Geraldo César Cantelli

2ª EDIÇÃO

Java

Uma Abordagem sobre Programação Java

editora VIENA

Geraldo César Cantelli

Java
Uma Abordagem sobre Programação Java

editora
VIENA
2ª Edição
Bauru/SP
Editora Viena
2022

Sumário

Lista de Siglas e Abreviaturas

↵	*Indica que a linha de comando é uma continuação da linha anterior, ou seja, as duas linhas são uma única linha de comando.*
API	*Application Program Interface.*
IDE	*Integrated Development Environment.*
JVM	*Java Virtual Machine.*
OO	*Orientação a Objetos.*
POO	*Programação Orientada a Objetos.*
RAM	*Random Access Memory.*
SGBD	*Sistema Gerenciador de Banco de Dados.*
SI	*Sistema de Informação.*

1

Conhecendo a Tecnologia da Informação

1. Conhecendo a Tecnologia da Informação

Prezados leitores, a presente obra tem o objetivo de iniciar o aluno no ofício da programação de computadores, visto que vivemos numa sociedade em que as informações se tornaram ponto estratégico para o sucesso das organizações.

Administradores e executivos possuem a responsabilidade de tomar decisões que atendem à missão das empresas e determinar sua posição no mercado. Para tanto, é preciso receber informações gerenciais da maneira correta e no momento certo.

São os sistemas de informação que recebem os dados (informação no seu estado bruto) e os processam por meio dos programas gerando, assim, uma verdadeira informação que guiará os líderes das empresas em suas decisões.

O termo **Informática** surgiu das palavras "**Informação Automática**" e se consolidou como tecnologia vital para os negócios. Atualmente, existe a necessidade crescente de sistemas de informação, porém não há desenvolvedores suficientes para atender toda essa demanda.

O mercado de trabalho de informática oferece vagas atraentes, mas muitas delas não têm profissionais devidamente qualificados para assumi-las. As estatísticas indicam que esses profissionais são disputados pelas corporações.

O hardware (parte física) precisa ser programado para realizar suas tarefas; é nesse momento que surge o conceito de software como a parte lógica do sistema (instruções em formato digital que o computador entende).

As Lógicas de Negócios são o "roteiro de ações" ou o algoritmo que ensina a máquina a trabalhar. Esse algoritmo é então transcrito para uma linguagem de programação de computadores. A linguagem humana é considerada "linguagem de alto nível", e a que o hardware é capaz de entender é a linguagem binária (zeros e uns), chamada "linguagem de baixo nível" (não no sentido pejorativo).

Portanto, o programador é aquele profissional que segue uma lógica (a partir das necessidades do cliente) e a transcreve em comandos de uma determinada linguagem de programação – código-fonte – que, por sua vez, é compilada para linguagem de máquina. Esse processo gera um código executável capaz de ser processado pela CPU (Central Processing Unit, em inglês, e Unidade Central de Processamento, em português) que nada mais é do que o microprocessador (cérebro do computador) que as coloca em execução.

1.1. Linguagens Compiladas versus Interpretadas

O processo de construção do compilador é complexo, pois nele os comandos do código-fonte, definidos pela empresa que fez a linguagem de programação, tornam-se "ordens em linguagem de máquina".

Quando as instruções do código-fonte de um programa são todas compiladas e geram-se um ou mais arquivos executáveis, essa linguagem é considerada compilada e esses arquivos são gerados de uma vez, completos.

Mas existe outro tipo de processo que consiste na compilação e na execução imediata de cada linha de comandos do código-fonte individualmente; quando aquele trecho de código termina de ser executado, o sistema segue para compilar a próxima linha, repetindo-se esse mecanismo até chegar ao fim do código-fonte. Nesse caso, fala-se em linguagem interpretada, já que ocorre uma interpretação sequencial de instruções, uma por vez.

Em ambos os casos, se houver necessidade de alteração (ampliação ou correção) do código-fonte, deverá ser repetido todo o processo de compilação.

Como exemplo de linguagem compilada temos: **Embarcadero Delphi**, **Visual Basic** e **C**. Já para as interpretadas podemos citar o assunto principal deste livro: a linguagem **Java**.

1.2. Portabilidade

A portabilidade é uma característica desejada em sistemas de informação; ela significa que uma linguagem de programação pode gerar arquivos executáveis para diferentes plataformas de computação. Por exemplo, o **Visual Basic** gera programas que só rodam em ambiente **Windows**; o **Delphi** também, porém apresenta um correspondente para **Linux** chamado **Kylix**, mas cada um deve compilar no seu ambiente nativo (o **Delphi** no **Windows** e o **Kylix** no **Linux**).

A linguagem **Java** é totalmente portável em função de ela ser interpretada por uma Máquina Virtual Java que, por sua vez, foi criada para várias arquiteturas. O código-fonte é compilado para um código chamado Byte Code e posteriormente executado pela JVM (Java Virtual Machine).

Existem Máquinas Virtuais Java para **Windows**, **Macintosh**, **Unix**, **Linux**, entre outros, e o código-fonte não precisa ser alterado para rodar em todas essas plataformas.

1.3. Termos e Definições em Programação

Paradigmas são as metodologias e técnicas utilizadas para uma determinada atividade. No desenvolvimento de sistemas de informação, podem-se exemplificar dois paradigmas:

- Programação Estruturada.
- Programação Orientada a Objeto.

Portanto, cada paradigma possui sua forma de trabalhar: tanto na fase de análise de sistemas quanto na programação em si.

Exercícios Práticos

1. Responda as questões:

a) Dê exemplos de linguagens de programação estruturadas e também de orientadas a objeto.

b) Seria **Java** pertencente a qual paradigma? Uma dica: **C#** pertence ao mesmo paradigma de **Java**.

c) Qual o nome da primeira linguagem orientada a objetos criada?

Anotações

Anotações

2

Conceitos Iniciais em Programação de Computadores

2. Conceitos Iniciais em Programação de Computadores

Os programas são sequências de instruções impostas aos microcomputadores visando realizar uma ou mais tarefas, que podem ser simples ou complexas. Como o computador só entende a linguagem binária ("bi" significa "dois"), essas mesmas instruções devem ser convertidas para o formato de zeros (0) e uns (1) e, então, entendidas pelo sistema operacional e executadas pelo hardware.

O sistema operacional é o "maestro" que coordena os demais programas e o hardware. É a interface entre os usuários e a máquina. Entre os usuários estão pessoas leigas em informática, que não saberiam operar um computador caso não existisse esse maestro.

Um conjunto de programas que interagem para uma finalidade comum chama-se Sistema de Informação (S.I.). Esses sistemas foram se tornando mais complexos com o passar do tempo e houve a necessidade da criação de uma Engenharia de Software para melhor programá-los, ou seja, em tempo hábil e de maneira eficaz (atingindo os objetivos) e eficiente (realizando os procedimentos corretamente).

Por exemplo, um Sistema Gerenciador de Banco de Dados (SGBD) é um conjunto de programas que visam acessar, consultar, alterar, incluir e excluir itens de uma coleção de dados. Lembrando que os dados são organizados de forma que o software que depende deles tenha um desempenho otimizado e seguro.

Dessa forma, pretende-se demonstrar não apenas que seres humanos são usuários de programas e bancos de dados, mas também que outros sistemas e programas podem interagir, solicitando serviços, dando ordens e recuperando informações entre si.

2.1. Hardware e Software

Houve um desenvolvimento importante no hardware e cabe aos programadores fazer evoluir o software também. Quando um programador otimiza seu código de maneira a fazer a mesma coisa, mas utilizando menos memória ou concluindo a tarefa em menos tempo, tem-se um passo no sentido da evolução de código.

A memória (seja primária como a RAM, ou secundária como o disco rígido) já foi mais cara e escassa, mas mesmo agora é reconhecida a importância vital da economia dela, pois imagine um processo executado em dois segundos para um determinado usuário. Se esse processo está rodando num servidor Web na Internet, o qual atende vários clientes, o que aconteceria se 200 clientes estivessem on-line solicitando esse serviço? Seria só multiplicar 2 segundos por 200 e se teria 400 segundos de tempo de resposta. Quem esperaria, na Web, pela resposta de um site que demorasse tanto?

Se o mesmo processo fosse concluído em 0,2 segundos, esse tempo cairia 90% podendo, assim, atender mais usuários, e a empresa em questão lucraria muito mais.

2.2. Algoritmos

Antes de digitar as linhas de código-fonte dos programas, é necessário ter claros os passos que o computador terá de dar para resolver o problema ou realizar uma tarefa. Esse conjunto de "ordens" e lógica chama-se algoritmo.

Desse modo, existe um algoritmo para, por exemplo, encontrar e exibir os números primos (divisíveis apenas por si mesmos e por 1); assim como também existem boas práticas de programação que ajudam os profissionais nesse trabalho.

Na primeira aula de programação, o professor pode pedir para os alunos fazerem um algoritmo a fim de pegar o maior pedaço de bolo numa festa de aniversário. Claro que não é possível programar esse algoritmo, mas os alunos podem escrevê-lo passo a passo, em português, no papel. O objetivo é fazê-los experimentar a ordenação lógica de passos para resolver um problema.

Quando se descreve em português, pode-se usar o termo "Portugol" (uma mistura de algoritmo e português). Porém, algumas escolas, logo depois da introdução ao tema, iniciam suas aulas aplicando esses algoritmos utilizando a linguagem **C** ou algum outro software próprio para o aprendizado.

No nosso caso, será aplicada a linguagem **Java**.

2.3. Estrutura de um Programa em Java

Como a linguagem **Java** segue a POO (Programação Orientada a Objetos), ela possui a seguinte nomenclatura: os programas são definidos em classes e organizados em pacotes, que são os subdiretórios ou as pastas que se encontram dentro do projeto.

É importante salientar que o nome do arquivo no qual é gravada a classe deve ser idêntico ao nome da classe. Por exemplo: Principal.java → Principal (nome da classe).

Abaixo temos um exemplo da estrutura de uma primeira classe em **Java**:

```
package capitulo02;
public class Principal {
   public static void main(String[] args) {
      System.out.println("Olá Mundo.");
   }
}
```

O termo **package** indica o pacote a que pertence a classe. Podemos ter nomes de classes iguais, porém em pacotes distintos.

Por exemplo, existe a interface **List**, que vem do pacote **java.util**, e a classe **List**, que foi construída em **java.awt**. A primeira indica uma lista de elementos na memória e a segunda pertence a um esquema gráfico.

Na sequência, temos a declaração do nome da classe (Principal) que é pública; isso será explicado na seção **Orientação a Objetos**, mas por enquanto revelamos que **public** é a palavra-chave que indica a maior abrangência e visibilidade da classe, podendo esta ser "vista" de qualquer ponto do sistema.

As classes que estão no mesmo pacote compartilham a visibilidade de seus atributos e métodos (ambos que forem públicos) porém, se estiverem em pacotes diferentes, é necessária a importação das classes, o que nada mais é do que declarar o caminho em que elas se encontram. Um exemplo de tal procedimento é:

```
import java.awt.List;
import java.awt.*;
```

O primeiro exemplo importa somente a classe **List**, enquanto o segundo importa todas as classes do pacote **java.awt**.

A instrução **import** deve vir logo após a declaração **package**, por exemplo:

```
package capitulo02;

import java.util.List;
import capitulo03.*;

public class Principal {
   public static void main(String[] args) {
            System.out.println("Olá Mundo.");
   }
}
```

O seguinte trecho é a assinatura de um importante método:

```
public static void main(String[] args)
```

A assinatura, como dizemos, é uma declaração que mostra, entre outros aspectos a visibilidade, o tipo de retorno (quando houver), o nome e os parâmetros passados como argumento por quem chamou o método.

Logo após, são abertas e fechadas as chaves { ... }, que indicam respectivamente o início e o término do corpo do método. Há linguagens que, em vez de utilizar chaves, utilizam as palavras-chave **begin** e **end** (como **Delphi**), mas não é o caso do **Java**, **C#**, **PHP**, **C++**, entre outras.

Mas o que é método?

Método é um bloco de código que realiza uma função específica passo a passo. Para o início do estudo de linguagens de programação, nossas classes terão apenas o método **main**, dentro do qual estarão todas as instruções.

É uma tradição na literatura de informática que o primeiro exemplo de um programa (ou no nosso caso classe) exiba a frase "Olá Mundo.", e é exatamente isso que o seguinte comando faz:

```
System.out.println("Olá Mundo.");
```

O texto a ser impresso deve vir entre aspas duplas (" "), pois isso significa que é uma String (tipo de dados dos textos) e, como a saída padrão do computador é o monitor, será apresentado na tela.

2.4. Case Sensitive

Algumas linguagens de programação são case sensitive, ou seja, são sensíveis a letras maiúsculas e minúsculas. O **Java** é uma delas.

Observe as seguintes palavras:

- “Tartaruga”
- “tartaruga”
- “tarTaruGa”

Elas são consideradas diferentes, e não geram conflito entre si.

Haveria conflito se duas variáveis camadas **numTotal** fossem declaradas no mesmo bloco de código. Porém, se forem declaradas em métodos diferentes ou classes diferentes, não haverá problema.

Observe o comando a seguir:

```
System.out.println("Olá Mundo.");
```

Note que o "S" de System é escrito em maiúsculo e o resto do comando em minúsculo. Se não for assim, a classe sequer será compilada (quanto mais executada). No conteúdo de texto entre aspas ("Olá Mundo."), não é obrigatória a escrita com as iniciais maiúsculas, porém será exibido na tela exatamente da forma que for escrito.

2.5. Declaração e Uso de Variáveis

Para explicar o conceito de variáveis, pode-se compará-las a caixinhas que contêm uma informação dentro. Por exemplo, se pedirmos para o usuário digitar sua idade, podemos armazená-la em uma variável do tipo primitivo para o número inteiro (int).

Para isso, ela deveria ser declarada, que consiste na apresentação que determina seu tipo e nome. Uma vez declarada, fica reservado um espaço na memória principal para esse valor e, enquanto ela for necessária, seu lugar fica fixo na memória, podendo seu conteúdo variar ou não de acordo com as instruções do programador.

Exemplo:

```
package capitulo02;
public class Declaracao {
   public static void main(String[] args) {
      int num;
      num = 0;
      int numero = 0;
   }
}
```

Nessa classe, são declaradas duas variáveis numéricas inteiras chamadas respectivamente **num** e **numero**.

Repare que em programação não devemos usar acentuação gráfica ou ç (c cedilha), por exemplo:

- Nome de pacotes (capitulo02).
- Classes (Declaracao).
- Métodos e variáveis (numero).

A declaração consiste em escrever primeiro o tipo da variável, espaço em branco e o nome da variável. No exemplo anterior, as duas são do tipo inteiro.

Java é uma linguagem "fortemente tipada", quer dizer, há um tipo para cada elemento da programação. Os tipos primitivos são predefinidos. São eles:

- **byte**.
- **short**.
- **int**.
- **long**.
- **float**.
- **double**.
- **char**.
- **boolean**.

Em que **byte**, **short**, **int** e **long** são números inteiros e ocupam, respectivamente, 1, 2, 4 e 8 bytes, portanto, o tipo **long** pode armazenar valores bem maiores que **int**. O objetivo dessa explicação é saber quando usar um ou outro, pois cada um, devido à sua capacidade de armazenamento, possui uma faixa de valores possíveis.

Cada byte possui 8 bits que podem valer 0 ou 1. Então 1 byte pode ter 256 combinações diferentes formando números inteiros que vão de -128 até 127.

O tipo **char** é para caracteres (apenas 1) e, quando receber valor, este deve vir entre aspas simples (' '). Por exemplo:

```
char letra = 'F';
```

Para finalizar a apresentação dos tipos primitivos, temos o **boolean**, que significa "valor lógico" e pode ser verdadeiro ou falso. Se for verdadeiro, será representado pela palavra reservada **true** e, se for falso, pela palavra reservada **false**.

Pode ser usado para variáveis no interior de instruções de decisão (teremos uma seção somente para explicá-las). Segue um exemplo:

```
boolean restituicao = true;
if (restituicao) {
   System.out.println("Receber restituição");
}
else {
   System.out.println("Não receber restituição");
}
```

Exercícios Práticos

1. Pesquisa para entender melhor:

a) Sabendo que a parte física é o Hardware e que a parte lógica é o Software, dê exemplos para cada uma dessas partes.

b) Não se esqueça de incluir aí o Peopleware (que somos nós pessoas), pois não existe apenas o usuário final, mas também vários profissionais envolvidos na computação. Dê exemplos deles. Você se considera um deles? Explique.

2. Crie uma tabela com a faixa de valores possíveis para os tipos **byte, short, int e long.**

3. O que significa **Casting** em programação?

4. Vamos estender nosso conhecimento para os números reais, respondendo quantos bytes ocupam na memória os tipos **float** e **double**.

Anotações