MODELAGEM DE SOFTWARE

Prof. Richard Henrique de Souza
Ministrante

Prof. Ricardo Ribeiro Assink
ricardo.assink@unisul.br

Prof. Rafael Lessa
rafael.lessa@unisul.br

Agenda

Modelagem UML e a Orientação a Objetos

- *Unified Modeling Language* – UML
- Paradigma: Orientação a Objetos
- Objetos e Classes de Objetos

Unified Modeling Language - UML

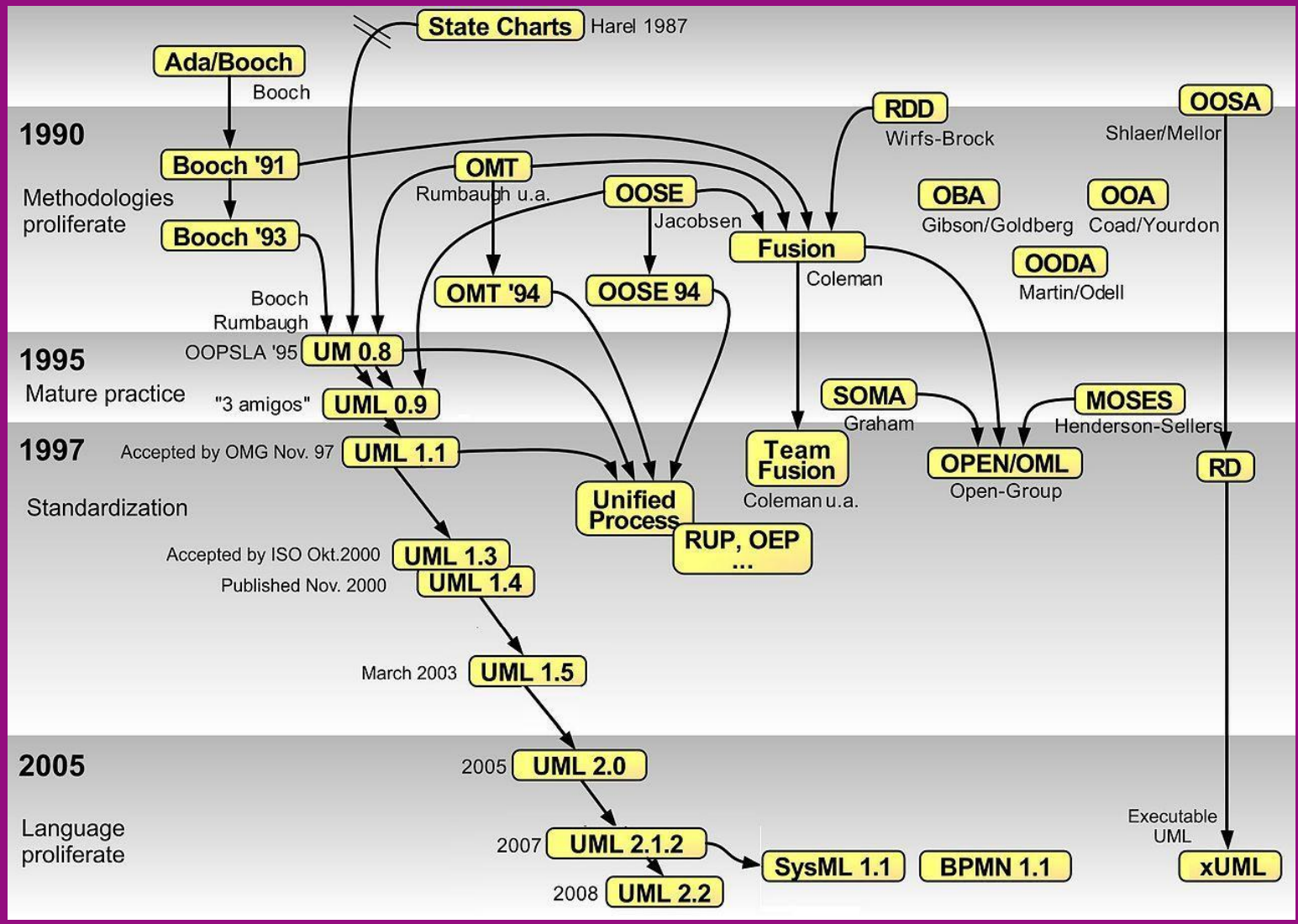
Diversas notação, processos, e ferramentas para descrição do projeto orientado a objetos foram proposta nos anos 80 e 90.

A UML era uma tentativa de padronizar a modelagem para que qualquer sistema possa ser descrito corretamente de forma consistente e clara. Existiam várias metodologias de modelagem orientada a objetos que causavam uma guerra entre a comunidade de desenvolvedores orientado a objetos. A UML acabou com esta guerra trazendo as melhores ideias de cada uma destas metodologias, e mostrando como deveria ser a migração de cada uma para a UML.

A UML foi desenvolvida por Grady Booch, James Rumbaugh, e Ivan Jacobson. Sendo que a OMG (*Object Management Group*), em 1997, unificou na UML diferentes notações existentes na época.

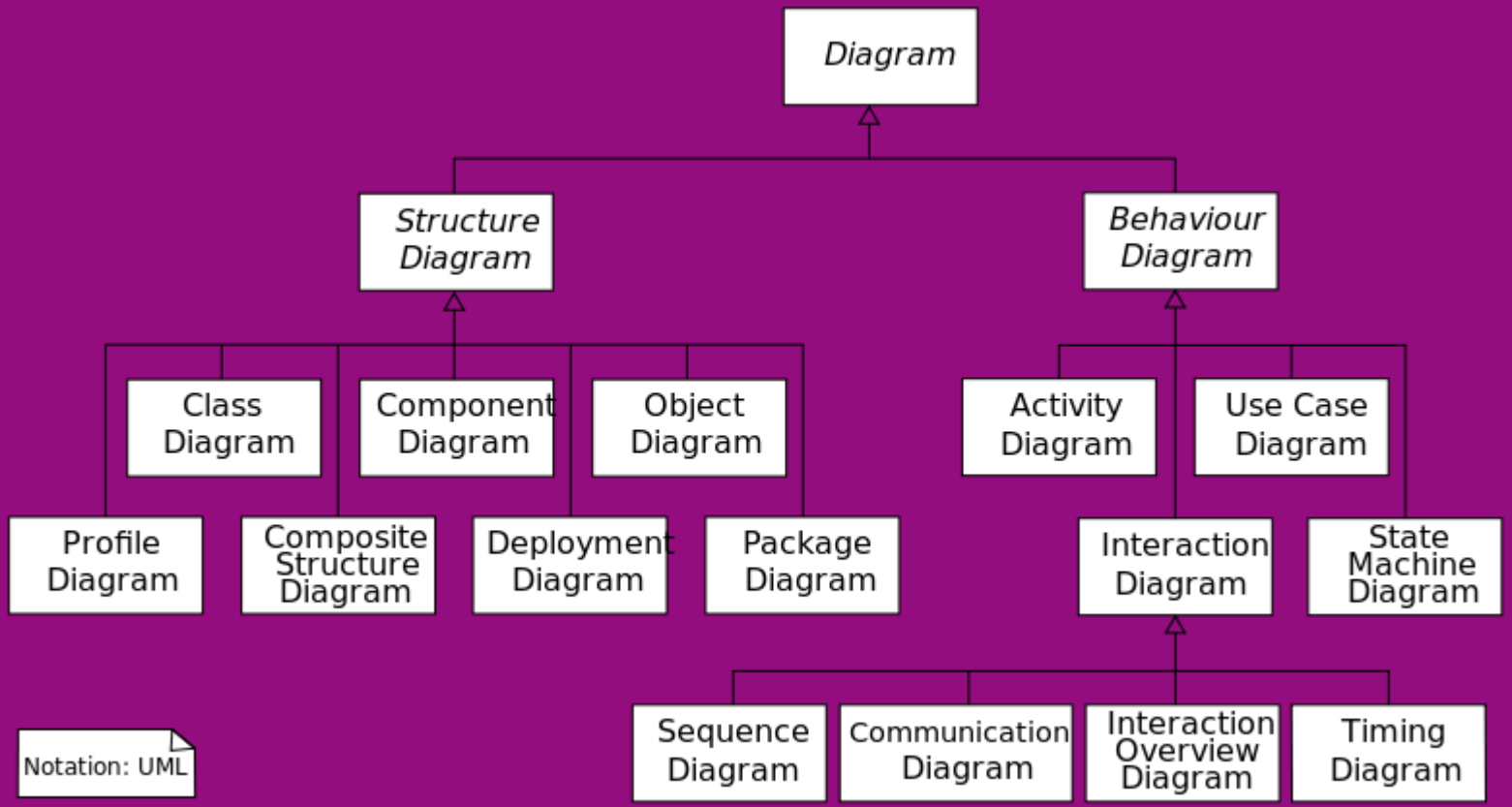
Versão atual da UML 2.0 <http://www.uml.org>

Unified Modeling Language - UML



Fonte: Guido Zockoll, Axel Scheithauer & Marcel Douwe Dekker (Mdd), 2009
Disponível em: <http://en.wikipedia.org/wiki/Unified_Modeling_Language>

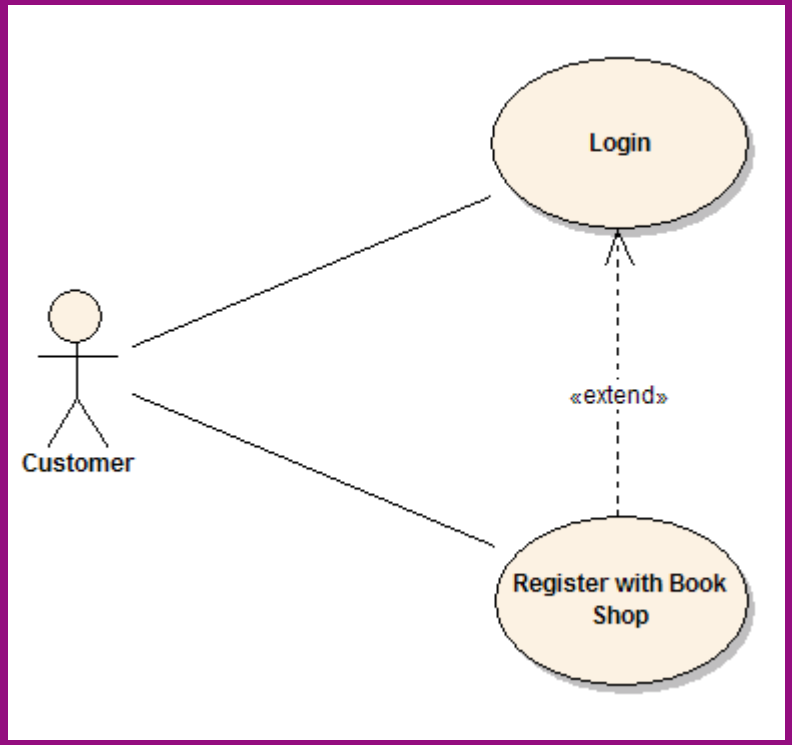
Unified Modeling Language - UML



Fonte: UML diagrams overview. Disponível em: <http://en.wikipedia.org/wiki/Unified_Modeling_Language>

Unified Modeling Language - UML

Diagrama de Use Case



Fonte: Sparx Systems. <http://www.sparxsystems.com/uml-tutorial.html>

Paradigma Orientado a Objetos e Modelagem

Paradigma: Orientação a Objetos

A orientação a objetos é um modelo de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos.

O paradigma da orientação a objeto tem como princípio a abstração de conceitos do mundo real.

Na qualidade de método de modelagem é tida como a melhor estratégia para se eliminar o "gap semântico", na medida em que a correlação da simbologia e conceitos abstratos do mundo real e das ferramentas (conceitos, terminologia, símbolos, grafismo e estratégias) fossem mais naturais e exatas.

Paradigma: Orientação a Objetos

Na programação orientada a objetos, implementa-se um conjunto de classes que definem os objetos presentes no sistema de software.

Cada classe determina o **comportamento** (definido nos métodos) e **estados possíveis** (atributos) de seus objetos, assim como o relacionamento com outros objetos.

Um projeto de software pode ser representado como um conjunto de objetos que interagem e gerenciam seus estados internos e suas operações.

Linguagem POO (*): C++, C♯, VB.NET, Java, Object Pascal, Python, SuperCollider, Ruby e Smalltalk.

* ActionScript, ColdFusion, Javascript, PHP (a partir da versão 4.0), Perl (a partir da versão 5) e Visual Basic (a partir da versão 4) são exemplos de linguagens de programação com suporte a orientação a objetos

Paradigma: Orientação a Objetos

Temos as seguintes características:

- Objetos são **abstrações** de entidades do mundo real (ou de algum sistema) que se auto gerenciam.
- Objetos são independentes e **encapsulam** suas representações de estado e de informações.
- A funcionalidade de um sistema é expressa em termos de serviços que os objetos prestam.
- Objetos se comunicam através do envio de **mensagens**.
- Objetos podem ser distribuídos.

Conceitos da POO: Classes e objetos

Classe: Uma classe descreve a estrutura e o comportamento de um conjunto de objetos similares.

Uma classe é a descrição de um grupo de objetos com propriedades similares, comportamento comum, relacionamentos comuns com outros objetos e semântica comum.

Uma classe deve ser a representação de apenas uma abstração.

Conceitos da POO: Classes e objetos

Uma classe deveria capturar uma e somente uma abstração chave.

Classes podem ser identificadas examinando-se os substantivos dentro do sistema a ser modelado.

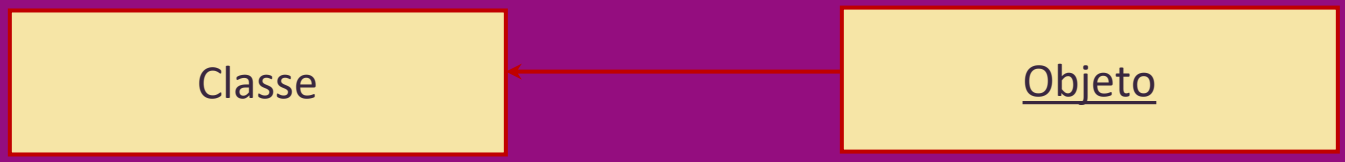
Uma classe deveria ser um substantivo singular que melhor caracteriza a abstração: Professor, Aluno, Turma, ...

Dificuldades na nomeação das classes podem indicar abstrações mal definidas.

Conceitos da POO: Classes e objetos

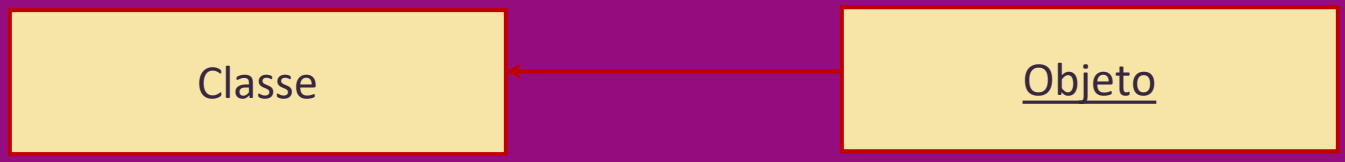
Objeto: É uma instância que está presente em tempo de execução e que se comporta de acordo com o protocolo da sua classe.

Todo objeto tem: Estado, Comportamento e Identidade



Conceitos da POO: Classes e objetos

A representação do relacionamento entre Classe e Objeto (Instância)

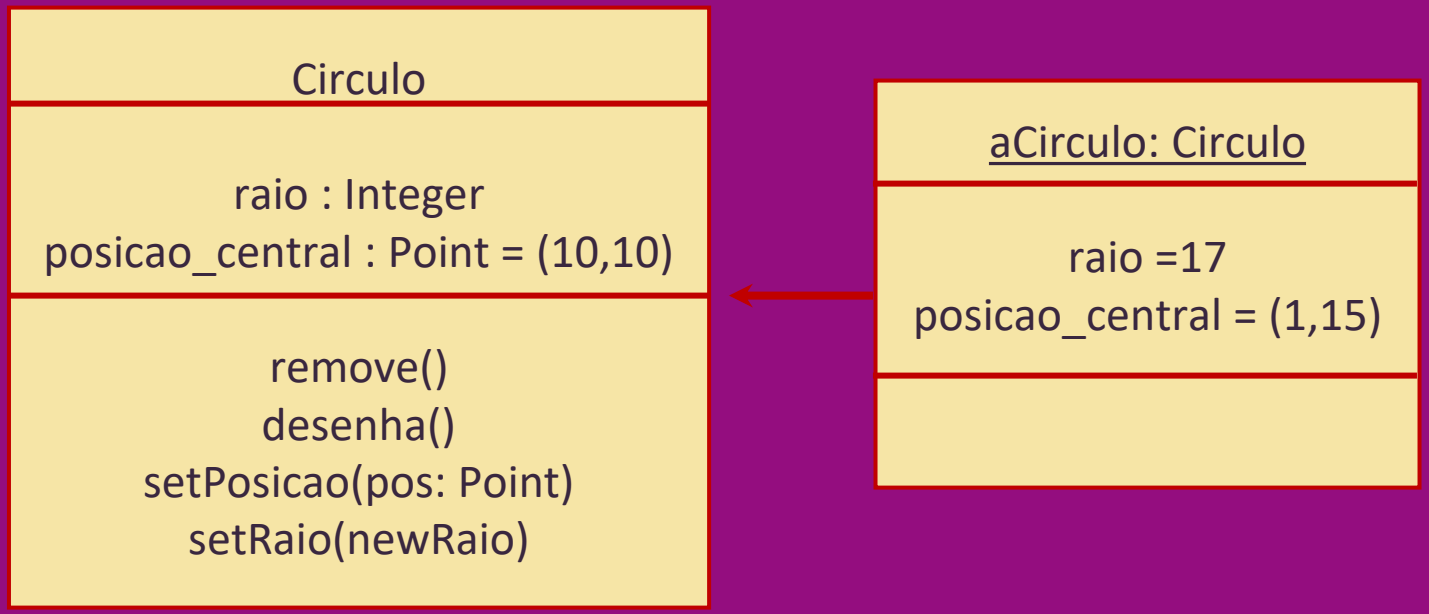


A classe “Pessoa” não existe como um objeto.

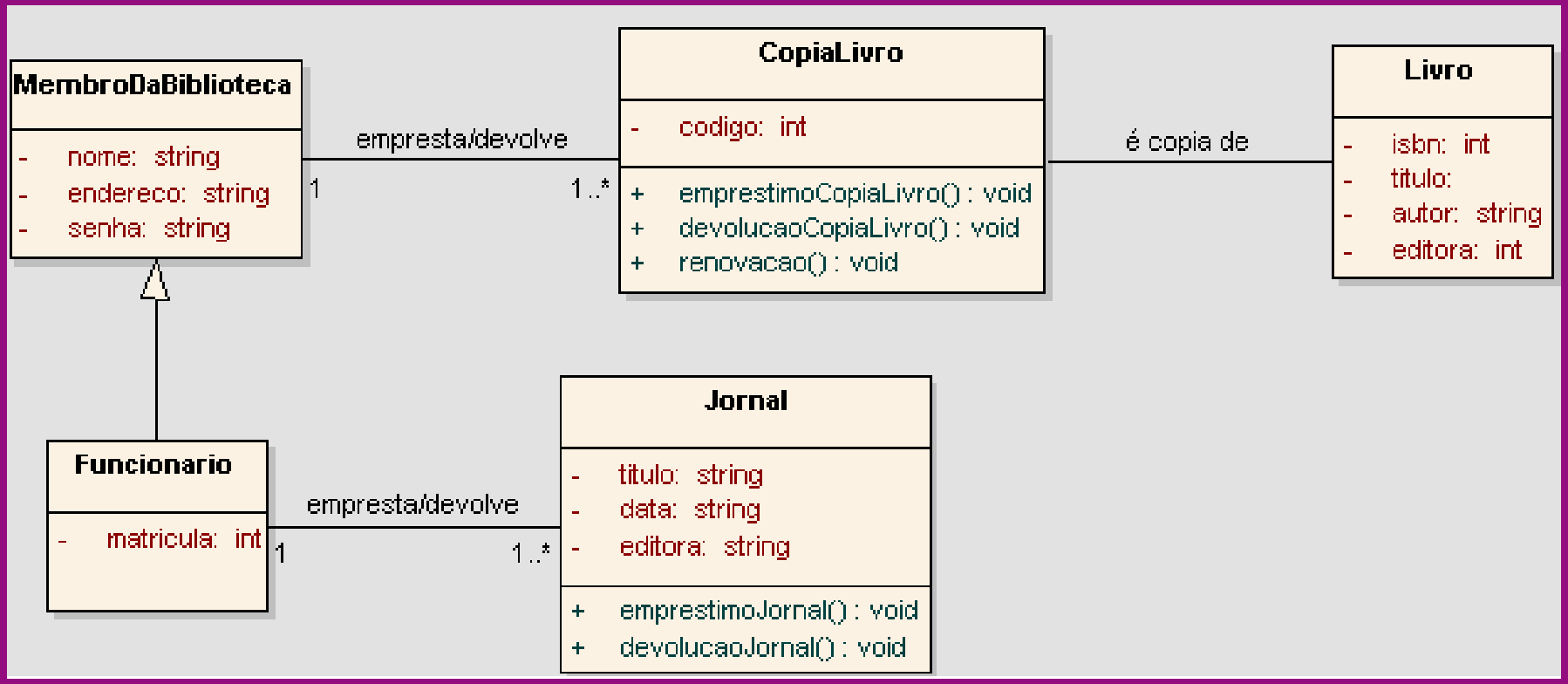
Conceitos da POO: Classes e objetos

Atributos: A estrutura do objeto: seus componentes e sua informação.

Operações: O comportamento dos objetos.



Modelos baseados em POO



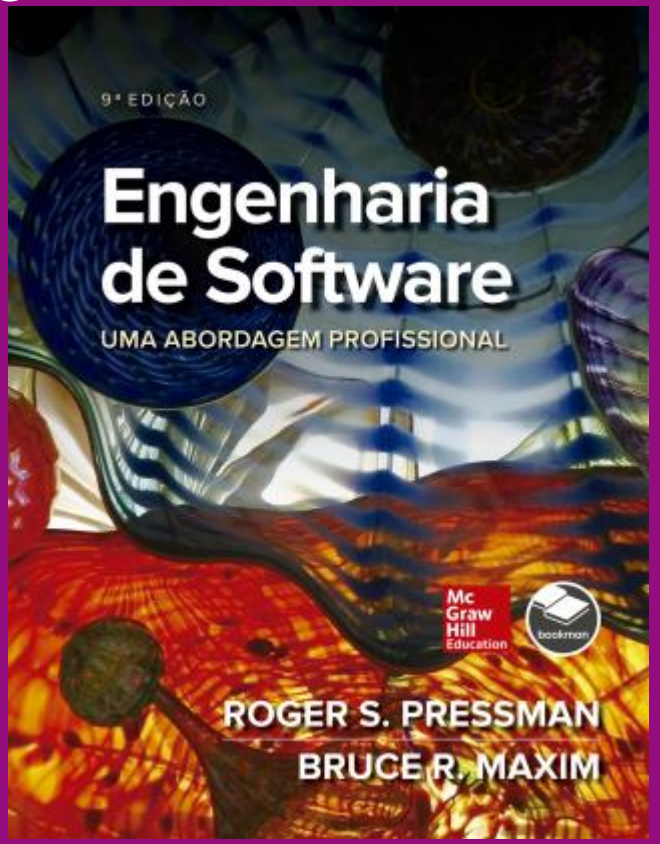
Busca Ativa

- Leitura das páginas 114 até a página 125 e

Das páginas 141 até 144 do livro

Pressman, Roger, S. e Bruce R. Maxim. *Engenharia de software*.

Disponível em: Minha Biblioteca, (9th edição). Grupo A, 2021.



Conceitos da POO: Classes e objetos

Visibilidade da classe

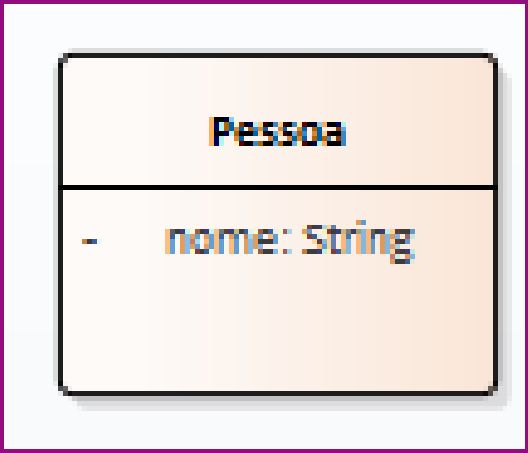
Especifica como os atributos e/ou operações poderão ser acessados por outros objetos

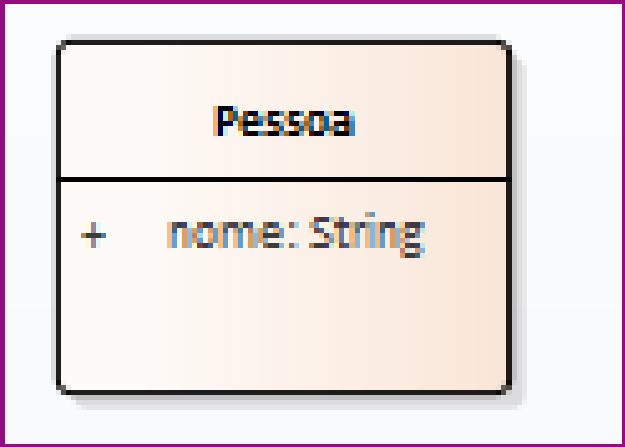
Atributos e operações podem ter visibilidades diferentes

- Privada (–) : atributos e operações só podem ser acessados na implementação da própria classe
- Pública (+) : atributos e operações são visíveis dentro da própria classe e para todas as outras classes
- Protegida (#) : será discutida após discutirmos o conceito de herança

Visibilidade

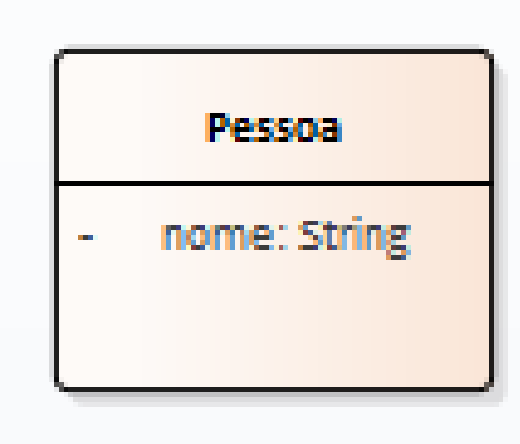
- Qual a diferença ?

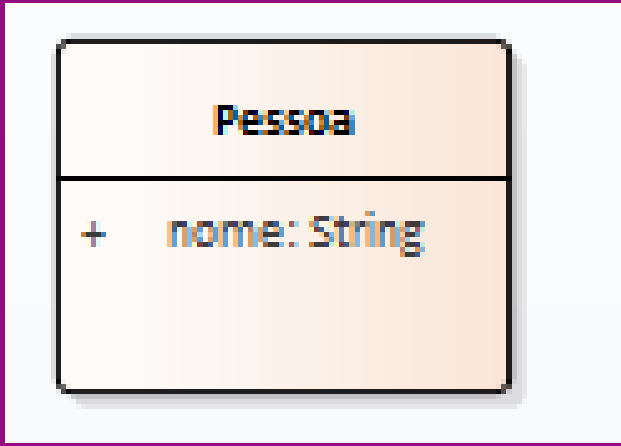




Visibilidade

- Qual a diferença ?
  - Encapsulamento





Visibilidade

- Qual a diferença ?
  - Encapsulamento

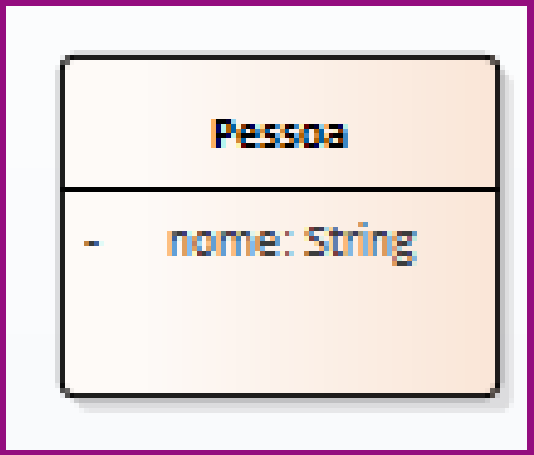
```
public class Pessoa {
public String nome;
}
```

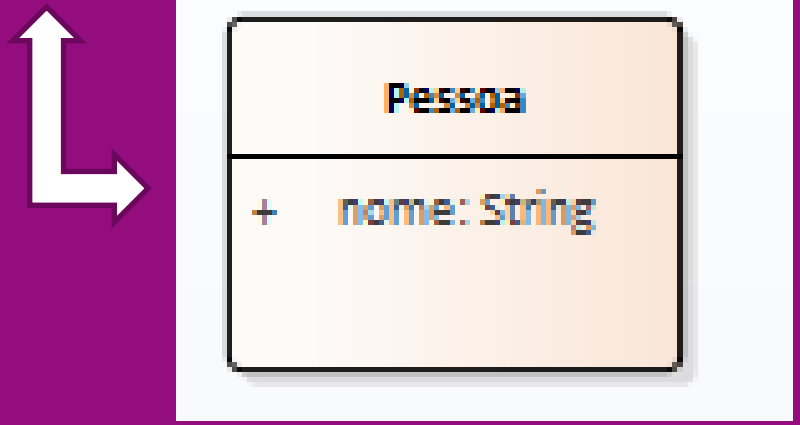
```
Pessoa p = new Pessoa();
p.nome = "Richard";
System.out.println(p.nome);
```

Problems | Javadoc | Declaration | Console

<terminated> Teste (1) [Java Application] C:\Program Files\Java\jre1.

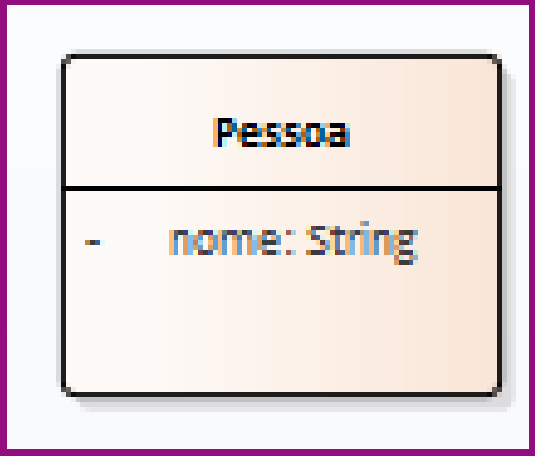
Richard





Visibilidade

- Qual a diferença ?
  - Encapsulamento




```
public class Pessoa {
  private String nome;
}
```

```
public static void main(String[] args) {
Pessoa p = new Pessoa();
p.nome = "Richard";
System.out.println(p.nome);
}
```

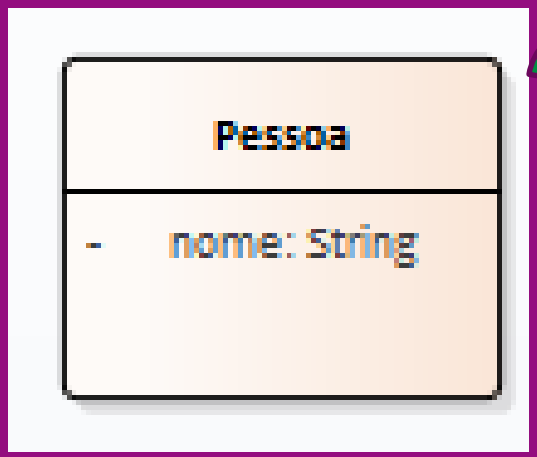
Problems @ Javadoc Declaration Console

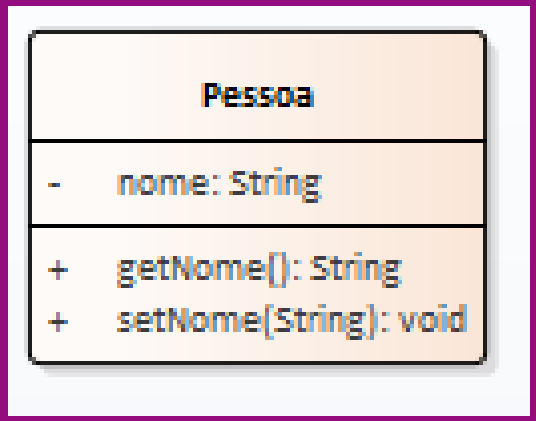
<terminated> Teste [Java Application] C:\Program Files\Java\jre1.8.0_321\bin\javaw.exe (28 de abr de 2022 12:50:28)

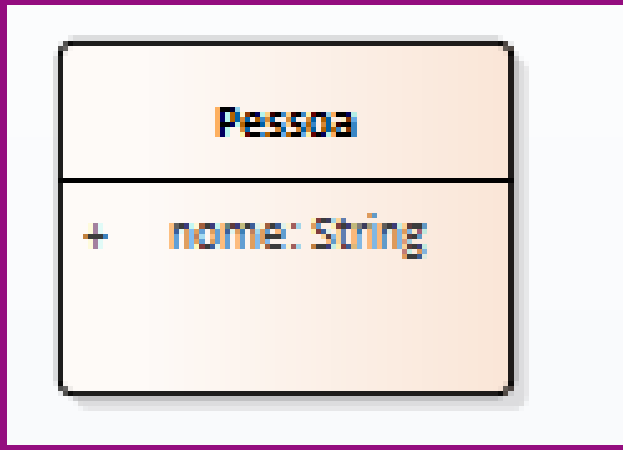
```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        The field Pessoa.nome is not visible

        at a.Teste.main(Teste.java:7)
```

Visibilidade

- Qual a diferença ?
  - Encapsulamento
    - O acesso externo somente poderá ser feito por operações para acessar.







Busca Ativa

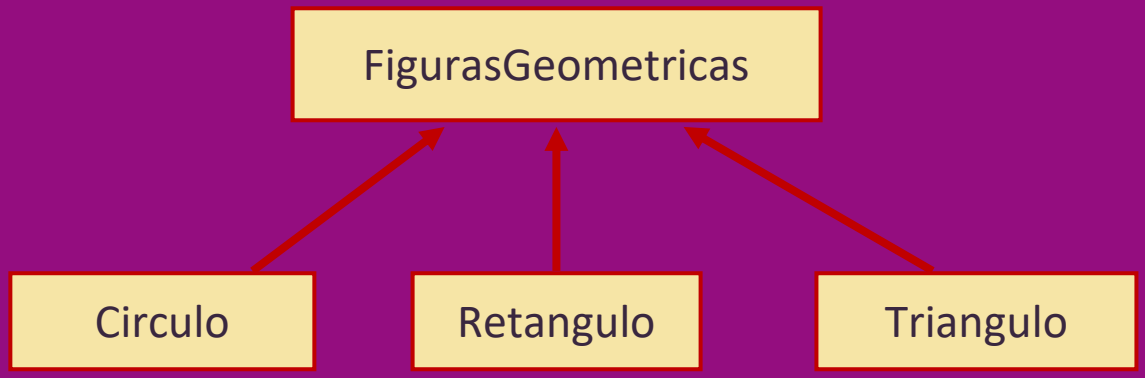
- Leitura da página 97 até a página 141 do livro

Horstmann, Cay. *Conceitos de Computação com Java*. Disponível em: Minha Biblioteca, Grupo A, 2009.

Conceitos da POO: Classes e objetos

Na hierarquia todas as classes possuem propriedades idênticas e sua especialização são representadas pelas subclasses.

Subclasses derivadas da classe possuem automaticamente todas as propriedades da sua classe superior.



Conceitos da POO: Classes e objetos

Superclasse é a classe que fornece os atributos e operações.

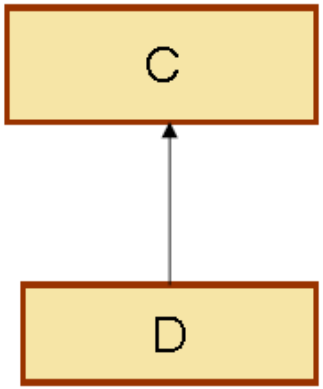
As relações:

- FigurasGeometricas é uma **generalização** de Retangulo.

- Retangulo é uma **especialização** de FigurasGeometricas

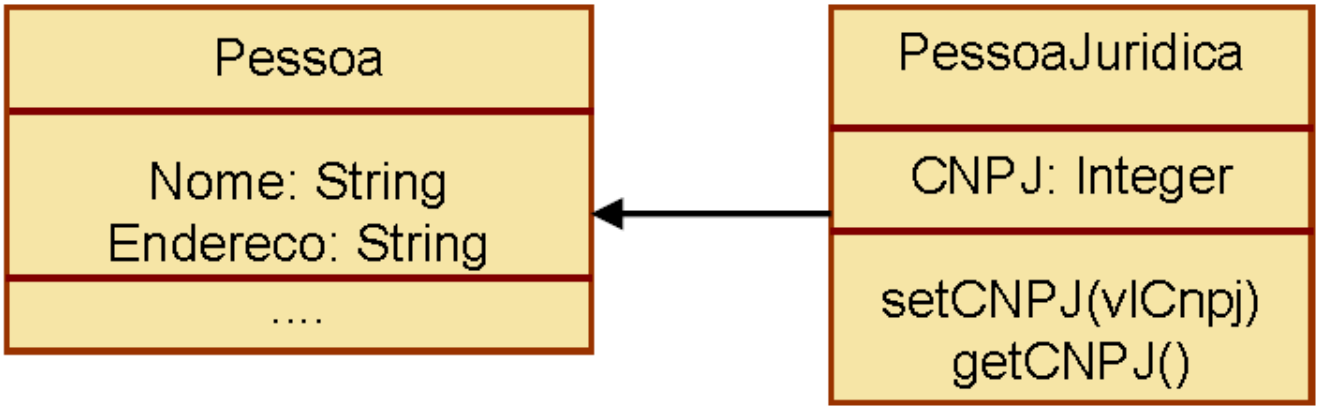
Conceitos da POO: Herança

- O que você faria se tivesse escrito uma classe C e depois descobrisse uma classe D quase igual a C, exceto com alguns métodos extras?
  - Uma solução seria duplicar todos os métodos de C e colocá-los em D. (Isso não seria retrabalho? E a manutenção?)
  - A *Melhor Solução* seria fazer com que a classe D de certa forma "pedisse para usar os métodos" da classe C. Chamamos isso de **Herança.**



Conceitos da POO: Herança

- **Herança** é o dispositivo pelo qual objetos de uma classe D podem usar as operações ou atributos que de outro modo estaria disponível somente para os objetos da classe C, como se aquele métodos ou variável tivesse sido definido através de D.



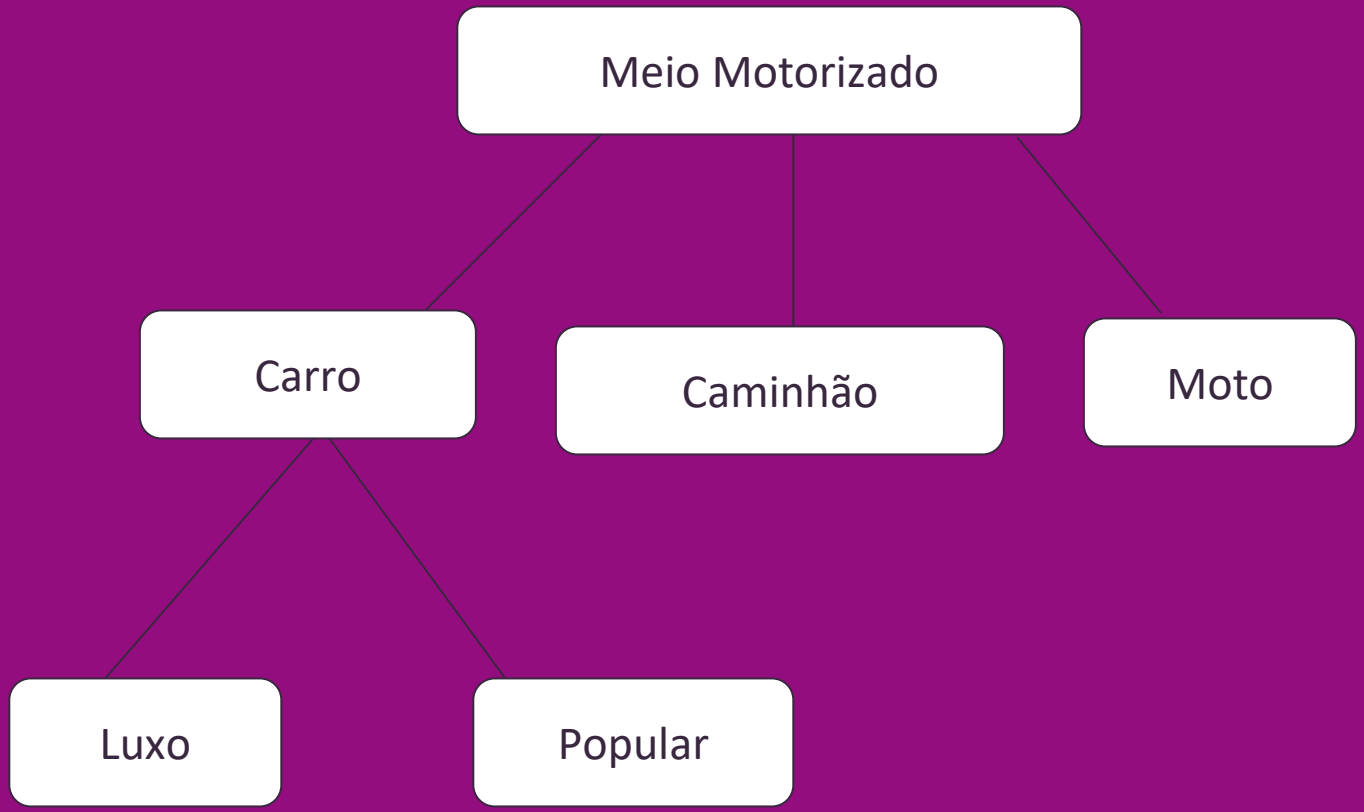
Conceitos da POO: Herança

É um relacionamento de **especialização** e **generalização**(“é um” ou “tipo de”).

Herança define uma hierarquia de abstrações na qual uma subclasse herda de uma ou mais superclasses:

- **Herança simples**: a subclasse herda de uma única superclasse.
- **Herança múltipla**: a subclasse herda de duas ou mais Superclasses.

Conceitos da POO: Herança

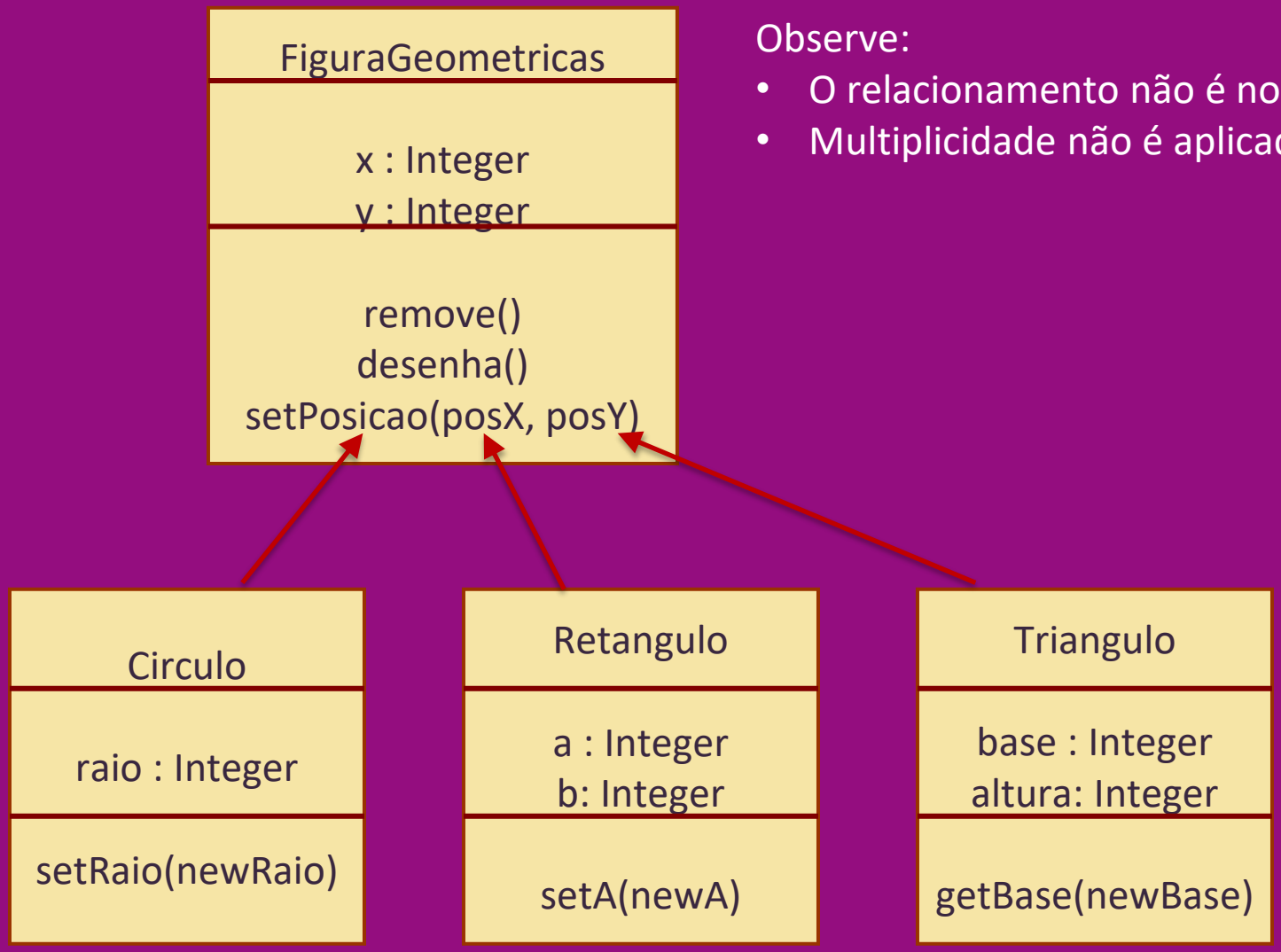


Conceitos da POO: Herança

Na hierarquia todas as classes possuem propriedades idênticas e sua especialização são representadas pelas subclasses. A hierarquia de classes segue a visão individual sobre o domínio a ser modelado.

Subclasses derivadas da classe possuem automaticamente todas as propriedades da sua classe superior.

Conceitos da POO: Herança



Observe:

- O relacionamento não é nomeado.
- Multiplicidade não é aplicada.

Conceitos da POO: Herança

Herança não relaciona objetos individuais

- O relacionamento não é nomeado.
- Multiplicidade não é aplicada.

Uma subclasse pode:

- Adicionar atributos, operações e relacionamentos.
- Redefinir operações herdadas.

Conceitos da POO: Classes, Objetos e Herança

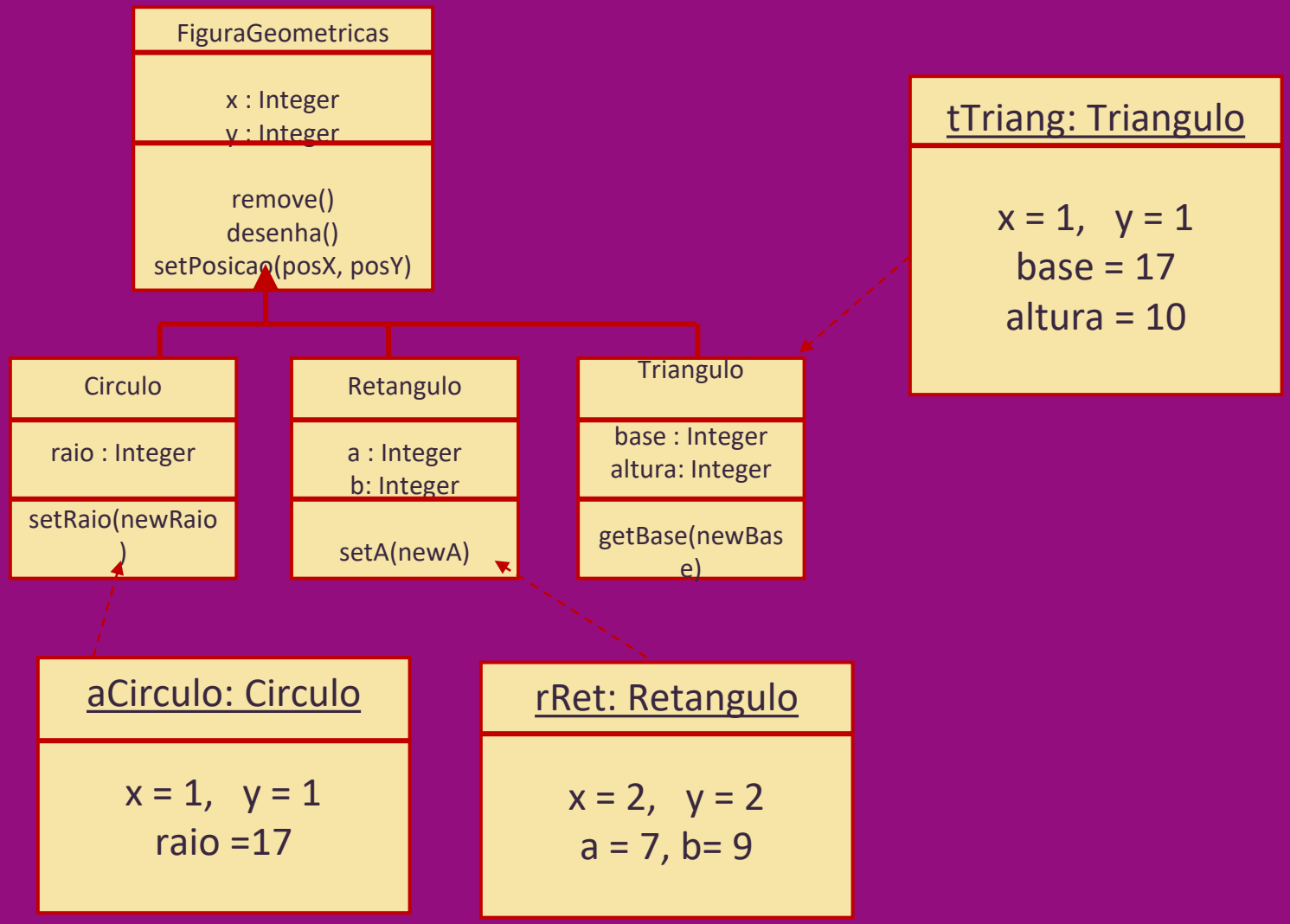


Diagrama de classes

Objetos são as instâncias de classe.

Atributos de classe são informações compartilhadas por todos os objetos da classe.

Atributos de objeto são informações específicas do objeto, usado para definir seu estado.

Operações são serviços que podem ser requeridos de um objeto Sendo descrito por sua assinatura (nome da operação)

Exercício

Considere que as classes Pessoa, Aluno e Professor,

Faça um diagrama de classes usando herança, indique possíveis atributos e operações para cada classe.

Ferrramenta (https://app.diagrams.net/)

Referência bibliográfica

FOWLER, Martin e SCOTT, Kendall. Uml Essencial. 2a. Edição. Bookman. Porto Alegre, 2000.

SCHNEIDER, Geri. Applying use case: a practical guide. Addison-Wesley, 1998.

OESTEREICH, Bernd. Developing Software with UML. Addison-Wesley, 1999.

CRÉDITOS

COORDENAÇÃO

Vera Rejane Niedersberg Schuhmacher

PROFESSORES

Rafael Lessa
Daniella Vieira