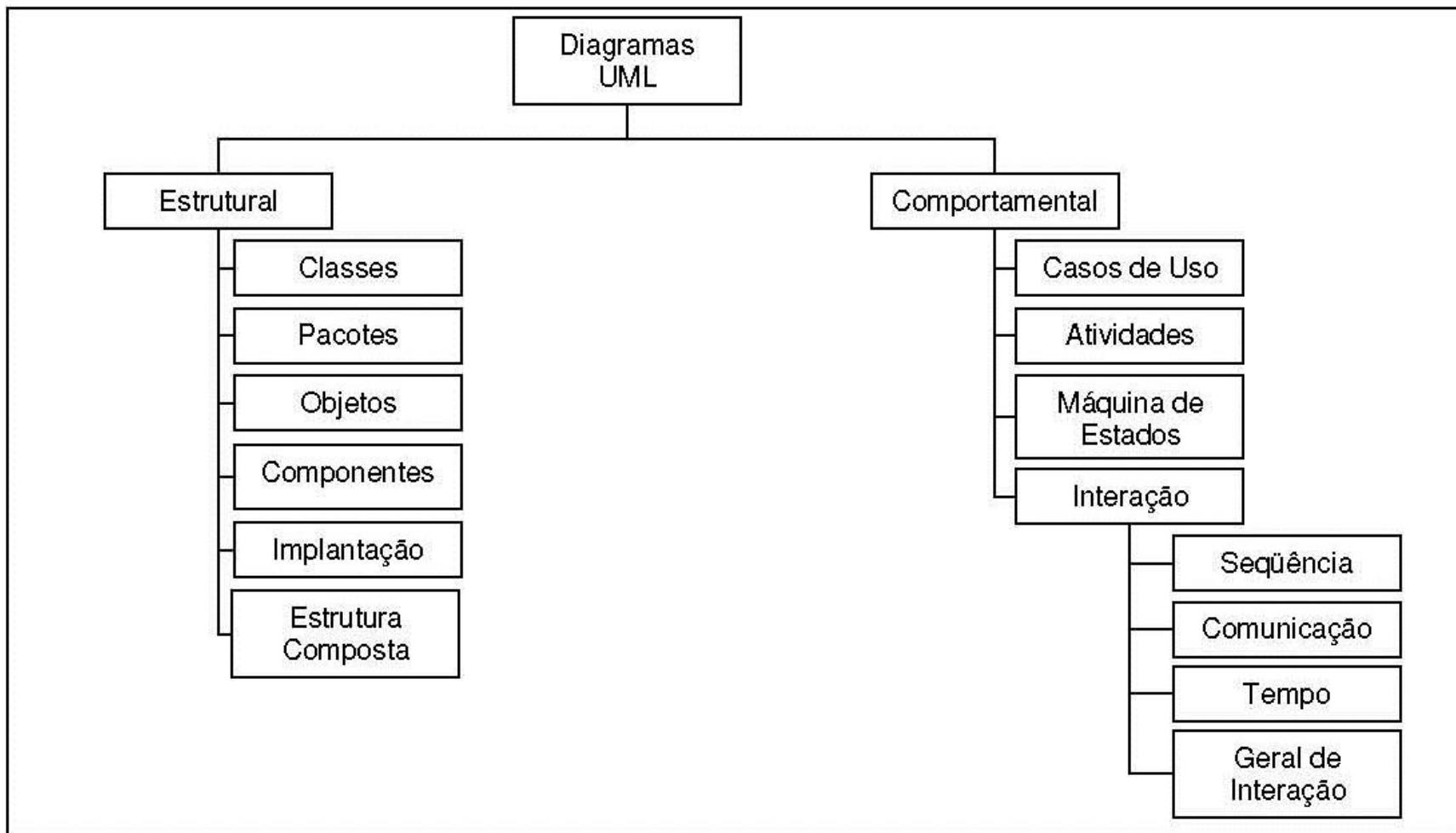
UML

# FUNÇÃO DO DIAGRAMA

- CONGREGA UMA DETERMINADA VISÃO QUE PODE ENGLOBAR VÁRIOS DIAGRAMAS.
- O DIAGRAMA DE VISÃO GERAL É SEMELHANTE AO DIAGRAMA DE ATIVIDADES, CONFORME VERSÃO ANTIGA DO UML.
- ELE APRESENTA AS EXECUÇÕES, AÇÕES E TRANSAÇÕES DA ATIVIDADE DESCrita.



# ESTRUTURA



# COMO DEVE SER USADO?

- PODE SER USADO PARA DEMONSTRAR DE FORMA SINTÉTICA SITUAÇÕES QUE TENHAM INTERAÇÕES COMPLEXAS.
  
- COM OS DIAGRAMAS SE TEM UMA FÁCIL VISUALIZAÇÃO DAS ATIVIDADES COMO UM TODO, ÓTIMO MATERIAL PARA SER USADA EM REUNIÕES.

# EXEMPLO DE UTILIZAÇÃO

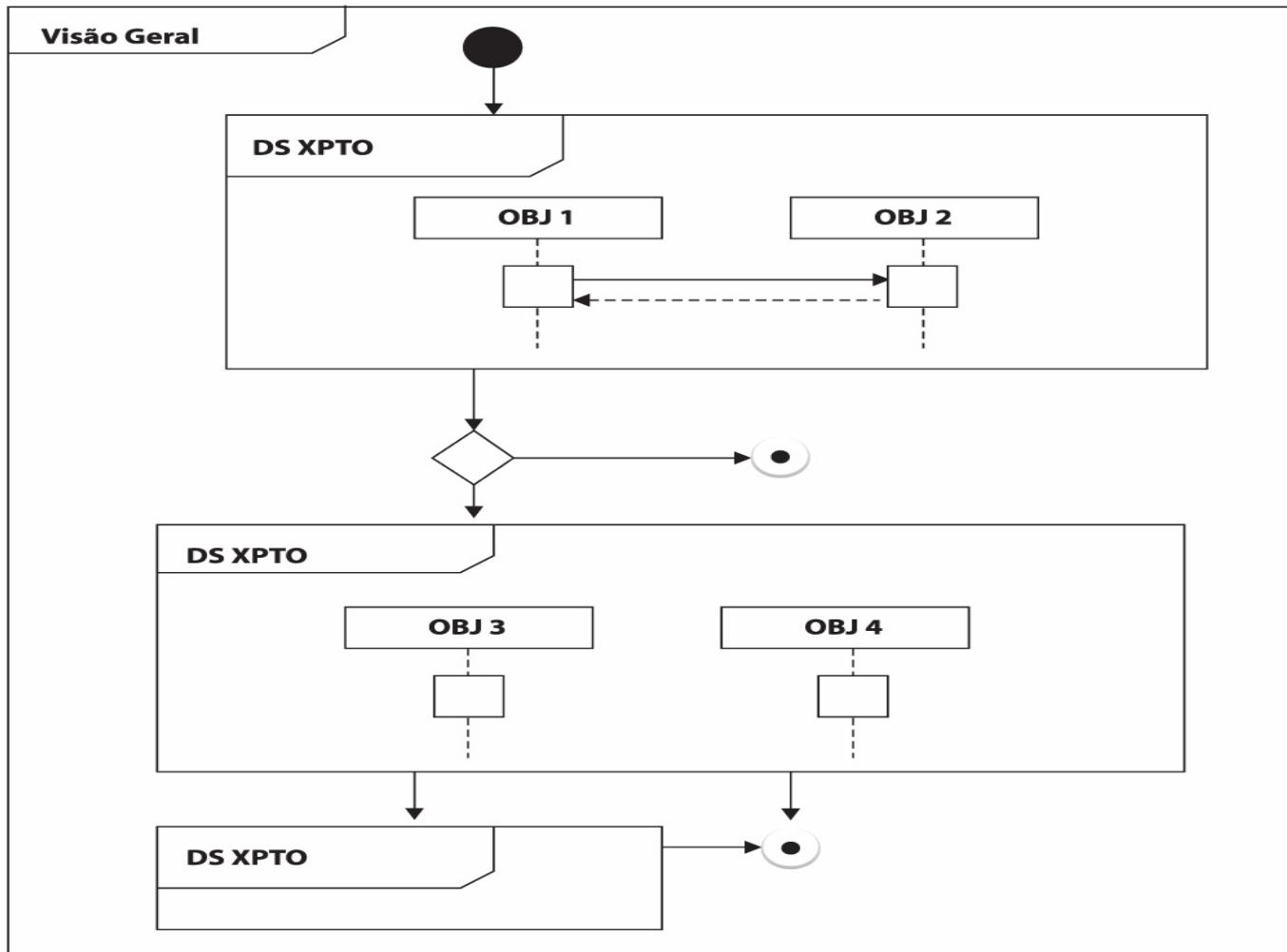
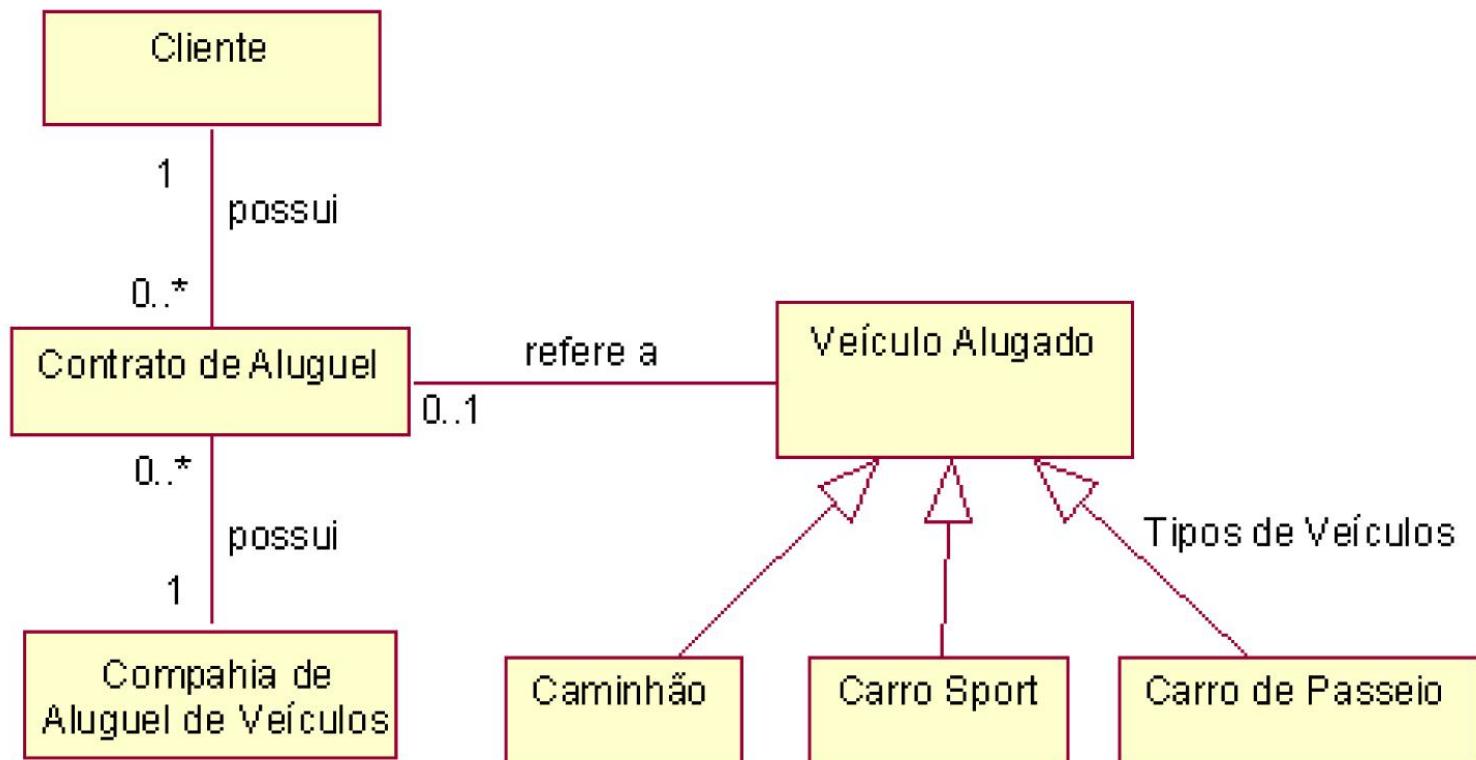


Diagrama de interação – Visão Geral.



# APLICANDO PROJETOS



# DIAGRAMA DE COMUNICAÇÃO

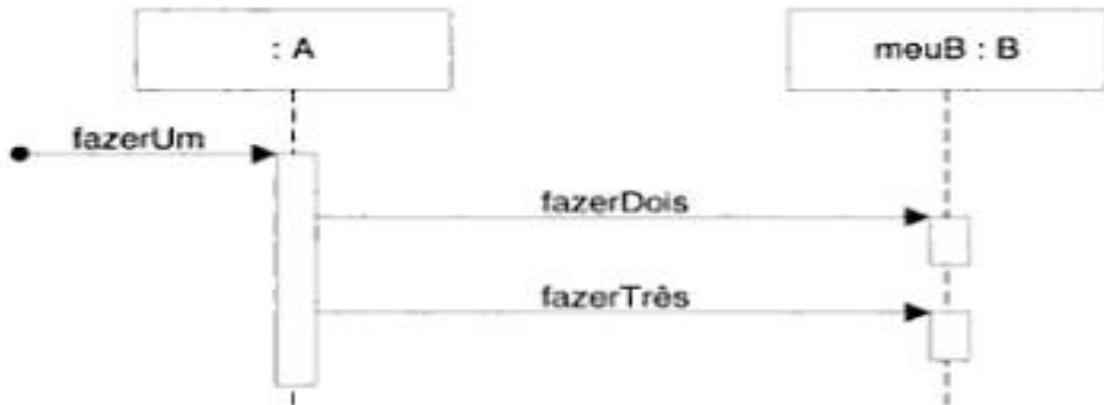
UML

# **FUNÇÃO DO DIAGRAMA**

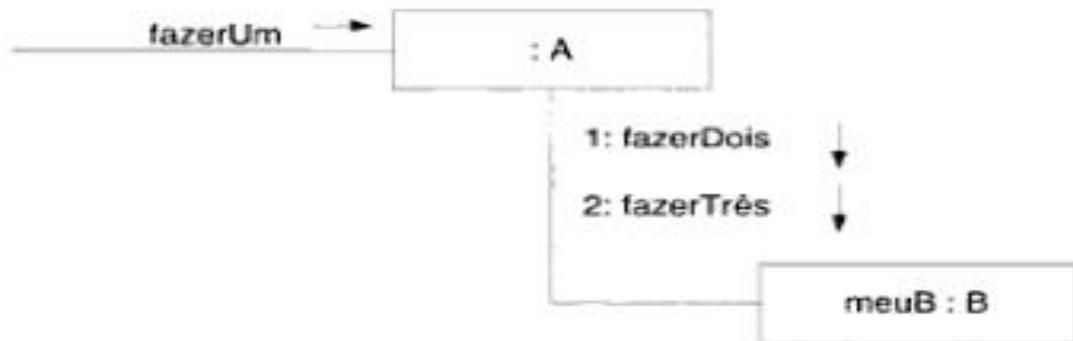
- SIMILAR COM O DIAGRAMA DE SEQUENCIA ELE TRAZ AS SEQUENCIAS DAS MENSAGENS.
  
- IDENTIFICAÇÃO É FEITA POR MEIO DE SEQUENCIA NUMÉRICAS.
  
- DIAGRAMAS DE COMUNICAÇÃO ILUSTRAM AS INTERAÇÕES ENTRE OBJETOS.



# ESTRUTURA



**Figura 15.1** Diagrama de seqüência.



**Figura 15.2** Diagrama de comunicação.

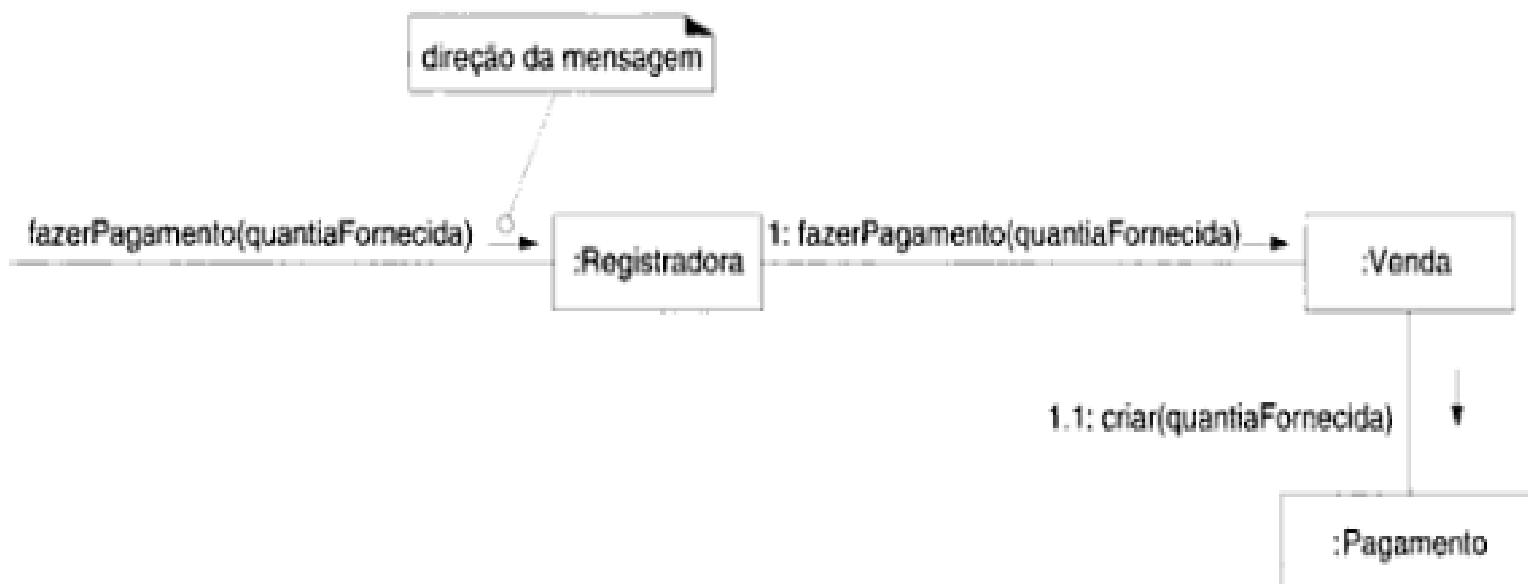
# COMO DEVE SER USADO ?

- USADOS EM GRAFO OU REDE
- PODE SER ALTERADO A QUALQUER MOMENTO SEM AFETAR OS PROCESSOS.
- DIAGRAMA DE COMUNICAÇÃO TEM VANTAGEM, SOBRE O DIAGRAMA DE SEQUENCIA, DE PERMITIR A EXPANSÃO VERTICAL PARA NOVOS OBJETOS.

# EXEMPLO DE UTILIZAÇÃO

*Exemplo de diagrama de comunicação: fazerPagamento*

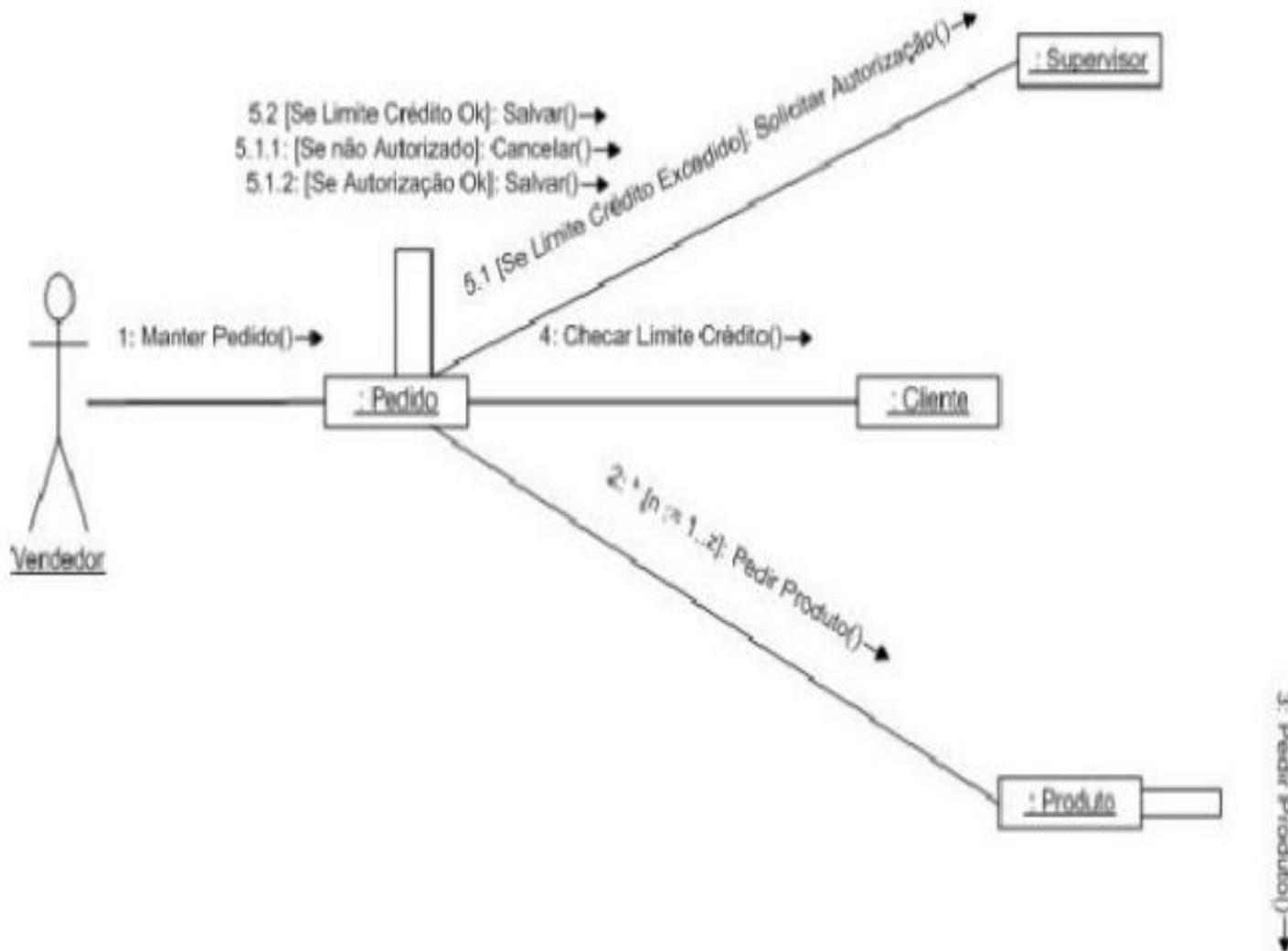
O diagrama de comunicação mostrado na Figura 15.4 tem a mesma finalidade que o diagrama de seqüência anterior.



**Figura 15.4** Diagrama de comunicação.



# APLICANDO PROJETOS



# **Diagrama de Máquina de Estados**

# Grupo

- 10100814 GILVAN VELAMES DA SILVA
- 10100845 RICARDO NOGUEIRA MENTA DE CARVALHO
- 10100843 RODINEI MIGUEL DOS SANTOS
- 10101211 RODRIGO GOERING OLIVEIRA
- 10101627 PALOMA HARUMI DA SILVA NISHIMURA

# **Função do diagrama**

- Representar o estado ou situação em que um objeto pode se encontrar no decorrer da execução de processos de um sistema. Com isso, o objeto pode passar de um estado inicial para um estado final através de uma transição.

# Estrutura

- Estado (Inicial/Final)
- Estados
- Eventos
- Barra Sincronização
- Guarda
- Ações
- Transição
- Condição



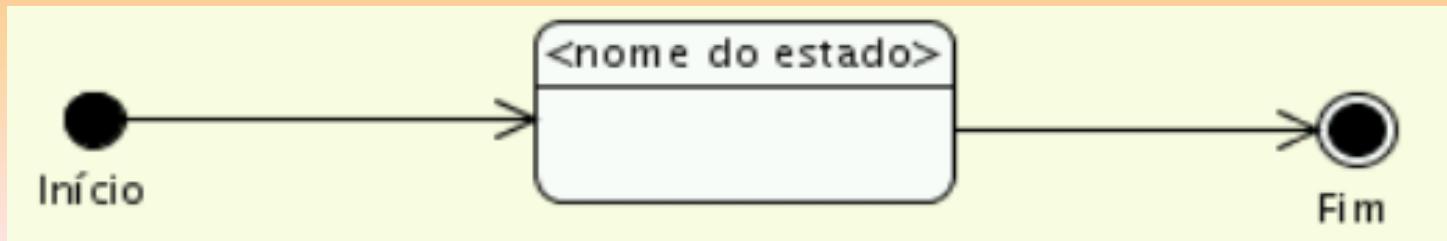
# Estado Inicial e Final

- Inicial: é utilizado para representar o início da modelagem de estados de um objeto, representado por um círculo preenchido.
- Final: é utilizado para representar o fim dos estados modelados, representado por um círculo preenchido envolvido por outro círculo não preenchido.



# Estados

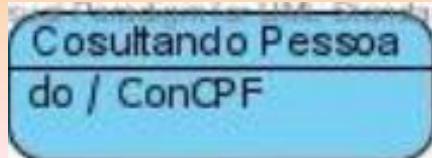
- Um objeto pode passar por diversos estados dentro de um processo.
- Um estado pode demonstrar:
- a espera pela ocorrência de um evento.
- a reação a um estímulo.
- a execução de alguma atividade.
- a satisfação de alguma condição.
- Ex:



# Estados

- Segunda divisão do retângulo pode armazenar três cláusulas:
  - Entry: representa as ações realizadas no momento em que o objeto assume o Estado em questão;
  - Exit: identifica as ações executadas antes do objeto mudar de Estado;
  - Do: ilustra as atividades executadas enquanto o objeto se encontra em um determinado Estado.

Ex:

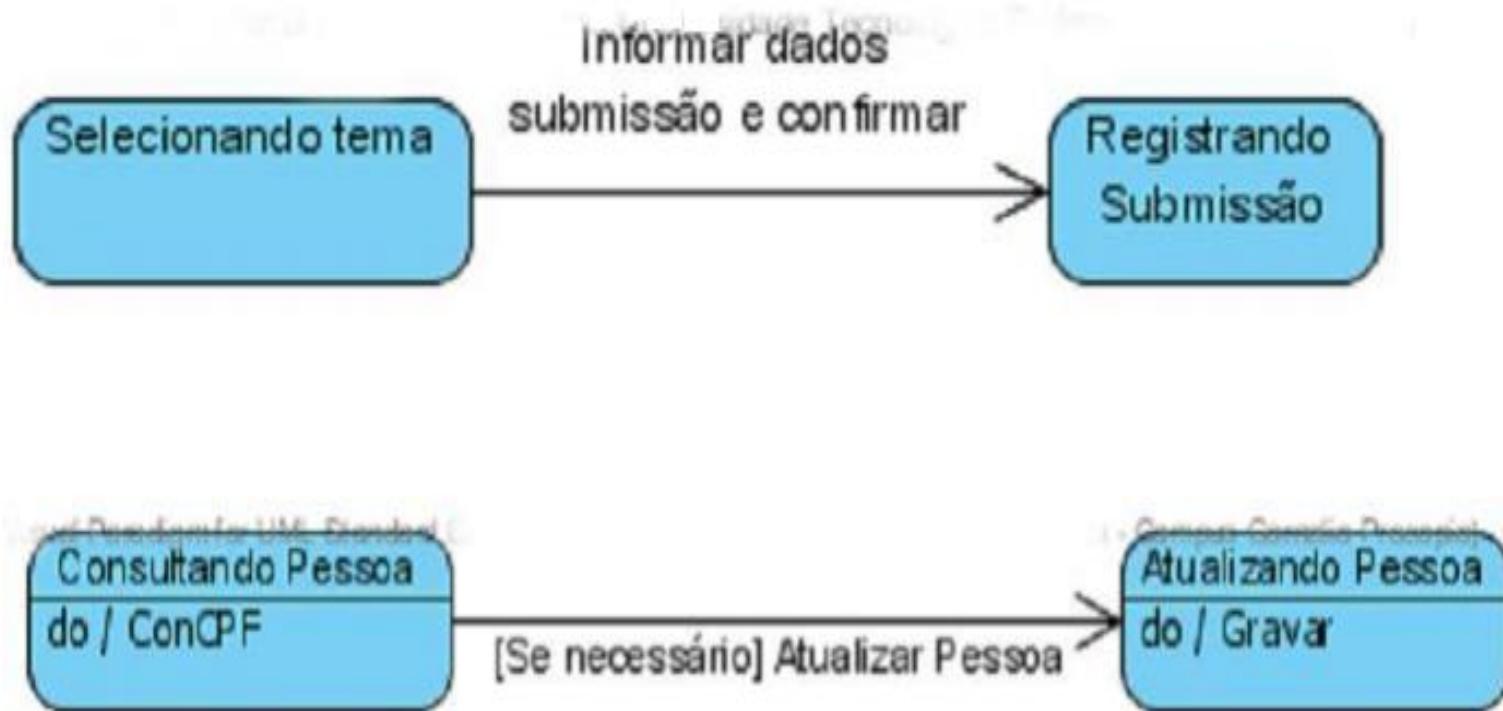


# Transições

- Uma transição representa um evento que causa uma mudança no Estado de um objeto, gerando um novo Estado (evento de ativação).
- Transições podem possuir condições de guarda e descrições, se isto for considerado necessário.
- Flecha que liga dois estados:  
**estado origem -> estado destino**
  - Evento - Provoca a transição de estado.
  - Guarda - Condição que restringe a ocorrência da transição.
  - Ação - Operação decorrente da transição de estado.

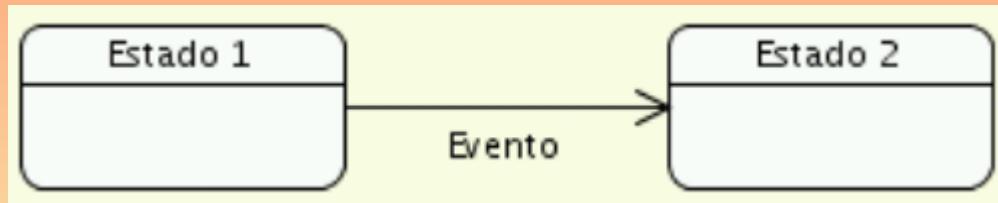
# Transições

- Ex:



# Eventos

- Ocorrência que deve ser reconhecida e gerar uma reação pelo sistema em estudo.
- A ocorrência de um evento provoca a transição entre estados de instâncias de alguma classe pertencente ao sistema.

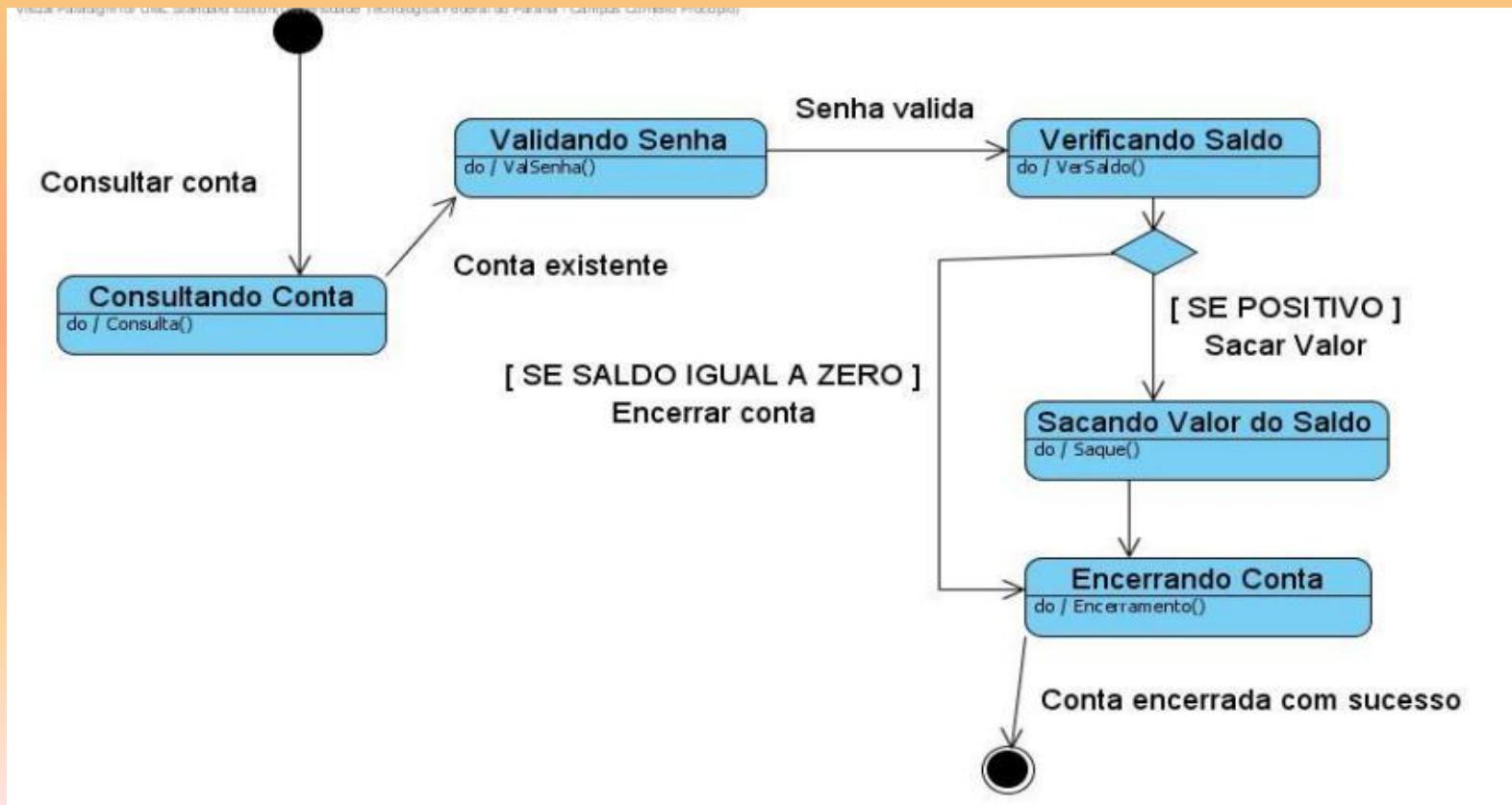


# **Guarda e Ações**

- A guarda é uma expressão que resulta em um valor booleano.
- É representada entre colchetes [expressão].
- A ação é representada pela invocação de uma operação:  
**operacao(parametros)**
- É precedida por uma barra inclinada ou um acento circunflexo.

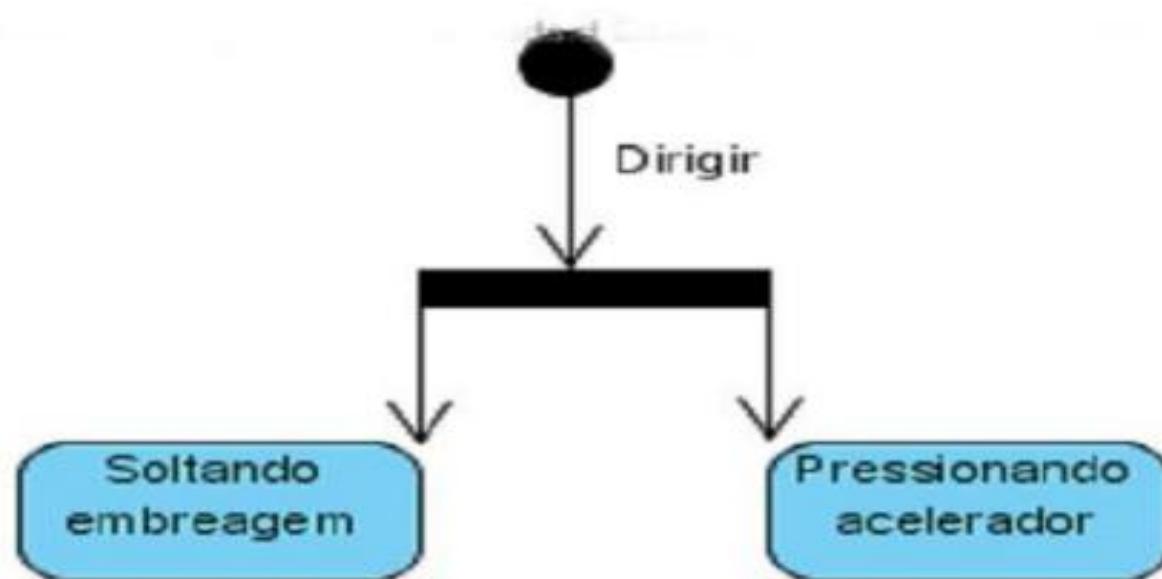
# Condicão

- Representa um ponto na transição de estados de um objeto em que deve ser tomada uma decisão, a partir da qual um determinado estado será ou não gerado.



# Barra Sincronização

- Utilizada quando da ocorrências de estados paralelos, causados por transições concorrentes.
- Ex:



# **Utilização:**

- Máquinas de estado comportamentais.
- Máquinas de estado para protocolos.

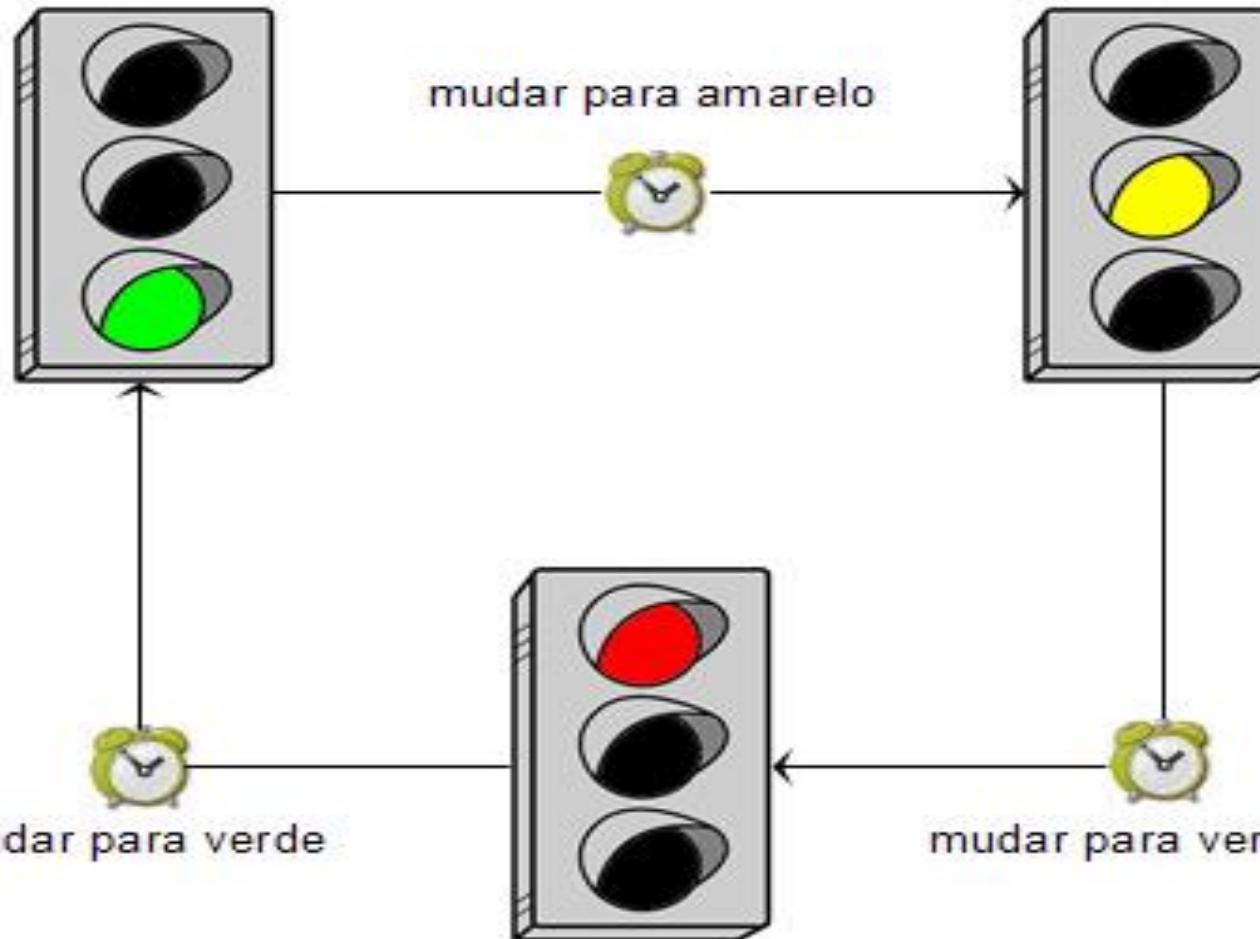
# **Máquinas de estado Comportamentais:**

- Podem ser utilizadas para especificar o comportamento de vários tipos de elementos. Por exemplo, podem ser utilizadas para modelar o comportamento de entidades individuais (objetos), por meio da modificação dos valores de seus atributos. O formalismo de máquina de estados neste caso é uma variante orientada a objetos.

# **Máquinas de estado para protocolos:**

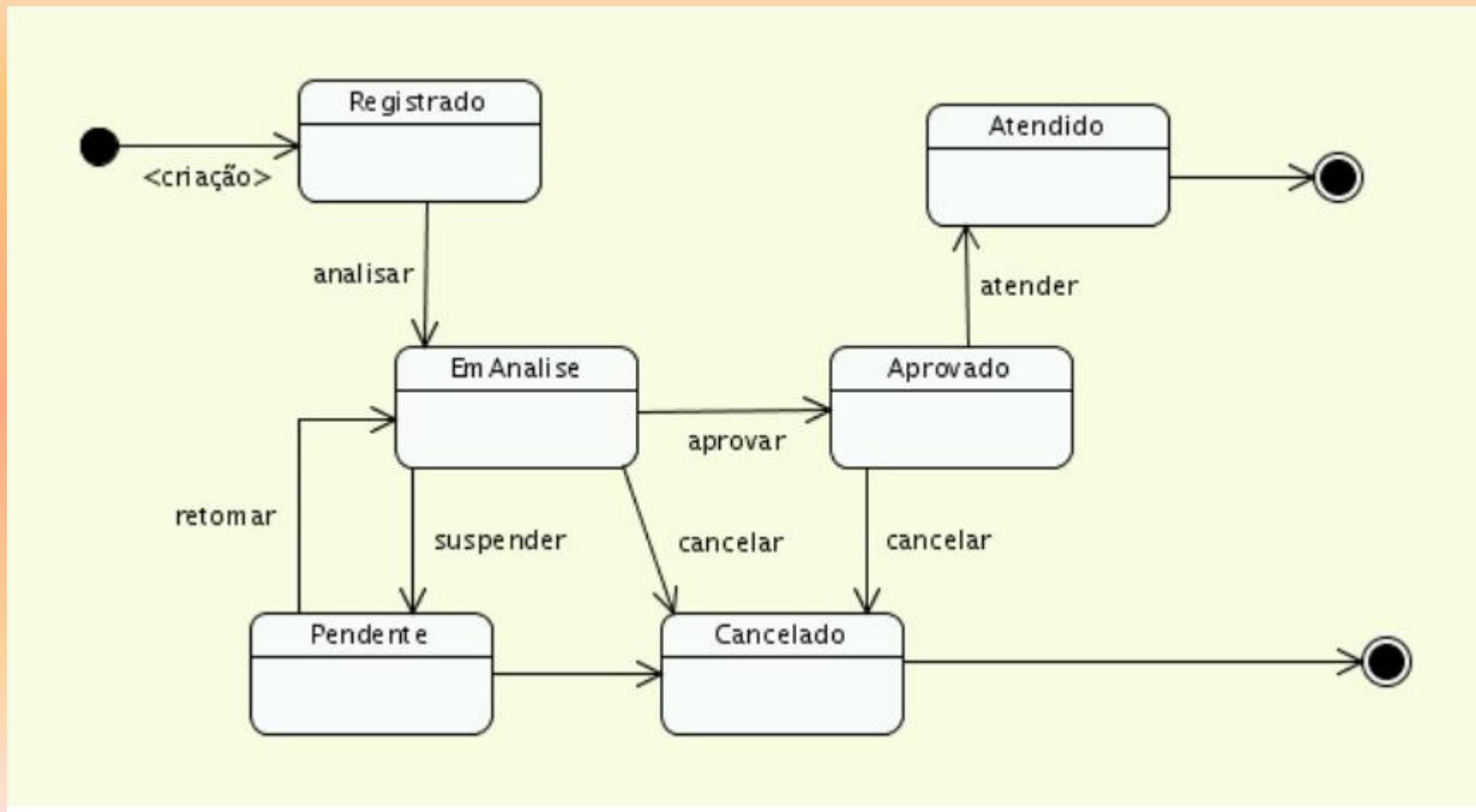
- Máquinas de estado para representar protocolos expressam as transições legais que um objeto pode desenvolver. Com seu uso, pode-se definir o ciclo de vida de objetos, ou uma determinada ordem na invocação de suas operações. Para este tipo de máquina de estado, interfaces e portas podem estar associados.

# Exemplo de Utilização:



# Exemplo de Utilização:

- Pedido de Compra:



# Exemplo de Utilização:

## Diagramas de Estado



Uma lâmpada: que evolui entre os estados “acesa” e “apagada”, conforme se liga e desliga um interruptor

# **Diagramas de implementação**

Cesar Augusto Romão – 10103024

Lucas de Melo Favaretto – 10102153

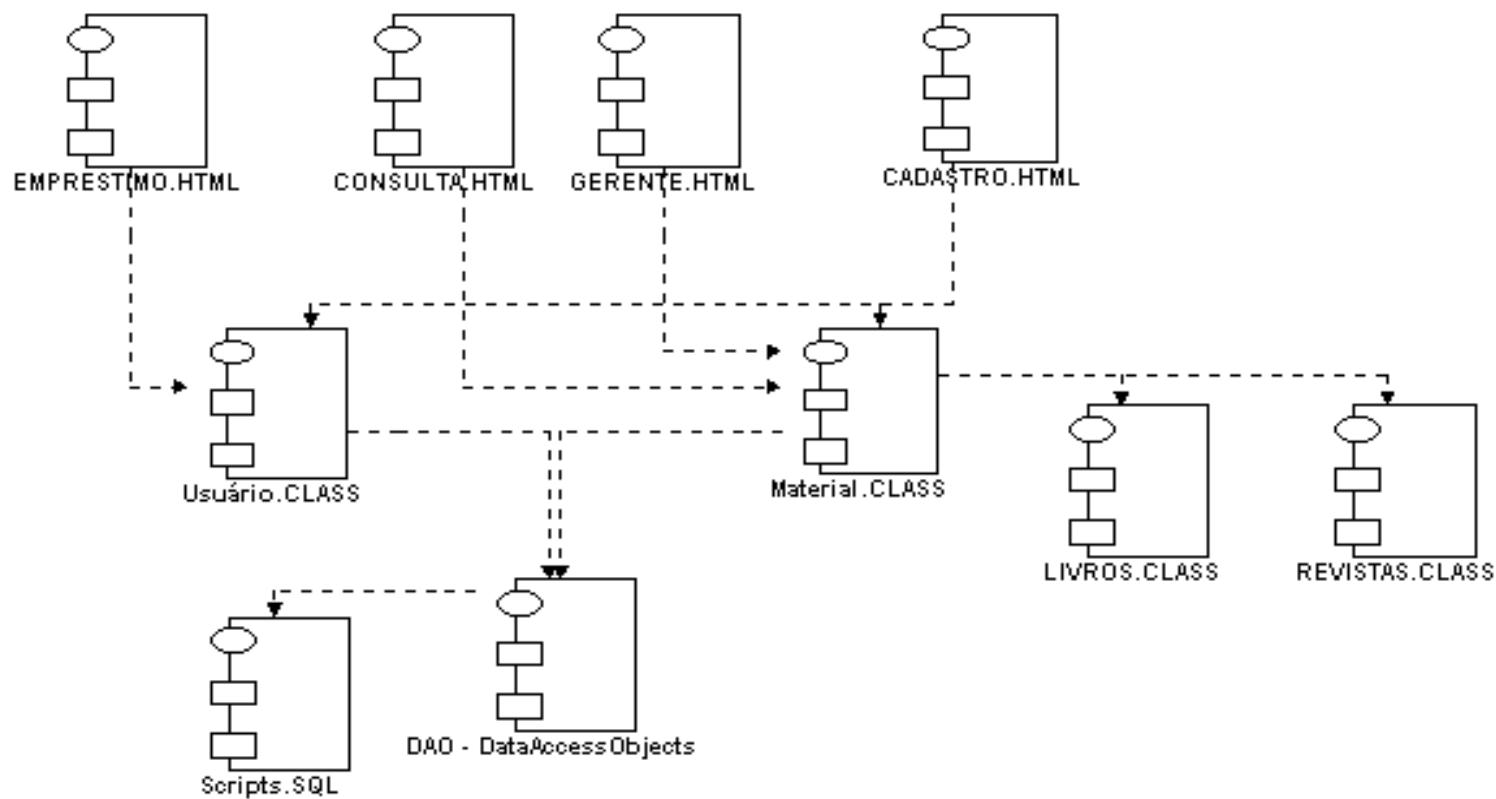
Thiago Pereira de Campos - 10100804

# Definição

- São utilizados como uma representação gráfica da visão estática de um sistema em funcionamento. Um único diagrama não deve capturar tudo sobre a visão de implantação do sistema. Neste estágio é onde organizamos o código fonte para ambiente de trabalho e realizamos o executável (ambiente de instalação)
- Foco na comunicação de um aspecto na visão estática de implantação do sistema e deve somente conter elementos essenciais à compreensão desse aspecto. Deve-se fornecer detalhes consistentes com seu nível de abstração além de informar bem ao leitor sobre a semântica importante.
- Podemos dividir o Diagrama de Implementação em dois, o Diagrama de Componentes e o Diagrama de Distribuição:

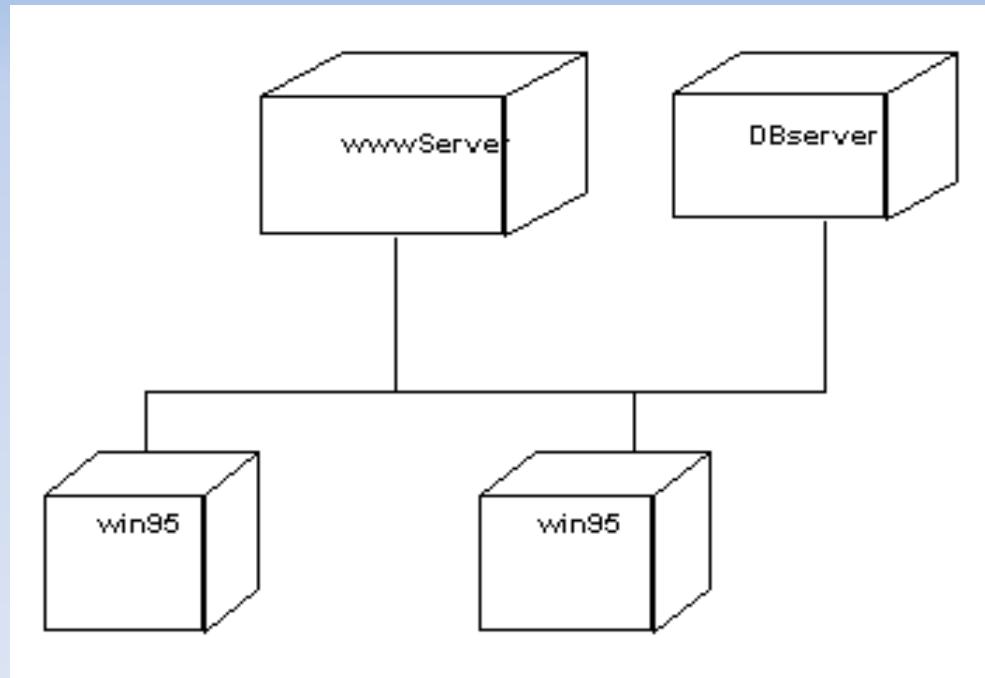
# Diagrama de Componentes

- Representa a parte física do sistema, exibindo os componentes do sistema e a dependência entre eles. Cada componente é composto por uma ou mais classes. Propõe uma visão estática do sistema. O diagrama de componentes é utilizado para modelar a arquitetura e solução de implementação do sistema. Através do diagrama de componentes pode-se gerar pseudo-código em linguagens de programação. No diagrama de componentes também é possível mostrar a configuração de um sistema de software mostrando, graficamente, a dependência entre os diversos arquivos que compõem o sistema.



# Diagrama de Distribuição (Implantação)

- Os diagramas de distribuição mostram a distribuição de hardware do sistema, identificando os servidores como nós do diagrama e a rede que relaciona os nós. Os componentes de software vão estar mapeados nestes nós.

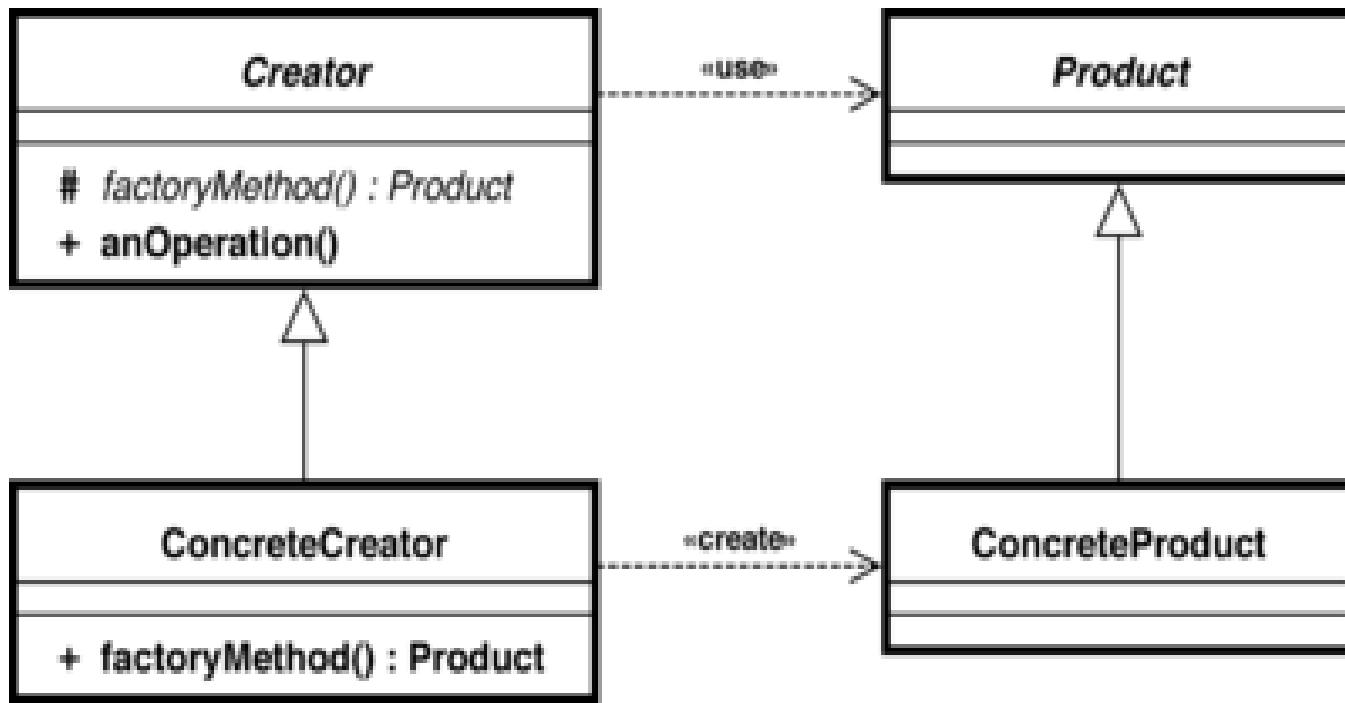


# Estrutura

- Abaixo estão alguns itens importantes para uma definição da estrutura:
- Devemos nomeá-lo de forma capaz de se comunicar com seu propósito
- Organize seus elementos de modo que os itens que são semanticamente afins fiquem próximos fisicamente.
- Usar notas e cores como indicações visuais para destacar itens importantes.
- Defina um conjunto de ícones comuns e utilize de forma consistente, evitando assim qualquer dificuldade no entendimento.

# Estrutura no padrão “Factory method”

- O padrão Factory Method, da forma como foi descrito no livro Design Patterns: Elements of Reusable Object-Oriented Software, contém os seguintes elementos:
- Creator — declara o factory method (método de fabricação) que retorna o objeto da classe Product (produto). Este elemento também pode definir uma implementação básica que retorna um objeto de uma classe ConcreteProduct (produto concreto) básica;
- ConcreteCreator — sobrescreve o factory method e retorna um objeto da classe ConcreteProduct;
- Product — define uma interface para os objectos criados pelo factory method;
- ConcreteProduct — uma implementação para a interface Product.



# Utilização

- Este padrão pode ser utilizado na construção de um framework que suporta aplicações que apresentam múltiplos documentos ao usuário. Normalmente este tipo de aplicação manipula um número variável de formatos de documento e, por isso, este framework deve ser flexível o bastante para suportar qualquer formato. Uma solução para este problema poderia disponibilizar, no framework, o código para alguns dos formatos mais utilizados. O padrão Factory Method propõe uma solução que deixa para o cliente (a implementação da aplicação) a tarefa de suportar os formatos necessários e para o framework o papel de definição de uma abstração que oferece uma interface única para criação de documentos. Este framework seria baseado em duas classes abstratas, que representam a Aplicação e o Documento.

# Conclusão

- Uma visão clara do sistema como um todo
- Avaliar se os processos estão organizados de forma mais efetiva
- Definir estratégias ou otimizar as já existentes dentro deste diagrama.

# Referencial teórico

- <http://www.klebermota.eti.br/2011/11/22/unified-modeling-language-a-linguagem-unificada-de-modelagem/>

alessandro.almeida@uol.com.br  
[www.slideshare.net/alessandroalmeida](http://www.slideshare.net/alessandroalmeida)

**Muito obrigado!**