Slide do
Prof. MSc. Raoni Oliveira

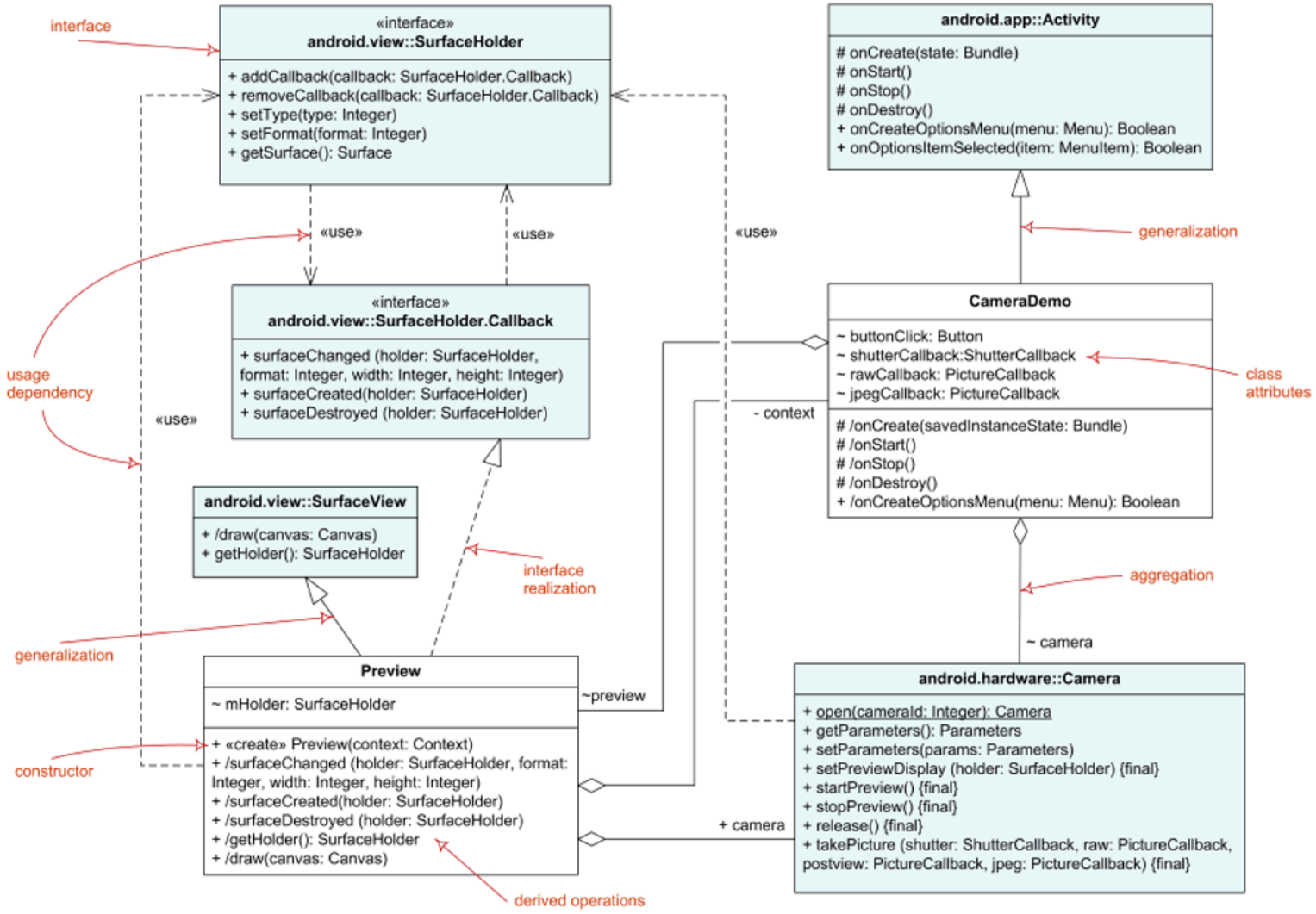
DIAGRAMAS UML DE CLASSES

Prof. MSc.
Danilo Farias

UML – DIAGRAMA DE CLASSES

1. Introdução
2. Orientação a objetos
3. Paradigma
4. Classes
5. Objetos
6. Atributos
7. Operações
8. Visibilidade
9. Relacionamentos

INTRODUÇÃO


interface
«interface»
android.view::SurfaceHolder
+ addCallback(callback: SurfaceHolder.Callback)
+ removeCallback(callback: SurfaceHolder.Callback)
+ setType(type: Integer)
+ setFormat(format: Integer)
+ getSurface(): Surface
android.app::Activity
onCreate(state: Bundle)
onStart()
onStop()
onDestroy()
+ onCreateOptionsMenu(menu: Menu): Boolean
+ onOptionsItemSelected(item: MenuItem): Boolean
«use»
«use»
«use»
generalization
«interface»
android.view::SurfaceHolder.Callback
+ surfaceChanged (holder: SurfaceHolder, format: Integer, width: Integer, height: Integer)
+ surfaceCreated(holder: SurfaceHolder)
+ surfaceDestroyed (holder: SurfaceHolder)
CameraDemo
~ buttonClick: Button
~ shutterCallback:ShutterCallback
~ rawCallback: PictureCallback
~ jpegCallback: PictureCallback
# /onCreate(savedInstanceState: Bundle)
# /onStart()
# /onStop()
# /onDestroy()
+ /onCreateOptionsMenu(menu: Menu): Boolean
class attributes
usage dependency
«use»
- context
android.view::SurfaceView
+ /draw(canvas: Canvas)
+ getHolder(): SurfaceHolder
interface realization
aggregation
~ camera
generalization
Preview
~ mHolder: SurfaceHolder
~preview
+ «create» Preview(context: Context)
+ /surfaceChanged (holder: SurfaceHolder, format: Integer, width: Integer, height: Integer)
+ /surfaceCreated(holder: SurfaceHolder)
+ /surfaceDestroyed (holder: SurfaceHolder)
+ /getHolder(): SurfaceHolder
+ /draw(canvas: Canvas)
android.hardware::Camera
+ open(cameraId: Integer): Camera
+ getParameters(): Parameters
+ setParameters(params: Parameters)
+ setPreviewDisplay (holder: SurfaceHolder) {final}
+ startPreview() {final}
+ stopPreview() {final}
+ release() {final}
+ takePicture (shutter: ShutterCallback, raw: PictureCallback, postview: PictureCallback, jpeg: PictureCallback) {final}
constructor
+ camera
derived operations

INTRODUÇÃO

- **Diagrama de Classes** descreve as **classes** e **interfaces** presentes no sistema, suas **características, restrições** e os vários **tipos de relacionamentos** estáticos entre seus objetos.

Representam-se também as **propriedades** e as **operações** de uma Classe, assim como as **restrições** que se aplicam maneira como os objetos estão conectados.

KEEP
CALM
AND
OBJECT-ORIENTED
PROGRAM

ORIENTAÇÃO A OBJETO

Toda a concepção de Allan Kay sobre o papel do computador na sociedade foi alterada quando ele viu Seymour Papert e seus colegas **ensinando crianças a programar** em Logo.

ORIENTAÇÃO A OBJETO

Então, Kay pensou em como construir um sistema de agentes autônomos que interagissem entre si, estabelecendo os seguintes princípios da orientação a objetos:

1. Qualquer coisa é um objeto.
2. Objetos realizam tarefas através da requisição de serviços.
3. Cada objeto pertence a uma determinada classe.
4. Uma classe agrupa objetos similares.
5. Um classe possui comportamentos associados ao objeto.
6. Classes são organizadas em hierarquias.

ORIENTAÇÃO A OBJETO

A orientação a objetos é uma **ponte entre o mundo real e virtual**, a partir desta é possível transcrever a forma como enxergamos os elementos do mundo real em código fonte, a fim de nos possibilitar a construção de sistemas complexos baseados em objetos.

ORIENTAÇÃO A OBJETO

Vejamos como descrever um objeto “Pessoa”.

As perguntas a serem feitas são: “O que é uma pessoa?”,”Quais são as características de uma pessoa?”,”Como uma pessoa se comporta?”.

O que é uma pessoa?

Quais são as características de uma pessoa?

Como uma pessoa se comporta?

ORIENTAÇÃO A OBJETO

Vejamos como descrever um objeto “Pessoa”.

As perguntas a serem feitas são: “O que é uma pessoa?”,”Quais são as características de uma pessoa?”,”Como uma pessoa se comporta?”.

O que é uma pessoa?

Resposta: A pessoa é um ser do mundo real que interage com toda a natureza.

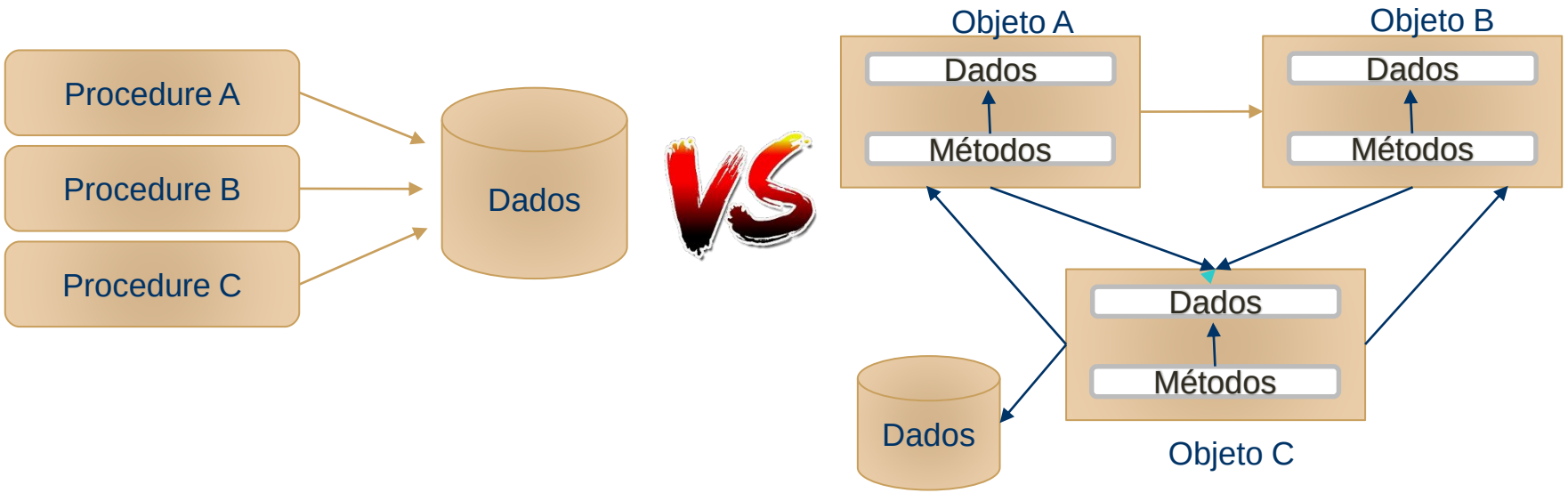
Quais são as características de uma pessoa?

Resposta: Uma pessoa possui: nome, olhos, boca, braços, pernas, cabelos e etc.

Como uma pessoa se comporta?

Resposta: Uma pessoa corre, anda, fala, pula, come e etc.

PARADIGMA TRADICIONAL X ORIENTADO A OBJETOS



O QUE SÃO CLASSES?

Classe é uma **estrutura classificadora** que **abstrai** um **conjunto de** objetos **que compartilham características, restrições** e **semânticas similares**.

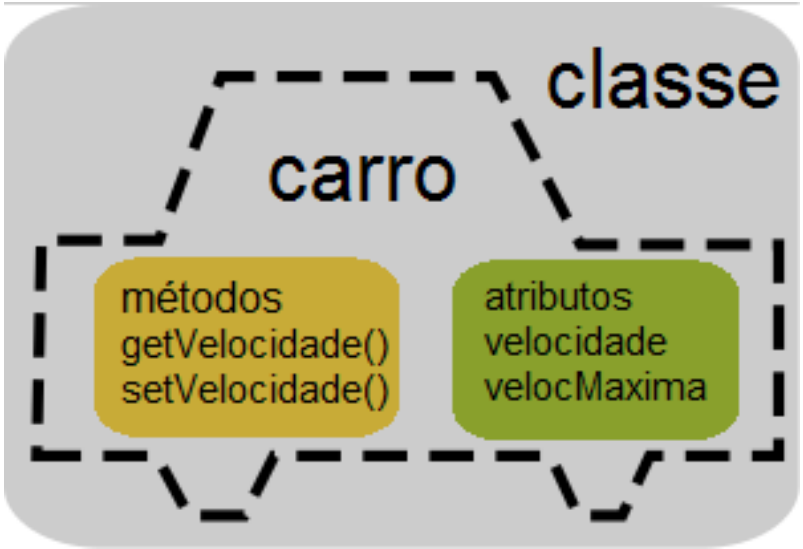
Ela define, também, o comportamento de seus objetos através de métodos e o estado por meio de atributos.

O QUE SÃO CLASSES?

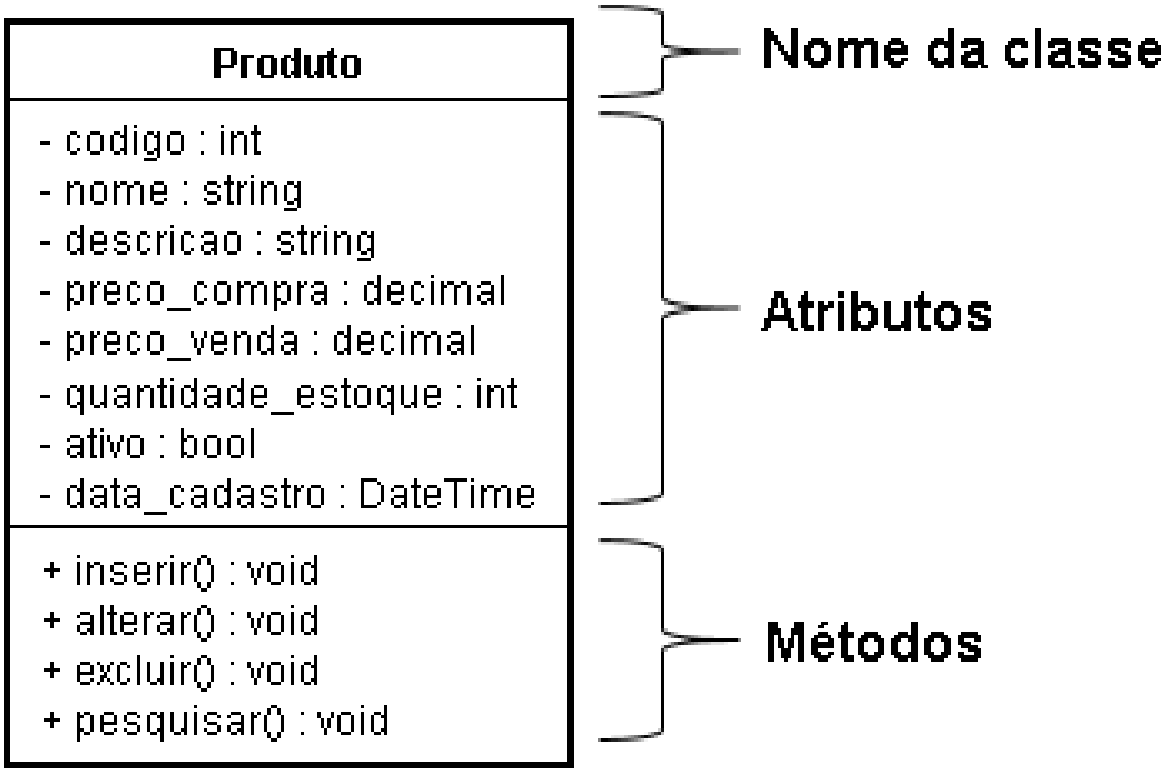
A classe é a descrição dos atributos (características) e ações comuns a um grupo de objetos (reais ou abstratos).

Podemos dizer que é um modelo a partir do qual objetos são construídos.

O QUE SÃO CLASSES?



O QUE SÃO CLASSES?



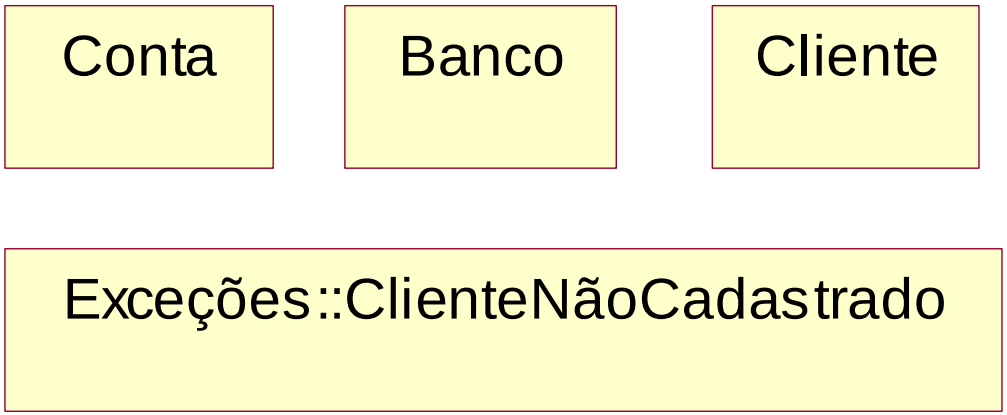
O QUE SÃO CLASSES?


```
public class Produto
{
    int codigo;
    string nome;
    string descrição;
...
    public void inserir()
    {
        ... // Código para inserir
    }
...
}
```

NOMES

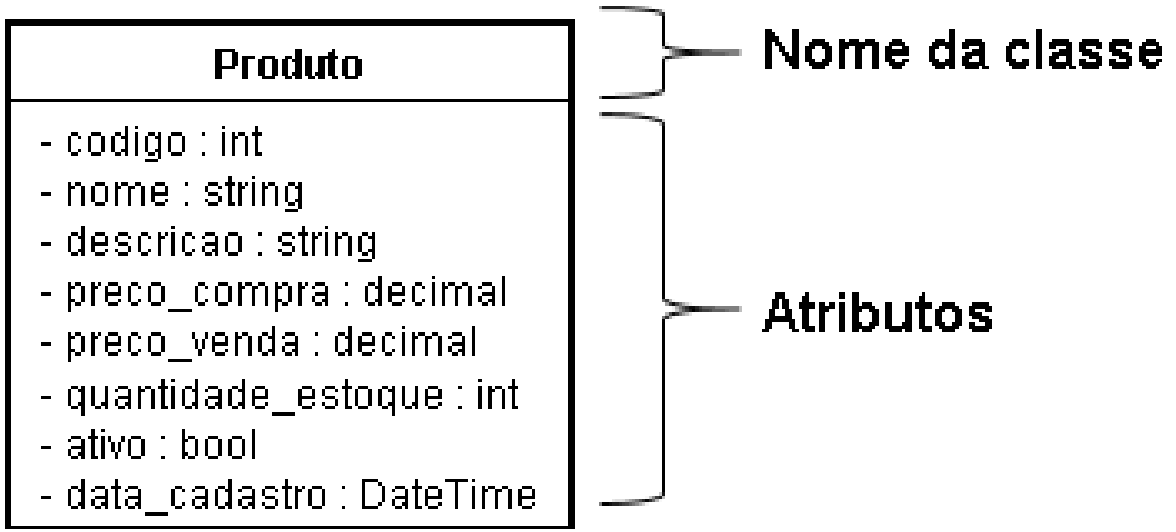
Toda classe deve ter um nome que faça a distinção entre as outras classes.

Um nome pode ser simples (apenas o nome), ou pode ser precedido pelo nome do pacote em que a classe está contida:



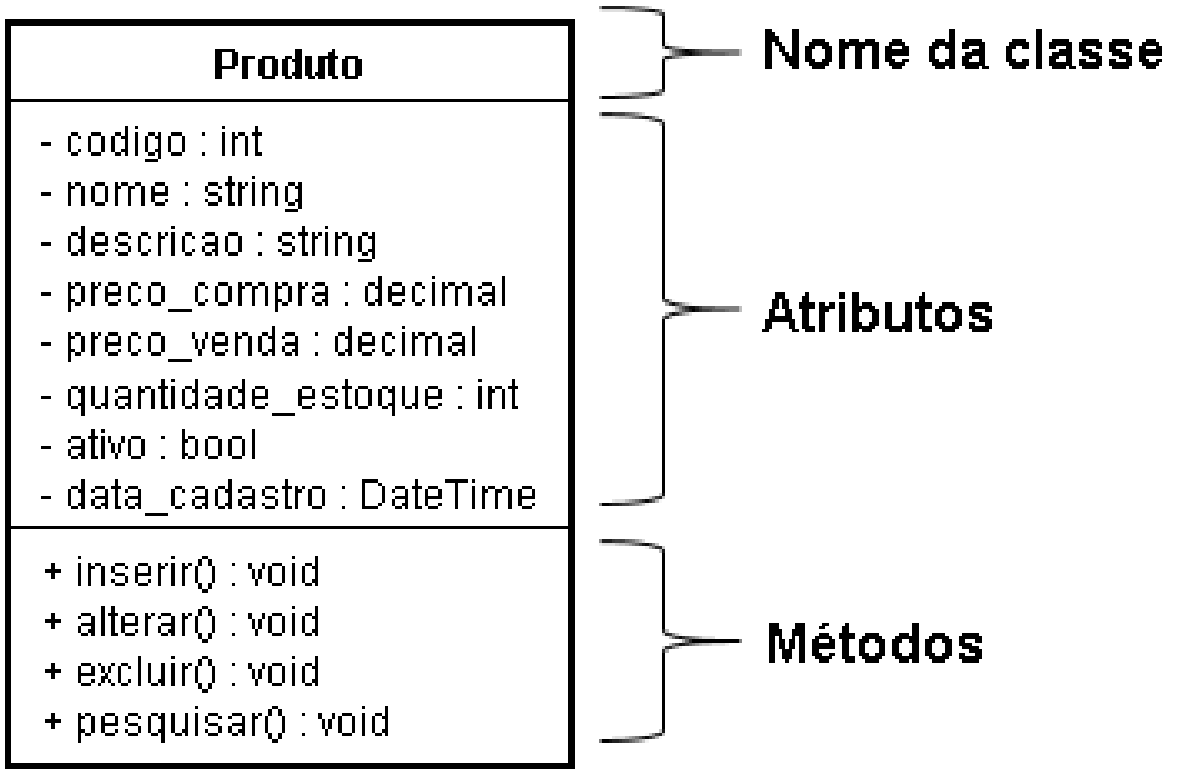
O QUE SÃO CLASSES?

```
public class Produto
{
    int codigo;
    string nome;
    string descrição;
...
    public void inserir()
    {
        ... // Código para inserir
    }
...
}
```

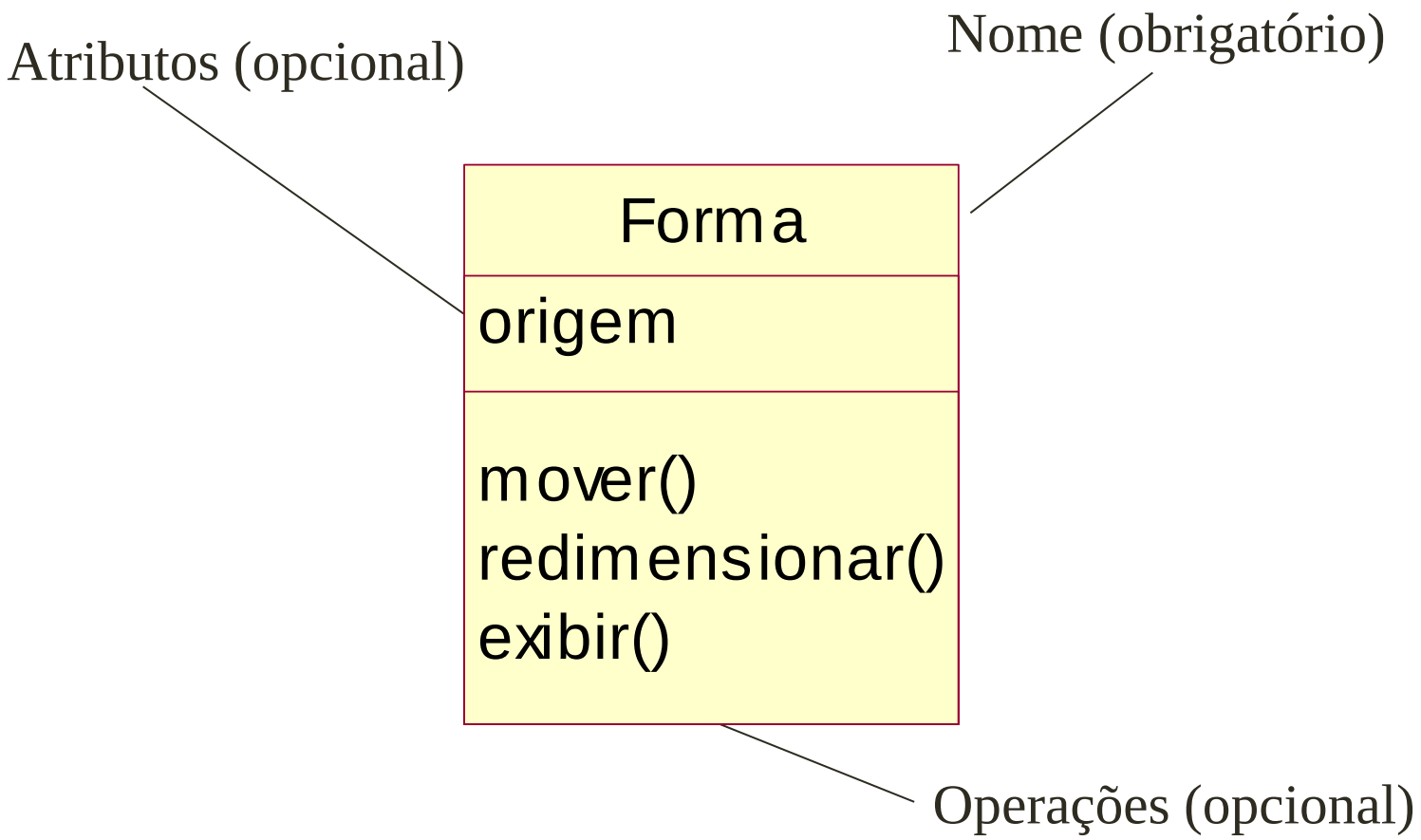


O QUE SÃO CLASSES?

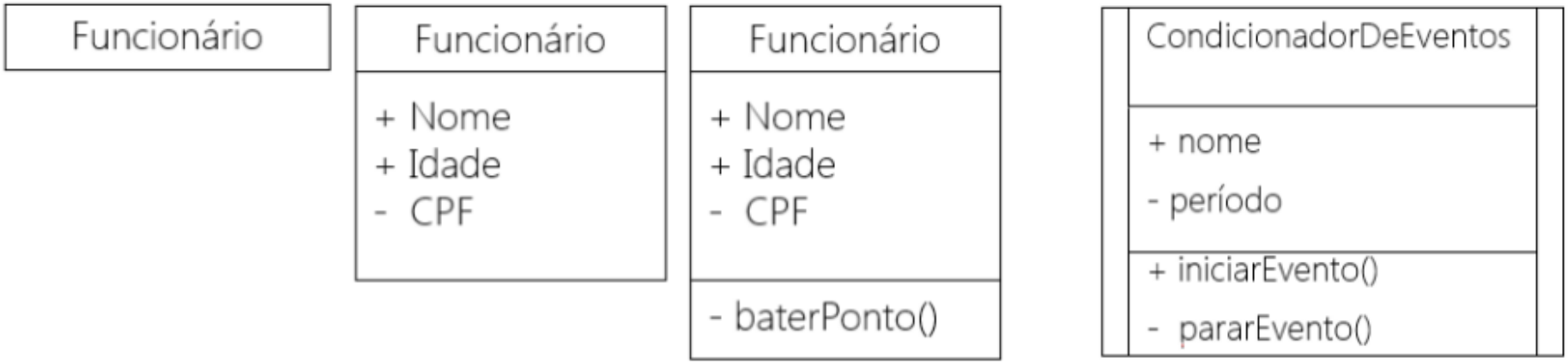
```
public class Produto
{
    int codigo;
    string nome;
    string descrição;
...
    public void inserir()
    {
        ... // Código para inserir
    }
...
}
```

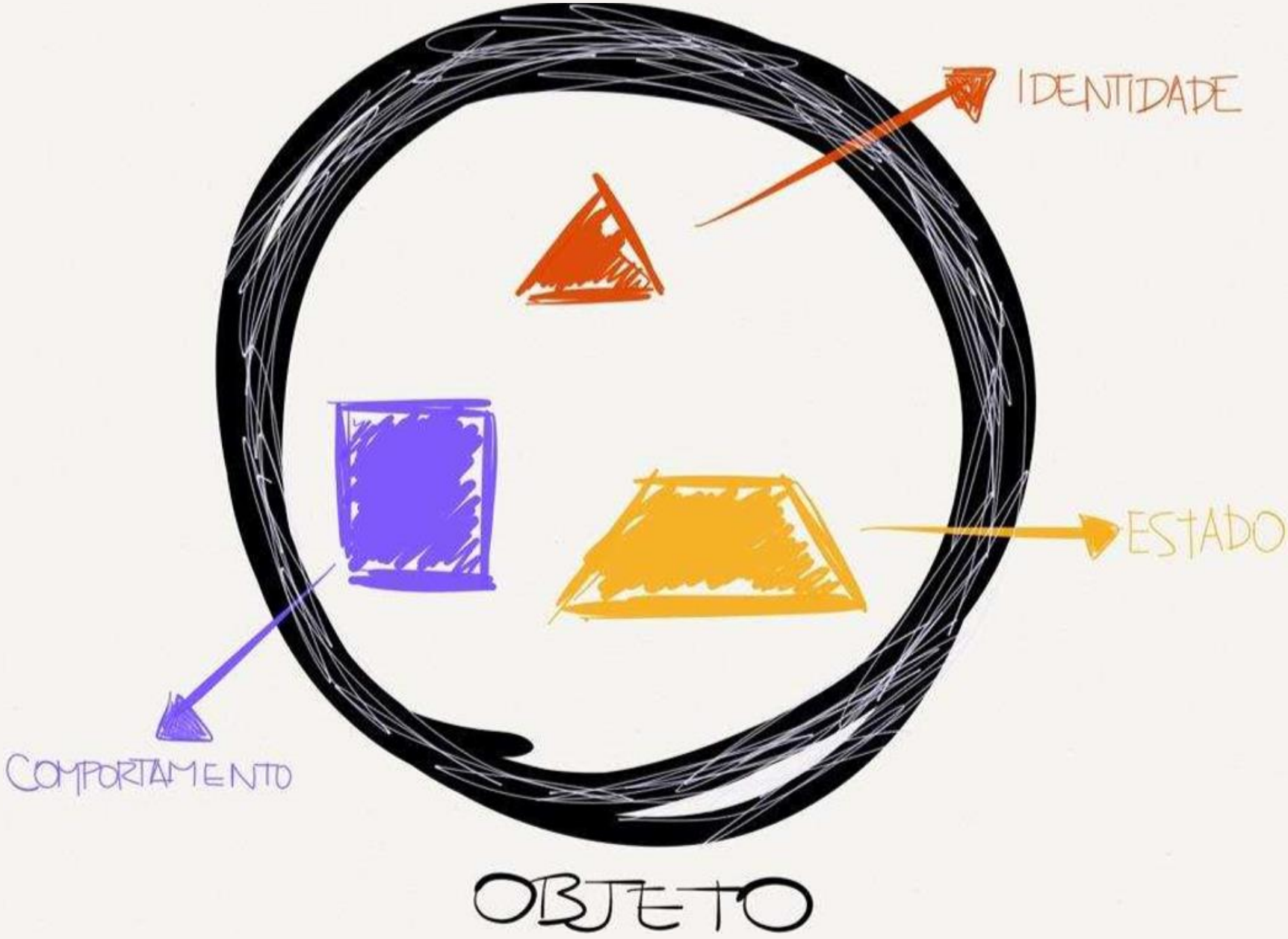


NOTAÇÃO BÁSICA



NOTAÇÃO BÁSICA (EXEMPLOS)



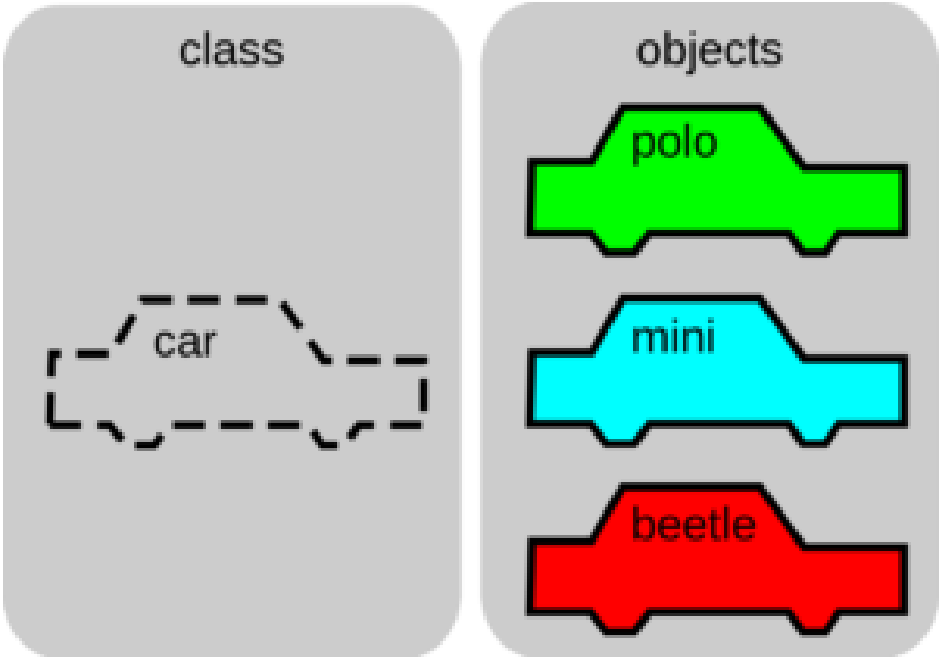

IDENTIDADE
ESTADO
COMPORTAMENTO
OBJETO

O QUE SÃO OBJETOS?

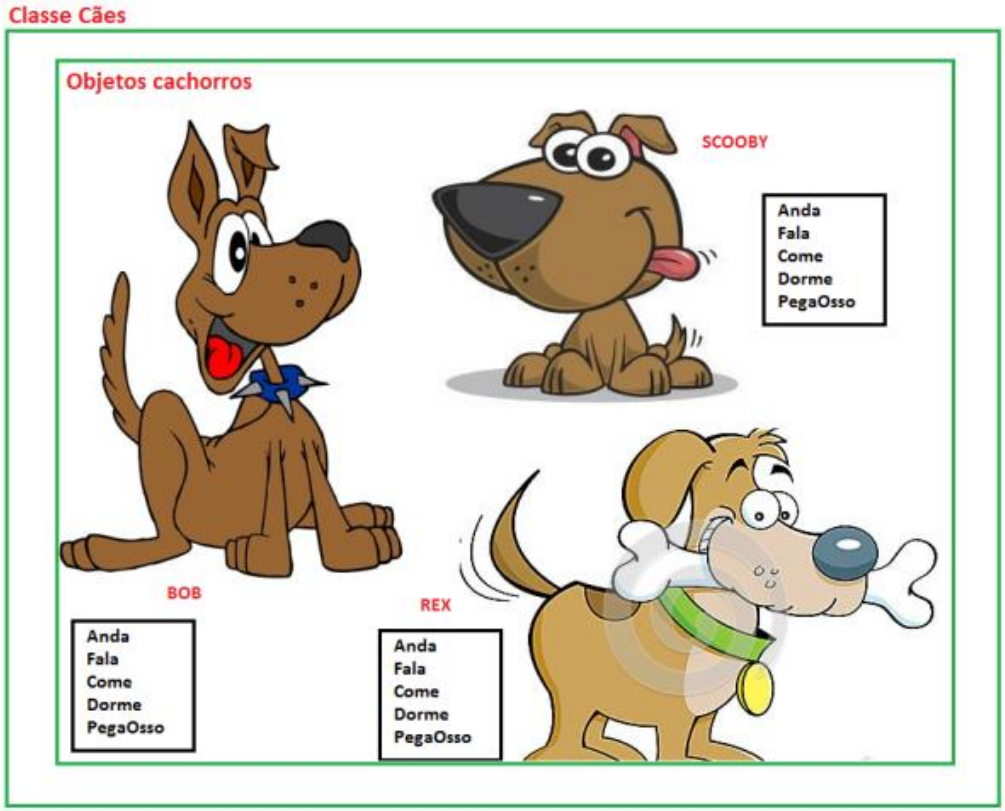
Objetos são instancias de classes.

O que é um carro? Posso abstrair um carro como um objeto que tem motor, volante, porta, rodas, etc.

O QUE SÃO OBJETOS?



O QUE SÃO OBJETOS?



ATRIBUTOS

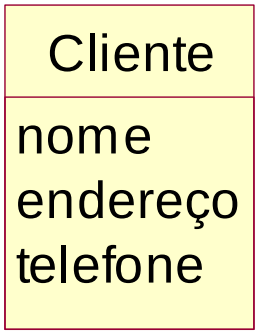
Um atributo representa alguma propriedade do que está sendo modelado, que é compartilhada por todos os objetos da classe.

Os atributos descrevem os dados contidos nas instâncias de uma classe.

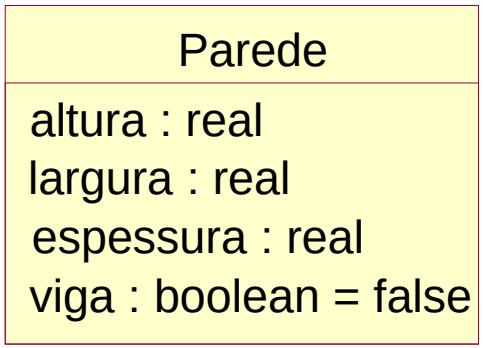
Em um momento dado, um objeto de uma classe conterá valores para todos os atributos descritos na sua classe.

ATRIBUTOS - NOTAÇÃO

Atributos podem ser identificados apenas com nomes



Atributos podem ter seus tipos (ou classes) especificados e terem valores padrão definidos



OPERAÇÕES

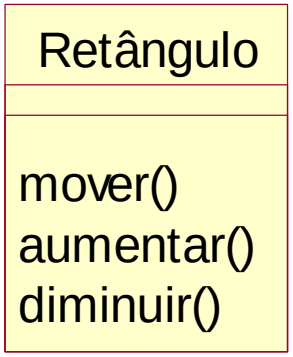
Uma operação é uma abstração de alguma coisa que se pode fazer com um objeto e que é compartilhada por todos os objetos da classe;

Um classe pode ter qualquer número de operações, inclusive nenhuma;

Operações são o meio de alterar os valores dos atributos ;

OPERAÇÕES - NOTAÇÃO

Como para os atributos, você pode especificar uma operação apenas com seu nome:



Você pode também especificar a **assinatura** da operação: seus parâmetros, o tipo desses parâmetros e o tipo de retorno

MODIFICADORES DE ACESSO

Declaração	Definição
public	Acesso ilimitado
private	Acesso limitado à classe e seus membros
internal	Acesso limitado ao programa (assembly)
protected	Acesso limitado à classe, seus membros e seus derivados
protected internal	Acesso limitado à classe, classes derivadas ou membros deste programa (assembly)

VISIBILIDADE

Você pode usar marcações de acesso para especificar o tipo de acesso permitido aos atributos e operações, utilizando modificadores:

MODIFICADOR			CLASSE	SUBCLASSE	PACOTE	TODOS
UML	PÚBLICO	+	X	X	X	X
	PROTEGIDO	#	X	X		
	PACOTE	~	X		X	
	PRIVADO	-	X			

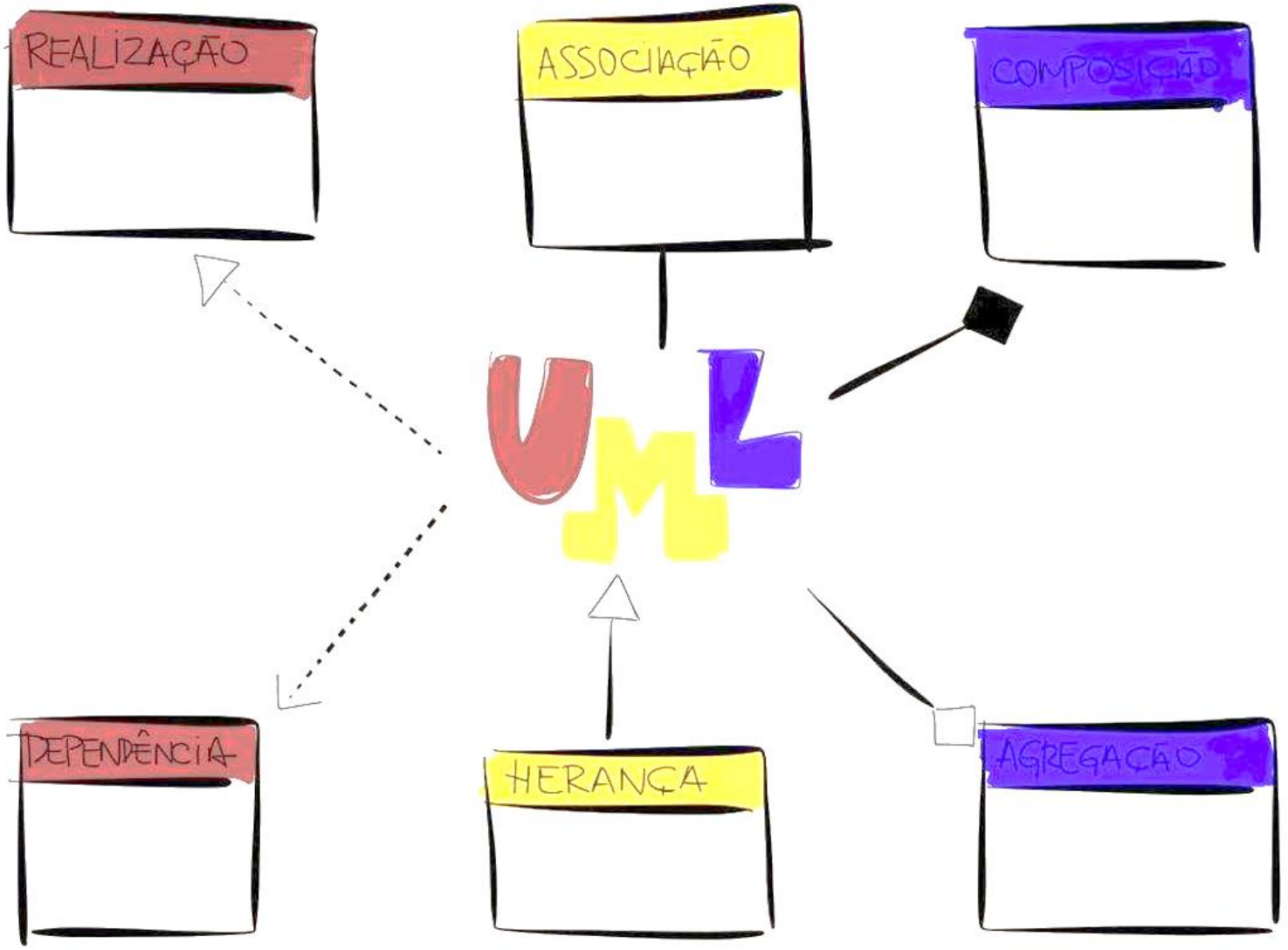
RELACIONAMENTOS

Poucas classes vivem sozinhas

Tipos de relacionamentos especialmente importantes na modelagem orientada a objetos:

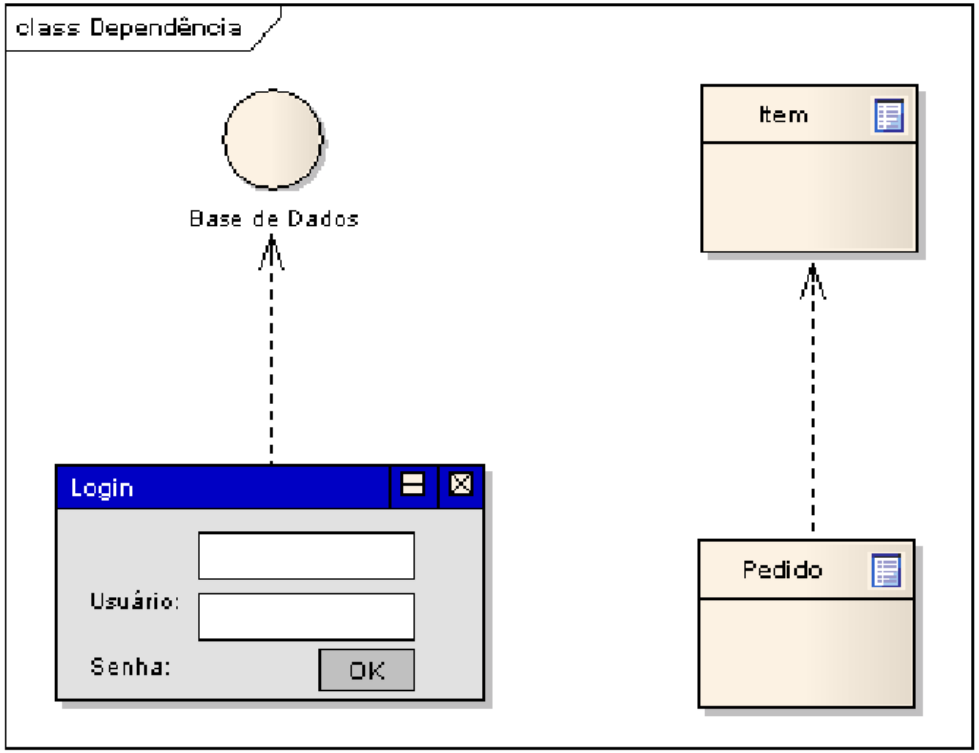
1. Associações
2. Agregação
3. Composição
4. Dependências
5. Generalizações
6. Realização

RELACIONAMENTOS



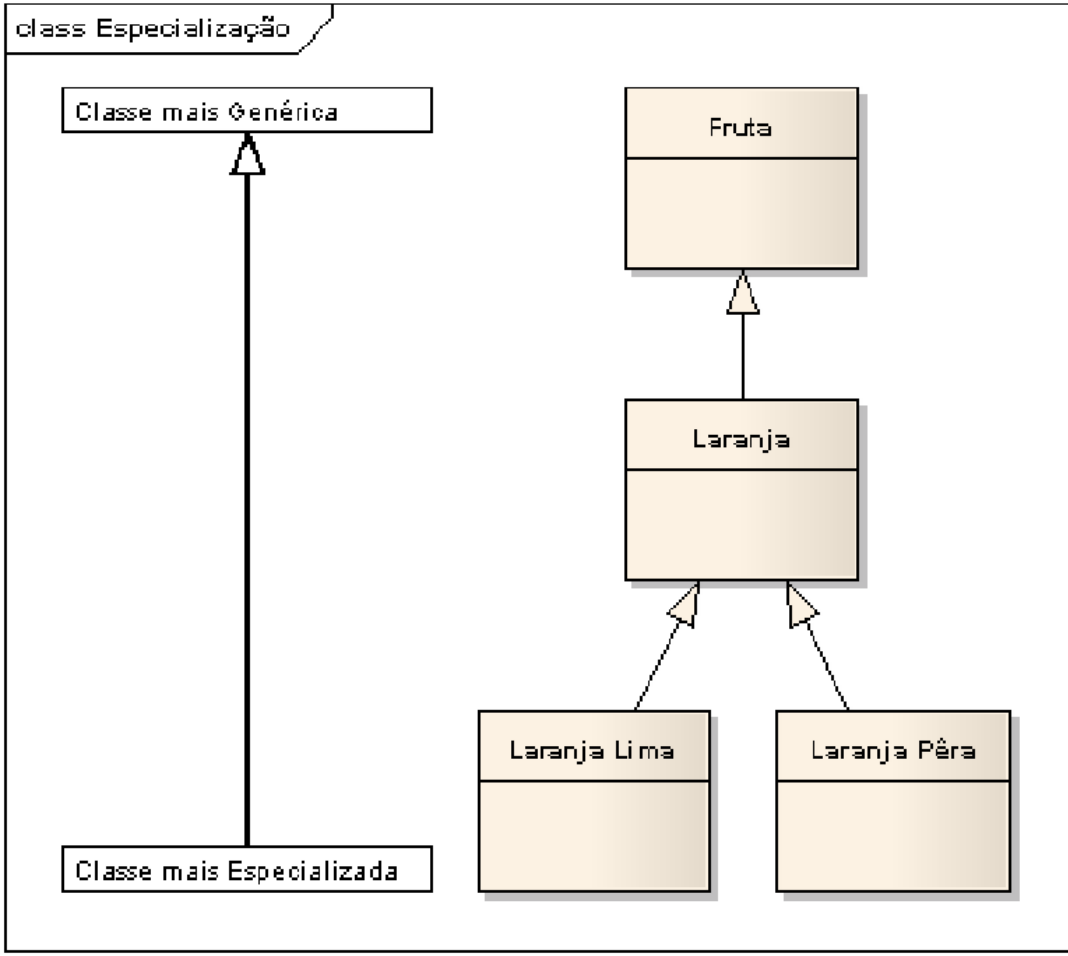
DEPENDÊNCIA

Relacionamento de Dependência: É um relacionamento direcionado e semântico entre dois ou mais elementos que ocorre se mudanças na definição de um elemento.



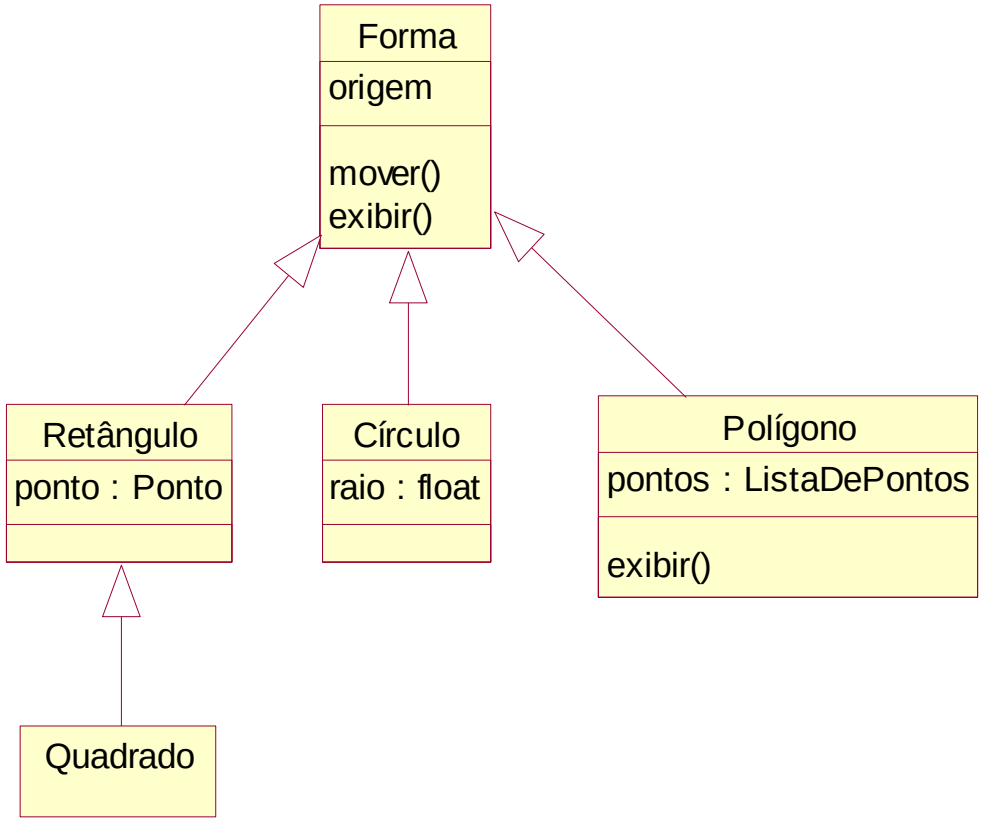
RELACIONAMENTO DE GENERALIZAÇÃO/ESPECIALIZAÇÃO (HERANÇA)

Relacionamento entre um elemento mais geral (chamado de superclasse ou pai) e um mais específico (chamado de subclasse ou filho)



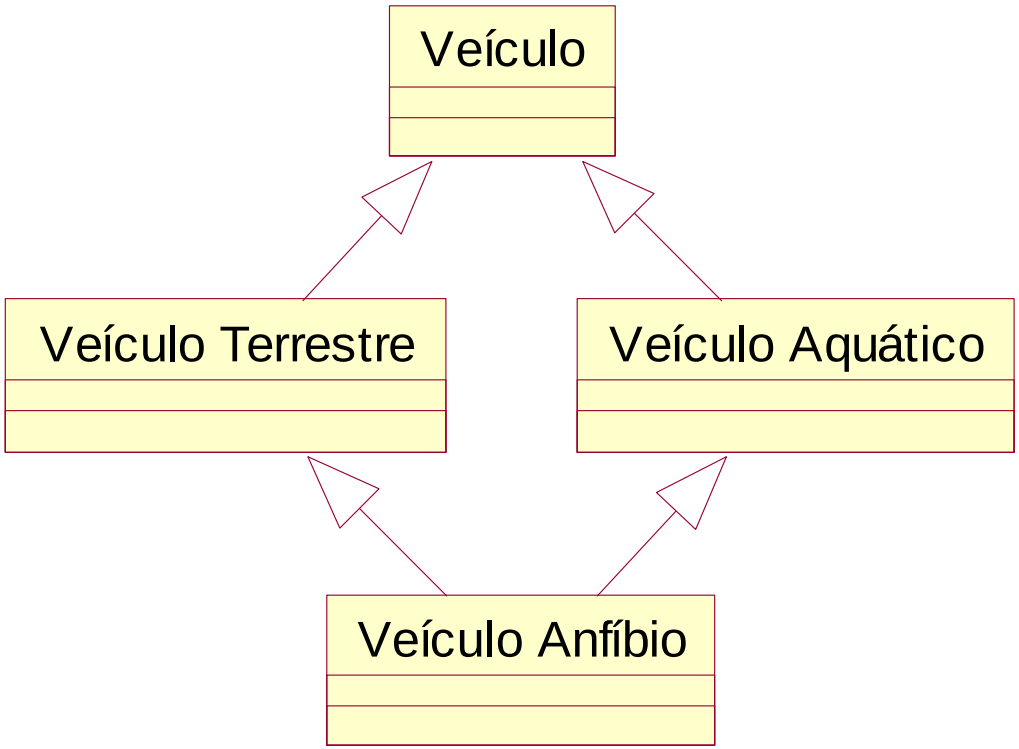
RELACIONAMENTO DE GENERALIZAÇÃO/ESPECIALIZAÇÃO (HERANÇA)

Relacionamento entre um elemento mais geral (chamado de superclasse ou pai) e um mais específico (chamado de subclasse ou filho)



HERANÇA MÚLTIPLA

Ocorrem múltiplas superclasses para uma mesma subclasse

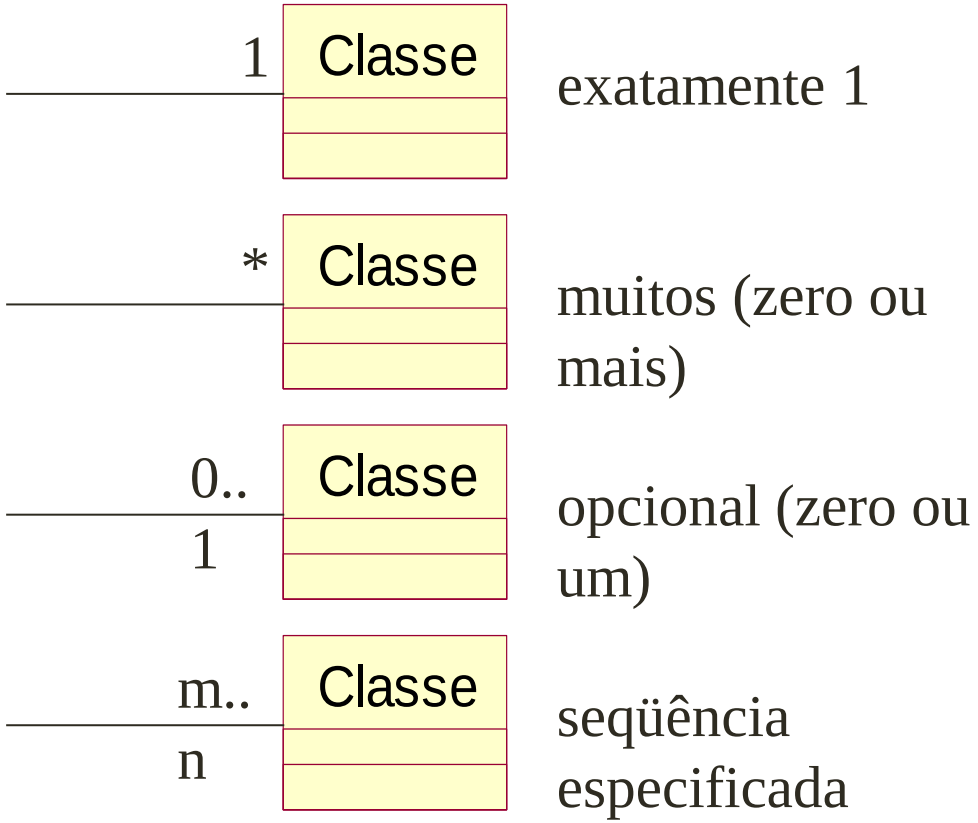


ASSOCIAÇÃO

A associação é um relacionamento estrutural que especifica que objetos de um elemento estão conectados a objetos de outro elemento.

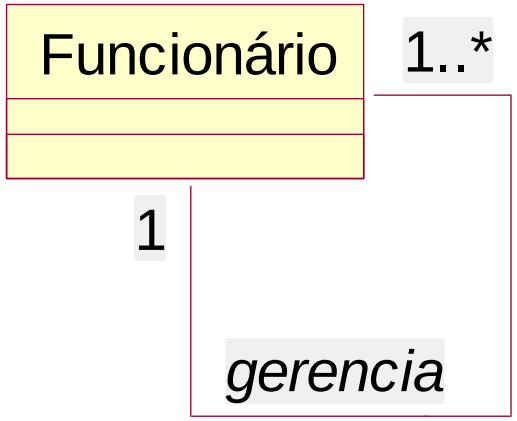
MULTIPLICIDADE

É a cardinalidade de uma associação



ASSOCIAÇÃO UNÁRIA

Quando há um relacionamento de uma classe para consigo própria



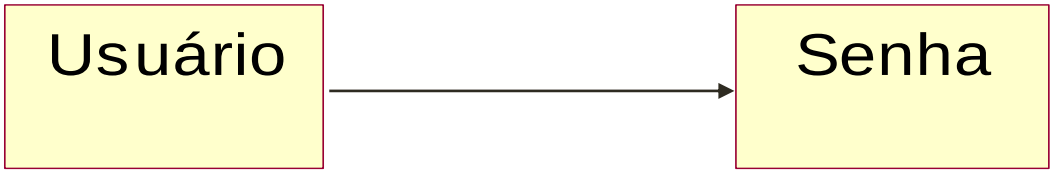
ASSOCIAÇÃO BINÁRIA

Quando há duas classes envolvidas na associação de forma direta de uma para a outra

ASSOCIAÇÃO: NAVEGABILIDADE

Em geral a navegação entre as classes de uma associação é bi-direcional

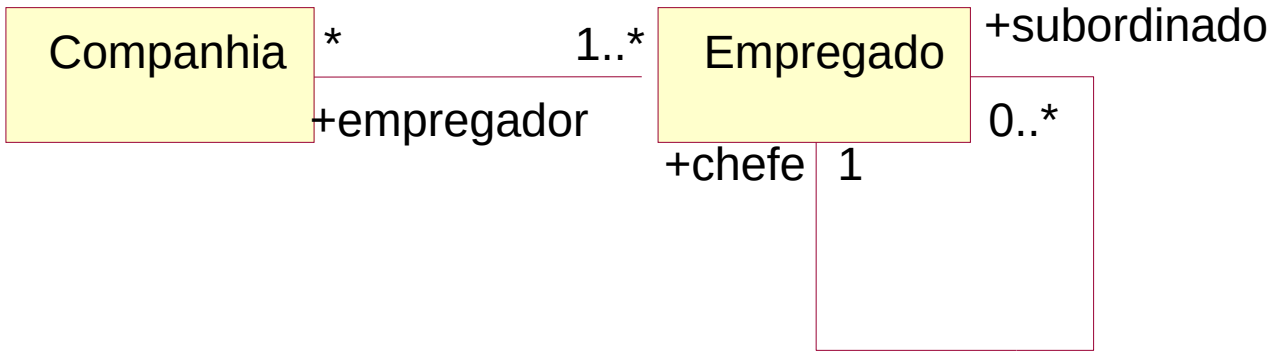
Porém é possível limitá-la a apenas uma direção



ASSOCIAÇÃO: PAPÉIS

Papéis: um dos lados da associação

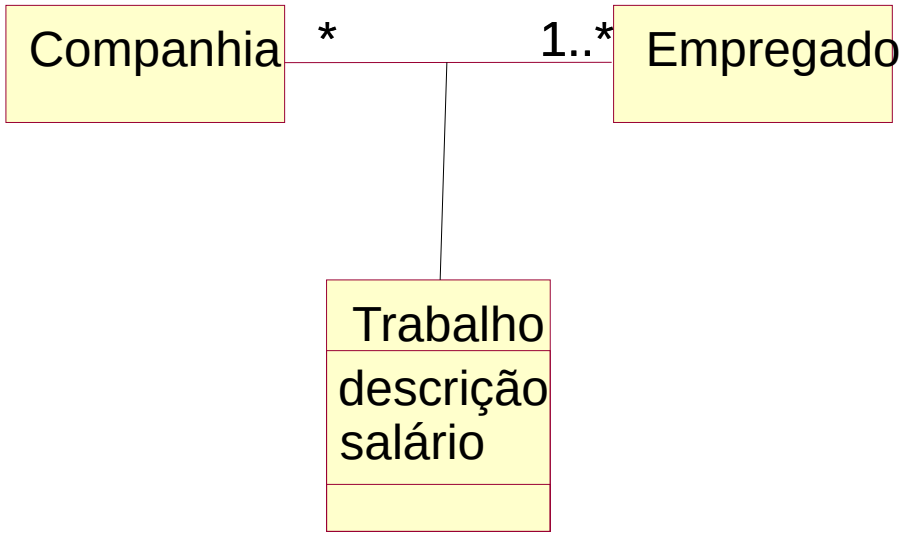
Nomes de papéis são necessários para associação entre dois objetos da mesma classe



ASSOCIAÇÃO COM ATRIBUTOS

Modela as propriedades associadas com uma associação

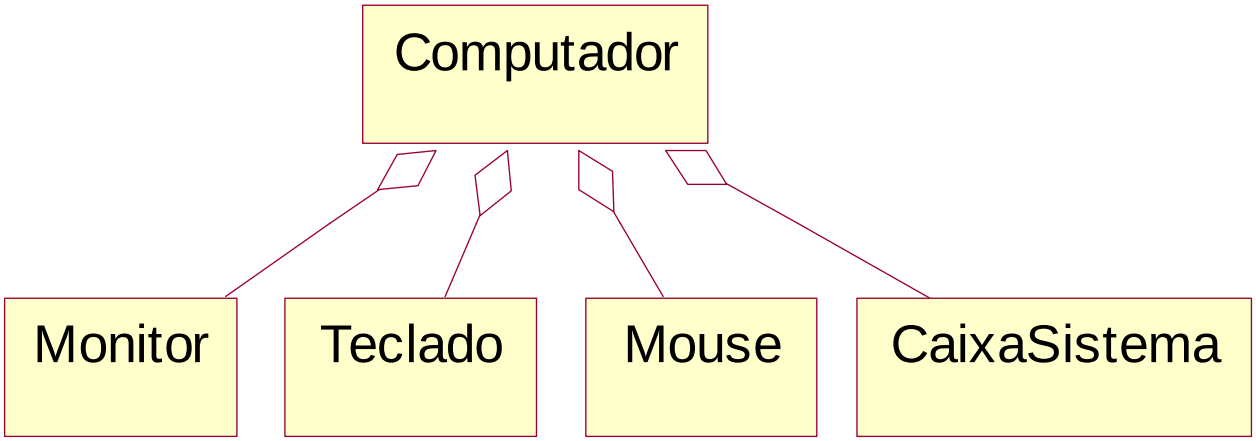
As propriedades devem ser representadas por uma classe



AGREGAÇÃO

Uma forma especial de associação entre o todo e suas partes, no qual o todo é composto de partes.

Não impõe que a vida das “Partes”’ esteja relacionado com a vida do “Todo”.



COMPOSIÇÃO

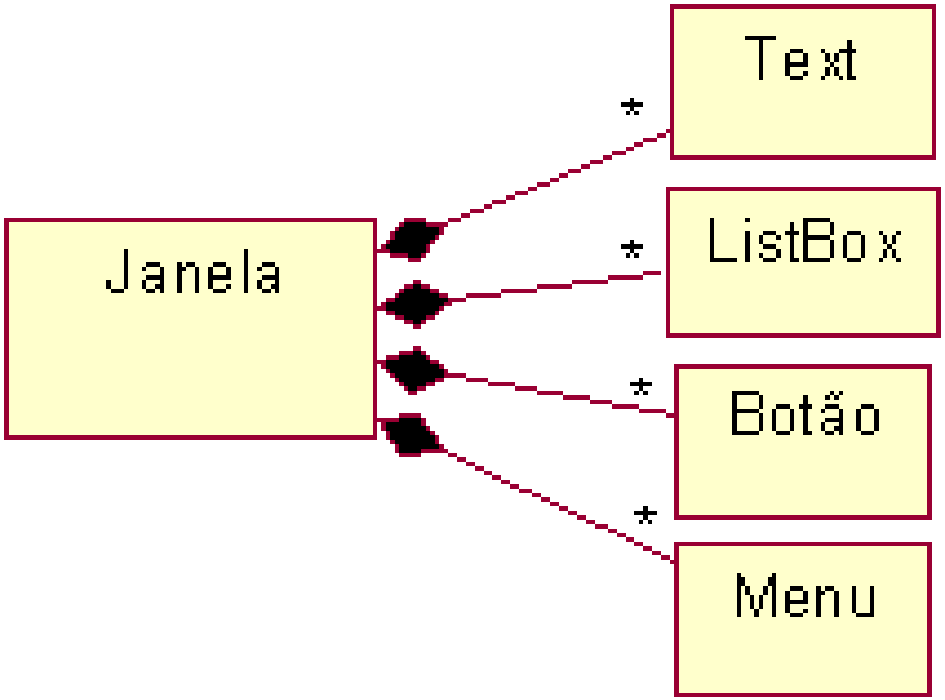
Uma forma mais forte de agregação

Há uma coincidência da vida das partes

Uma vez criada a parte ela irá viver e morrer com ele

O “Todo” é responsável pelo gerenciamento da criação e destruição das partes

COMPOSIÇÃO



INTERFACES

Uma interface é um conjunto de operações usado para especificar um serviço de uma classe ou componente.

Diferentemente das classes, as interfaces não especificam nenhuma estrutura.

Interfaces não podem conter atributos.

INTERFACES

Com as interfaces, é possível se concentrar apenas nos serviços oferecidos por classes ou componentes.

O uso de interfaces é uma maneira elegante e poderosa de isolar a especificação da implementação.

Uma interface especifica o contrato para uma classe ou componente, sem definir como ele será implementado.

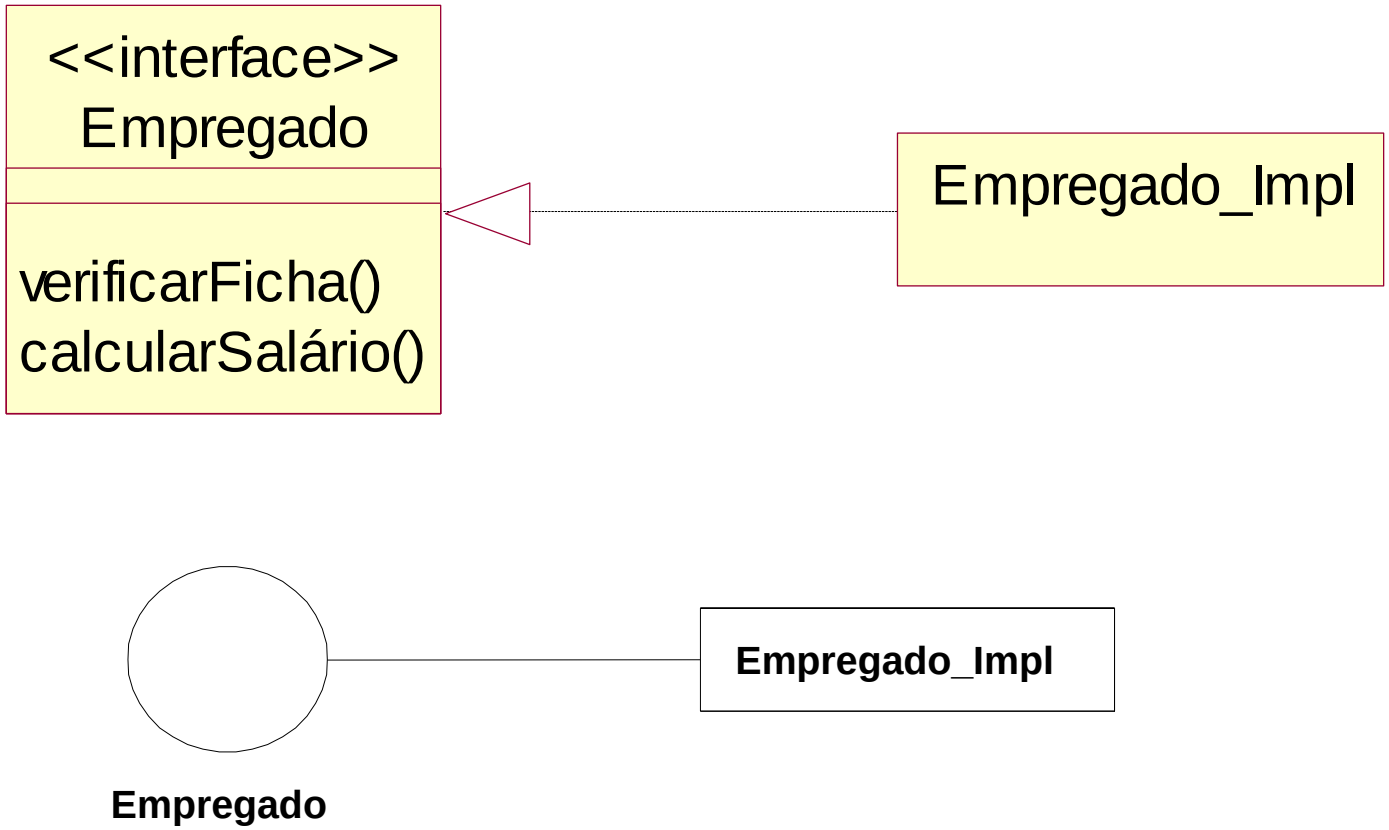
INTERFACES E REALIZAÇÃO

Realização é uma relação pela qual um elemento especifica o contrato que outro elemento deve implementar.

A realização é um relacionamento entre uma especificação e sua implementação.

É um relacionamento semântico entre classificadores no qual um classificador especifica um contrato que outro classificador garante cumprir.

REALIZAÇÃO - NOTAÇÃO



QUANDO UTILIZAR DIAGRAMAS DE CLASSE?

Os Diagramas de Classe são a espinha dorsal da UML; portanto, você irá utilizá-los o tempo todo.

O maior problema é que eles são tão ricos que podem ser complexos demais para usar. Dessa forma, não tente utilizar todas as notações de que você dispõe; uso de diagramas de classes conceituais são muito uteis na exploração da linguagem do negocio.

Busque manter o software fora da discussão e manter a notação mais simples; não desenhe modelos para tudo; em vez disso, concentre-se nas Áreas principais. melhor ter poucos diagramas que você utiliza e os mantém atualizados do que ter muitos modelos esquecidos e obsoletos. O maior perigo é que você pode focalizar exclusivamente a estrutura e ignorar o comportamento.

EXERCICIOS

1. **Pretende-se desenvolver um sistema para apoio à gestão de aluguéis de automóveis que permita efetuar, cancelar e modificar pedidos através da Internet. Após a análise inicial de requisitos do sistema foram levantadas as seguintes informações:**
   - O sistema só pode ser utilizado após cadastro prévio.
   - Os usuários individuais (clientes) podem introduzir, modificar, consultar e cancelar pedidos de aluguel.
   - Por outro lado, os agentes (empresas e bancos) podem modificar e avaliar pedidos.
   - Após introdução no sistema, os pedidos são analisados do ponto de vista financeiro pelos agentes e, em caso de parecer positivo, são colocados à sua consideração para execução do contrato.
   - Sobre os contratantes do aluguel, armazenam-se os dados de identificação (RG, CPF, Nome, Endereço), profissão, as entidades empregadoras e os respectivos rendimentos auferidos (máximo 3
   - Dependendo do tipo de contrato, os automóveis alugados podem ser registrados como propriedade dos clientes, empresas ou bancos.
   - Sobre os automóveis, o sistema registra a matrícula, ano, marca, modelo e placa.
   - O aluguel de um automóvel pode estar associado com um contrato de crédito, o qual foi concedido por um dos bancos agentes.
   - Em termos do sistema, o servidor central encontra-se ligado aos computadores locais dos clientes e aos diversos agentes aderentes através da Internet.
   - O sistema pode ser subdividido em dois subsistemas: um para gestão de pedidos e contratos; e outro para a construção dinâmica das páginas WWW.

Elabore um diagrama de casos de uso e um diagrama de classes do sistema.

EXERCICIOS

2. **Especificação dos Requisitos Sistema Bancário A. Lançamentos diversos:**
   - O sistema deve permitir o cadastro e alteração de clientes do banco os seguintes atributos: nome, endereço (rua, número, bairro, cep), telefone, data de nascimento para pessoa física, data de fundação para pessoa jurídica, e-mail, cpf (pessoa física) e cnpj (pessoa jurídica);
   - O sistema deve permitir o cadastro e alteração dos bancos com os seguintes atributos: código e nome;
   - O sistema deve permitir o cadastro e alteração das agências bancárias com os seguintes atributos: número da agência, nome, endereço (rua, número, bairro, cep), telefone, e-mail. Sabe-se que um banco pode ter várias agências. Uma agência pertence apenas a um banco;
   - O sistema deve permitir a criação de contas nos(as) bancos/agências com os seguintes atributos: número da conta e saldo. Sabe-se que um cliente pode ter várias contas e uma conta pode ter mais de um cliente como administrador (contas conjuntas, contas empresariais, etc).
   - Uma agência pode ter apenas dois tipos de contas: corrente e poupança. Para diferenciá-las é utilizado apenas a adição de (\1) no final da conta corrente. Exemplo: cc 5187, cp 5187\1;
   - O sistema deve permitir que os clientes efetuem operações de saque, depósito, transferências e agendamento (futuro) em uma conta. O sistema deve manter o registro de todas operações efetuadas pelos clientes;
   - Os agendamentos de operações devem verificar a data do lançamento da operação para que a data informada não seja inferior à data atual;

Faça os seguintes diagramas: a. Diagrama de classes do sistema.

BIBLIOGRAFIA

Cockburn, A., *Writing Effective Use Cases*, Addison-Wesley, 2001.

Fowler, M e Scott, K., *UML Distilled – A Brief Guide to the standard Object Modeling Language*, Addison Wesley Longman, 2002

Booch, G., Rumbaugh, J. and Jacobson, I., *Unified Modeling Language User Guide*, 2nd Edition, Addison-Wesley Object Technology Series, 2005.

DÚVIDAS ?

Aulas disponíveis em: professor.raonioliveira.com.br