

---

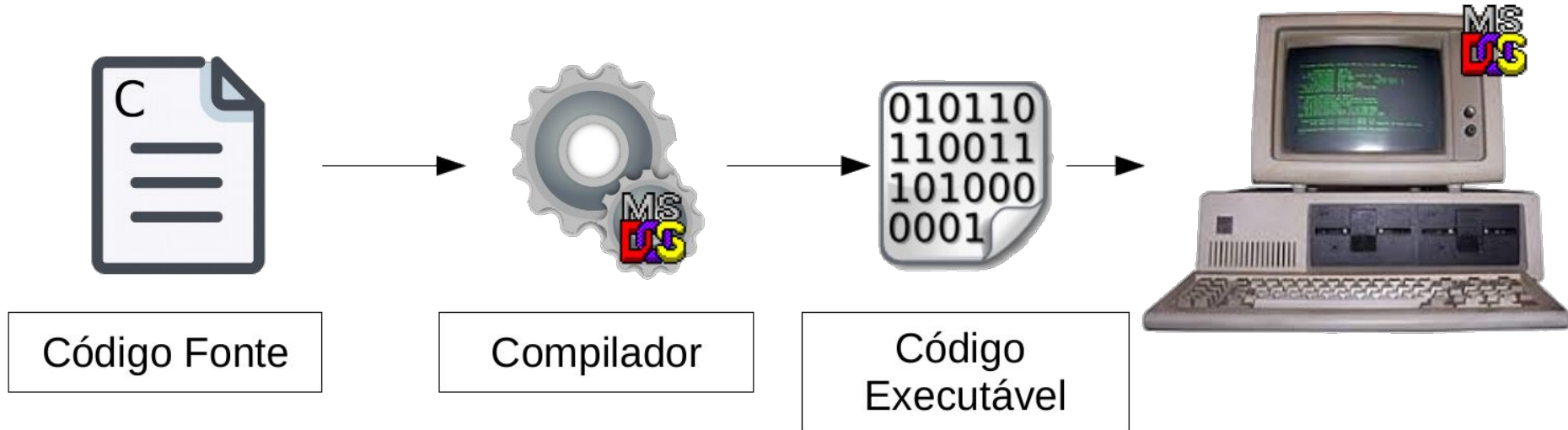
# Programação Orientada a Objetos

Introdução / JAVA

Prof. Leandro Rodrigues Pinto  
<leandrorodp@gmail.com>

---

# A Linguagem de programação Java



# A Linguagem de programação Java



Código Fonte



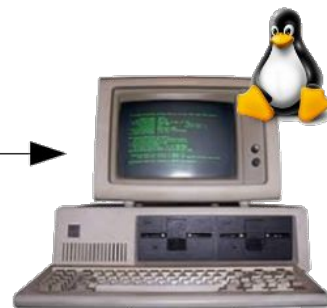
Javac



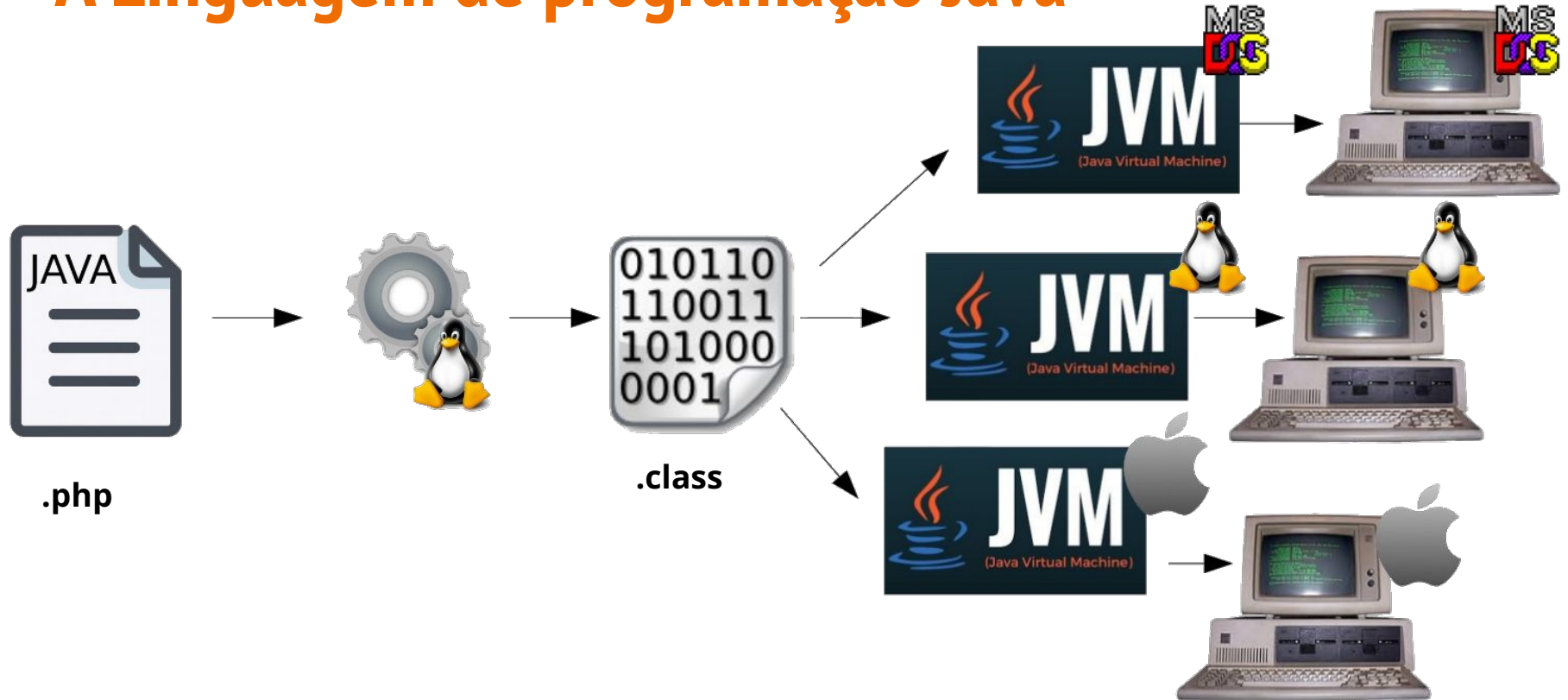
Bytecode



Interpretar



# A Linguagem de programação Java



# A Linguagem de programação Java

**WORA** (Write once, run anywhere) ('Escreva uma vez, execute onde quiser')

Fluxo básico de desenvolvimento e execução:

1. Escreva seu código *.java*
2. Compile com a **JDK** em bytecode *.class* com auxílio da **JRE**
3. Execute o *.class* com a **JVM** com auxílio da **JRE**

**JDK (Java Development Kit)** - é o Kit de Desenvolvimento Java responsável por compilar código-fonte (*.java*) em bytecode (*.class*)

**JVM (Java Virtual Machine)** - é a Máquina Virtual do Java responsável por executar o bytecode (*.class*)

**JRE (Java Runtime Environment)** - Ambiente de Execução do Java que fornece as bibliotecas padrões do Java para o **JDK** compilar o seu código e para a **JVM** executar o seu programa.

# A Linguagem de programação Java

## **Java SE : Standard Edition**

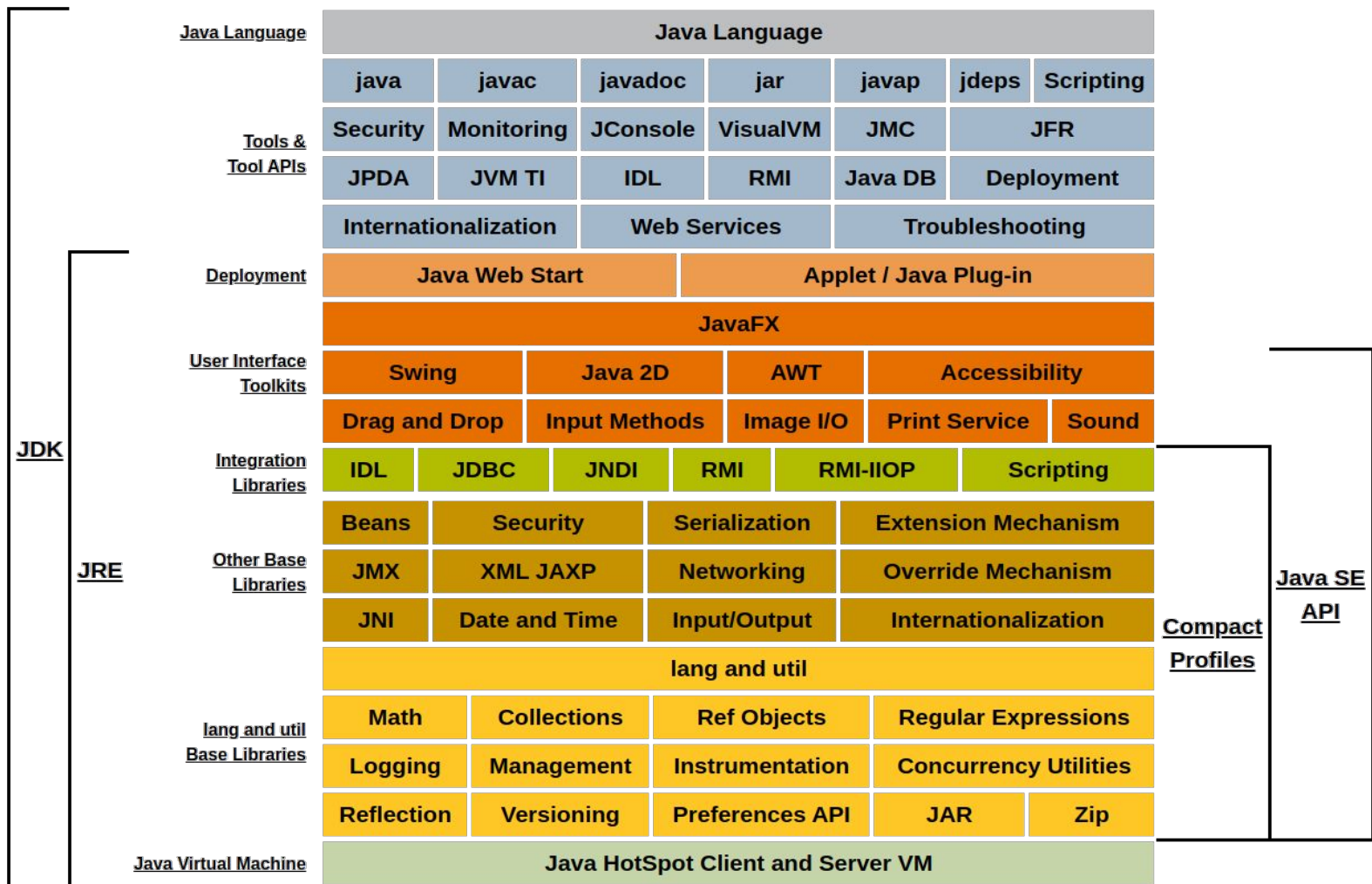
é um kit de desenvolvimento de software usado para criar mini-aplicativos e aplicativos que utilizam a linguagem de programação Java.

## **Java EE : Enterprise Edition**

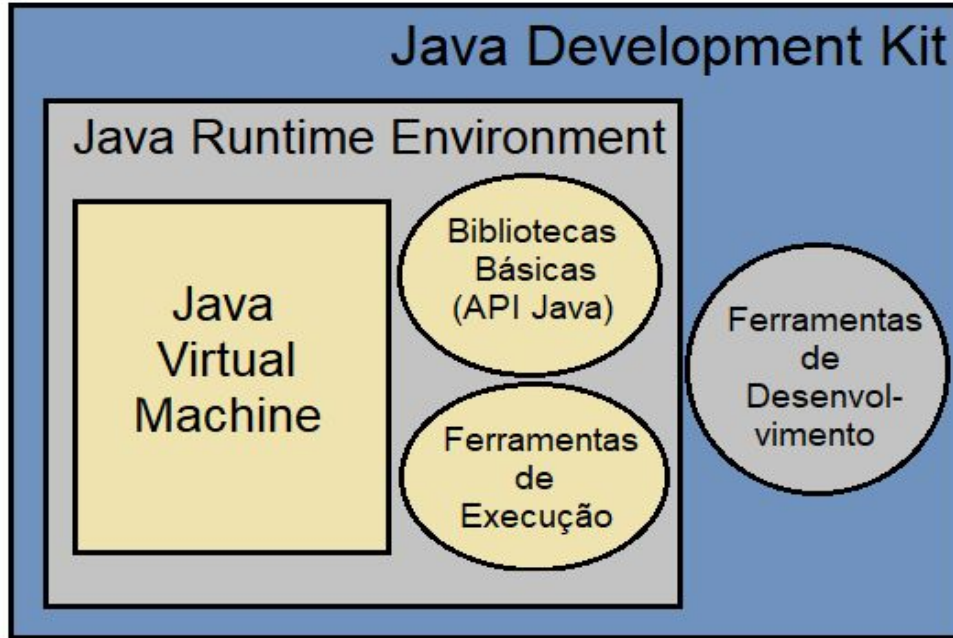
é um ambiente independente da plataforma, centrado em Java que cria e implementa aplicativos corporativos baseados na Web on-line.

## **Java ME : Micro Edition**

oferece um ambiente robusto e flexível para aplicativos executados em dispositivos móveis e integrados: celulares, set-top boxes, reprodutores de discos Blu-ray, dispositivos de mídia digital, módulos M2M, impressoras etc.



# A Linguagem de programação Java



**IDE: Integrated Development Environment**

**ou**

**Ambiente de Desenvolvimento Integrado**

**Editor, compilador, debugger, testes, gerador de código, etc...**



# A Linguagem de programação Java : IDE's

## O que é uma IDE?

É um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.

Geralmente os IDEs facilitam a técnica de RAD (de Rapid Application Development, ou "Desenvolvimento Rápido de Aplicativos"), que visa a maior produtividade dos desenvolvedores.

# A Linguagem de programação Java : IDE's

## Características

- **Editor** - edita o código-fonte do programa escrito na(s) linguagem(ns) suportada(s) pela IDE;
- **Compilador (compiler)** - compila o código-fonte do programa, editado em uma linguagem específica e a transforma em linguagem de máquina;
- **Linker** - liga (linka) os vários "pedaços" de código-fonte, compilados em linguagem de máquina, em um programa executável que pode ser executado em um computador ou outro dispositivo computacional;
- **Depurador (debugger)** - auxilia no processo de encontrar e corrigir defeitos no código-fonte do programa, na tentativa de aprimorar a qualidade de software;
- **Modelagem (modeling)** - criação do modelo de classes, objetos, interfaces, associações e interações dos artefatos envolvidos no software com o objetivo de solucionar as necessidades-alvo do software final;

# A Linguagem de programação Java : IDE's

## Características

- **Geração de código** - característica mais explorada em Ferramentas CASE, a geração de código também é encontrada em IDEs, contudo com um escopo mais direcionado a templates de código comumente utilizados para solucionar problemas rotineiros. Todavia, em conjunto com ferramentas de modelagem, a geração pode gerar praticamente todo o código-fonte do programa com base no modelo proposto, tornando muito mais rápido o processo de desenvolvimento e distribuição do software;
- **Distribuição (deploy)** - auxilia no processo de criação do instalador do software, ou outra forma de distribuição, seja discos ou via internet;
- **Testes Automatizados (automated tests)** - realiza testes no software de forma automatizada, com base em scripts ou programas de testes previamente especificados, gerando um relatório, assim auxiliando na análise do impacto das alterações no código-fonte. Ferramentas deste tipo mais comuns no mercado são chamadas robôs de testes;

# A Linguagem de programação Java : IDE's



# Exercícios de revisão

1. **Descreva através de um algoritmo o processo de troca de uma lâmpada queimada.**

Tempo: 10 min

# Exercícios de revisão

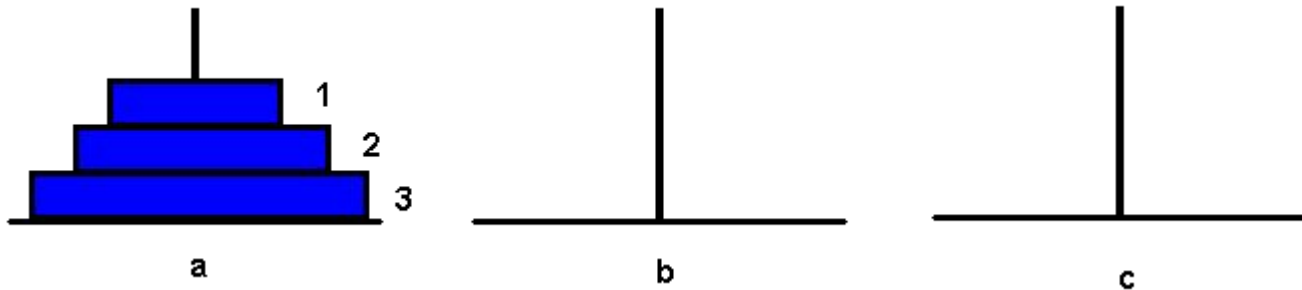
2. Um homem precisa atravessar um rio com um barco que possui capacidade de transportar apenas ele mesmo e mais uma de suas três cargas, que são: um lobo, um bode e uma caixa de alfafa. Indique as ações necessárias para que o homem consiga atravessar o rio sem perder suas cargas.
- a. O lobo não pode ficar sozinho com o bode, senão ele o come;
  - b. O bode não pode ficar sozinho com a caixa de alfafa, senão a come;

Tempo: 10 min

# Exercícios de revisão

3. Elabore um algoritmo que mova três discos de uma Torre de Hanói, que consiste em três hastes (a-b-c), uma das quais serve de suporte para três discos de tamanhos diferentes (1-2-3), os menores sobre os maiores. Pode-se mover um disco de cada vez para qualquer haste, contanto que nunca seja colocado um disco maior sobre um menor. O objetivo é transferir os três discos para outra haste.

Tempo: 10 min



# Exercícios de revisão

## 2. Atravessar o rio

início

atravessar homem e bode

voltar homem

atravessar homem e lobo

voltar homem e bode

atravessar homem e alfafa

voltar homem

atravessar homem e bode

fim

## 3. Torre de Hanói

início

mover o disco 1 para a haste b

mover o disco 2 para a haste c

mover o disco 1 para a haste c

mover o disco 3 para a haste b

mover o disco 1 para a haste a

mover o disco 2 para a haste b

mover o disco 1 para a haste b

fim

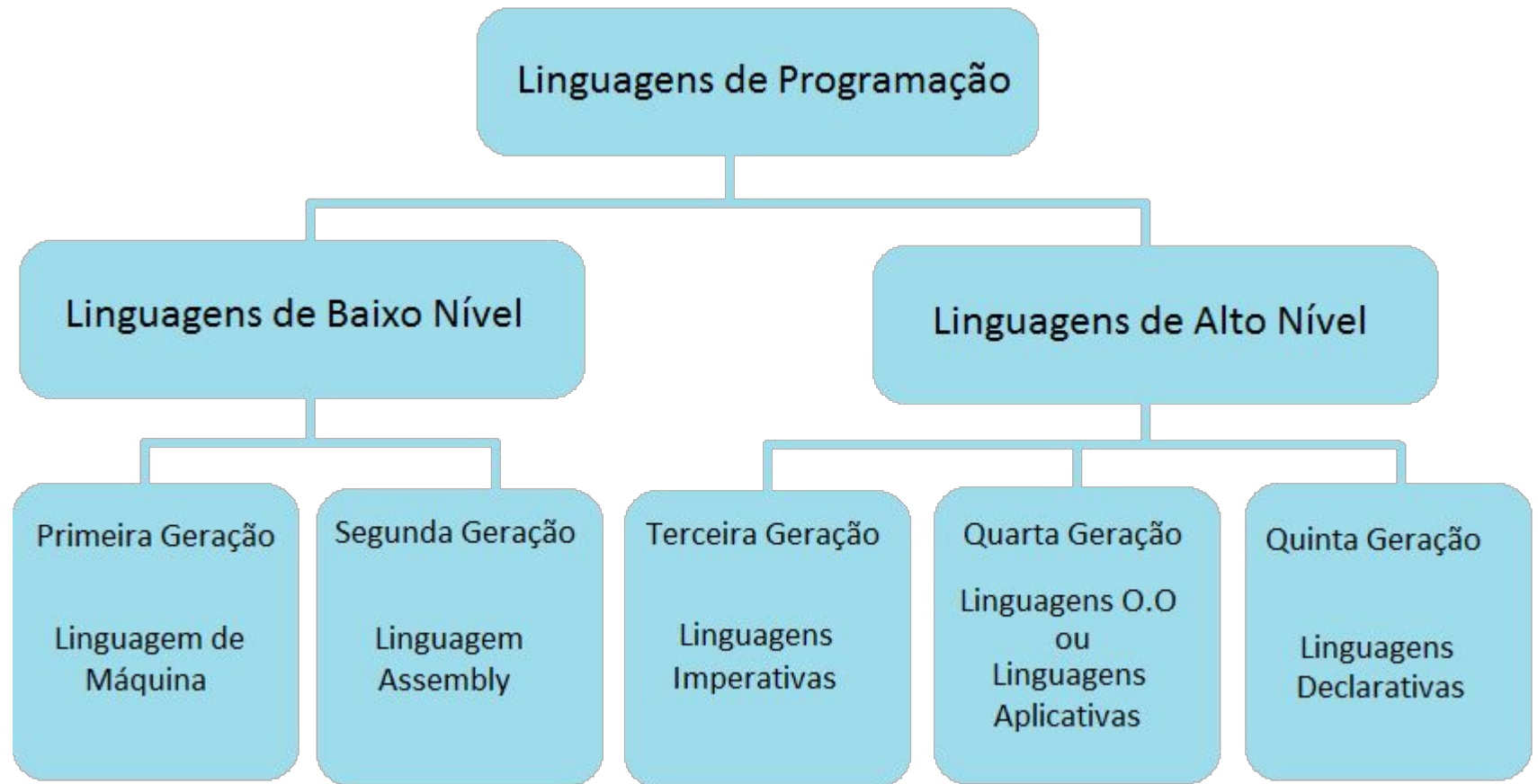


# Paradigmas de Programação

De acordo com Houaiss, Franco e Villar (2001, p. 329), paradigma significa modelo, padrão. No contexto da programação de computadores, um paradigma é um jeito, uma maneira, um estilo de se programar.

- **Paradigma é um padrão de raciocínio para resolução de problemas!**
- **Muitas linguagens suportam mais de um paradigma**
- **Estudaremos neste semestre o paradigma orientado a objetos**

# Paradigmas de Programação



# Paradigmas de Programação



# Paradigmas de Programação

## Modelo imperativo

- Linguagens expressam seqüências de comandos que realizam transformações sobre dados
- Base: máquina de von Neumann
  - orientadas a procedimentos
  - orientadas a objetos

## Modelo declarativo

- Linguagens que não possuem os conceitos de
  - seqüências de comandos
  - atribuição
- linguagens funcionais: ênfase em valores computados por funções
- linguagens lógicas: ênfase em axiomas lógicos

$x = x + 1$

[identificação] [atribuição] [expressão]

# Paradigmas de Programação : Imperativo

## Vantagens:

- Eficiência
- Paradigma dominante e bem estabelecido
- Método 'receita de bolo'

## Desvantagens

- Difícil legibilidade
- Descrições muito operacionais
- Foca muito no **COMO** e não no **O QUÊ**

# Histórico do paradigma de orientação a objetos

O criador da expressão programação orientada a objetos (POO) foi **Alan Kay**, o mesmo que criou a linguagem **Smalltalk**.

Antes de o termo ter sido criado, ideias conceituais sobre orientação a objetos já estavam sendo aplicadas na linguagem de programação **SIMULA 67** (DOUGLAS, 2015).

Essa linguagem era usada para criar simulações.


Alan Kay, que atuava na Universidade de Utah naquela época, gostou do que viu na SIMULA.

Consta que ele teria vislumbrado um computador pessoal que pudesse fornecer aplicações orientadas a gráficos e intuiu que uma linguagem como a SIMULA poderia oferecer bons recursos para leigos criarem tais aplicações.

No início dos **anos 1970** sua equipe criou o primeiro computador pessoal, o Dynabook.

A linguagem Smalltalk, que era orientada a objetos e também orientada a gráficos, foi desenvolvida para programar o Dynabook. Ela existe até hoje, embora não seja largamente usada para fins comerciais.

## Orientação a Objetos

A diagram illustrating the four pillars of Object-Oriented Programming. It features a classical architectural style with four yellow columns supporting a horizontal beam. The beam is labeled 'Orientação a Objetos'. The columns are labeled 'Encapsulamento', 'Herança', 'Composição', and 'Polimorfismo' from left to right. The columns rest on a base labeled 'Princípio da abstração'. The entire structure is set against a light blue circular background with a green oval base.

Encapsulamento

Herança

Composição

Polimorfismo

Princípio da abstração



**Encapsulamento**

**Herança**

**Composição**

**Polimorfismo**

**Abstração**

**Objetos**





**Encapsulamento** é a característica da orientação a objetos capaz de ocultar partes (dados e detalhes) de implementação interna de classes do mundo exterior.



**Herança** é o mecanismo pelo qual uma classe pode "**herdar**" as características e métodos de outra classe para expandi-la ou especializar de alguma forma.



**Composição** é a característica da orientação a objetos que permite combinar objetos simples em objetos mais complexos.



**Polimorfismo** é a habilidade de objetos de classes diferentes responderem a mesma mensagem de diferentes maneiras.

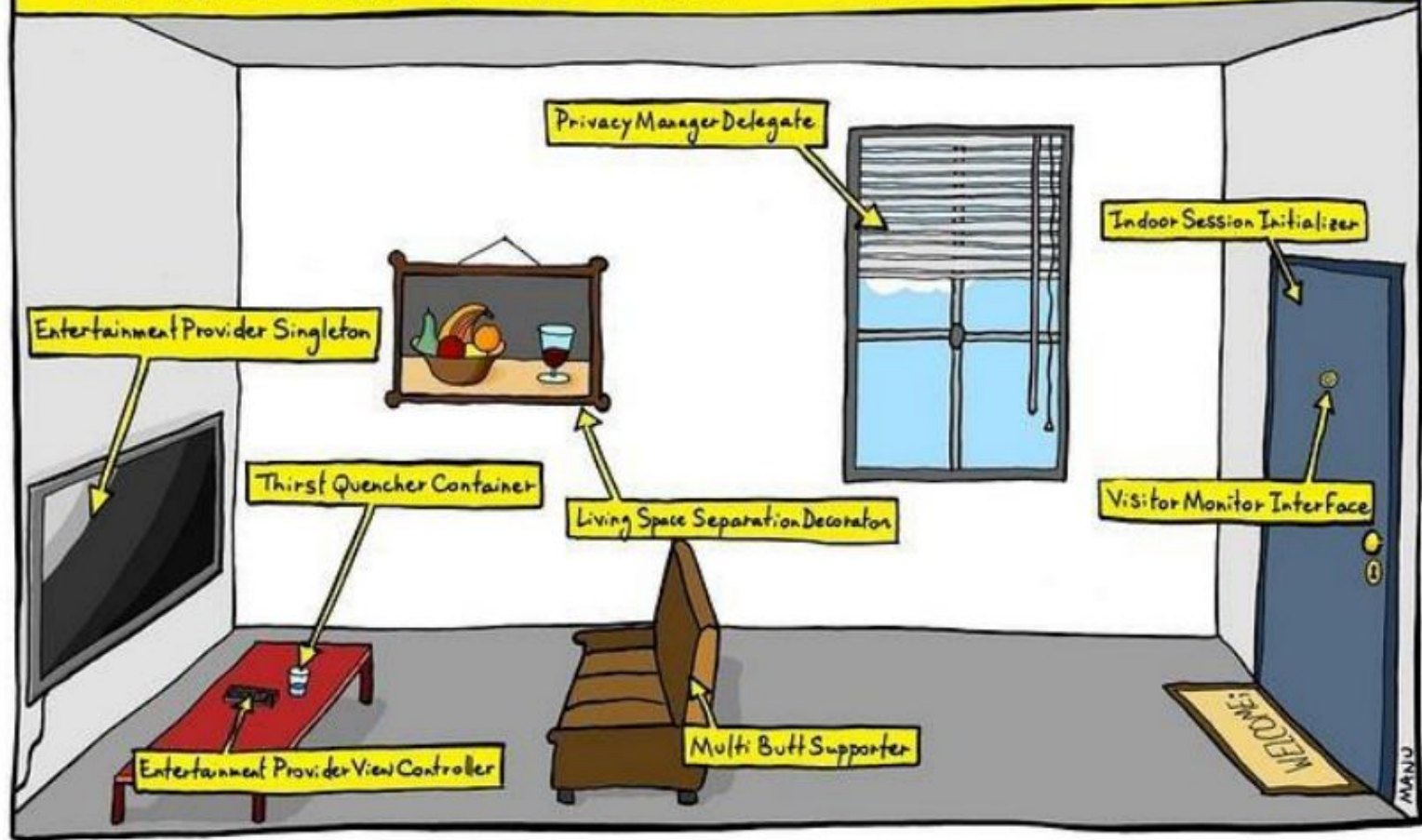
# Orientação a objetos

Consiste em um paradigma de análise, projeto e programação de sistemas baseado na composição e interação entre diversas unidades de software chamadas de objetos.

Sugere a **diminuição** da distância entre a modelagem computacional e o mundo real:

- O ser humano se relaciona com o mundo através de conceitos de objetos;
- Estamos sempre identificando qualquer objeto ao nosso redor;
- Para isso lhe damos nomes, e de acordo com suas características lhes classificamos em grupos;

# THE WORLD SEEN BY AN "OBJECT-ORIENTED" PROGRAMMER.



# Orientação a objetos

- Surgiu na tentativa de solucionar problemas complexos existentes através do desenvolvimento de sistema **menos complexos, confiáveis** e com **baixo custo** de desenvolvimento e manutenção.
- Por que usar a orientação a objetos?
  - Organização do código;
  - Aumenta a reutilização de código;
  - Reduz tempo de manutenção de código;
  - Reduz complexidade através da melhoria do grau de abstração;
  - Ampla utilização comercial;

# Introdução à UML

- UML (Unified Modelling Language)
- É uma linguagem para especificação, construção, visualização e documentação de sistemas.
- É uma evolução das linguagens para especificação de conceitos de Booch, OMT e OOSE e também de outros métodos de especificação de requisitos de software orientados a objetos ou não.



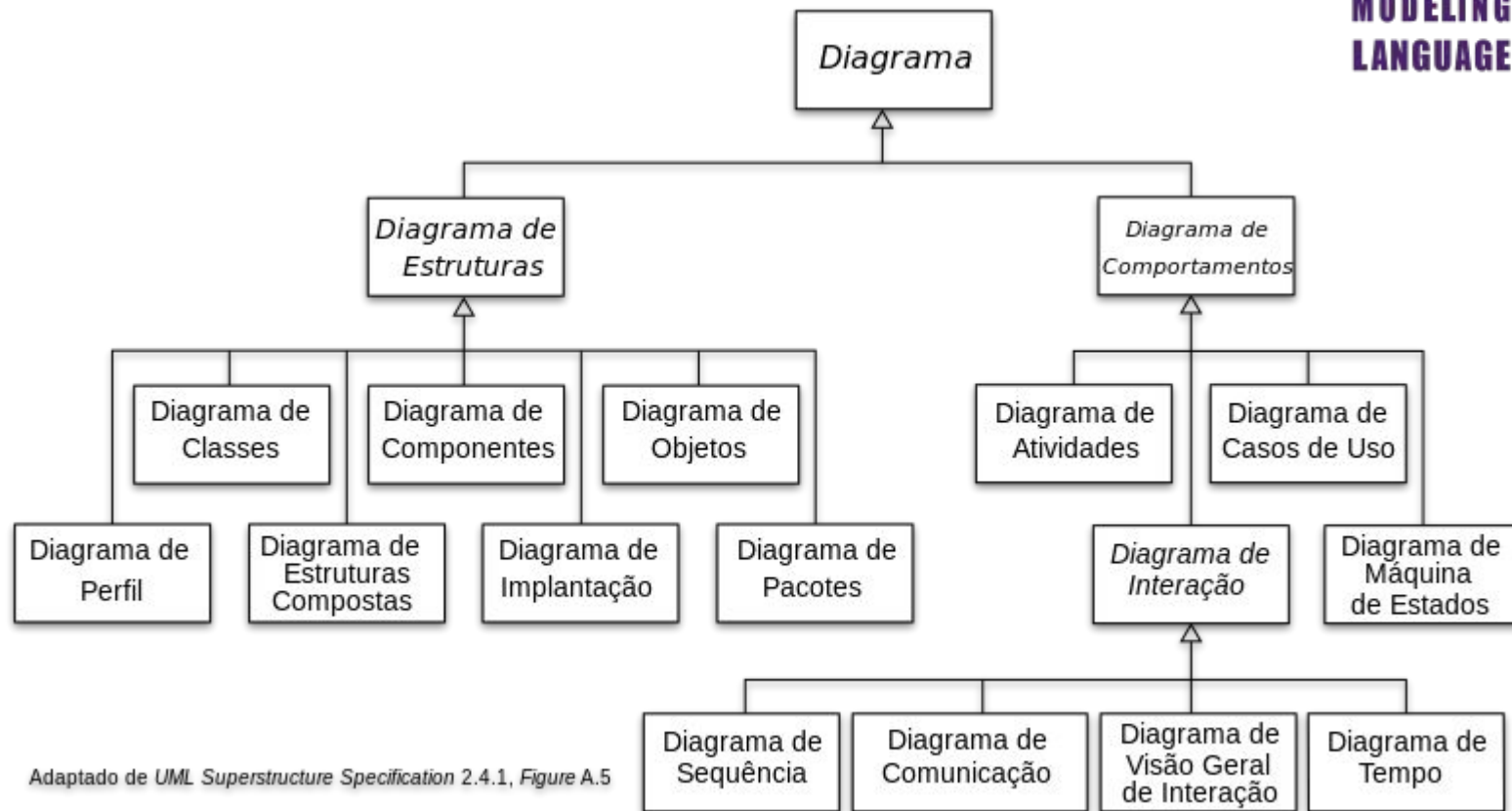


# Histórico da UML

- Início em Outubro de **1994**: Booch e Jim Rumbaugh começaram um esforço para unificar o método de Booch e OMT (Object Modeling Language).
- Uma primeira versão, chamada Unified Method, foi divulgada em outubro de **1995**.
- Jacobson juntou-se ao grupo, agregando o método OOSE (Object-Oriented Software Engineering) .
- O esforço dos três resultou na liberação da UML versão 0.9 e 0.91 em junho e outubro de **1996**. Em janeiro de **1997**, foi liberada a versão 1.0 da UML.
- Adotada como padrão segundo a OMG (Object Management Group, <http://www.omg.org/>) em Novembro de **1997**
- UML 2.0 em **2004**

# Introdução à UML

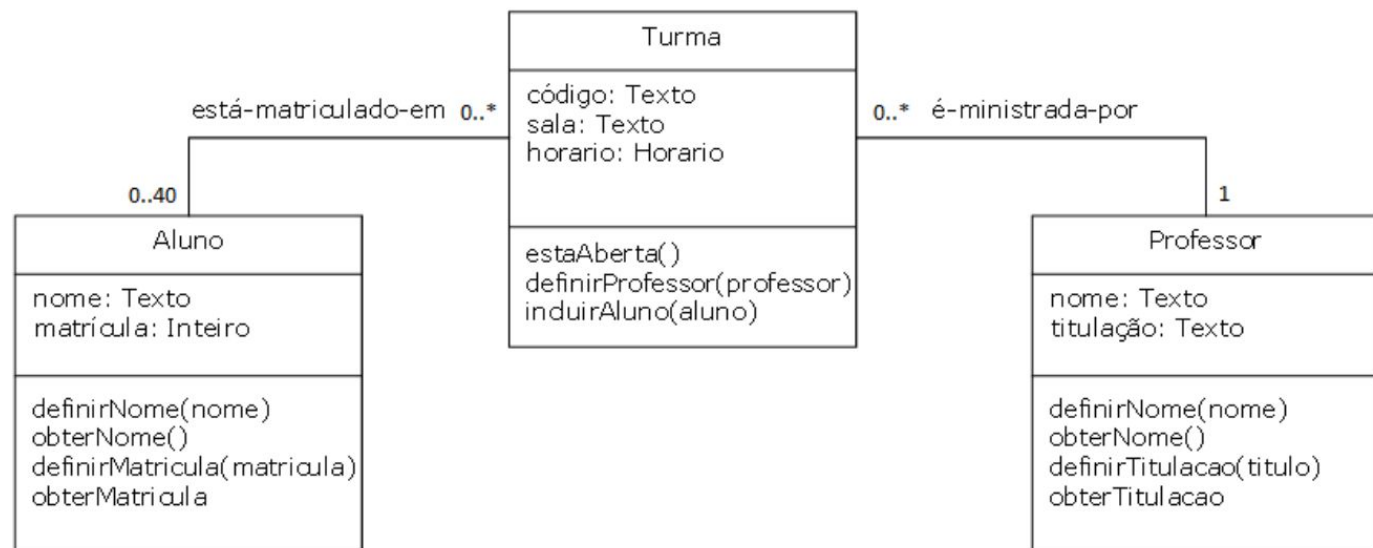
UNIFIED  
MODELING  
LANGUAGE™



Adaptado de UML Superstructure Specification 2.4.1, Figure A.5

# UML - Diagrama de classes

- Mostra um conjunto de classes e seus relacionamentos;
- É o diagrama central da modelagem orientada a objetos;



# Classes e Objetos

- A classe é o **modelo** ou **molde** de construção de objetos. Ela define as características e comportamentos que os objetos irão possuir.
- E sob o ponto de vista da programação:
  - O que é uma classe?
  - O que é um atributo?
  - O que é um método?
  - O que é um objeto?
  - Como o objeto é usado?
  - Como tudo isso é codificado?

# Classes e Objetos

```
public class Carro {
```

```
    private String marca;  
    private String cor;  
    private String placa;  
    private int portas;  
    private int marcha;  
    private double velocidade;
```

```
    public void Acelerar()  
    {  
        velocidade += marcha * 10;  
    }
```

```
    public void Frear()  
    {  
        velocidade -= marcha * 10;  
    }
```

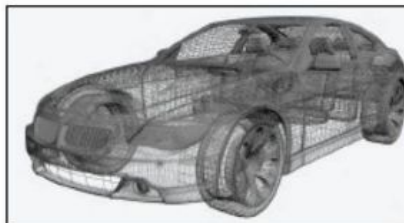
```
    ...
```

```
}
```

Declaração

Atributos

Métodos



Carro

- Marca: Texto
- Cor: Texto
- Placa: Texto
- N° Portas: Inteiro

...

- + Acelerar(): void
- + Frear(): void
- + TrocarMarcha(x): void
- + Buzinar(): void

...

# Perguntas?



**Obrigado!**