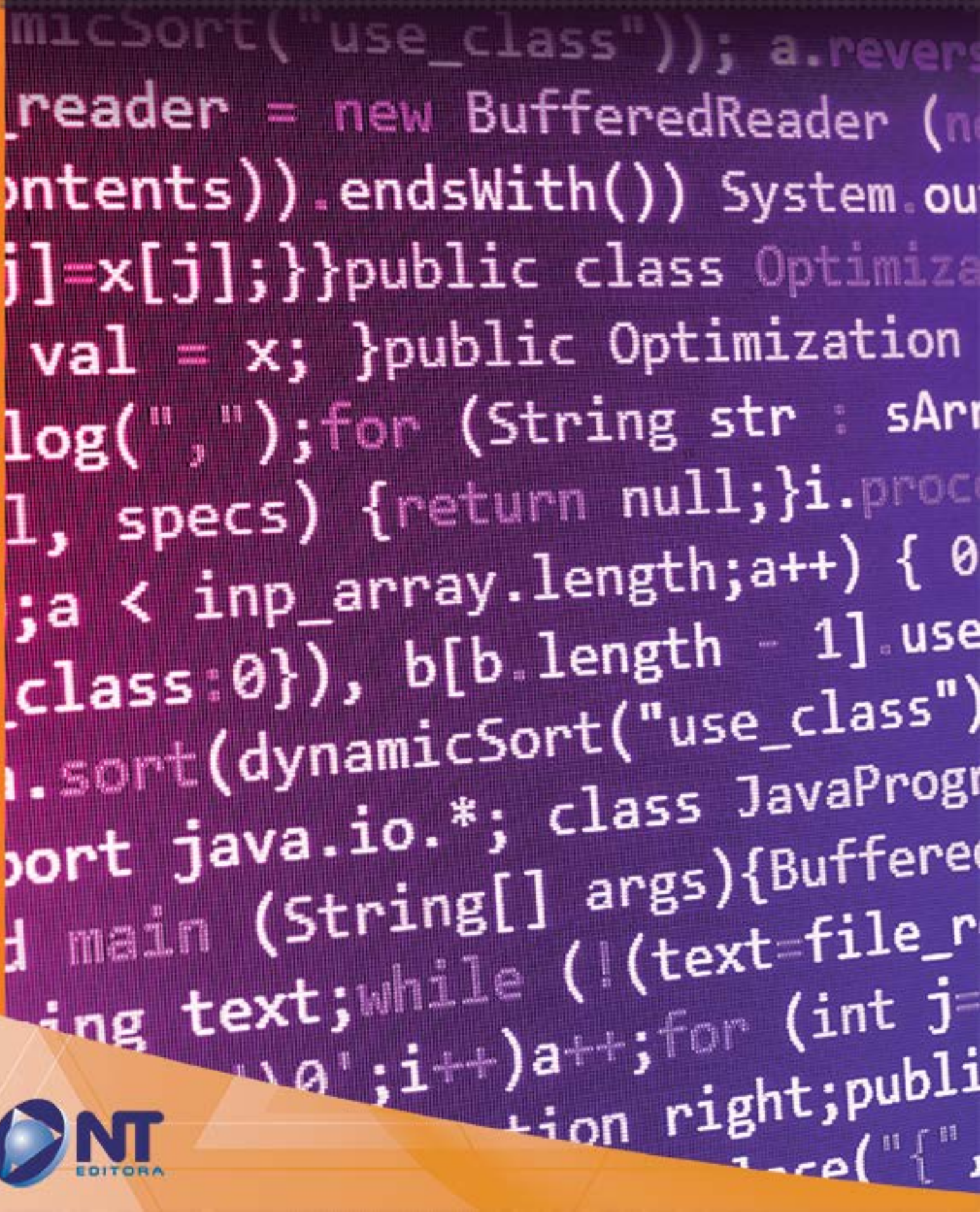
DESENVOLVIMENTO EM JAVA

Glauber Boff



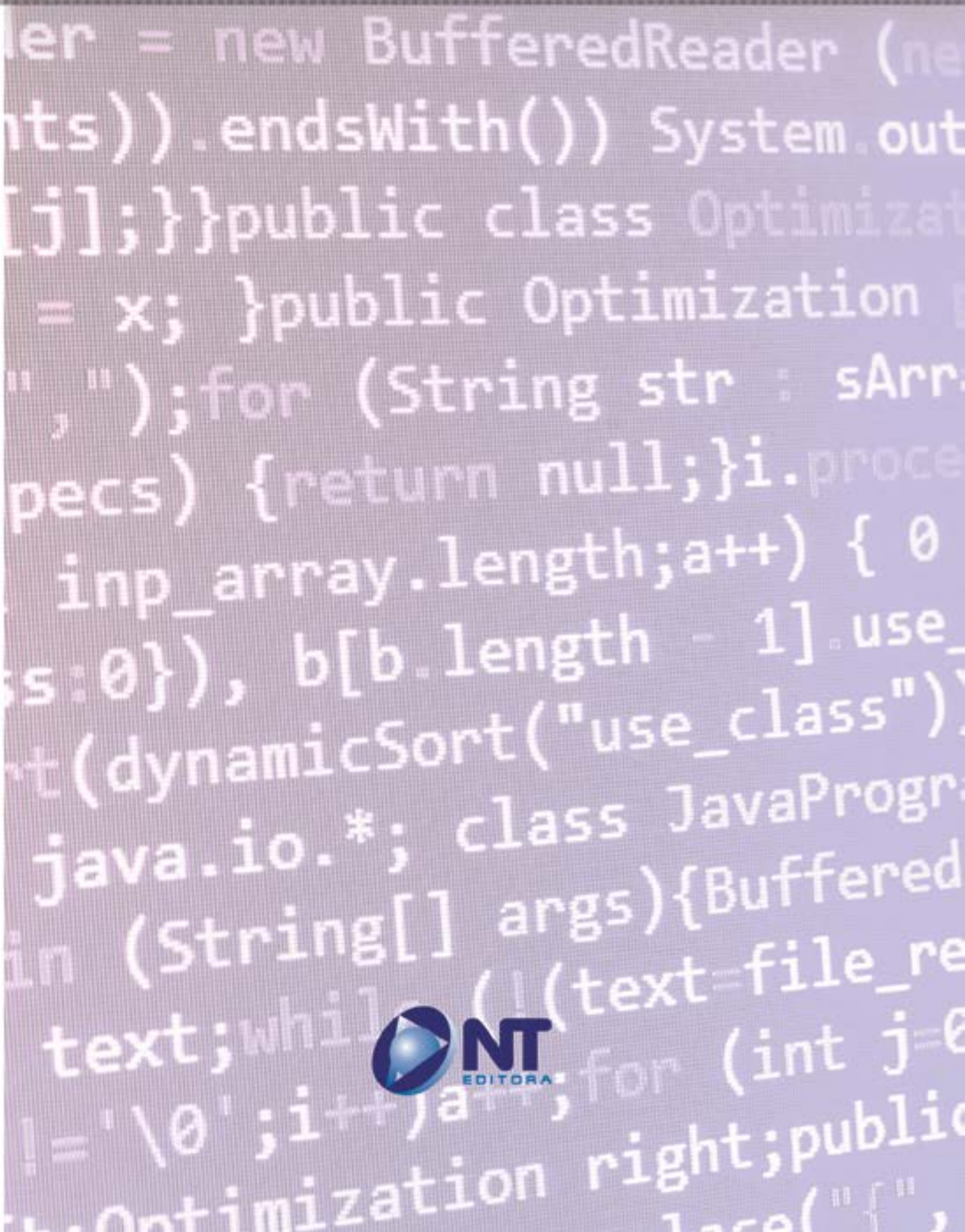
INFORMAÇÃO E COMUNICAÇÃO

NT EDITORA

DESENVOLVIMENTO EM JAVA

Glauber Boff

INFORMAÇÃO E COMUNICAÇÃO



NT EDITORA

Autor

Glauber Boff

Bacharel e mestre em Ciência da Computação pela Universidade Federal de Goiás na área de Engenharia de Software, além de especialização em Governança de TI pela Unieuro-DF. Atua na gestão estratégica da área de tecnologia da informação e na gestão de projetos de desenvolvimento de software no Corpo de Bombeiros Militar do DF. Tem experiência como professor/instrutor em cursos na área de tecnologia da informação e já trabalhou no Serviço Federal de Processamento de Dados (SERPRO) no Escritório de Projetos de desenvolvimento. Tem especialização e experiência nas áreas de Governança de TI, Planejamento Estratégico, Gestão de Projetos, Gestão de Processos de Negócio, Engenharia de Software, Qualidade de Software e Contratação de Bens e Serviços de TI, com as seguintes certificações: PMP, COBIT, ITIL, RUP e MPS.BR.

Design Instrucional
Vinicius Abreu

Projeto Gráfico
NT Editora

Revisão
Thaís Costa
Ricardo Moura
Mariana Carvalho

Capa
NT Editora

Ilustração
Thiago Ferreira

Editoração Eletrônica
Marcelo Moraes
Daniel Lopes
Nathália Nunes

NT Editora, uma empresa do Grupo NT
SCS Quadra 2 – Bl. C – 4° andar – Ed. Cedro II
CEP 70.302-914 – Brasília – DF
Fone: (61) 3421-9200
sac@grupont.com.br
www.nteditora.com.br e www.grupont.com.br

Boff, Glauber.

Desenvolvimento em Java / Glauber Boff – 1. ed. – Brasília: NT Editora, 2017.

224 p. il. ; 21,0 X 29,7 cm.

ISBN 978-85-8416-248-2

1. Desenvolvimento. 2. Linguagem.

I. Título

ÍCONES

Prezado(a) aluno(a),
Ao longo dos seus estudos, você encontrará alguns ícones na coluna lateral do material didático. A presença desses ícones o(a) ajudará a compreender melhor o conteúdo abordado e a fazer os exercícios propostos. Conheça os ícones logo abaixo:

Saiba mais
Esse ícone apontará para informações complementares sobre o assunto que você está estudando. Serão curiosidades, temas afins ou exemplos do cotidiano que o ajudarão a fixar o conteúdo estudado.

Importante
O conteúdo indicado com esse ícone tem bastante importância para seus estudos. Leia com atenção e, tendo dúvida, pergunte ao seu tutor.

Dicas
Esse ícone apresenta dicas de estudo.

Exercícios
Toda vez que você vir o ícone de exercícios, responda às questões propostas.

Exercícios
Ao final das lições, você deverá responder aos exercícios no seu livro.

Bons estudos!

Sumário

APRESENTAÇÃO

Caro(a) aluno (a),

Seja bem-vindo à **Desenvolvimento em Java**!

A linguagem de programação Java surgiu em 1995, criada pela Sun Micro*systems* Inc. e comprada em 2010 pela Oracle Corporation. Seu uso difundiu-se rapidamente na comunidade de desenvolvimento de *softwares* em razão de suas características e recursos, os quais atendiam às necessidades dos programadores e dos usuários de *softwares*. Entre essas características, citam-se a portabilidade e a segurança, que deram destaque à linguagem no cenário global.

A versão inicial dessa linguagem representou certa revolução no desenvolvimento de *software*, criando uma nova forma de projetar e desenvolver aplicações nos mais variados cenários. Ao longo dos anos, a linguagem Java sofreu uma série de modificações, seja para evoluir e melhorar os recursos existentes, seja para atender a demandas da comunidade de desenvolvimento de *software*. Atualmente, a linguagem está em sua versão 8, denominada Java SE 8 (Java Standard Edition 8), mas a versão 9 já está prevista para lançamento em breve.

Por ser uma linguagem bastante robusta e com grande diversidade de recursos, os quais foram evoluindo ao longo do tempo, muitos consideram Java uma linguagem complexa de ser aprendida. Portanto, este material destina-se àqueles que buscam uma forma simplificada de conhecer essa linguagem e ter uma visão geral de seus recursos. Não se pretende tratar de forma exaustiva todos os aspectos da linguagem; acredita-se, contudo, que o leitor conseguirá aprender os fundamentos e os principais recursos que o habilitem a se tornar um programador Java.

Para fixar o aprendizado, são trazidos exercícios e exemplos práticos de programas escritos na linguagem Java, o que permitirá melhor entendimento do conteúdo. Ao término deste estudo, o leitor terá a base teórica necessária para investigação de outros recursos mais avançados.

Bons estudos!

Glauber Boff

1 APRESENTAÇÃO DA LINGUAGEM JAVA

Olá! Está preparado para conhecer a linguagem Java? Então, vamos começar!

Esta primeira lição apresenta os conceitos iniciais sobre a linguagem, trazendo um pouco da sua evolução desde seu projeto inicial, bem como as principais características que a tornaram uma linguagem bastante reconhecida na comunidade de desenvolvedores de *software*. Além disso, serão dadas algumas instruções de como você deve proceder para realizar a configuração do ambiente de desenvolvimento necessário às atividades práticas do curso.

Objetivos

Ao finalizar esta lição, você deverá ser capaz de:

- conhecer a história da linguagem de programação Java e suas origens;
- entender os conceitos gerais da linguagem Java;
- compreender as principais características da plataforma Java;
- dominar algumas das ferramentas utilizadas com a linguagem Java e suas aplicações.

1.1 Histórico da linguagem

Java é uma linguagem de programação que, atualmente, é bastante poderosa, oferecendo diversos recursos para o desenvolvimento de sistemas de informações, desde aplicativos simples até sistemas corporativos robustos. Ela se baseia no paradigma de programação orientado a objetos e permite que os sistemas desenvolvidos sejam utilizados em diversos ambientes, como Windows, Linux, Mac OS X ou Solaris.

Saiba mais

A linguagem Java inicialmente foi denominada Oak pelo seu criador; mas, depois, verificou-se que já existia uma linguagem de programação com o mesmo nome. A equipe, então, ao visitar uma cafeteria da cidade, resolveu colocar o nome Java, em referência ao nome de uma cidade que produzia um determinado tipo de café importado.

Você sabia que a linguagem Java não é tão nova assim?

Seu desenvolvimento começou a partir de um projeto de pesquisa iniciado em 1991 pela empresa Sun Micro*systems*. A equipe desse projeto, denominada Green Team, era liderada por James Gosling, reconhecido como o pai do Java.

O projeto, criado inicialmente pela Sun, tinha como objetivo principal criar um sistema que permitisse a comunicação entre computadores e dispositivos eletrônicos utilizados no dia a dia, como a TV a cabo. Entretanto, por volta de 1992, esse projeto enfrentou algumas dificuldades, visto que o mercado de dispositivos eletrônicos não se desenvolveu como era esperado pela equipe. Com o avanço da internet, a partir de 1993, o projeto começou a tomar novos rumos, focando na utilização do Java para a criação e a publicação de conteúdo interativo e dinâmico (ORACLE CORPORATION, 2014).

De 1991 a 1995, o projeto da linguagem Java evoluiu, e novas características foram acrescentadas para atender às necessidades do mercado e para acompanhar a evolução tecnológica. Foi em 1995, em uma conferência na área de tecnologia, que a Sun anunciou o Java como uma linguagem de programação pela primeira vez. Surgiu, então, o Java 1.0, versão que permitia o desenvolvimento de aplicações simples, os *Applets*, que podiam ser executadas via navegador *web*, característica ainda inexistente na época.

Como forma de promover a divulgação dessa versão, a Sun passou a utilizar o slogan "*Write once, run anywhere*" (WORA), significando "Escreva uma vez, execute em qualquer lugar". Esse *slogan* fazia referência a uma das características principais dessa versão, que era o fato de a linguagem ser multiplataforma, isto é, permita a execução de um mesmo programa em diferentes plataformas, como Windows e Linux.

Mas o Java foi a primeira linguagem de programação a ser criada?

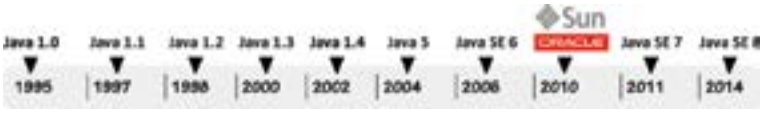
Não. Antes do surgimento do Java, diversas linguagens de programação foram criadas, e muitas delas continuavam a ser utilizadas para o desenvolvimento de aplicações variadas. Na verdade, o Java não foi criado a partir de ideias exclusivas da equipe envolvida no seu projeto. A linguagem foi projetada a partir da evolução das linguagens C e C++, que eram linguagens bastante utilizadas à época; portanto suas estruturas e regras assemelham-se bastante às dessas linguagens (FURGERI, 2015).

Atenção

O surgimento do Java não fez com que as demais linguagens desaparecessem do mundo da programação. As diversas linguagens de programação continuam sendo utilizadas paralelamente ao Java.

Ao longo dos anos, o Java continuou evoluindo e se tornou uma linguagem bastante completa e com grande variedade de recursos para os desenvolvedores. Essa evolução é apresentada, de forma simplificada, na figura a seguir.

Evolução histórica das versões da plataforma Java



Em 2010, houve um fato importante na história da linguagem: a aquisição da Sun Micro*Systems* pela Oracle Corporation. A Oracle, juntamente com a empresa IBM, sempre investiu em negócios relacionados ao uso da plataforma Java e continuou investindo na evolução e na expansão de uso da linguagem. De forma resumida, seguem algumas modificações realizadas em cada uma das versões anteriores da linguagem:

Versão	Características
Java Development Kit (JDK ou Java 1.0)	Primeira versão da linguagem, criada para o desenvolvimento de aplicações na internet baseadas em *Applets*.
Java Development Kit (JDK ou Java 1.1)	Foram inclusas várias bibliotecas, entre elas: Java Database Connectivity (JDBC), utilizada para conexão com bancos de dados, e Java Remote Method Invocation (RMI).
Java Standard Edition (J2SE 1.2 ou Java2)	Foram incluídas novas bibliotecas, como a API gráfica *Swing* e a API *Collections*. A partir dessa versão, houve uma mudança na plataforma padrão, sendo então subdividida em três edições: J2SE (Java 2 Platform, Standard Edition), J2EE (Java 2 Platform, Enterprise Edition) e J2ME (Java 2 Platform, Micro Edition).
Java Standard Edition – J2SE 1.3	Apelidada de Kestrel, a versão incluiu algumas novas bibliotecas, como Java Naming and Directory Interface (JNDI) e Java Sound, além do HotSpot JVM, que é uma máquina virtual Java para computadores desktops e servidores.
Java Standard Edition J2SE 1.4	Apelidada de Merlin, incluiu algumas mudanças na definição da linguagem, como a melhoria no modelo de expressões regulares, API de *log* e manipulação de arquivos XML.
Java Standard Edition J2SE 5.0	Apelidada de Tiger, incluiu elementos significantes para a linguagem, como *Generics,* Enumeration e Annotations.
Java Standard Edition JSE 6	Apelidada de Mustang, incluiu várias melhorias, como suporte ao JDBC 4.0, suporte a linguagens de *scripts* e melhorias nos componentes de interface gráfica. Além disso, houve melhora significativa no desempenho e na estabilidade.
Java Standard Edition JSE 7	Apelidada de Dolphin, incluiu algumas pequenas mudanças na linguagem, adição de uma nova biblioteca de manipulação de arquivos e gerenciamento automático de recursos, como conexões de bancos de dados, entre outras alterações.

Atualmente, o Java encontra-se na versão 8 (Java SE 8), lançada em 2014. Essa versão trouxe várias mudanças significativas na linguagem, não apenas na API da linguagem, mas também na Máquina Virtual Java. Algumas dessas mudanças estão descritas a seguir (ORACLE CORPORATION, 2015).

- Expressões lambda: novo recurso da linguagem que permite que o programador defina métodos anônimos diretamente no local de sua utilização, podendo utilizar uma funcionalidade ou um trecho de código Java como um parâmetro para esse método.
- API de data e hora: essa nova API foi inclusa para manipulação de data e hora, tornando a programação mais simples e fácil de entender. As classes para manipulação de datas existentes nas versões anteriores foram mantidas na versão 8 do Java.
- *Collections*: foi inclusa a nova API *stream*, integrada à API *collections*, para auxiliar o desenvolvimento de programas que realizem o tratamento de fluxos de elementos, inclusive melhorando o desempenho.

- Interface funcional: foi inclusa a anotação *functional interface* para identificar aquelas interfaces que definem apenas um método abstrato.

Saiba mais

Além das mudanças no Java 8 citadas acima, várias outras foram incorporadas. Para conhecer a lista completa dessas mudanças, veja as notas de versões *publi*cadas pela Oracle na página a seguir:

https://www.java.com/pt_BR/*download*/faq/release_changes.xml

Programando o conhecimento

Em relação à evolução histórica do Java, escolha a alternativa que melhor representa os marcos da linguagem.

a) A linguagem Java foi projetada a partir de conceitos e características de linguagens anteriores, dentre as quais se destacam as linguagens C e Javascript.

b) Patrocinado pela Sun Microsystems, o projeto inicial que deu origem ao Java fracassou, em decorrência do surgimento da internet.

c) Das versões da linguagem, o lançamento do J2SE 1.2 representou um importante marco, pois, nessa época, a plataforma foi dividida em três edições: J2SE, J2EE e J2ME.

d) A versão atual do Java, J2SE 9, incorporou novos recursos que permitem manipular de forma mais simples dados do tipo data e hora, bem como estruturas do tipo collections.

Comentário: o lançamento da terceira versão do Java (J2SE 1.2 ou Java 2) foi um importante marco para a linguagem Java, pois, além de adicionar novas bibliotecas, trouxe uma mudança na plataforma padrão, o que revolucionou todo o sistema. Portanto, a reposta correta é a letra "c".

1.2 Visão geral da linguagem

A linguagem de programação Java é uma linguagem de alto nível que permite escrever programas de computador para variadas áreas de aplicação. Ela apresenta diversos benefícios em relação às linguagens anteriores, e, devido a isso, que teve grande destaque na área de desenvolvimento de sistemas de informação.

As principais características do Java foram elencadas em um documento elaborado pelos criadores da linguagem, denominado *The Java Language Environment: a white paper* (GOSLING; MCGILTON, 1995). São elas:

Simples	Segura
Orientadora de objetos	Interpretada
Robusta	Alto desempenho
Portável	Compatível com redes
Arquitetura neutra	Múltiplos *threads*

Vamos, então, entender cada uma dessas características.

Simples

A ideia inicial era criar uma linguagem que permitisse desenvolver sistemas de forma fácil e que não exigisse muito treinamento por parte dos programadores. Apesar de ser baseada no C++, a linguagem Java omitiu vários recursos que, na visão dos técnicos, eram complexos e traziam poucos benefícios. Isso permitiu que os programadores que trabalhavam com o C++ também passassem a utilizar o Java.

Além da criação de uma linguagem simples, pretendia-se desenvolver aplicações que pudessem ser executadas em equipamentos pequenos e simples, com poucos recursos de *hardware*. Essa necessidade já existia na época em que o Java surgiu e, alguns anos depois, deu origem à Java Micro Edition (Java ME), utilizada em sistemas embutidos em equipamentos eletrônicos.

Orientada a objetos

Ser orientada a objetos foi uma característica que esteve sempre presente na linguagem, desde sua criação. Mas o que significa isso? O que é ser orientada a objetos?

A orientação a objetos é uma forma de se projetar e desenvolver um programa de computador, utilizando, para isso, conceitos e comportamentos do mundo real, que são mapeados para dar origem aos comportamentos no mundo computacional. Essa forma de se desenvolver programas é utilizada pelas principais linguagens de programação atualmente, e facilita bastante o entendimento das necessidades dos clientes.

Por exemplo, consideremos o carro como um objeto que possui algumas informações básicas (marca, modelo, ano etc.) e possui como comportamentos acelerar e frear. Na orientação a objetos, esses conceitos e comportamentos devem ser modelados de forma que possuam uma ou mais funcionalidades dentro de um sistema que controla um carro. A ideia é que cada objeto do mundo real que queiramos controlar por meio de um sistema deve ser modelado computacionalmente.

Robusta

A linguagem Java é considerada robusta, pois foi projetada para permitir a escrita de programas de diversas complexidades e tamanhos, ou seja, desde programas pequenos e simples, como um programa de agenda telefônica, até programas muito grandes e de grande complexidade, como um programa de controle de aeronaves e aeroportos.

Além disso, o Java possui recursos que realizam a verificação de problemas em programas, inclusive em tempo de execução, permitindo realizar o tratamento e a eliminação de situações indesejáveis em um programa. Isso faz com que a linguagem seja mais estável e confiável.

Portável e de arquitetura neutra

A portabilidade do Java deve-se à existência da Máquina Virtual Java (JVM), que realiza a interpretação dos chamados *bytecodes*, que, depois, serão executados em um computador de qualquer arquitetura.

Bytecodes: códigos intermediários entre o código-fonte Java, escritos pelo programador, e o código em linguagem de máquina, resultante do processo de compilação.

Dizer que o Java possui arquitetura neutra significa que um programa qualquer escrito nessa linguagem pode ser executado em qualquer plataforma ou equipamento. Por exemplo, um mesmo programa pode ser executado tanto em um computador com sistema operacional Windows como em outro computador com Mac OS X, mantendo a mesma aparência e o mesmo comportamento.

Segura

Desde seu projeto inicial, os criadores da linguagem tiveram a preocupação com o quesito de segurança.

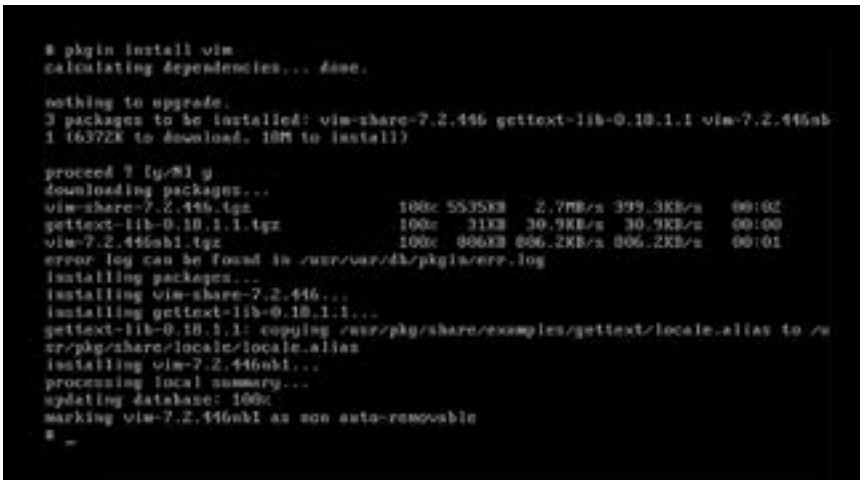
Para refletir

Você acha que segurança é realmente um item que exige preocupação por parte dos projetistas de uma linguagem de programação e dos programadores que a utilizam?

A resposta para essa pergunta é sim! Segurança é essencial, pois a falta dela pode trazer sérias consequências para um sistema e, consequentemente, para a organização que o utiliza.

Como a linguagem Java foi pensada para o uso em ambientes de rede, como a internet, a segurança das informações é um ponto fundamental. A ideia dos criadores era permitir o desenvolvimento e a execução de aplicações que fossem o mais confiáveis possível, evitando que elas tivessem acesso a recursos computacionais indevidamente.

Interpretada



A interpretação está diretamente relacionada ao funcionamento da Máquina Virtual Java (JVM).

O código-fonte de um programa escrito em Java, inicialmente, deve ser compilado, ou seja, analisado e transformado em um código mais simples (*bytecode*). Esse é o código que deve ser interpretado pela JVM antes de entrar em execução. A interpretação, em conjunto com a JVM, tem como principal benefício fornecer portabilidade à linguagem.

Dinâmica

O fato de o Java ser uma linguagem dinâmica significa que foi projetada para se adaptar a um ambiente em evolução. Por exemplo, nas bibliotecas da linguagem, é possível incluir novos métodos (operações) e variáveis (atributos), mesmo que o interpretador Java já esteja em execução, sem que isso acarrete algum problema para seus usuários. Essa característica é fundamental quando se deseja adicionar um determinado trecho de código a um programa em execução.

Alto desempenho

O projeto inicial do Java previa sua utilização em dispositivos de pequeno porte, logo exigia pouco espaço de armazenamento e pouca memória para executar os programas. Todavia, com a sua evolução e a ampliação do seu uso, observou-se que os programas passaram a exigir mais recursos computacionais. Entretanto, com a utilização da Máquina Virtual Java, é possível obter melhor desempenho na aplicação, visto que os *bytecodes* podem ser convertidos em tempo de execução para código de máquina específico de determinado tipo de sistema operacional, como Windows ou Linux.

Compatível com redes

Como surgiu em meio aos avanços da internet, o Java possui uma série de recursos que facilitam a programação em redes de computadores. Entre eles, podemos citar as bibliotecas de rotinas usadas para tratar os protocolos de rede *Transport Control Protocol/Internet Protocol* (TCP/IP), como o *Hypertext Transfer Protocol* (HTTP) e *File Transfer Protocol* (FTP). Com isso, os programas em Java podem, por exemplo, abrir e acessar objetos distribuídos na internet simplesmente utilizando suas URLs – é como se uma pessoa estivesse acessando um arquivo em seu próprio computador.

Múltiplos *threads*

Para entendermos essa característica, é necessário definir o que é um *thread*. De forma geral, um programa de computador é capaz de executar apenas um comando específico de cada vez. Entretanto, às vezes, precisamos que um programa execute mais de um comando ao mesmo tempo.

Vamos supor que você esteja utilizando um determinado programa de computador cuja função é controlar vendas de uma loja. Suponha, ainda, que ele tenha uma funcionalidade que permite consultar todo o histórico de vendas dessa loja. É bem provável que a loja tenha muitas vendas registradas no programa, correto? Portanto, essa funcionalidade pode levar bastante tempo para responder, isto é, para exibir todos os registros de vendas existentes. Assim, para evitar que o usuário do programa fique aguardando o término da operação, é interessante permitir que ele execute outra operação no sistema ao mesmo tempo. A essa ideia de executar duas ou mais operações em um programa damos o nome de paralelismo.

Mas o que isso tem a ver com *threads*?

Threads são os recursos disponíveis na linguagem Java para que o paralelismo possa ser colocado em prática. De maneira simples, podemos considerar que cada operação executada em paralelo em um programa corresponde a uma *thread* diferente. Elas permitem uma melhor interação do usuário com o programa, bem como um melhor comportamento em tempo real.

Programando o conhecimento

Correlacione as características do Java com suas definições. Em seguida, assinale a alternativa correta.

I. Robusta

II. Portável

III. Interpretada

IV. Dinâmica

( ) Permite que um programa qualquer seja executado em qualquer platafoma ou equipamento.

( ) Permite que novos recursos sejam adicionados ao programa em tempo de execução, adaptando-se ao ambiente.

( ) Permite a escrita de programas de diferentes graus de complexidade e com mecanismos de tratamento de erros.

( ) O código-fonte do programa, após ser compilado, resulta nos chamados *bytecodes*, que podem ser executados a partir da JVM.

A sequência correta de associações é:

a) I – II – III – IV.

b) II – IV – I – III.

c) II – III – IV – I.

d) IV – II – III – I.

Comentário: o Java possui diversas características que realizam funções distintas no processo de programação. A alternativa que relaciona corretamente as características com sua definição é a "b".

1.3 Plataforma Java

Quando utilizamos a palavra Java, muitas vezes estamos no referindo à linguagem de programação especificamente. Entretanto, Java é muito mais do que simplesmente uma linguagem de programação. A tecnologia envolve uma série de recursos integrados em um ambiente computacional, ou plataforma, criado para dar suporte ao desenvolvimento de programas, utilizando as diversas linguagens disponíveis para a plataforma, sendo a linguagem Java a mais comum. Mas, antes de continuar, vamos entender o que vem a ser uma plataforma.

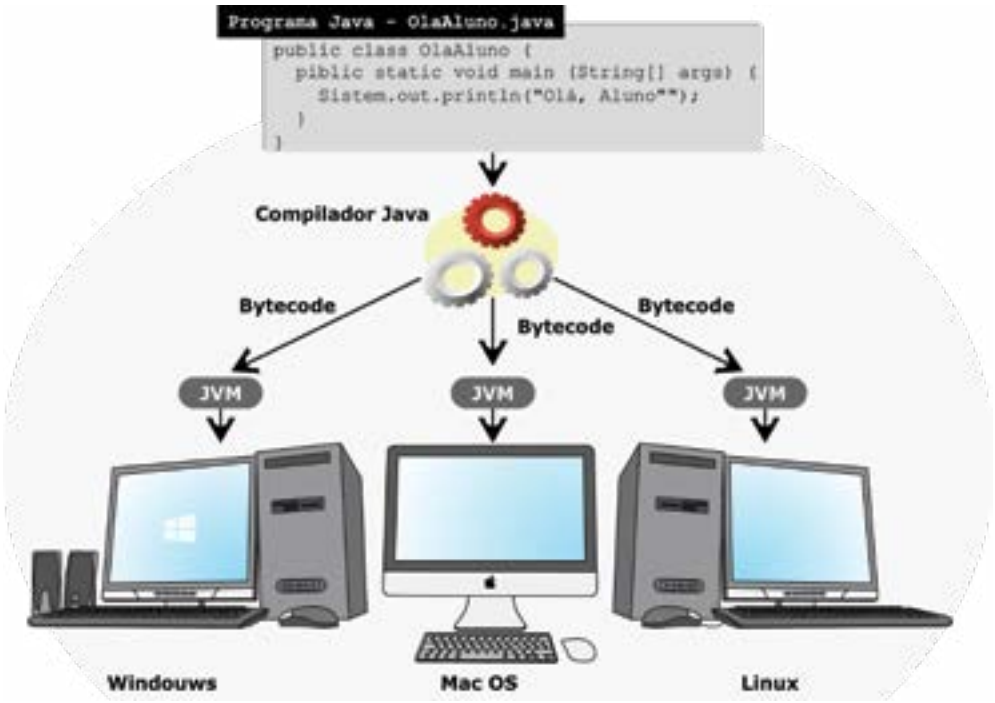
Uma plataforma pode ser definida como um ambiente de *hardware* ou no qual um programa é executado. Como exemplos, podemos citar algumas plataformas conhecidas, como: Microsoft Windows, Linux, Solaris e Mac, as quais envolvem tanto *hardware* como *software*. Por outro lado, a plataforma Java difere um pouco dessas outras, já que é totalmente baseada em , ou seja, é independente do *hardware* do dispositivo que roda o programa (GALLARDO et al, 2014).

A plataforma Java envolve dois componentes principais, listados a seguir.

- **Máquina Virtual Java (*Java Virtual Machine* – JVM)**: realiza a interpretação do *bytecode*, um código intermediário obtido a partir da compilação do código-fonte Java, para depois ser executado em um computador de qualquer arquitetura. Isto é, o mesmo *bytecode* Java pode ser executado em computadores com sistemas operacionais diferentes, sem a necessidade de fazer alterações no programa, exceto quando se tratar de alguma biblioteca específica de determinada plataforma. Para facilitar o entendimento, veja a figura a seguir, que mostra uma visão geral da operação da JVM. Inicialmente, o código-fonte do programa Java escrito pelo programador é compilado, produzindo como resultado os *bytecodes* correspondentes em um arquivo com extensão ".class". Em seguida, a JVM lê e interpreta esses *bytecodes*, gerando o código de máquina específico para a plataforma na qual o programa será executado.
- **Interface de Programação de Aplicações Java (*Application Programming Interface* – API)**: fornece um conjunto de funcionalidades essenciais para o desenvolvimento de aplicações em Java. Essa API envolve um conjunto grande e bastante variado de classes e interfaces que podem ser utilizadas pelo programador. Entre os recursos disponíveis, podemos citar: opera-

ções de manipulação de números e textos, recursos de gerenciamento de banco de dados, redes e segurança.

Visão geral da operação da Máquina Virtual Java



Como já vimos no histórico da linguagem, a evolução do Java levou à divisão da plataforma em três edições principais a partir da versão 2:

- **Java Standard Edition (JSE)**: é a edição principal da linguagem e serve de base para o JEE e JME. É utilizada para o desenvolvimento de aplicações para PCs e servidores, fornecendo vários tipos básicos de dados, bibliotecas gráficas, acesso a bancos de dados, entre outros elementos;

- **Java Enterprise Edition (JEE)**: edição voltada para o desenvolvimento de aplicações distribuídas em redes, como internet e intranet, que exigem o acesso simultâneo de usuários. Contém diversas bibliotecas que permitem o acesso a servidores de aplicação e servidores de *e-mail*, entre outros recursos, como suporte interno à manipulação de *Java Server Pages* (JSP) e *extensible markup language* (XML);

- **Java Micro Edition (JME)**: edição voltada para o desenvolvimento de aplicações que são executadas em pequenos dispositivos móveis ou portáteis, com poucos recursos computacionais (memória e processamento), na forma de embutido, como celulares, impressoras, dispositivos *Blu-ray* e controles remotos.

A figura a seguir apresenta o diagrama conceitual da plataforma Java, sendo possível ver que ela envolve uma série de tecnologias, bibliotecas e ferramentas adicionais. Como exemplos, podemos citar o Java Web Start, uma tecnologia utilizada para instalar e executar aplicações, o *swing* e o *abstract window toolkit* (AWT), duas bibliotecas gráficas, e a biblioteca *Java Database Connectivity* (JDBC), utilizada para manipulação de bancos de dados.

Diagrama conceitual da plataforma Java

É importante destacar que não é necessário utilizar todos os elementos da plataforma para criar um programa em Java. Você pode aplicar apenas um subconjunto desses elementos de acordo com a necessidade do seu programa. Por exemplo, das bibliotecas de interface gráfica de usuário, você poderia escolher utilizar apenas a *swing*, deixando de lado as demais bibliotecas, como o Java 2D e o AWT.

Saiba mais

A partir da versão 8 da plataforma Java, foi integrado a ela o JavaFX, que é uma tecnologia de *software* que pode ser utilizada em conjunto com os demais recursos da plataforma Java para a criação de aplicações com interfaces gráficas modernas e com conteúdo rico de áudio e vídeo. Com essa nova tecnologia, é possível desenvolver aplicações para computador *desktop*, celular, *web* e televisão digital, por exemplo. Ela é gratuita e está disponível para os seguintes sistemas operacionais: Windows, Linux e Mac OS X. Para saber mais informações sobre o JavaFX, acesse a página oficial pelo *link*: <http://docs.oracle.com/javase/8/javase-clienttechnologies.htm>.

Programando o conhecimento

Java consiste em uma plataforma robusta com diversos componentes e bibliotecas integradas para permitir o desenvolvimento de portável e flexível. Sobre essa plataforma, assinale a alternativa correta.

a) A JVM é um dos componentes principais da plataforma e tem como principal finalidade a análise e a compilação de código-fonte para posterior execução da aplicação.

b) Das três edições principais da plataforma, o JavaFX foi incluso mais recentemente, permitindo o desenvolvimento de aplicações para dispositivos móveis e portáteis, como os celulares.

c) A API Java fornece um conjunto de funcionalidades essenciais para o desenvolvimento de aplicações, incluindo uma grande variedade de classes e interfaces que podem ser utilizadas pelo programador para auxiliá-lo no desenvolvimento.

d) A edição JSE é a edição principal da linguagem e serve de base para o JEE e JME. Sua principal finalidade é o desenvolvimento de aplicações para utilização em redes de computadores, como a internet.

Comentário: a API contém um conjunto grande e essencial de recursos que podem ser utilizados pelo programador no desenvolvimento de seu trabalho. Portanto, a alternativa correta é a "c".

Compilador: programa de computador que lê e analisa o código-fonte de um programa e o converte para linguagem de máquina.

1.4 Ambiente de desenvolvimento

Antes de começarmos, você sabe o que é um ambiente de desenvolvimento?

Ambiente de desenvolvimento pode ser definido como o conjunto de elementos necessários para o desenvolvimento de um *software*. Ele envolve *softwares* como editores de texto, compiladores, interpretadores, entre outros.

Para o desenvolvimento de aplicações em Java, é necessário utilizar o chamado Java Development Kit (JDK), que inclui um conjunto de recursos essenciais, como o Java Runtime Environment (JRE), o compilador Java e as APIs Java (ORACLE CORPORATION, 2016). Além do JDK, vamos precisar utilizar alguma ferramenta para a edição de código-fonte, podendo ser um editor de texto simples, como o bloco de notas do Windows ou alguma ferramenta com mais recursos, como um ambiente de desenvolvimento integrado (IDE).

Atenção

Antes de realizarmos a preparação do ambiente, é necessário definir o sistema operacional que será utilizado, visto que o JDK deve ser específico para a plataforma utilizada: Windows, Linux, Mac OS X ou Solaris.

A preparação do nosso ambiente de desenvolvimento seguirá quatro etapas:

1. obtenção e instalação do JDK;

2. configuração de variáveis de ambiente;

3. teste da instalação;

4. instalação e configuração da ferramenta IDE.

Obtenção do JDK

Para obter a versão mais recente do JDK, a maneira mais simples e confiável é acessar a página de *download* do Java SE, disponível no seguinte sítio da Oracle:

http://www.oracle.com/technetwork/java/javase/downloads/index.html.

Nessa página, você verá esta imagem ao lado, contendo o *link* para a página de *download* da última versão disponível do JDK. O nome da versão geralmente possui o formato: **Java Platform 8u121**, ressaltando que o número **8** corresponde ao número da edição do Java, e a sequência **u121** indica o número de atualizações feitas nessa edição. Ao fazer o *download*, procure sempre a última versão disponível, que já contém diversas melhorias e correções em relação às anteriores.

Saiba mais

A plataforma Java possui diversos pacotes para *download*, cada um deles com um propósito específico: JDK, JRE e Server JRE (ORACLE CORPORATION, 2017).

JDK (Java SE Development Kit): destinado a desenvolvedores Java, inclui um conjunto de ferramentas para desenvolver, testar e monitorar aplicações Java.

JRE (Java Runtime Environment): destinado para usuários finais do Java, contém tudo o que é necessário para executar aplicações Java.

Server JRE (Server Java Runtime Environment): destinado a desenvolvedores de aplicações Java para servidores, inclui monitoramento da JVM e ferramentas necessárias para as aplicações.

Após ter acessado o *link*, você será redirecionado para uma página contendo uma lista de versões do JDK para *download*. Cada item dessa lista corresponde a uma plataforma de sistema operacional diferente, como mostrado na figura a seguir. Portanto, dependendo do sistema operacional (Windows, Linux, Mac OS X ou Solaris) e da arquitetura (x86 ou 64 *bits*) em utilização, será necessário escolher um pacote específico para *download* para não ter problemas na instalação. Após definir o sistema operacional a ser utilizado, selecione a opção "Accept Licence Agreement" na parte de cima dessa lista (destacada em vermelho na imagem a seguir) e clique no *link* para *download* do pacote de instalação disponível ao lado do nome da versão. Pronto! O *download* será iniciado na sequência.

Versões do JDK disponíveis para *download*

Instalação do JDK

A instalação do JDK varia de acordo com a plataforma escolhida. Iremos, então, explicar como fazer a instalação em Windows 10 e Linux, que são as plataformas mais utilizadas.

Saiba mais

Para realizar a instalação do JDK para as plataformas Mac OS X e Solaris, você pode seguir as orientações contidas no *link* a seguir:

- Como instalar o Java para o meu Mac?

https://www.java.com/pt_BR/*download*/help/mac_install.xml

- Como fazer *download* e instalar Java para Linux?

https://www.java.com/pt_BR/*download*/help/solaris_install.xml

Instalação do JDK no Windows 10

Pronto para instalar o JDK? Siga, então, cada um dos passos apresentados a seguir.

1º passo – vá ao diretório em que salvou o arquivo de instalação baixado e execute o programa para iniciar o processo de instalação.

> **Atenção**
>
> O programa de instalação deve ser executado como usuário Administrador do sistema operacional.

2º passo – será aberta uma janela inicial apenas com informações básicas sobre a instalação, conforme imagem a seguir. Basta clicar no botão "*Next*" para continuar.

3º passo – a próxima janela, apresentada na figura a seguir, exibe os itens que serão instalados e indica o diretório para a instalação. Caso queira, você pode fazer a alteração do diretório utilizando o botão "Change", mas não é necessário. Em seguida, clique no botão "*Next*" para seguir ao passo seguinte.

4º passo – após indicar o diretório, a instalação será iniciada. Uma tela com a barra de progresso será exibida. Por fim, será exibida uma tela indicando que a instalação foi realizada com sucesso, clique no botão "Close" para finalizar a instalação, conforme a última figura.

Informações básicas

Itens da instalação

Progresso da instalação e tela de encerramento

Pronto! Você já está com o JDK instalado em sua máquina. Agora precisamos fazer a configuração das variáveis de ambiente do sistema operacional.

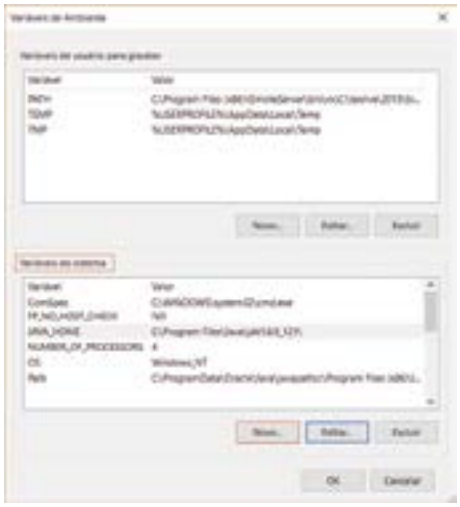
Configuração de variáveis de ambiente

1º passo – localize o item "Meu Computador" do Windows, clique com o botão direito do *mouse* sobre ele e selecione a opção "Propriedades".

2º passo – selecione o item "Configurações avançadas do sistema". Uma nova janela será aberta, e você deverá selecionar a aba "Avançado".

3º passo – clique no botão "Variáveis de Ambiente". Será aberta uma nova janela.

4º passo – clique no botão "Novo..." na parte da tela correspondente a "Variáveis do sistema". Uma nova janela será aberta para informar o nome e o valor das variáveis.



Variáveis de ambiente

5º passo – preencha os campos da seguinte forma:

- Nome da variável: **JAVA_HOME**
- Valor da variável: preencha aqui com o endereço de instalação do JDK, de acordo com o que foi informado na sua instalação, conforme a figura a seguir. No nosso caso, seria "C:\Program Files\Java\jdk1.8.0_121\".

Edição de variável de sistema

Variáveis de ambiente: variável dinâmica de um sistema operacional que permite armazenar informações úteis para a execução de programas, como caminhos e diretórios específicos para determinadas aplicações.

6º passo – clique no botão "OK".

7º passo – na tela com a lista de variáveis de ambiente, localize a variável "*Path*", selecione-a, e clique no botão "Editar...". Uma nova janela será aberta com os valores já cadastrados para essa variável. Então, clique no botão "Novo".

8º passo – preencha o campo com o seguinte valor: "%JAVA_HOME%\bin" e clique no botão "OK".

9º passo – a instalação foi concluída. Agora vamos testar se ela foi feita corretamente. A partir do menu "Iniciar" do Windows, localize e execute o programa "*Prompt* de Comando". Uma nova janela será aberta, na qual você deverá digitar os comandos das figuras a seguir.

```
java -version
```

Informações java -version

Informações javac -version

Se você executou os comandos e obteve as mesmas informações, então sua instalação está correta e pronta para uso. O primeiro comando mostra as versões dos componentes instalados do Java, e o segundo expõe a versão do compilador Java.

Instalação do JDK no Linux

A versão do Linux utilizada neste roteiro de instalação será o Debian. Caso você utilize outra versão de Linux, consulte as instruções de instalação disponíveis em:

https://docs.oracle.com/javase/8/docs/technotes/guides/install/linux_jdk.html

1º passo – abra o aplicativo "Terminal".

2º passo – torne-se superusuário (root) com o comando:

```
su -
```

3º passo – adicione o repositório de binários.

```
echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" | tee /
etc/apt/sources.list.d/webupd8team-java.list
```

4º passo – adicione o repositório de código-fonte.

```
echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main" |
tee -a /etc/apt/sources.list.d/webupd8team-java.list
```

5º passo – adicione a chave do repositório.

```
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886
```

6º passo – atualize o Apt.

```
apt-get update
```

7º passo – agora instale o programa com o seguinte comando:

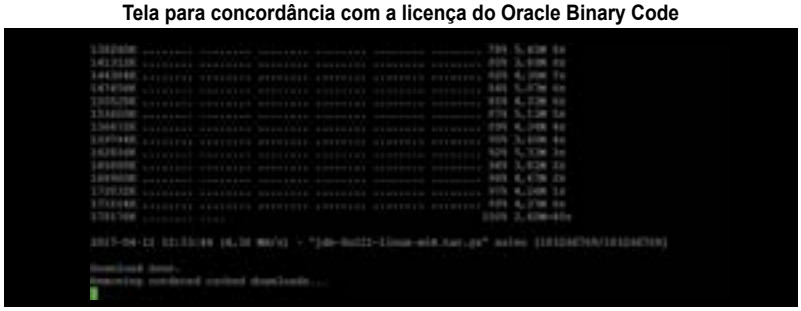
```
apt-get install oracle-java8-installer
```

8º passo – ao executar o instalador, você deverá concordar com os termos de uso, para poder, finalmente, baixar e instalar o Java. Em seguida, será realizada a instalação.

Repositório: local de armazenamento de pacotes *softwares* destinados a auxiliar o processo de instalação e configuração de aplicativos e ferramentas de *software*.

Tela para concordância com a licença do Oracle JDK

Tela para concordância com a licença do Oracle Binary Code



Tela que exibe o progresso da instalação

9º passo – há um pacote no repositório que define automaticamente as variáveis de ambiente Java 8 e define JDK8 como o JDK padrão. Para instalá-lo, use o seguinte comando:

```
sudo apt-get install oracle-java8-set-default
```

Programando o conhecimento

A plataforma Java está disponível em diversas edições, e cada uma delas é utilizada para objetivos específicos. Sobre a preparação do ambiente de desenvolvimento Java, assinale a alternativa correta.

a) A instalação do JRE é suficiente para o desenvolvimento de *software* em Java, pois já inclui o compilador da linguagem.

b) Para a instalação do JDK, é necessário fazer o *download* do arquivo de instalação, que é o mesmo para qualquer sistema operacional utilizado.

c) A configuração de variáveis de ambiente, que geralmente contém informações sobre o sistema e sobre caminhos de diretórios específicos no sistema de arquivos, é uma etapa opcional do processo de instalação do Java.

d) O JDK é um dos componentes do Java que inclui um conjunto de recursos essenciais, como o JRE, o compilador Java e as APIs Java.

Comentário: o JDK é destinado a desenvolvedores Java, incluindo ferramentas para desenvolver, testar e monitorar aplicações Java, como o JRE, os compiladores e as APIs. Portanto, a alternativa correta é a "d".

1.5 Ferramentas IDE

De acordo o Glossário de Tecnologia da Informação da Gartner (2017), o *integrated development environment* (IDE) – em português, ambiente de desenvolvimento integrado – corresponde a um ambiente para a escrita da lógica e projeção de interfaces para aplicações. Nesse ambiente, as ferramentas são utilizadas como uma forma de auxiliar o processo de desenvolvimento de diversas formas e, para isso, geralmente possuem recursos integrados, como um editor de código-fonte, um compilador ou interpretador e um depurador (*debugger*).

Depurador: programa de computador utilizado para auxiliar a identificação de defeitos em códigos-fonte de outros programas.

Existem diversas ferramentas IDE disponíveis para trabalhar com desenvolvimento de aplicações em Java. Entre elas, destacam-se duas ferramentas livres: Eclipse e NetBeans.

A ferramenta NetBeans é totalmente feita em Java, motivo pelo qual é independente de plataforma, funcionando em qualquer sistema operacional que tenha a Máquina Virtual Java instalada. Ela fornece recursos aos programadores para a escrita, a compilação e a depuração de aplicações, bem como oferece variadas bibliotecas adicionais que facilitam o desenvolvimento de aplicações e aumentam a produtividade da equipe de desenvolvimento.

Neste material, adotaremos o Eclipse como ferramenta IDE, pois, entre as duas opções, é a preferida e mais utilizada entre os desenvolvedores, além de possuir versões para Windows, Linux, Mac OS X e Solaris.

Agora vamos, então, aprender como instalar e configurar o Eclipse para executar nosso primeiro programa em Java.

Instalação do Eclipse no Windows

1º passo – para realizar o *download* do arquivo de instalação do Eclipse, acesse a página http://www.eclipse.org/*downloads*/eclipse-*packages*. Será, com isso, exibida uma nova página com uma lista de versões do Eclipse. Selecione a versão a ser utilizada, que, no nosso caso, será a versão Eclipse IDE *for* Java Developers. Escolha, então, a plataforma correspondente ao seu sistema operacional (32 *bit* ou 64 *bit*) e inicie o *download*.

2º passo – após a conclusão do *download*, vá até o local onde salvou o arquivo (normalmente na pasta de *downloads*) e descompacte o arquivo em uma pasta do seu computador. A pasta extraída já contém os arquivos para executar o Eclipse, isto é, não é necessário fazer a instalação do programa.

Atenção

Para descompactar o arquivo baixado, você vai precisar de um programa de descompactação, como o Winrar, que pode ser obtido na página http://www.win-rar.com.

3º passo – para iniciar o Eclipse, basta executar o arquivo chamado "eclipse.exe" dentro da pasta onde descompactou os arquivos baixados.

Instalação do Eclipse no Linux

A versão do Linux utilizada neste roteiro de instalação será o Debian. Lembre-se de alternar para o usuário com permissões de superusuário (*root*) antes de executar os passos a seguir.

1º passo – inicie o aplicativo "Terminal".

2º passo – primeiramente, vamos remover instalações antigas do Eclipse que, eventualmente, estejam instaladas na máquina. Para isso, execute os dois comandos que seguem.

```
> rm -Rf /opt/eclipse/
> rm -Rf /usr/share/applications/eclipse.desktop
```

3º passo – verifique a versão do seu sistema operacional utilizando o seguinte comando:

```
> uname -m
```

4º passo – para obter a última versão da ferramenta, utilize um dos comandos abaixo, de acordo com a versão do seu sistema operacional. Caso o *link* esteja desatualizado, acesse a página <http://eclipse.c3sl.ufpr.br/technology/epp/*download*s/release/neon/3/> para baixar a última versão, e salve o arquivo com o nome "eclipse.tar.gz".

- Versão 32 *bit*

```
> wget http://eclipse.c3sl.ufpr.br/technology/epp/downloads/release/neon/3/eclipse-java-neon-3-linux-gtk.tar.gz -O eclipse.tar.gz
```

- Versão 64 *bit*

```
> wget http://eclipse.c3sl.ufpr.br/technology/epp/downloads/release/neon/3/eclipse-java-neon-3-linux-gtk-x86_64.tar.gz -O eclipse.tar.gz
```

5º passo – depois de completo o *download*, execute o comando a seguir para descompactar os arquivos:

```
> sudo tar -zxvf eclipse.tar.gz -C /opt/
```

6º passo – faça o *download* do ícone do Eclipse e salve-o na mesma pasta:

```
> sudo wget https://d12.macupdate.com/images/icons128/11662.png -O /opt/eclipse/eclipse.png
```

Após seguir esses passos, você pode iniciar o Eclipse utilizando o seguinte comando:

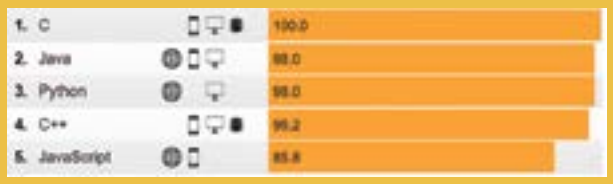
```
> /opt/eclipse/eclipse
```

Outra opção para executar o programa é utilizar o gerenciador de arquivos do sistema, e, para isso, basta abrir sua pasta de instalação e clicar em seu arquivo executável.

Saiba mais

Atualmente, existem diversas linguagens de programação disponíveis, cada uma com seus propósitos específicos. Dependendo do tipo de programa desenvolvido (aplicação *web* ou aplicativo para dispositivos móveis, por exemplo), os programadores preferem utilizar uma linguagem específica. Para se ter uma ideia das linguagens mais utilizadas, o Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE), uma organização internacional sem fins lucrativos, fundada nos Estados Unidos, criou o *ranking* das linguagens de programação, disponível na revista IEEE Spectrum (DIAKOPOULOS; CASS, 2016). Nesse *ranking*, o Java aparece como a segunda linguagem mais importante para o desenvolvimento de aplicações para internet, dispositivos móveis e computadores, ficando atrás apenas da linguagem C, conforme pode ser visto na figura a seguir.

Ranking das linguagens de programação



Programando o conhecimento

Um dos elementos que compõem o ambiente de desenvolvimento Java é a ferramenta IDE. Sobre esse tema, assinale a alternativa correta.

a) Eclipse e Netbeans são duas IDEs bastante utilizadas para o desenvolvimento em Java, observando-se que, enquanto o Eclipse é gratuito, o Netbeans deve ser adquirido para uso.

b) As IDEs possuem recursos integrados, como um editor de código-fonte, um compilador ou interpretador e um depurador.

c) Eclipse permite ao desenvolvedor escrever código-fonte de programas em diversas linguagens de programação, como Java e PHP, não sendo necessário instalar nenhuma versão diferenciada ou biblioteca para isso.

d) A escolha do sistema operacional a ser utilizado com a ferramenta IDE, como Windows ou Linux, é de fundamental importância, mas a versão deles não impacta na instalação.

Comentário: as IDEs visam auxiliar no processo de desenvolvimento de diversas formas e, para isso, possuem recursos integrados, como os mencionados na alternativa "b", que é a resposta correta.

Resumindo

Nesta lição, estudamos os conceitos iniciais relacionados à linguagem Java que serão necessários para você compreender questões mais avançadas. A partir do conhecimento do histórico da linguagem Java, você pôde perceber como ela evoluiu até alcançar o nível atual, que disponibiliza uma grande quantidade de recursos. Além disso, aprendemos como preparar o ambiente básico para o desenvolvimento de aplicações Java, incluindo a instalação e a configuração do JDK e do Eclipse.

Veja se você se sente apto a:

- relatar a história da linguagem de programação Java e suas origens;
- explicar os conceitos gerais da linguagem Java;
- demonstrar as principais características da plataforma Java;
- preparar o ambiente de desenvolvimento Java e instalar a ferramenta Eclipse.

Parabéns, você finalizou esta lição!

Agora responda às questões ao lado.

Exercícios

Questão 1 – Antes do surgimento do Java, diversas linguagens de programação foram criadas. Sobre o surgimento do Java, marque a alternativa correta.

a) A linguagem foi criada a partir de um projeto iniciado pela Sun Micro*systems* e tinha como objetivo desenvolver aplicações para grandes computadores.

b) A linguagem foi projetada a partir da evolução das linguagens C e C++, que eram bastante utilizadas na época.

c) A linguagem, apesar de utilizar conceitos já existentes, veio para substituir completamente outras linguagens de programação que estavam ficando ultrapassadas.

d) A linguagem Java foi uma ideia totalmente inovadora, sendo criada sem nenhuma base preexistente ou influência de códigos de outras linguagens.

Questão 2 – A respeito das origens e das características da linguagem Java, assinale a alternativa correta.

a) A linguagem, inicialmente, foi denominada de Oak e permitia o desenvolvimento de aplicações em *Applets*, que podiam ser executados em navegador *web*.

b) O surgimento da linguagem ocorreu a partir de um projeto desenvolvido pela Oracle Corporation, mas que não obteve sucesso em decorrência do surgimento da internet, sendo abandonado no ano seguinte.

c) Desde o início, a linguagem tinha como objetivo auxiliar o desenvolvimento de aplicações para dispositivos móveis, como celulares e *tablets*.

d) A utilização da linguagem Java para a comunicação em redes via internet não era uma das intenções da equipe envolvida nesse projeto, já que a internet ainda era muito simples e não oferecia recursos interessantes.

Questão 3 – A linguagem Java surgiu na década de 90, e sua criação foi motivada principalmente:

a) pelo enfraquecimento das outras linguagens existentes à época.

b) pelo desenvolvimento e avanço de tecnologias relacionadas à internet.

c) pela baixa eficiência da linguagem.

d) pela falta de uma linguagem orientada a objetos.

Questão 4 – Sobre as características da linguagem Java, assinale a opção incorreta.

a) Permite o desenvolvimento de programas de diferentes complexidades.

b) É baseada no paradigma orientado a objetos.

c) Pode ser executada em qualquer computador com qualquer sistema operacional, independentemente de ter a máquina virtual Java instalada.

d) Possui diversas bibliotecas de código que facilitam o desenvolvimento de aplicações.

Questão 5 – Entre as características gerais do Java, podemos citar sua robustez, seu alto desempenho, o fato de ser interpretado e de ser compatível com redes. Sobre essas características, assinale a alternativa que define corretamente a característica apresentada.

a) É possível a escrita de programas em Java que atendam a diversos tipos de clientes, com diversos tipos de aplicações, simples ou complexas, inclusive com mecanismos de tratamento de erros, característica relacionada ao fato de a linguagem ser interpretada.

b) A Máquina Virtual Java é um componente essencial da plataforma Java, que permite interpretar código-fonte compilado, oferecendo portabilidade à linguagem, devido à sua robustez.

c) A compatibilidade com redes provida pelo Java permite realizar programas de computador que acessem ou compartilhem informações via redes de dados utilizando protocolos de comunicação, como o TCP/IP e FTP.

d) A linguagem Java foi projetada inicialmente para oferecer alto desempenho às aplicações, mas, com a incorporação da Máquina Virtual Java, esse desempenho foi afeitado consideravelmente.

Questão 6 – Ao longo da sua evolução, o Java sofreu várias modificações, visando tornar a linguagem cada vez mais robusta e de maior potencial. Sobre as versões da linguagem, assinale a alternativa correta.

a) A versão 1.0 foi a primeira versão da linguagem e era basicamente voltada para o desenvolvimento de aplicações comerciais.

b) A versão 1.2 representou um marco, uma vez que, a partir dela, houve a divisão da plataforma em três edições: J2SE, J2ME e J2EE.

c) A versão 5, apelidada de Tiger, trouxe poucas alterações para a linguagem e não teve grande repercussão entre os desenvolvedores.

d) A versão atual da linguagem (versão 7) incluiu uma nova biblioteca de manipulação de arquivos e gerenciamento automático de recursos.

Questão 7 – O Java possui diversas características que o levaram a uma posição de destaque na comunidade de programação. Entre as opções que seguem, qual delas não faz parte dessas características?

a) Orientada a objetos.

b) Portabilidade.

c) Compatibilidade com redes.

d) *Threads* simples.

Questão 8 – Sobre as características do Java, assinale a opção correta.

a) Simplicidade corresponde ao desenvolvimento de programas de forma rápida e fácil, porém somente após intensos treinamentos da equipe.

b) Dizer que a linguagem é robusta significa dizer que é possível escrever com ela programas de diferentes complexidades e que sejam estáveis e confiáveis.

c) Portabilidade está relacionada ao desenvolvimento de aplicações para computadores que possam ser deslocados para diferentes locais, dando maior mobilidade ao usuário.

d) Não é possível desenvolver programas que realizem mais de uma operação ao mesmo tempo, visto que a linguagem não permite paralelismo com múltiplos *threads*.

Questão 9 – Uma das características da linguagem Java é ser orientada a objetos. Sobre esse conceito, assinale a opção que contém as palavras que completam adequadamente a afirmação a seguir.

A orientação a objetos é uma forma de projetar e desenvolver um _____ de computador, utilizando, para isso, conceitos e comportamentos do _____ que deverão ser mapeados para orientar os comportamentos no _____.

a) programa / mundo computacional / mundo real.

b) programa / mundo real / mundo computacional.

c) *software* / cenário imaginário / cenário real.

d) aplicativo / mundo imaginário / mundo computacional.

Questão 10 – A versão atual do Java (Java SE 8) foi lançada no ano de 2014 e incorporou alguns recursos importantes para a linguagem. Entre eles, podemos citar:

a) tipos enumerados, usados para criar estruturas de dados organizados compostas de valores que representem tipos de informações semelhantes.

b) biblioteca JDBC (Java Database Connectivity), utilizada para o acesso e a manipulação de bancos de dados.

c) *Applets*, uma pequena aplicação que roda em um navegador *web* utilizando a Máquina Virtual Java (JVM).

d) expressões lambda, que permitem utilizar uma funcionalidade como argumento de método, por exemplo.

GLOSSÁRIO

APPLICATION PROGRAMMING INTERFACE (API) – Conjunto de rotinas e padrões de programação utilizados para a comunicação entre programas, aplicativos ou componentes de programas

BANCO DE DADOS RELACIONAL – Recurso que permite o armazenamento de dados na forma de tabelas organizadas em linhas e colunas, ressaltando que cada coluna corresponde a um determinado tipo de dado, e cada linha corresponde a um registro distinto.

BYTECODES **–** Códigos intermediários entre o código-fonte Java, escritos pelo programador, e o código em linguagem de máquina, resultante do processo de compilação.

COMPILADOR – Programa de computador que lê e analisa o código-fonte de um programa e o converte para linguagem de máquina.

CONSOLE – O console do Java fornece informações sobre a versão do Java, o diretório principal do usuário e as mensagens de erro exibidas durante a execução de um programa. Serve também como recurso para a entrada e a saída de dados de usuário.

DEPURADOR – Programa de computador utilizado para auxiliar a identificação de defeitos em códigos-fonte de outros programas.

DRIVER JDBC – Componente de *software* que contém as operações necessárias para a interação entre uma aplicação Java e um banco de dados relacional.

ESCOPO DE VARIÁVEL – Representa o contexto em que uma variável de um programa foi criada, e o trecho do programa em que ela é visível pode ser acessado.

FILE TRANSFER PROTOCOLO (FTP) – Protocolo utilizado para transferência de arquivos em redes de computadores.

FLUXOGRAMA – Tipo de diagrama que permite representar graficamente processos, fluxos de trabalho ou algoritmos de um programa de computador. Para isso, geralmente, são utilizados em conjunto símbolos como círculos, triângulos, retângulos, linhas e setas.

HYPERTEXT TRANSFER PROTOCOL (HTTP) – Protocolo de comunicação utilizado para troca de informação hipermídia (imagens, sons e textos) na internet.

JAVA DATABASE CONNECTIVITY (JDBC) – Conjunto de classes e interfaces Java que permitem realizar operações em bancos de dados, como inclusão, alteração, exclusão e consulta.

JAVA NAMING AND DIRECTORY INTERFACE (JNDI) – Conjunto de classes e interfaces Java que permitem realizar serviços de consulta em diretórios, como de dados de usuários e *e-mail*, por meio de nomes na aplicação Java.

JAVA VIRTUAL MACHINE (JVM) – Programa responsável por carregar e executar aplicativos escritos na linguagem Java, convertendo os *bytecodes* em código executável de máquina.

REPOSITÓRIO – Local de armazenamento de pacotes *softwares* destinados a auxiliar o processo de instalação e configuração de aplicativos e ferramentas de *software*.

TRANSPORT CONTROL/INTERNET PROTOCOL (TCP/IP) – Protocolo de rede utilizado para gerenciar a comunicação em redes de computadores, envolvendo uma série de outros protocolos.

VARIÁVEL DE AMBIENTE – Variável dinâmica de um sistema operacional que permite armazenar informações úteis para a execução de programas, como caminhos e diretórios específicos para determinadas aplicações.

BIBLIOGRAFIA

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML**: Guia do Usuário. Tradução da segunda edição: Fábio Freitas da Silva. Rio de Janeiro: Campus, 2006.

DEITEL, H. M.; DEITEL P. J. **Java**: como programar. 10. ed. São Paulo: Pearson Education do Brasil, 2016.

DIAKOPOULOS, N.; CASS, S. **Interactive**: The Top Programming Languages 2016. **IEEE Spectrum,** New York, 26 jul. 2016. Disponível em: <http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016>. Acesso em: 28 mar. 2017.

FURGERI, S. **Java 8 – Ensino Didático**: Desenvolvimento e Implementação de Aplicações. São Paulo: Érica/Saraiva, 2015.

GALLARDO, R. et al. **The Java Tutorial**: A Short Course on the Basics. 6. ed. New York: Addison-Wesley Professional, 2014.

GARTNER. **Gartner IT** Glossary - IDE (integrated development environment). 2017. Disponível em: <http://www.gartner.com/it-glossary/ide-integrated-development-environment/>. Acesso em: 5 abr. 2017.

GOSLING, J.; MCGILTON, H. **The Java Language Environment**: Contents. Mountain View: Sun Microsystems Inc, 1995.

INSTITUTO DE ENGENHEIROS ELETRICISTAS E ELETRÔNICOS. **IEEE 754: IEEE** Standard for Floating-Point Arithmetic. New York: IEEE, 2008.

KAY, A. C. The Early History of Smalltalk. **ACM SIGPLAN Notices**, New York, p. 69-95. mar. 1993.

MANZANO, J. A. N. G.; COSTA JÚNIOR, R. A. da. **Programação de Computadores com Java**. São Paulo: Érica, 2014.

ORACLE CORPORATION. Java History Timeline. **Oracle Corporation**, Redwood City, 2014. Disponível em: http://oracle.com.edgesuite.net/timeline/java/. Acesso em: 28 de março de 2017.

_____. Informações do Java 8. **Oracle Corporation**, São Paulo, 2015. Disponível em: <https://www.java.com/pt_BR/download/faq/java8.xml>. Acesso em: 2 abr. 2017.

_____. Como posso começar a desenvolver programas Java com o Java Development Kit (JDK)? **Oracle Corporation**, São Paulo, 2016. Disponível em: <https://www.java.com/pt_BR/download/faq/develop.xml>. Acesso em: 2 abr. 2017.

_____. Java SE Downloads. Oracle Corporation, Redwood City, 2017. Disponível em: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. Acesso em: 2 abr. 2017.

PRESSMAN, R. S.; MAXIM, B. **Engenharia de Software**. 8. ed. Porto Alegre: Amgh Editora, 2016.

SCHILDT, H. **Java para Iniciantes.** 6. ed. Porto Alegre: Bookman, 2015.

SEBESTA, R. W. **Conceitos de Linguagens de Programação**. 9. ed. Porto Alegre: Bookman, 2011.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. Sistemas de Bancos de Dados. 6. ed. Amsterdã: Elsevier, 2012.