- Introdução
- Pacote
- Classe
- Visibilidade
- Atributo
- Método

Encapsulamento de Classes

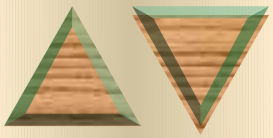
Prof. Dr. Enzo Seraphim

Projeto de Software

Introdução

Arte em se definir fronteiras no mundo computacional que representa o mundo real.

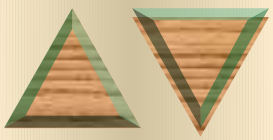
Prof.Enzo Seraphim

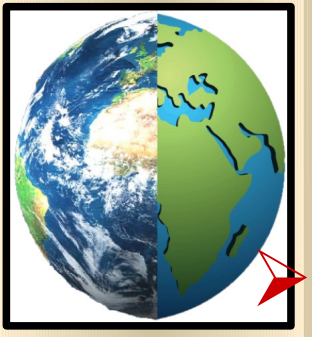


Ciclo desenvolvimento Engenharia de Softwate

Introdução

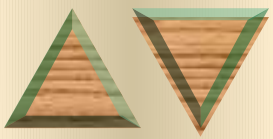
- Análise e Requisitos
- Projeto
- Implementação
- Teste
- Manutenção





Paradigmas Projeto Software

- Análise e Projeto Estruturado
  - 60-70: COBOL, FORTRAN, C, Pascal
- Projeto OO
  - 80: Smalltalk, ADA, C++, Object Pascal
  - 90: Java, C#
- Multiparadigma
  - 2000: C++, Groovy, Oz, Ruby ,Scala , Swift ,Lua, Python, JavaScript

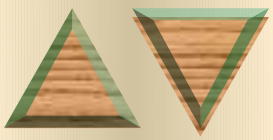


Tecnologia OO

Introdução

- Mais do que um **Modo** de **Programar**
- Modo de pensar abstrato sobre um domínio de problemas
- Usa conceitos do mundo real ao invés de conceitos computacionais
- Baseada em construções chamadas objetos, proporciona um paradigma evolucionário para:

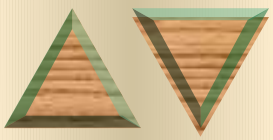
  criar modelos do mundo real em computador
- Usa modelo para simular o mundo real



Projeto OO

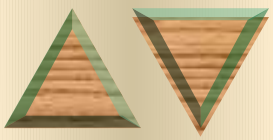
Introdução

- **CRC (Classe-Responsabilidade-Colaborador) 1989 - Wirfs-Brock**
  - *Designing Object-Oriented Software, Prentice-Hall, 1990.*
- **Modelos OOA e OOD (1991) - Coad/Yourdon**
  - *Análise Baseada em Objetos, Editora Campus, 1991*
  - *Projeto Baseado em Objetos, Editora Campus, 1991*
- **OMT (*Object Modeling Technique*) (1991) – James Rumbaugh**
  - *Modelagem Baseada em Objetos - Editora Campus, 1991*
- **BOOCH (1991) – Grady Booch**
  - *Object-Oriented Design with Applications, Benjamin Cummings, 1991*
- **OBJECTORY OOSE (1992) – Ivar Jacobson**
  - *Object-Oriented Software Engineering, Addison-Wesley*
- **Fusion (Booch, OMT, CRC, Métodos Formais) 1994 - Colemann**



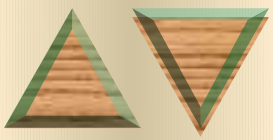
Como surgiu UML?

- 94/95: Parceria de metodologistas:
  - James Rumbaugh (OMT)
  - Grady Booch (Booch Method)
  - Ivar Jacobson (Objectory OOSE Process)
  - Unified Modeling Language (UML)
- 98: Reengenharia de livros, métodos e cases OO para incluir UML



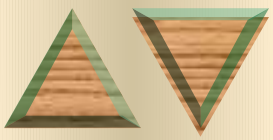
O que é a UML?

- É um padrão aberto
  - versão 1.1 aprovada pelo OMG (Object Management Group) em Novembro de 1997
  - versão 1.3 aprovada em Junho de 1999
- Suporta todo o ciclo de vida do software
  - modelagem do negócio (processos e objetos do negócio)
  - modelagem de requisitos alocados ao software
  - modelagem da solução de software



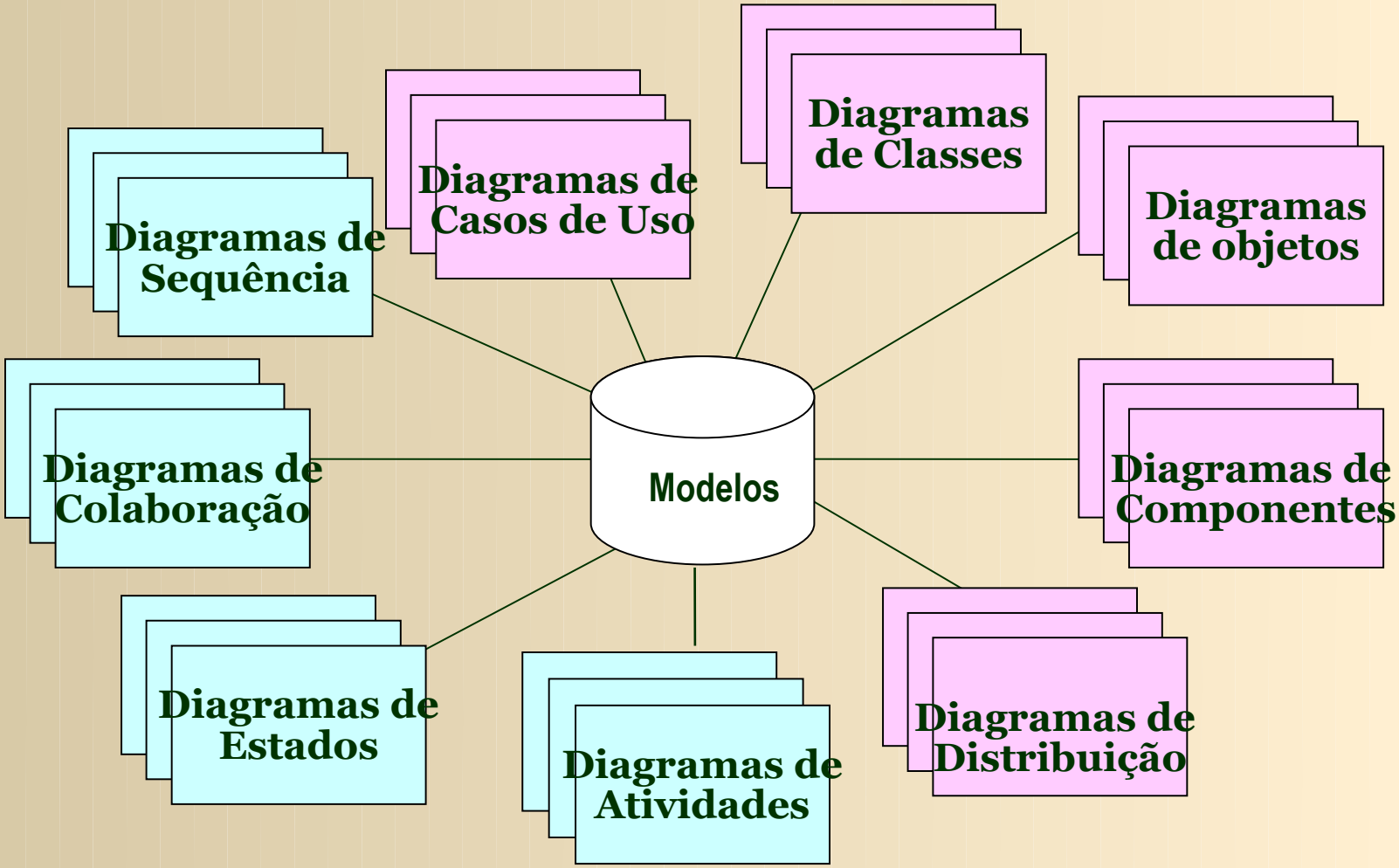
Parceiros da UML

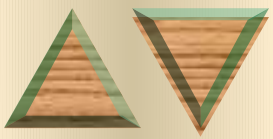
- Rational Software Corporation
- Hewlett-Packard
- I-Logix
- IBM
- ICON Computing
- Intellicorp
- MCI Systemhouse
- Microsoft
- ObjecTime
- Oracle
- Platinum Technology
- Taskon
- Texas Instruments/ Sterling Software
- Unisys



Modelos e Diagramas

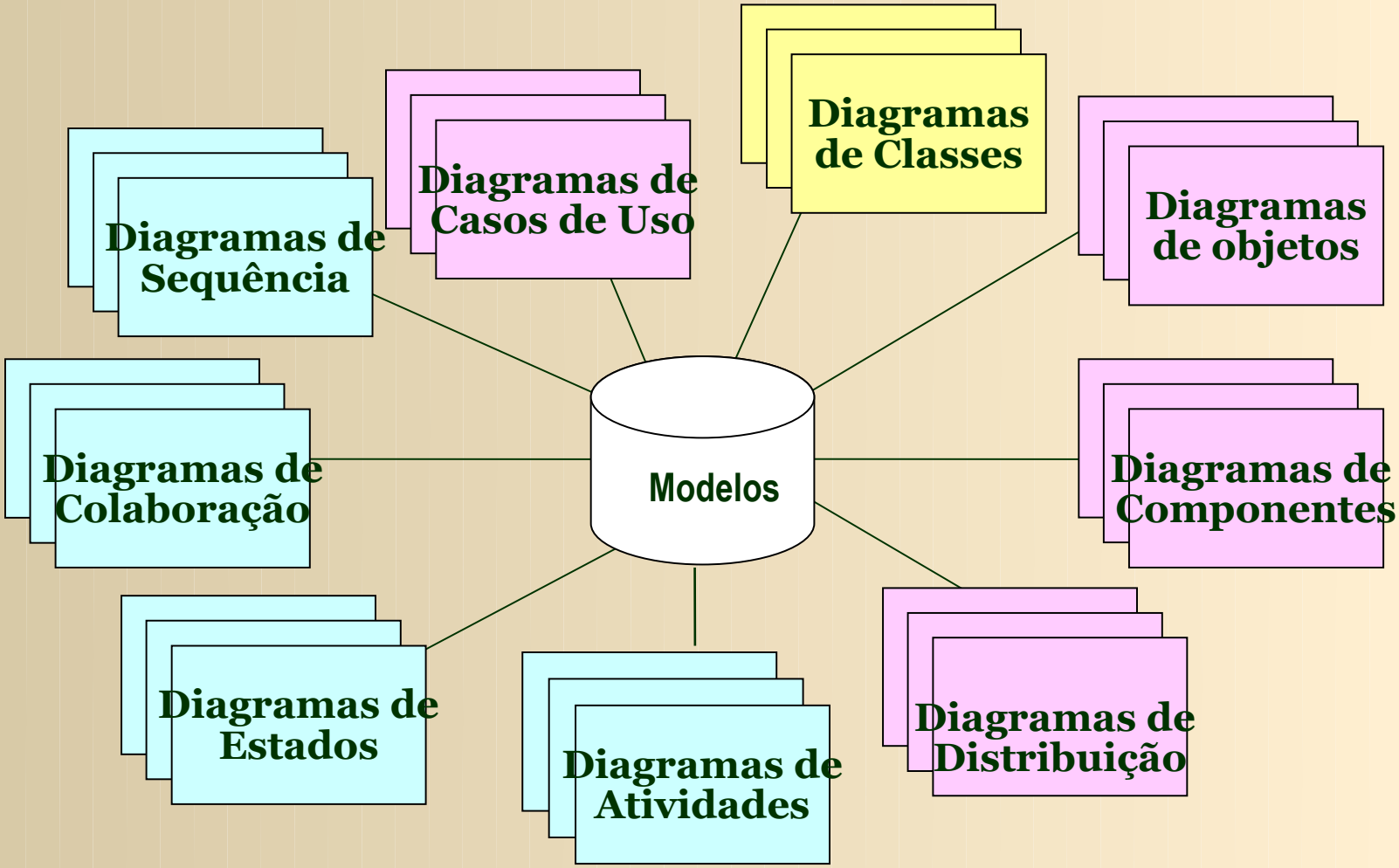
Introdução





Modelos e Diagramas

Introdução



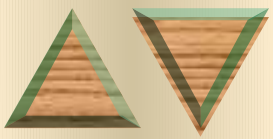


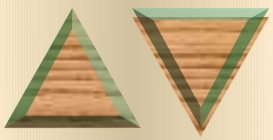
Diagrama de Classes

Introdução

- Modela o vocabulário de um sistema, do ponto de vista do cliente/problema ou do desenvolvedor/solução
- Ponto de vista do cliente/problema
  - Fase de captura e análise de requisitos, em paralelo com a identificação dos casos de uso
- Vocabulário do desenvolvedor/solução
  - Fase de projeto
- Modelos de objetos de domínio representa o negócio do cliente
  - Especifica esquemas lógicos de bases de dados

Objetivo

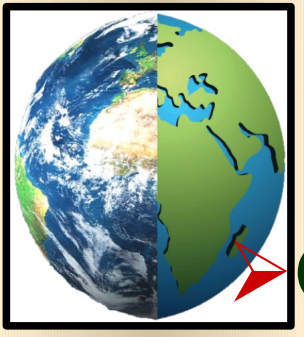
- Também serve para:
  - Especificar colaborações (no âmbito de um caso de utilização ou mecanismo)
  - Especificar esquemas lógicos de bases de dados
  - Especificar visões (estrutura de dados de formulários, relatórios, etc.)
- Modelos de objetos de domínio, negócio, análise e design



- Introdução
- Pacote
- Classe
- Visibilidade
- Atributo
- Método

Encapsulamento de Classes

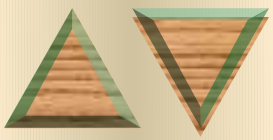
Prof. Dr. Enzo Seraphim



Pacotes

- Organiza as classes de objetos em grupos.
- Melhorar a organização do sistema sub-sistemas
- Estrutura hieraquicamente o projeto
- Estrutura fisica dos arquivos do projeto
- Nomenclatura
  - Minúsculo
  - Endereços de web

Pacote



Regra para Pacote

Pacote

Nunca definirás uma classe

sem antes definir um pacote

Pacotes

Pacote

br.edu.unifei.academico

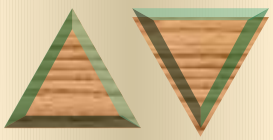
```
package br.edu.unifei.academico;
```

br.com.softwarehouse.sistema

```
package br.com.softwarehouse.sistema;
```

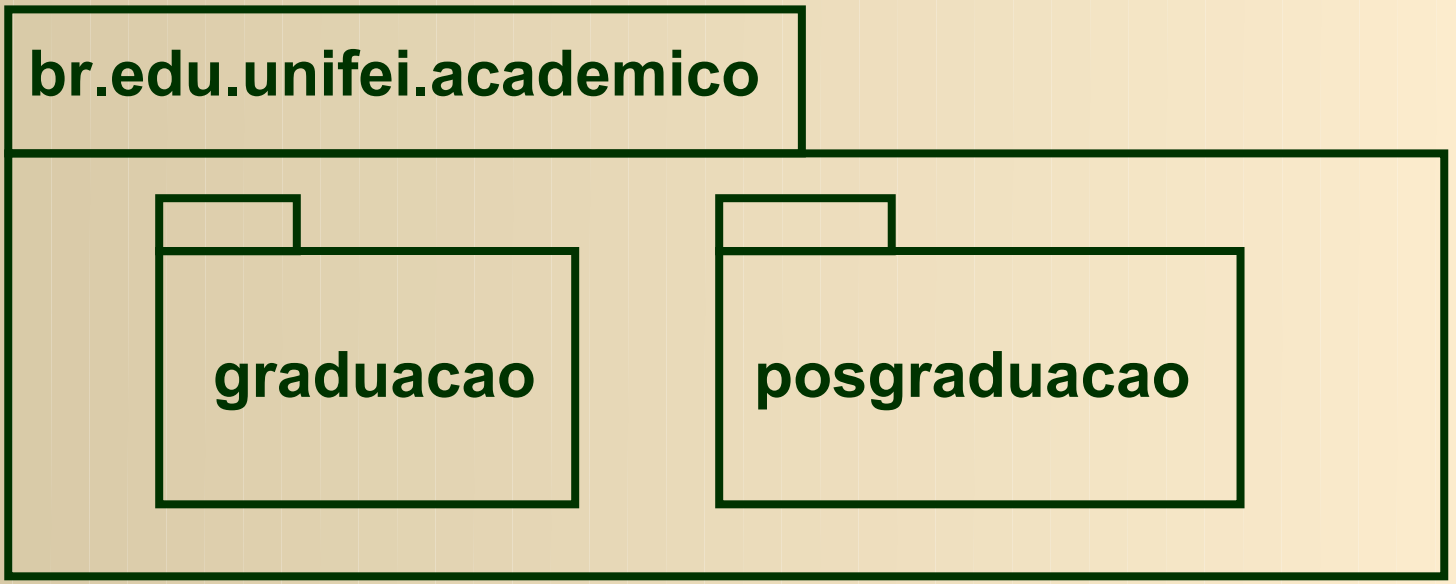
org.kernel.drivers

```
package org.kernel.drivers;
```



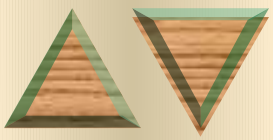
Pacotes

Pacote




```
package br.edu.unifei.academico.graduacao;
```

```
package br.edu.unifei.academico.posgraduacao;
```



- Introdução
- Pacote
- Classe
- Visibilidade
- Atributo
- Método

Encapsulamento de Classes

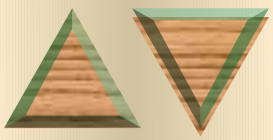
Prof. Dr. Enzo Seraphim

Objetos

Objetos computacionais são imagens de objetos do mundo real que interagem entre si

- Um objeto é algo com fronteiras bem definidas, relevante para o problema em causa
- Exemplos de objetos do mundo real:
  - o Sr. João
  - a aula de ES no dia 11/10/2000 às 11h

Classe

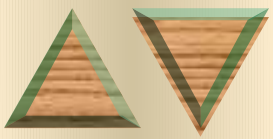


Classes

Classe

- Uma classe é um descritor de um conjunto de objetos que partilham as mesmas propriedades (semântica, atributos, operações e relações)

- Um objeto de uma classe é uma instância da classe



Classes

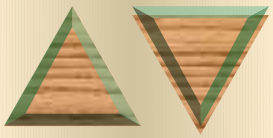
- Em UML, uma classe é representada por um retângulo com o nome da classe
- Escreve-se o nome da classe no singular (nome de uma instância), com a 1ª letra em maiúscula

Classe


```
public class Aluno{}
```


```
public class Triangulo{}
```


```
public class ExpressaoSegundoGrau{}
```



- Introdução
- Pacote
- Classe
- Visibilidade
- Atributo
- Método

Encapsulamento de Classes

Prof. Dr. Enzo Seraphim

Visibilidade atributos e métodos

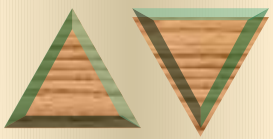
Visibilidade

- Esconder detalhes de implementação que não interessam aos que utilizam as classe
- Permite alterar representação do estado sem afetar clientes
- Tipos de visibilidade:

  \+ (public) : visível na classe, subclasse e objeto

  – (private): visível só na própria classe

  # (protected): visível na classe, subclasse
- Java protected estende visibilidade para todos objetos que estão no mesmo pacote (friend)

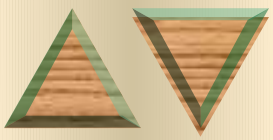


Exemplo

Visibilidade

Circulo
-centroX : double -centroY : double -raio : double
+getCentroX() : double +setCentroX(v:double) : void +getCentroY() : double +setCentroY(v:double) : void +getRaio() : double +setRaio(v:double) : void +area() : double +perimetro() : double

Pessoa
-nome : String -altura : float -peso : float
+getNome():String +setNome(nome:String):void +getAltura() :float +setAltura(altura:float):void +getPeso():float +setPeso(peso:float):void +imc() : double



- Introdução
- Pacote
- Classe
- Visibilidade
- Atributo
- Método

Encapsulamento de Classes

Prof. Dr. Enzo Seraphim

Atributos de instância

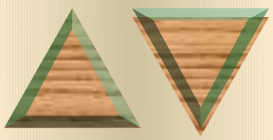
Atributo

- O estado de um objeto é dados por valores de atributos (e por ligações com outros objetos)
- Todos os objetos da classe tem os mesmos atributos, provavelmente com valores diferentes
- Atributos são definidos ao nível de classe
- Valores atributos são definidos ao nível objeto
- Tipos atributos não estão pré-definidos UML
- Nome do atributo (POJO) minísculo e primeira letra da concatenicação de palavra em maiúscula
- Classe não tem dois atributos com mesmo nome
- Visibilidade de atributo, normalmente privada.
- Acesso por método get (leitura) e set (escrita)

Atributos em Classe

Pessoa
-nome : String -altura : float -peso : float
+getNome():String +setNome(nome:String):void +getAltura() :float +setAltura(altura:float):void +getPeso():float +setPeso(peso:float):void +imc() : double

```
public class Pessoa{
 private String nome;
 private float altura;
 private float peso;
 public String getNome(){
  return this.nome;}
 public void setNome(String nome){
  this.nome=nome;}
 public float getAltura(){
  return this.altura;}
 public void setAltura(float altura){
  this.altura=altura;}
 public float getPeso(){
  return this.peso;}
 public void setPeso(float peso){
  this.peso=peso;}
 public double imc(){
  return altura/(peso*peso); } }
```



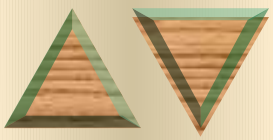
Exemplo atribuição de valores para atributos de instância

João (objeto) é uma pessoa com nome “Enzo”, altura 1,75 e peso “70 Kg”

Pessoa
-nome: String -altura: float -peso: float
+gets +sets

Atributo

```
public class App{
 public static void main(String args[]){
   Pessoa p = new Pessoa();
   p.setNome("Enzo");
   p.setAltura(1.75f);
   p.setPeso(70f);
   System.out.println(p.getNome());
 }  }
```

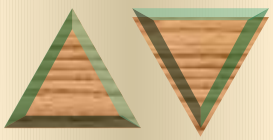


Exemplo

Atributo

ExpressaoSegundoGrau
-a:double -b:double -c:double
+gets +sets

```
public class ExpressaoSegundoGrau{
 private double a;
 private double b;
 private double c;
 //gets sets
}
```

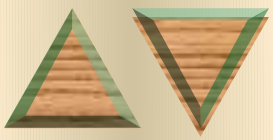


Exemplo

Atributo

Triangulo
-ladoA:double
-ladoB:double
-ladoC:double
-anguloAB:double
-anguloBC:double
-anguloCA:double
+gets +sets

```
public class Triangulo{
 private double ladoA;
 private double ladoB;
 private double ladoC;
 private double anguloAB;
 private double anguloBC;
 private double anguloCA;
 //gets sets
}
```



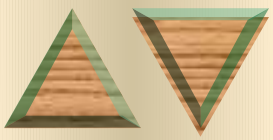
Atributos estáticos

Atributo

- Atributo estático: um único valor para todas instâncias da classe
  - valor definido ao nível da classe e não ao nível das instâncias
- Sublinha o atributo estático

Aluno
-matricula: long -nome: String -totalOnline: int
+gets +sets

```
public class Aluno{
 private long matricula;
 private String nome;
 private static int totalOnline;
 //gets sets matricula e nome
 public static int getTotalOnline(){
  return this.totalOnline;}
 public static void setTotalOnline(int totalOnline){
  this.nome=nome;}
}
```

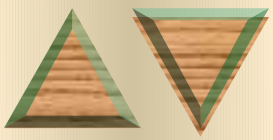


Atributos estáticos

Aluno
-matricula: long -nome: String -totalOnline: int
+gets +sets

Atributo

```
public class App{
 public static void main(String args[]){
  //cadê a instância?
  Aluno.setTotalOnline(10);
  System.out.println(Aluno.getTotalOnline());
 }
}
```



- Introdução
- Pacote
- Classe
- Visibilidade
- Atributo
- Método

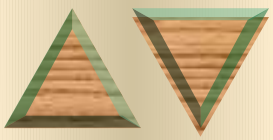
Encapsulamento de Classes

Prof. Dr. Enzo Seraphim

Método

- Comportamento invocável de objetos
- Manipula atributos da classe
- Todos objetos da mesma classe têm as mesmas operações
- Operações são definidos ao nível da classe
- Invocações são definida ao nível do objeto
- Nome do método (POJO) minísculo e primeira letra da concatenicação de palavra em maiúscula

Métodos



Método em Classe

Pessoa
-nome : String -altura : float -peso : float
+getNome():String +setNome(nome:String):void +getAltura() :float +setAltura(altura:float):void +getPeso():float +setPeso(peso:float):void +imc() : double

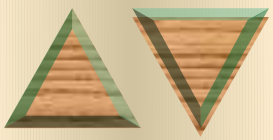
```
public class Pessoa{
 private String nome;
 private float altura;
 private float peso;
 public String getNome(){
  return this.nome;}
 public void setNome(String nome){
  this.nome=nome;}
 public float getAltura(){
  return this.altura;}
 public void setAltura(float altura){
  this.altura=altura;}
 public float getPeso(){
  return this.peso;}
 public void setPeso(float peso){
  this.peso=peso;}
 public double imc(){
  return altura/(peso*peso); } }
```

Exemplo

Métodos

Circulo
-centroX : double -centroY : double -raio : double
+getCentroX() : double +setCentroX(v:double) : void +getCentroY() : double +setCentroY(v:double) : void +getRaio() : double +setRaio(v:double) : void +area() : double +perimetro() : double

Pessoa
-nome : String -altura : float -peso : float
+getNome():String +setNome(nome:String):void +getAltura() :float +setAltura(altura:float):void +getPeso():float +setPeso(peso:float):void +imc() : double



Métodos estáticos

- Não é invocada para um objeto específico da classe
- Não tem instância da classe para invocar o método.
- Sublinha o método estático
- Uso em a cesso leitura e escrita de atributo estático.
- Uso em método que não usa atributo da classe.

Métodos

Triangulo
ladoA : double ladoB : double ladoC : double
hipotenusa(catA: double, catB:dobule):double

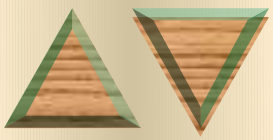
```
public class Triangulo{
private double ladoA, ladoB, ladoC;
//gets sets
public static double pitagora(double catA,double catB)
 return Math.sqrt((catA*catA)+(catB*catB)); } }
```

Métodos estáticos

Triangulo
ladoA : double ladoB : double ladoC : double
hipotenusa(catA: double, catB:dobule):double

Métodos

```
public class App{
 public static void main(String args[]){
  //cadê a instância?
  System.out.println(Triangulo.hipotenusa(3, 4));
 } }
```



Prof.Dr.Enzo Seraphim
seraphim@unifei.edu.br
IESTI/UNIFEI

Encapsulamento de Classes