Introdução à linguagem Java

Apresentação

O mundo está se transformando rapidamente e acompanhar a evolução tecnológica tem sido um desafio não trivial para muitas pessoas.
O mercado de trabalho está em busca de profissionais cada vez mais qualificados.

Quando se fala sobre o mercado de trabalho na área
de tecnologia, de forma geral, e em programação de computadores,
de forma mais específica, a situação é boa e ruim ao mesmo tempo.
Por um lado, são necessários profissionais altamente qualificados, porém a demanda é gigantesca e a tendência é de crescimento.

Nesta Unidade de Aprendizagem, você começará sua jornada por
uma das linguagens mais populares do mercado: Java. Primeiramente, você verá um pouco da história e da filosofia dessa linguagem e, depois, poderá ver um programa simples em Java sendo executado, o qual será explicado em detalhes. Por último, você estudará sobre os fundamentos básicos da linguagem Java e suas principais características.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Descrever a história, a filosofia e a terminologia da linguagem Java.
- Desenvolver um programa simples em Java.
- Explicar a sintaxe da linguagem Java.

Desafio

A lógica de programação (LP) é, talvez, o mais importante assunto dentro das disciplinas da área de tecnologia. Qualquer pessoa que deseja ser programador, analista de sistema ou arquiteto de *software* deve resolver problemas, computacionalmente viáveis, por meio de um algoritmo. Com essa lógica, se aprende facilmente qualquer linguagem de programação, pois a base de todas as linguagens é justamente a LP.

A LP é uma metodologia para a resolução de problemas que podem ser resolvidos por um computador. Além disso, ela visa a fazer isso de forma sistemática e com passos finitos. Um profissional que domina LP consegue resolver problemas de forma sistemática, criativa e lógica. Muitas vezes, é preciso realizar tarefas com poucos recursos e fazer valer a LP.

Sendo assim, imagine que o líder da sua equipe de trabalho lhe solicitou um método que fizesse algo bem específico: um programa em Java que receba do usuário um número com 3 algarismos (123,
435 ou 786, etc.) e imprima esse número ao contrário (321, 534 ou 687,
e assim por diante). Contudo, você só pode usar operações aritméticas (+, -, *, /, %).

Descreva como você faria e mostre esse código.

Primeiramente, vamos examinar o problema. Há um número para inverter as posições de seus algarismos. Talvez a melhor abordagem seria testar um número simples e, depois, tentar generalizar.

Para simplificar, suponha que o número dado pelo usuário seja 123.
A operação aritmética para obter 3, que é o último número, será: 123 dividido por 10 é igual a 12 e o resto é 3. Então, tem-se uma operação que dá o resto. 123%10 = 3.

Logo, recebe-se do usuário o número; depois, obtém-se o resto da divisão por 10 e divide-se o próprio número por 10; imprime-se o
resto, até que o valor seja menor que 10, pois, desse modo, não se pode mais dividir por 10; logo, imprime-se o que sobrou do valor.
Assim, nesse caso, obtém-se 321.

O código seria:

```
package Testes;
import java.util.Scanner;
public class Desafio {
public static void main(String[] args) {
Scanner numero = new Scanner(System.in);
System.out.println("Digite um número por favor: ");
//Digamos que o valor é 123
int valor = numero.nextInt();
int resto=0;

resto=valor%10; //resto=3
valor=valor/10;// valor=12
System.out.print(resto); //imprime 3

resto=valor%10; //resto=2
valor=valor/10;// valor=1
System.out.print(resto); //imprime 2
System.out.print(valor); //imprime 1
}
}
```

Infográfico

A linguagem Java foi pioneira em estabelecer uma linguagem híbrida (compilada e interpretada), conseguindo, assim, ser multiplataforma. Depois de ter compreendido isso, você deve voltar sua atenção para
a estrutura de classes do Java, sendo que cada componente tem
um significado.

Neste Infográfico, você verá como é uma classe Java em detalhes,
seu processo de compilação e interpretação até o código ser executado.

```
import java.util.Scanner;

public class ReverseNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Digite uma sequência de 3 números: ");
        String input = scanner.nextLine();
        scanner.close();

        String[] numArray = input.split(" ");

        if (numArray.length == 3) {
            for (int i = numArray.length - 1; i >= 0; i--) {
                int num = Integer.parseInt(numArray[i]);
                System.out.println(num);
            }
        } else {
            System.out.println("Por favor, digite exatamente 3 números separados por espaço.");
        }
    }
}
```

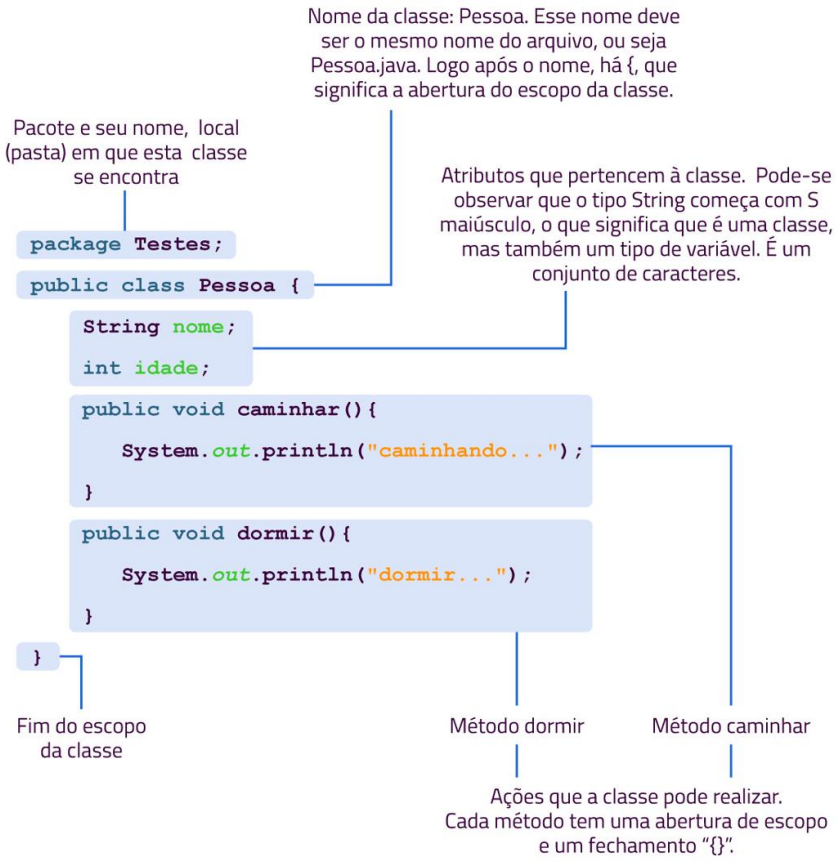
ESTRUTURA DE UMA CLASSE EM JAVA

Cada linguagem de programação tem uma sintaxe própria. Às vezes, parte dessa sintaxe é herdada de outras linguagens. No caso da linguagem Java, grande parte de sua sintaxe advém da linguagem C e C++.

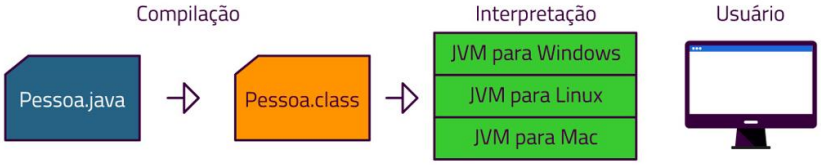
A sintaxe de uma linguagem de programação é um conjunto de regras que determina a posição de palavras-chaves, símbolos e suas combinações, que regem a composição das linhas de comando. A sintaxe diz respeito à estrutura da linguagem.

Saiba mais:

Estrutura básica de uma classe Java



Processo de compilação e interpretação



O processo de compilação e interpretação da linguagem Java é o mecanismo que possibilita a **portabilidade do código**. O arquivo fonte Pessoa.java é compilado para uma linguagem intermediária chamada bytecode (Pessoa.class).

Esse arquivo é interpretado pela máquina virtual do Java (JVM) específica para o computador do usuário.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conteúdo do Livro

A história do Java começou em meados dos anos 1990 e trouxe
uma nova maneira de programar para a Web. Sendo multiplataforma, gratuita e com a sintaxe similar a linguagens já consagradas como C e C++, ela logo ganhou adeptos no mundo inteiro. Sua estrutura orientada a objetos e a simplicidade na comunicação entre eles foram aspectos que a tornaram popular. Ainda hoje, Java é uma das linguagens mais utilizadas no mercado e a demanda por profissionais qualificados
só cresce.

No capítulo **Introdução à linguagem Java**, base teórica desta Unidade de Aprendizagem, você aprenderá um pouco da história da linguagem Java. É necessário conhecer a sua história para compreender suas origens, suas comunidades e seus objetivos. Logo após, você verá os elementos básicos dela, os conceitos iniciais e alguns exemplos que servirão
de guia para que você, futuramente, escreva seus próprios códigos.

Boa leitura.

ESTRUTURA DE DADOS EM JAVA

SOLUÇÕES EDUCACIONAIS INTEGRADAS

Introdução à linguagem Java

Pedro Kislansky

OBJETIVOS DE APRENDIZAGEM

- Descrever a história, a filosofia e a terminologia da linguagem Java.
- Desenvolver um programa simples em Java.
- Explicar a sintaxe da linguagem Java.

Introdução

Você sabia que a linguagem Java é uma das linguagens mais populares do mercado? Isso ocorre porque, em primeiro lugar, é uma linguagem simples, fácil de aprender, de forma que sua curva de aprendizado, em relação a outras linguagens, é bem menor. Em segundo lugar, a linguagem Java possui uma grande comunidade de colaboradores: ou seja, você terá o apoio de milhões de usuários e inúmeras linhas de código que poderão facilitar seu aprendizado. Além disso, a linguagem Java possui ótimas ferramentas de desenvolvimento (Eclipse, NetBeans, etc.), que ajudam você a codificar e tornar seu código mais limpo, legível, organizado e, por consequência, fácil de ser mantido, bem como uma grande coleção de bibliotecas e de *frameworks* prontas para uso, como Maven, Struts, etc.

Neste capítulo, você vai aprender um pouco sobre a história do Java e como ela se tornou, rapidamente, uma das linguagens de programação mais importantes da atualidade. Em seguida, vamos desenvolver um programa simples em Java, mas o suficiente para que você possa reconhecer o formato básico de um programa em Java. Por fim, você vai conhecer a sintaxe básica dessa linguagem, bem como alguns conceitos iniciais importantes.

História, filosofia e terminologia da linguagem Java

Em 1991, a Sun Microsystems financiou uma pesquisa que levaria ao desenvolvimento da linguagem Java, uma linguagem baseada em C e C++. A principal motivação foi a necessidade de uma linguagem independente de plataforma que pudesse ser usada na criação de *software* embutido em vários dispositivos eletrônicos domésticos, como torradeiras, fornos de micro-ondas e controles remotos. O projeto foi liderado por James Gosling, programador canadense nascido em 19 de maio de 1956, que trabalhou na Sun Microsystems de 1984 até abril de 2010. A versão inicial do Java foi chamada de "Oak" (carvalho, em inglês), pois havia um carvalho em frente à janela onde a equipe do projeto trabalhava. Contudo, já havia uma linguagem de programação com esse nome, então os envolvidos foram obrigados a modificar para "Java", uma referência à forma coloquial de chamar o café em algumas partes dos Estados Unidos, para contornar o problema de patente registrada (SCHILDT, 2015).

Mais ou menos na mesma época quando os detalhes de Java estavam sendo esboçados, surgia a World Wide Web, um fator que desempenharia papel crucial no futuro da linguagem. Se a *web* não estivesse se formando quase ao mesmo tempo em que Java estava sendo implementada, talvez ela continuasse sendo uma linguagem útil, mas obscura para a programação de utensílios eletrônicos domésticos. No entanto, com o surgimento da *web*, Java foi impulsionada para a dianteira do *design* das linguagens de computador, porque a *web* também precisava de programas portáteis (SCHILDT, 2015).

Fique atento

Uma novidade que o Java trouxe foi o gerenciamento de memória por "garbage collector", coletor de lixo que se encarrega de se desfazer de posições de memória que não estão sendo utilizadas. Até aquele momento, o programador era responsável por esse gerenciamento, o que induzia a vários erros lógicos dentro do programa (FINEGAN; LIGUORI, 2018).

A principal filosofia que Gosling queria implementar na linguagem Java era o fato de ela poder ser compilada uma vez e executada em qualquer ambiente. Daí saiu o famoso *slogan* da linguagem: "write once, run anywhere". O modelo desenvolvido sugere que a linguagem Java, quando compilada, é transformada em um código intermediário, chamado de **bytecode**. Depois, o bytecode pode ser interpretado por qualquer máquina que tenha uma **JVM (máquina virtual Java)** específica. Logo, se estiver usando o Linux, você precisa

de uma JVM para Linux; se estiver usando o Windows, precisará de uma JVM para Windows e assim sucessivamente.

De fato, converter um programa Java em bytecode facilita muito sua execução em uma grande variedade de ambientes, porque só a JVM tem que ser implementada para cada plataforma. Uma vez que o pacote de tempo de execução estiver presente em determinado sistema, qualquer programa Java poderá ser executado nele. Lembre-se: embora os detalhes da JVM sejam diferentes de uma plataforma para outra, todas entendem o mesmo bytecode Java. Além disso, o fato de um programa Java ser executado pela JVM também ajuda a torná-lo seguro. Como a JVM está no controle, ela pode reter o programa e impedi-lo de gerar efeitos colaterais fora do sistema (SCHILDT, 2015).

Quando você estiver aprendendo Java, verá que existe um mar de abreviações com que terá que se acostumar: JDK, JRE, JVM, JSE, JEE, JSP e muitas outras. Mas não se apavore! Vamos explicar algumas delas no decorrer deste capítulo e, na dúvida, uma simples busca na internet resolve o problema.

Antes de mais nada, para que você possa iniciar seu caminho no aprendizado da linguagem, você precisa de duas coisas: JDK e JRE. **JDK** (*Java development kit*), ou *kit* de desenvolvimento do Java, é um conjunto de utilitários que permitem criar sistemas de *software* para a plataforma Java. É composto pelo compilador e por bibliotecas. Por sua vez, o **JRE** (*Java runtime environment*), ou ambiente de execução do Java, é o ambiente utilizado para executar as aplicações da plataforma Java. É composto por bibliotecas (APIs, do inglês *application programming interfaces*) e pela JVM (HORSTMANN, 2009).

Para codificar programas em Java, pode-se usar qualquer editor de textos; porém, a melhor opção é utilizar um **IDE** (*integrated development environment*), ou ambiente de desenvolvimento integrado. Trata-se um *software* que possui um editor de texto e várias outras ferramentas, como automação de compilação local, *debugger* e análise de código em tempo real, que ajudam o programador em suas tarefas e agilizam o processo, possibilitando trabalho colaborativo, integração com outras ferramentas, etc.

Os dois IDEs mais populares são o Eclipse (ECLIPSE FOUNDATION, c2020) e o Netbeans (NETBEANS, c2020). O **projeto Eclipse** foi criado, originalmente, pela IBM em 2001, mas, em 2004, foi criada a fundação Eclipse, independente e sem fins lucrativos. Hoje, o Eclipse é utilizado por milhões de pessoas e possui uma comunidade bastante ativa. Por sua vez, o **NetBeans** começou como um projeto de estudantes da República Tcheca em 1996. O objetivo era escrever um IDE similar ao Delphi no Java, e, assim com o Eclipse, ele também é gratuito. Com a aquisição da Sun Microsystems pela Oracle em 2010, o NetBeans se tornou parte da Oracle também.

Fique atento

Não existe um IDE melhor que outro, assim como não existe uma linguagem melhor que outra. O que vai determinar o IDE mais adequado para o projeto é sua afinidade com ele, e a linguagem geralmente é definida pela natureza do problema ou pela *expertise* da equipe de trabalho.

Segundo o *ranking* da RedMonk (RENATO, 2020), hoje a linguagem Java é a terceira linguagem de programação mais popular do planeta, perdendo somente para C e Python. Desde que foi lançada, foi rapidamente adotada por uma legião de fãs. Sua gratuidade e portabilidade fizeram com que fosse adotada pela maioria das instituições públicas. Hoje, ela está presente em *notebooks*, *tablets*, celulares, *mainframes*, sistemas de automação e muito mais. De fato, a comunidade Java é uma das comunidades mais ativas entre as linguagens de programação (GUJ, c2020; JAVA, c2020; SANTANA, 2020).

Como desenvolver um programa simples em Java?

Nesta seção, vamos desenvolver um programa simples em Java. Primeiramente, descreveremos um pequeno problema e, depois, veremos sua solução em forma de programa codificado na linguagem Java. Por fim, analisaremos cada linha do programa. Não se assuste caso não compreenda todos os detalhes: o importante, por enquanto, é entender os conceitos de forma geral.

Problema

Digamos que uma pessoa trabalhe com importação e exportação. Certo dia, seu chefe pede que ela faça um programa que converta valores em dólar para real, uma vez que consiste em uma demanda diária do trabalho, pois ele precisa saber quanto ganha em real nas exportações.

Solução

```
1 package Testes;

2 import java.util.Scanner;

3 public class Main {

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

```
5       final double CAMBIO_ATUAL = 5.4;
6          Scanner s = new Scanner(System.in);
7         System.out.println("Qual e o valor em dolar?");
8                 double valorEmDolar = s.nextDouble();
9            double valorEmReal = valorEmDolar*CAMBIO_ATUAL;
10       System.out.println("Valor em real: R$ "+valorEmReal);
11    }
12 }
```

A estrutura de um projeto em Java segue certa hierarquia. Primeiramente, temos, é claro, o projeto. Dentro do projeto, temos pacotes (*packages*, em inglês). Entenda **pacote** como se fosse uma pasta. Na pasta, podemos ter subpastas ou arquivos de diferentes formatos. Um desses arquivo é chamado de "classe". **Classe** é uma abstração de algo que existe no mundo real, então uma classe pode ser uma pessoa, um objeto ou parte de um objeto.

No código acima, a linha 1 identifica o pacote em que essa classe se encontra. Na linha 3 está a declaração da classe e seu nome, `Main` (o nome da classe tem que ser o mesmo nome do arquivo). A classe sempre começará com letra maiúscula. Por quê? Convenção.

Na linha 4, temos um **método** (também chamado de função), muito especial em Java. Um projeto Java pode ter dezenas, centenas, milhares de classes, mas pelo uma delas deve conter esse método. Quando você executa um projeto em Java, o que será executado é esse método.

Na linha 5, é declarada uma **constante** (variável que não se pode modificar seu valor depois de ele ter sido estabelecido) do tipo "double" (ponto flutuante). Na linha 6, criamos um objeto de nome "s", que tem a capacidade de gerar "input" do usuário no sistema. Com ele, é possível receber valores de usuários externos ao sistema.

Na linha 7, é impresso, na tela, "Qual é o valor em dólar?", e, na linha 8, recebemos, do usuário, um valor em ponto flutuante (valor decimal) e colocamos em uma variável chamada de `valorEmDolar`. Nas linhas 13 e 14, respectivamente, é feito o cálculo de conversão e ele é impresso na tela.

Note que toda linha de código termina em ponto e vírgula e que, nas linhas 3 e 12, temos `{` abrindo o escopo da classe e `}` fechando esse escopo. Assim como nas linhas 4 e 11, esses estão abrindo e fechando o escopo do método `Main`.

Sintaxe da linguagem Java

Java é uma linguagem *case sensitive*, o que quer dizer que ela diferencia caracteres maiúsculos de minúsculos. Logo, o nome Pedro é diferente de PeDro em Java. Em todas as linguagens de programação, existem certas convenções quanto à escrita do código, e elas devem ser seguidas. Ao longo desta seção, explicaremos algumas delas. A seguir, você verá os elementos básicos da linguagem Java.

Estrutura da linguagem Java

Existem elementos básicos na estrutura de um programa em Java: pacotes, classes e importações. Os pacotes servem para você organizar suas classes, e geralmente o nome dos pacotes tende a fazer referência a suas funcionalidades, como o pacote "persistência" pode se referir a um pacote com classes que manipulam base de dados.

As classes são as unidades mais importantes. Cada classe deve realizar tarefas especificas a seu propósito. Logo, se tenho uma classe chamada de `ContaBancaria`, ela é uma abstração de uma conta bancária, então ela terá saldo, número da conta, titular e tudo mais que faça sentido para ela. As importações são a maneira como temos de importar código de bibliotecas externas. No código da seção anterior, por exemplo, importamos a classe `Scanner` do pacote java.util, e essa classe nos permite receber entradas de usuários externos.

A classe tem um escopo que é delimitado por `{}`. As classes possuem três coisas:

1. nome;
2. atributos;
3. métodos.

O nome da classe sempre começará com letra maiúscula (convenção), como já comentamos. Os atributos, assim como o próprio nome sugere, são os atributos daquela abstração que chamamos de classe. Então, se você tem uma classe `ContaBancaria`, os atributos seriam: saldo, número da conta, titular, tipo da conta (poupança, corrente), *status* (ativa, desativada), limite e assim por diante.

Os métodos são as ações que a classe pode realizar. Em nossa classe `ContaBancaria`, poderiam ser métodos dela: `depositar`, `retirar`, `transferir`, `visualizarSaldo`, etc. Veja que métodos em Java começam sempre com letras minúsculas, como já foi dito anteriormente.

Comentários

Você pode inserir comentários em seu código. Comentários são muito bem-vindos, pois esclarecem o código para pessoas que o veem pela primeira vez ou que realizarão sua manutenção. Os comentários em Java podem ser feitos de duas maneiras: em uma linha ou em duas linhas. Veja:

```
// Isso é um comentário de apenas uma linha
Caso você queira comentar mais de uma linha, deverá usar assim:
/*
Várias linhas de comentário
são possíveis de fazer aqui
*/
```

É importante ressaltar que o comentário não deve ser feito de maneira trivial: ele só deve ser inserido no código se consideramos que o código em si não seja autoexplicativo. Mas lembre-se: o que é claro para você, talvez não seja para outra pessoa.

Variáveis

Variáveis são endereços de memória onde você pode armazenar informações. É claro que não vamos manipular endereços de memória, o que seria muito trabalhoso. As linguagens de programação fornecem uma abstração para isso. Há a possibilidade de dar um nome a esse endereço e atribuir a ele um tipo. Logo, se declaro `int numero=10;`, isso quer dizer que estou declarando uma variável chamada de `numero` do tipo inteiro e que está sendo atribuído a ela o valor 10. Agora, quando quiser imprimir a variável `numero`, o Java vai imprimir 10 (FINEGAN; LIGUORI, 2018).

Na seção anterior, vimos um código que possui uma variável especial, chamada de constante. Esse tipo de variável não pode ser modificado durante o programa. Nenhuma variável em Java pode começar com número ou caracteres especiais, exceto *underline*. Variáveis em Java começam sempre com letra minúscula, e caso haja um nome composto por dois ou mais nomes, todos os outros nome, com exceção do primeiro, virão com letra maiúscula (esse padrão é chamado de `camelCase`). Esse padrão pode ser percebido na variável `valorEmDolar`, no código da seção anterior.

Fique atento

Com relação a acentos, o Java aceita se você declarar uma variável como `você`, por exemplo. Contudo, não é uma boa prática e deve ser evitada.

Como pode ser percebido, as variáveis possuem um tipo. O tipo serve para dizer ao computador qual é o domínio daquela variável, ou seja, quais valores aquela variável pode assumir. O tipo também determina qual faixa de valores a variável pode ter. No Quadro 1, você pode observar os principais tipos primitivos da linguagem Java, sua faixa de valores e seu tamanho.

Quadro 1. Tipos primitivos em Java

Tipo	Descrição	Faixa		
		Menor	Maior	Tamanho
Int	Valores inteiros	–2.147.483648	2.147.483647	32 bits
Float	Valores com casas decimais	–1.402E-37	3.40282347E+38	32 bits
Double	Valores com casas decimais	–4.94E-307	1.79769313486231570E+308	64 bits
Char	Um caractere	0	65535	16 bits
Boolean	Boleano	False	True	1 bit

Fonte: Adaptado de Caelum (c2020).

Observe que, se atribuirmos um valor a uma variável e esse valor estiver fora da faixa, ocasionará um erro de compilação:

```
int numero = 21855002252;
```

Esse código não funcionará.

Operadores aritméticos

Os operadores aritméticos em Java são + (adição), – (subtração), * (multiplicação), / (divisão) e % (resto da divisão). Com relação a operações aritméticas, é importante lembrar duas coisas. Vejamos.

1. Seus cálculos estão sob jurisdição das leis que regem a matemática e a precedência das operações.
2. É importante verificar a compatibilidade de tipos antes de realizar cálculos. Por exemplo: você não pode realizar uma conta de dividir e colocar o resultado em uma variável inteira, pois, se houver casas decimais, elas se perderão.

Caso você precise fazer uma conta do tipo `int resultado = numero / outroNumero`, onde `numero` é inteiro e `outroNumero` é double, você pode fazer um "cast". *Casting* é uma maneira de dizer ao compilador que você sabe o que você está fazendo e que você só quer a parte inteira do cálculo. Então como você faria? Simples: `int resultado = (int) (numero / outroNumero)`. O que você está dizendo é: faça o cálculo e depois me retorne somente a parte inteira da divisão. Assim, se o resultado do cálculo é 13,254, a variável `resultado` só receberá 13 (HORSTMANN, 2009; SCHILDT, 2015).

Fique atento

O operador + tem outra funcionalidade: concatenar um texto com uma variável. Por exemplo: `System.out.print(meu saldo é: +saldo)`. Esse fenômeno é chamado de **sobrecarga de operadores**.

Palavras reservadas

No Quadro 2, você pode observar um conjunto de palavras que são reservadas pela linguagem Java. Essas palavras, e ainda `true`, `false` e `null`, são palavras que não podem ser usadas como nome de pacote, de classe, de métodos ou de variáveis.

Quadro 2. Palavras reservadas na linguagem Java

abstract	do	import	short	volatile
assert	double	instanceof	static	while
boolean	else	int	strictfp	
break	enum	interface	super	
byte	extends	long	switch	
case	final	native	synchronized	
catch	finally	new	this	
char	float	package	throw	
class	para	private	throws	
const	goto	protected	transient	
continue	if	public	try	
default	implements	return	void	

Fonte: Adaptado de Perry (2010).

String

Como vimos anteriormente, temos vários tipos primitivos. Perceba que todos eles começam com letra minúscula. "String" também é um tipo, só que não primitivo: veja que ele começa com letra maiúscula e, em Java, isso quer dizer que se trata de uma classe (HORSTMANN, 2009).

O tipo String é usado para estabelecer uma cadeia de caracteres (texto, se preferir). Você a declara como uma variável qualquer (veja no exemplo abaixo). Contudo, você não deve confundir a String "a" do char 'a'. No primeiro, você tem uma cadeia de caracteres com um caractere; no segundo, você tem um caractere do tipo "char".

Declarando uma String:

```
String texto = "eu gosto muito de programar";
```

Por ser uma classe, String possui métodos, ou seja, ações que podem ser realizadas porseu objeto. Existem diversos métodos úteis na classe String. No Quadro 3, você pode ver alguns desses métodos. Na primeira coluna, temos o nome do método; na segunda, um exemplo e, na terceira, o que será impresso, com uma curta explicação entre parênteses. Você

pode ver todos os métodos dessa classe na documentação oficial da Oracle (ORACLE, c2020).

Quadro 3. Alguns métodos da classe String

Método	Exemplo	Saída
`length()`	`String texto="eu amo java"; System.out.println(texto.length());`	`11` (tamanho da String; inclui os espaços em branco)

`endsWith()`	`String texto="eu amo java"; System.out.println(texto.endsWith("java")); System.out.println(texto.endsWith("jAva"));`	`true`
		`false`
		(retorna `true` se a String termina com o texto mencionado. Perceba que "java" é diferente de "jAva")

`substring()`	`String texto="eu amo java"; System.out.println(texto.substring(0,6));`	`eu amo` (retorna uma parte da String; nesse caso, da posição zero [primeira posição] até 6, inclui espaços em branco)

`toLowerCase()`
`toUpperCase()`
	`String texto="eu AMO java"; System.out.println(texto.toLowerCase()); System.out.println(texto.toUpperCase());`	`eu amo java`
		`EU AMO JAVA`
		(a primeira retorna a String em letras minúsculas, e a segunda retorna em maiúsculas)

Neste capítulo, estudamos um pouco sobre a linguagem Java, sua estrutura, seus elementos mais básicos, etc. Vimos os principais tipos primitivos da linguagem e descobrimos que "String" não é um tipo primitivo, mas uma classe. Ainda, aprendemos que devemos obedecer a certas convenções, porque isso leva a um código limpo e fácil de manter. Estudamos, ainda, um pouco sobre a origem do Java, sua história e seu caminho desde sua concepção até os dias de hoje.

A linguagem Java é muito rica em recursos e possui uma grande e ativa comunidade. Participe de fóruns e de debates *on-line* sobre a linguagem e lembre-se: para saber programar, é necessário praticar o máximo possível. Desenhe um projeto e comece a desenvolver em Java, procurando os recursos de que você necessita. Perseverança, curiosidade e foco são os três pilares de um bom programador.

Referências

CAELUM. *Variáveis primitivas e controle de fluxo*. c2020. Disponível em: https://www.caelum.com.br/apostila-java-orientacao-objetos/variaveis-primitivas-e-controle-de-fluxo#casting-e-promocao. Acesso em: 4 jan. 2021.

ECLIPSE FOUNDATION. *Eclipse IDE for Java developers*. 2018. Disponível em: https://www.eclipse.org/downloads/packages/release/oxygen/3a/eclipse-ide-java-developers. Acesso em: 3 jan. 2021.

FINEGAN, E.; LIGUORI, R. *OCA Java SE 8*: guia de estudos para o exame 1Z0-808. Porto Alegre: Bookman, 2018.

GUJ. *Categorias*. c2020. Disponível em: https://www.guj.com.br/. Acesso em: 3 jan. 2021.

HORSTMANN, C. *Conceitos de computação com Java*: compatível com Java 5 & 6. 5. ed. Porto Alegre: Bookman, 2009.

JAVA. *Por que aprender Java?* c2020. Disponível em: https://www.javafree.org/. Acesso em: 3 jan. 2021.

NETBEANS. *Apache NetBeans*. c2020. Disponível em: https://netbeans.org/. Acesso em: 3 jan. 2021.

ORACLE. *Java™ platform, standard edition 7*: API specification. c2020. Disponível em: https://docs.oracle.com/javase/7/docs/api/java/lang/String.html. Acesso em: 3 jan. 2021.

PERRY, J. S. *Fundamentos da linguagem Java*. 2010. Disponível em: https://developer.ibm.com/br/tutorials/j-introtojava1/. Acesso em: 4 jan. 2021.

RENATO, S. *Confira as 20 linguagens de programação mais populares do momento*. 2020. Disponível em: https://olhardigital.com.br/2020/03/03/pro/confira-as-20-linguagens-de-programacao-mais-populares-do-momento/. Acesso em: 3 jan. 2021.

SANTANA, P. *SouJava, comunidades Java e o que você tem a ver com isso?* 2020. Disponível em: https://soujava.org.br/. Acesso em: 3 jan. 2021.

SCHILDT, H. *Java para iniciantes*: crie, compile e execute programas Java rapidamente. 6. ed. Porto Alegre: Bookman, 2015.

Leitura recomendada

GOODRICH, M. T.; TAMASSIA, R. *Estrutura de dados e algoritmos com Java*. 5. ed. Porto Alegre: Bookman, 2013.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Dica do Professor

Java é uma linguagem com uma comunidade muito ativa. Isso possibilita que você tenha acesso a fóruns, repositório de códigos, eventos, e muito mais. Aprender uma linguagem de programação envolve mergulhar em seu ecossistema.

Nesta Dica do Professor, você vai conhecer as comunidades Java formadas ao longo do tempo e que oferecem muitos recursos e formas de se manter atualizado nos estudos da linguagem.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **A linguagem Java tem muitos recursos e uma comunidade muito ativa. Com relação a esses recursos, é possível afirmar:**

I. Java é uma linguagem compilada. Por isso, sua execução é tão rápida e sua portabilidade é muito boa.

II. Java não diferencia entre letras maiúsculas e minúsculas; daí seu alto grau de legibilidade.

III. A sintaxe da linguagem Java é muito parecida com a linguagem C.

Assinale a opção correta de acordo com o que foi dito anteriormente:

A) Somente a III está correta.

B) Somente I e III estão corretas.

C) Somente a II está correta.

D) Somente II e III estão corretas.

E) Somente a I está correta.

2) **Um bom programador deve ser capaz de analisar códigos de terceiros e saber como reaproveitá-los em seu projeto. Tendo isso em mente, analise o código a seguir e assinale a alternativa que corresponde ao que será impresso pelo programa:**

```
package Testes;

public class Main {
public static void main(String[] args) {

int valor = 12345;
valor=valor%10;
valor=valor*2;
valor=valor+100;
valor=valor%10;
int voce=10;
System.out.println("Valor: "+valor);
}
```

}

A) 12.

B) 0.

C) 25.

D) 123.

E) 152.

3) **Programar não é só saber codificar, e, sim, ter um conhecimento abrangente e robusto da linguagem e das tecnologias que estão sendo usadas no projeto, incluindo coisas triviais como nome de variáveis. Existem regras em Java para criar nomes de variáveis.**

Assinale a opção em que todos os nomes de variáveis são possíveis na linguagem Java:

A) pedro, casa, _sentimento, 1voce.

B) melao, criado, banco, *double*.

C) senha, livre, mamao0023, casa_grande.

D) supermercado, salve, *for*, inteiro.

E) sao_paulo, *voce, limpar, decimal.

4) **O Java foi criado para ser multiplataforma. Daí o *slogan*: "*Write once, run anywhere*".**

Assinale a alternativa que justifica esse *slogan*:

A) O Java pode ser compilado para qualquer máquina, independente do sistema operacional dela. Assim, ela gera um arquivo.exe, que pode ser executado em qualquer lugar.

B) Quando se compila um programa em Java, é gerado um código intermediário. Esse código pode ser interpretado por qualquer sistema operacional, contanto que ele tenha uma JVM específica para ele.

C) Quando se compila um programa em Java, é gerado um código intermediário. Esse código pode ser recompilado para qualquer sistema operacional e, assim, executado.

D) Para que o Java seja utilizado por qualquer máquina, ele tem de ser compilado para um arquivo especial executável. Isso faz com que o código Java possa ser executado em qualquer lugar.

E) O Java foi idealizado para trabalhar somente com o Linux e o Windows, pois são os sistemas operacionais mais utilizados no mercado. Essa é uma das grandes vantagens do Java.

5) **Classes devem começar com letras maiúsculas e métodos, com letras minúsculas.**

Como você justifica isso, sabendo que, se não o fizer, o programa funcionará da mesma forma?

A) Não importa como se escreve. Isso é apenas uma recomendação. E como é uma recomendação, pode ser ignorada sem prejuízos.

B) Irá funcionar, mas é importante para a legibilidade do código e serve como linguagem comum entre profissionais da área.

C) Serve para que o código fique mais elegante, mais charmoso. A beleza do código é muito importante.

D) Se não fizer assim, o código compila, mas pode não funcionar da maneira que se quer.

E) Um programador experiente em Java não precisa dessas convenções. Ele sabe onde estão todas as funcionalidades.

Na prática

Classe é uma abstração de algo que existe no mundo real. Isso quer dizer que qualquer coisa pode ser uma classe: um carro, uma caneta e até um fenômeno físico como um tornado. Quando se modela um projeto em uma linguagem orientada a objetos como Java, se modelam as classes que esse projeto irá conter.

Por exemplo, um projeto relacionado a bancos teria classes como: agência, conta, cliente e outras. Faz parte do estudo teórico conhecer os conceitos básicos de programação. Contudo, nada substitui a prática. Os acontecimentos diários no mundo trazem novos conhecimentos, que são relacionados ao que se estuda e, assim, se evolui profissionalmente.

Confira, em Na Prática, o processo de modelagem de uma classe para um projeto de *software*, que será desenvolvido usando o paradigma de orientação a objetos e a linguagem de programação Java.

PROCESSO DE MODELAGEM DE UMA CLASSE

David é um programador iniciante e fará parte da equipe que projetará um novo sistema financeiro que determinada empresa deseja lançar ao mercado.

No projeto, cada pessoa desenvolverá uma funcionalidade do novo sistema. David ficou encarregado de fazer a criação, cujos atributos e métodos foram detalhados na seguinte ficha:

Ficha para desenvolvimento:

Classe: ContaBancaria

Atributos:
numero (inteiro)
titular (String)
saldo (double)

Métodos:
imprimirSaldo
depositar
sacar

David, então, definiu os passos para a resolução do problema:

1. Criar, primeiramente, a classe ContaBancaria.java, tomando cuidado para seguir o padrão camelCase, lembrando que classes começam com letra maiúscula.

```
package Testes;
public class ContaBancaria {
}
```

2. Acrescentar os atributos como pedido, lembrando que atributos começam com letra minúscula.

```
package Testes;
public class ContaBancaria {
    int numero;
    String titular;
    double saldo;
}
```

3. Escrever os métodos, ou seja, o que a classe deve fazer, suas ações. A primeira ação ou método é imprimir o saldo. Para imprimir, usa-se o comando System.out.print("").

Método 1

```
public void imprimirSaldo(){
    System.out.print("Seu saldo: "+saldo);
}
```

Método 2

Para depositar, é necessário receber um valor e acrescentá-lo ao saldo.

```
public void depositar(double valor){
    saldo = saldo + valor;
}
```

O saldo será o valor que está no saldo + o valor recebido no método.

Método 3

Para sacar, é preciso retirar o valor do saldo, ou seja, o inverso de depositar.

```
public void sacar(double valor){
    saldo = saldo - valor;
}
```

O saldo será o valor que está no saldo - o valor recebido no método.

```
package Testes;
public class ContaBancaria {
int numero;
String titular;
double saldo;

public void imprimirSaldo(){
    System.out.print("Seu saldo: "+saldo);
}
public void depositar(double valor){
    saldo = saldo + valor;
}
public void sacar(double valor){
saldo = saldo - valor;
}
}
```

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Seja reconhecido como um líder do setor

Neste *site*, você poderá obter todas as informações necessárias caso deseje fazer um certificado Java. Certificações são importantes para o mercado de trabalho. Elas atestam que você domina aquela tecnologia.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Classes e objetos

Assista neste vídeo os conceitos básicos de classes e objetos, além dos pilares da Programação Orientada a Objetos.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Java para iniciantes - Crie, compile e execute programas Java rapidamente

Esta obra, de Herbert Schildt, é um excelente ponto de partida para quem está começando sua jornada em programação Java. O livro começa do zero e avança de forma didática e legível.

Conteúdo interativo disponível na plataforma de ensino!