



# Programação de Computadores em Java

## Informações em

<http://www.dcc.ufmg.br/~camarao/ipcj/>

Prefácio      Sumário      Errata

Transparências

Livros de referência à linguagem Java

Tutoriais      Bibliotecas (*APIs*)

Ambientes de Programação

Artigos sobre POO e Java





---

---

Visão geral  
contextualização



---

---

Cap. 1: Computadores e Programas

Visão geral

Cap. 2: Paradigmas de Programação

contextualização

Programação baseada em objetos

---



---

---

Cap. 1: Computadores e Programas

Visão geral

Cap. 2: Paradigmas de Programação

contextualização

Programação baseada em objetos

---

Programação imperativa

passo

a passo

---



---

---

Cap. 1: Computadores e Programas

Visão geral

Cap. 2: Paradigmas de Programação

contextualização

Programação baseada em objetos

---

Cap. 3: Primeiros Problemas

Programação imperativa

Cap. 4: Entrada e Saída: Parte I

passo

Cap. 5: Recursão e Iteração

a passo

---



---

---

Cap. 1: Computadores e Programas

Visão geral

Cap. 2: Paradigmas de Programação

contextualização

Programação baseada em objetos

---

Cap. 3: Primeiros Problemas

Programação imperativa

Cap. 4: Entrada e Saída: Parte I

passo

Cap. 5: Recursão e Iteração

a passo

---

Visão geral: POO

---



---

Cap. 1: Computadores e Programas	Visão geral
Cap. 2: Paradigmas de Programação	contextualização
Programação baseada em objetos	
<hr/>	
Cap. 3: Primeiros Problemas	Programação imperativa
Cap. 4: Entrada e Saída: Parte I	passo
Cap. 5: Recursão e Iteração	a passo
<hr/>	
Cap. 6: Classes, Subclasses e Herança	Visão geral: POO

---



---

---

Cap. 1: Computadores e Programas	Visão geral
Cap. 2: Paradigmas de Programação	contextualização
Programação baseada em objetos	

---

Cap. 3: Primeiros Problemas	Programação imperativa
Cap. 4: Entrada e Saída: Parte I	passo
Cap. 5: Recursão e Iteração	a passo

---

Cap. 6: Classes, Subclasses e Herança	Visão geral: POO
---------------------------------------	------------------

---

Mais sobre  
programação  
imperativa

---

---



---

Cap. 1: Computadores e Programas	Visão geral
Cap. 2: Paradigmas de Programação	contextualização
Programação baseada em objetos	
<hr/>	
Cap. 3: Primeiros Problemas	Programação imperativa
Cap. 4: Entrada e Saída: Parte I	passo
Cap. 5: Recursão e Iteração	a passo
<hr/>	
Cap. 6: Classes, Subclasses e Herança	Visão geral: POO
<hr/>	
Cap. 7: Exceções	Mais sobre
Cap. 8: Entrada e Saída: Parte II	programação
Cap. 9: Arranjos	imperativa

---



# Computadores e Programas

- Algoritmos e programas
- Organização de computadores
- Linguagens, compiladores e interpretadores



# Algoritmo e Programa

- **Algoritmo:** Conjunto de regras especificando como realizar uma tarefa
- **Programa:** Representação desse conjunto de regras

Exemplos de “programas”:



# Algoritmo e Programa

- **Algoritmo:** Conjunto de regras especificando como realizar uma tarefa
- **Programa:** Representação desse conjunto de regras

Exemplos de “programas”:

receitas culinárias, instruções de montagem (brinquedos, aparelhos), partituras musicais, programas de computadores.



# Organização de computadores

Arquitetura de von-Neumann:



# Organização de computadores

## Arquitetura de von-Neumann:

- programa a ser executado armazenado na memória



# Organização de computadores

## Arquitetura de von-Neumann:

- programa a ser executado armazenado na memória
- instrução buscada na memória, armazenada em um registrador



# Organização de computadores

## Arquitetura de von-Neumann:

- programa a ser executado armazenado na memória
- instrução buscada na memória, armazenada em um registrador
- e executada



# Organização de computadores

## Arquitetura de von-Neumann:

- programa a ser executado armazenado na memória
- instrução buscada na memória, armazenada em um registrador
- e executada
- (endereço da próxima instrução a ser executada, armazenado em outro registrador, atualizado)



Computadores e programas

# Linguagens, Compiladores e Interpretadores





# Linguagens, Compiladores e Interpretadores

Compiladores são tradutores

- Programas gerados geralmente em linguagem de mais baixo nível.
- Compilação inclui em geral “montagem”: tradução de linguagem de montagem para linguagem de máquina.
- Interpretação pode ser usada em vez de ligação + execução.
- Execução pode ser vista como *interpretação em hardware*



# Linguagens, Compiladores e Interpretadores

Ambientes de programação usuais
editar–compilar–ligar–executar ou
editar–compilar–interpretar

Outras “ferramentas” de software úteis:

- **Depurador:** Ajuda na correção de erros de execução (funções para acompanhamento e impressão de dados de programa em execução)
- **Profilador:** ajuda na análise de gastos em tempo e memória

# Paradigmas



# Paradigmas



**Tradicional** Fortran, Algol, Algol-68, Pascal, C, Cobol, PL/I

# Paradigmas



Tradicional  
OO

Fortran, Algol, Algol-68, Pascal, C, Cobol, PL/I  
Simula-67, Smalltalk, C++, Eiffel, Object Pascal, Java, C#

# Paradigmas



Tradicional	Fortran, Algol, Algol-68, Pascal, C, Cobol, PL/I
OO	Simula-67, Smalltalk, C++, Eiffel, Object Pascal, Java, C#
Funcional	Lisp, ML, Scheme, Miranda, Haskell

# Paradigmas



<b>Tradicional</b>	Fortran, Algol, Algol-68, Pascal, C, Cobol, PL/I
<b>OO</b>	Simula-67, Smalltalk, C++, Eiffel, Object Pascal, Java, C#
<b>Funcional</b>	Lisp, ML, Scheme, Miranda, Haskell
<b>Lógico</b>	Prolog, Mercury