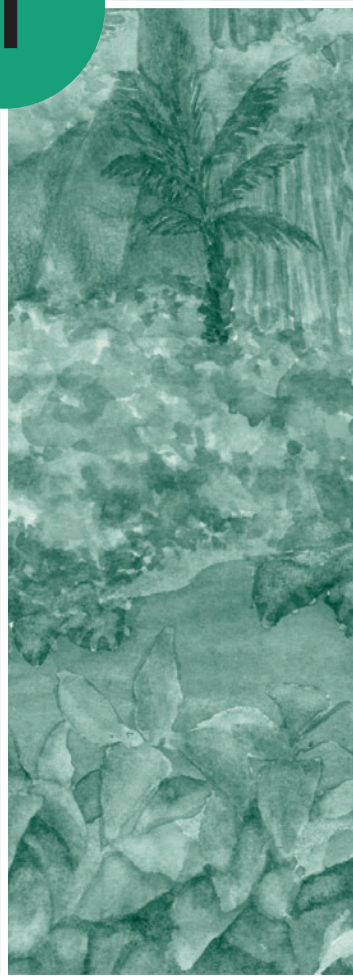
Capítulo 1

Introdução



Objetivos do capítulo

- Entender a atividade de programação
- Aprender sobre arquitetura e computadores
- Aprender sobre código de máquina e linguagens de programação de alto nível
- Familiarizar-se com seu ambiente de computação e seu compilador
- Compilar e executar seu primeiro programa em Java
- Reconhecer erros de sintaxe e de lógica

O objetivo deste capítulo é familiarizá-lo com o conceito de programação. Ele revisa a arquitetura de um computador e discute a diferença entre código de máquina e as linguagens de programação de alto nível. Por fim, você verá como compilar e executar seu primeiro programa em Java e como diagnosticar erros que podem ocorrer quando um programa é compilado ou executado.

Conteúdo do Capítulo

1.1 O que é programação?

Provavelmente você já utilizou um computador para trabalho ou lazer. Muitas pessoas utilizam computadores para tarefas cotidianas como consultar o saldo bancário ou escrever um trabalho escolar. Computadores são bons para essas tarefas. Eles podem tratar tarefas repetitivas, como somar números ou inserir palavras em uma página, sem ficar entediados ou exaustos. Os computadores também são bons para jogos porque podem reproduzir seqüências de sons e imagens, envolvendo o usuário humano no processo.

A flexibilidade de um computador é um fenômeno bem surpreendente. A mesma máquina pode ser usada para consultar um saldo bancário, imprimir um trabalho escolar e jogar um jogo. Em comparação, outros tipos de máquinas executam uma variedade bem menor de tarefas – um carro anda e uma torradeira torra.

Um computador precisa ser programado para realizar tarefas. Diferentes tarefas requerem diferentes programas.

Um programa de computador executa uma seqüência de operações muito básicas em rápida sucessão.

Para alcançar essa flexibilidade, o computador deve ser *programado* para realizar cada tarefa. O computador é uma máquina que armazena dados (números, palavras, imagens), interage com dispositivos (monitor, sistema de som, impressora) e executa programas. Programas são seqüências de instruções e decisões que o computador executa para realizar uma tarefa. Um programa calcula o saldo bancário; um outro, talvez projetado e construído por uma empresa diferente, processa texto; e um terceiro programa, provavelmente de outra empresa, executa um jogo.

Os programas dos computadores atuais são tão sofisticados que é difícil acreditar que são todos compostos a partir de operações extremamente primitivas.

Uma operação típica pode ser uma das listadas abaixo:

- Colocar um ponto vermelho nesta posição da tela.
- Enviar a letra A para a impressora.
- Obter um número a partir desta posição de memória.
- Somar dois números.
- Se este valor for negativo, continuar o programa a partir desta instrução.

Um programa de computador contém as seqüências de instruções para todas as tarefas que pode executar.

Um programa informa para um computador, em detalhes, a seqüência dos passos necessários para completar uma tarefa. Um programa contém um número enorme de operações simples e o computador as executa em grande velocidade. O computador não possui inteligência – ele simplesmente executa seqüências de instruções que foram previamente preparadas.

Para utilizar um computador, não é necessário ter conhecimento sobre programação. Quando você escreve um trabalho escolar usando um processador de texto, esse pacote de software foi programado pelo fabricante e está pronto para seu uso. Isso é apenas o esperado – você pode dirigir um carro sem ser mecânico e tostar uma fatia de pão sem ser eletricista.

O principal propósito deste livro é ensinar como projetar e implementar programas de computador. Você aprenderá a formular instruções para todas as tarefas que seus programas precisam executar.

Tenha em mente que programar um jogo de computador sofisticado ou um processador de texto requer uma equipe de muitos programadores altamente qualificados, artistas gráficos e outros profissionais. Seus primeiros esforços de programação serão mais triviais. Os conceitos e as habilidades que você aprenderá neste livro formam uma base importante, mas não espere produzir imediatamente softwares profissionais. Um curso universitário em ciência de computação ou engenharia de software típico leva quatro anos para ser concluído. O objetivo deste livro é ser um curso introdutório em um curso universitário completo.

Muitos alunos acham que há uma grande emoção mesmo nas tarefas mais simples de programação. É uma experiência incrível ver o computador executar de maneira tão precisa e rápida uma tarefa que demandaria horas de trabalho árduo.

Autoverificação da aprendizagem

1. O que é necessário para tocar um CD de músicas em um computador?
2. Por que um CD player é menos flexível que um computador?
3. Um programa de computador pode tomar a iniciativa de executar tarefas de uma maneira melhor do que seus programadores imaginaram?

1.2 A anatomia de um computador

Para entender o processo de programação, você precisa ter pelo menos um entendimento rudimentar dos blocos de construção que compõem um computador. Esta seção descreverá um computador pessoal. Computadores maiores têm componentes maiores, mais poderosos ou mais rápidos, mas fundamentalmente todos possuem as mesmas partes.

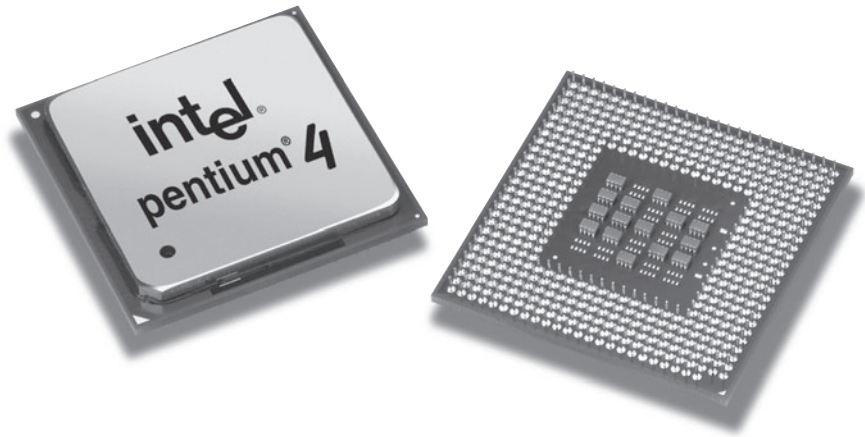


Figura 1 Unidade central de processamento.

No coração do computador está a CPU (unidade central de processamento).

No coração do computador está a *unidade central de processamento* – UCP (*central processing unit* – CPU*) (veja Figura 1). Ela consiste em um único *chip* (circuito integrado) ou em um pequeno número de chips. Um chip de computador é um componente com uma embalagem plástica ou metálica, com conectores de metal e fiação interna feita principalmente de silício. Em um chip de CPU, a fiação interna é muito complicada. Por exemplo, o chip do Pentium 4 (uma CPU popular em computadores pessoais no momento em que escrevíamos este livro) contém mais de 50 milhões de elementos estruturais chamados *transistores* – elementos que permitem que sinais elétricos controlem outros sinais elétricos, tornando possível a computação automática. A CPU localiza e executa instruções de programa; executa operações aritméticas como adição, subtração, multiplicação e divisão; busca dados nas unidades de armazenamento e dispositivos de entrada/saída e envia esses dados de volta.

Dados e programas são mantidos na unidade de armazenamento primária (memória) e nas unidades de armazenamento secundárias (por exemplo, um disco rígido).

O computador mantém os dados e os programas em unidades de *armazenamento*. Há dois tipos de armazenamento. O *armazenamento primário*, também chamado *memória de acesso aleatório* (*random-access memory* – RAM) ou simplesmente *memória*, é rápido, porém, caro e consiste em chips de memória (veja Figura 2). O armazenamento primário tem duas desvantagens: é comparativamente caro e perde todos os dados quando o computador é desligado. O *armazenamento secundário*, normalmente um *disco rígido* (veja Figura 3), oferece um armazenamento mais barato que persiste sem eletricidade. Um disco rígido consiste em lâminas giratórias revestidas por um material magnético e em cabeças de leitura/gravação que podem detectar e alterar os padrões de variação no fluxo magnético sobre as lâminas. Basicamente, é o mesmo processo de gravação e reprodução utilizado em fitas de áudio ou vídeo.

* N. de R.T.: Adotaremos o acrônimo em inglês, CPU, por ser bastante popular no Brasil.

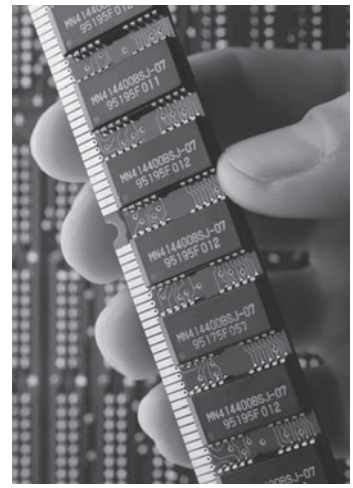


Figura 2 Módulo de memória com chips de memória.

Alguns computadores são unidades autocontidas, enquanto outros são interconectados em *redes*. Na maioria das vezes, os computadores domésticos permanecem intermitentemente conectados à Internet via conexão discada ou de banda larga. Computadores em um laboratório de computação provavelmente permanecem conectados a uma rede local. Por meio do cabeamento de rede, o computador pode ler programas armazenados

Figura 3 Disco rígido.

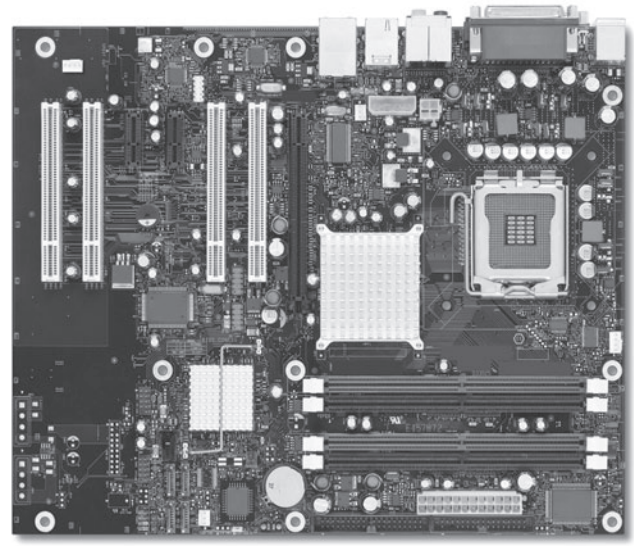


Figura 4 Placa-mãe.

em unidades de armazenamento centralizadas ou enviar dados a outros computadores. Para o usuário de um computador em rede, pode não ser óbvio distinguir quais dados residem no próprio computador e quais são transmitidos pela rede.

A maioria dos computadores tem dispositivos de *armazenamento removíveis* que permitem acessar dados ou programas em mídias como disquetes, fitas ou discos compactos (*compact discs* – CDs).

Para interagir com um usuário humano, um computador requer outros dispositivos periféricos. O computador transmite as informações ao usuário através de uma tela de vídeo, alto-falantes e impressoras. O usuário pode inserir informações e instruções para o computador utilizando um teclado ou um dispositivo de apontamento como um mouse.

A CPU, a RAM e os circuitos eletrônicos que controlam o disco rígido e outros dispositivos são interconectados por um conjunto de linhas elétricas chamado *barramento*. Os dados trafegam através do barramento indo e voltando entre a memória do sistema ou os dispositivos periféricos e a CPU. A Figura 4 mostra uma *placa-mãe*, que contém a CPU, a RAM e conectores para dispositivos periféricos.

A CPU lê as instruções de máquina a partir da memória. As instruções a orientam a se comunicar com a memória, os dispositivos de armazenamento secundário e os periféricos.

A Figura 5 fornece uma visão esquemática geral da arquitetura de um computador. As instruções e os dados de um programa (como texto, números, áudio ou vídeo) são armazenados no disco rígido, em um CD ou estão disponíveis pela rede. Quando um programa é iniciado, ele é carregado na memória onde pode ser lido pela CPU. A CPU lê o programa uma instrução por vez. Guiada por essas instruções, a CPU lê os dados, modifica-os e grava-os de volta na RAM ou no dispositivo de armazenamento secundário. Algumas instruções de programa farão a CPU inte-

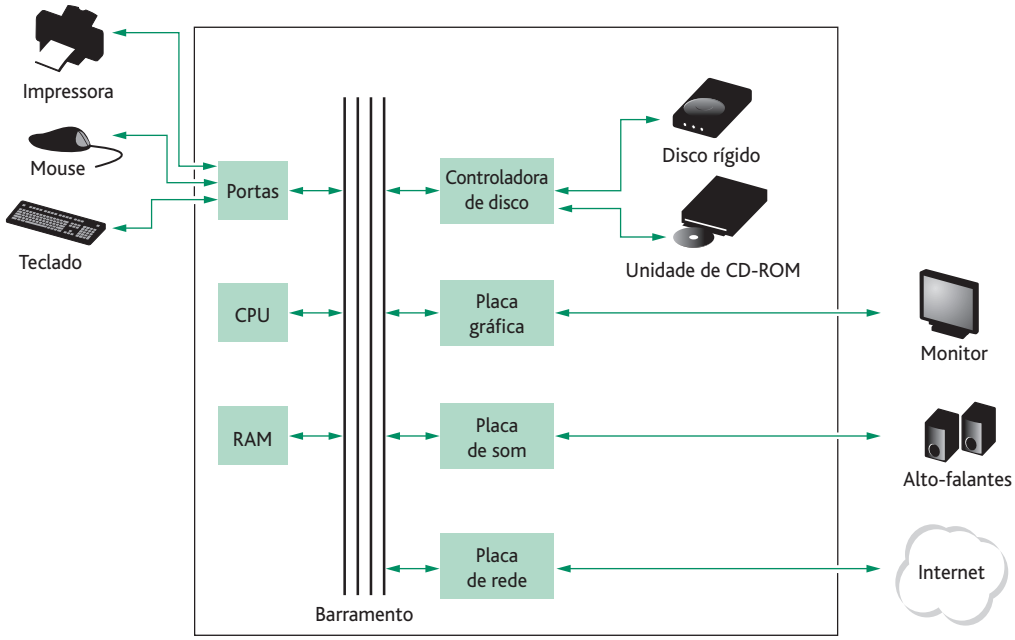


Figura 5 Diagrama esquemático de um computador.

ragir com os dispositivos que controlam a tela de vídeo ou o alto-falante. Como essas ações acontecem muitas vezes e em grande velocidade, o usuário perceberá imagens e som. De maneira semelhante, a CPU pode enviar instruções a uma impressora para marcar o papel com padrões de pontos tão pequenos e próximos que a visão humana reconhece como caracteres de texto e imagens. Algumas instruções de programa lêem a entrada do usuário a partir do teclado ou do mouse. O programa analisa a natureza dessas entradas e então executa as próximas instruções apropriadas.

AUTOVERIFICAÇÃO DA APRENDIZAGEM

4. Onde um programa é armazenado quando não está em execução?
5. Qual parte do computador realiza operações aritméticas, como adição e multiplicação?

FATO ALEATÓRIO 1.1

O ENIAC e a alvorada da computação

O Fato Aleatório 1.1 comenta a história do ENIAC, o primeiro computador eletrônico utilizável. O ENIAC, concluído em 1946, continha aproximadamente 18.000 válvulas a vácuo e ocupava uma sala grande inteira.

1.3 Traduzindo programas legíveis pelo homem em código de máquina

Geralmente, o código de máquina depende do tipo de CPU. Entretanto, o conjunto de instruções da máquina virtual Java (JVM – Java Virtual Machine) pode ser executado em muitas CPUs.

No nível mais básico, instruções de computador são extremamente primitivas. O processador executa *instruções de máquina.* As CPUs de diferentes fornecedores, como o Pentium da Intel ou a SPARC da Sun, têm conjuntos de instruções de máquina diferentes. Para que aplicativos Java possam executar em múltiplas CPUs sem modificação, programas Java contêm instruções de máquina para uma "Máquina Virtual Java" (JVM ou Java Virtual Machine), uma CPU idealizada e que é simulada pela execução de um programa na CPU real. A diferença entre as instruções de máquina virtual e de máquina real não é importante – tudo o que você precisa saber é que instruções de máquina são muito simples; são codificadas como números, armazenadas na memória e podem ser executadas muito rapidamente.
Uma seqüência típica de instruções de máquina é

1. Carregue o conteúdo armazenado na posição 40 da memória.
2. Carregue o valor 100.
3. Se o primeiro valor for maior que o segundo, continue com a instrução que está armazenada na posição 240 da memória.

Na verdade, instruções de máquina são codificadas como números para que possam ser armazenadas na memória. Na máquina virtual Java, essa seqüência de instruções é codificada como a seqüência de números

```
21 40
16 100
163 240
```

Quando a máquina virtual busca essa seqüência de números, ela a decodifica e executa a seqüência de comandos associada.

Uma vez que as instruções de máquina são codificadas como números, é difícil escrever programas em código de máquina.

Como você pode fornecer a seqüência de comando para o computador? O método mais direto é colocar os próprios números na memória do computador. Essa é, na realidade, a maneira como os primeiros computadores funcionavam. Mas um programa longo é composto de milhares de comandos individuais e é entediante e suscetível a erros pesquisar os códigos numéricos para todos os comandos e colocá-los manualmente na memória. Como dissemos anteriormente, computadores são realmente bons para automatizar atividades entediantes e sujeitas a erros, e não demorou muito para que os programadores percebessem que os computadores poderiam ajudar no próprio processo de programação.

Linguagens de alto nível permitem descrever tarefas em um nível conceitual mais alto que o do código de máquina.

Na metade dos anos 1950, começaram a aparecer linguagens de programação *de alto nível*. Nessas linguagens, o programador expressa a idéia por trás da tarefa que precisa ser realizada e um programa de computador especial, chamado *compilador*, traduz a descrição de alto nível em instruções de máquina para um processador particular.

Por exemplo, em Java, a linguagem de programação de alto nível utilizada neste livro, você poderia escrever a seguinte instrução:

```
if (intRate > 100)
   System.out.println("Interest rate error");
```

Isso significa "Se a taxa de juros estiver acima de 100, exiba uma mensagem de erro". É tarefa do programa compilador examinar a seqüência de caracteres `if (intRate > 100)` e traduzi-la em:

```
21 40 16 100 163 240 . . .
```

Um compilador traduz programas escritos em uma linguagem de alto nível para código de máquina.

Compiladores são programas bem sofisticados. Eles traduzem instruções lógicas, como a instrução `if`, em seqüências de cálculos, testes e saltos. Eles atribuem posições de memória a *variáveis* – partes de informação identificadas por nomes simbólicos – como `intRate`. Neste curso, a existência de um compilador estará subentendida na maioria das vezes. Se decidir tornar-se um cientista da computação profissional, você irá aprender mais sobre como escrever programas compiladores mais adiante em seus estudos.

AUTOVERIFICAÇÃO DA APRENDIZAGEM

6. O que é o código para a instrução da máquina virtual Java "Carregue o conteúdo armazenado na posição 100 da memória"?
7. Uma pessoa que usa um computador para trabalho administrativo precisará executar um compilador?

1.4 A linguagem de programação Java

Java foi originalmente projetado para a programação de dispositivos de consumo popular, mas inicialmente foi utilizado com sucesso para escrever *applets* para Internet.

Em 1991, um grupo liderado por James Gosling e Patrick Naughton, da Sun Microsystems, projetou uma linguagem de programação denominada "Green" para uso em dispositivos de consumo popular, como "set-top boxes" de televisores inteligentes. Essa linguagem foi projetada para ser simples e de arquitetura neutra para que pudesse ser executada em vários tipos de hardware. Não havia clientes para essa tecnologia.

Gosling relembra que, em 1994, a equipe percebeu que "Poderíamos escrever um navegador realmente legal. Era uma das poucas coisas na tendência dominante cliente/servidor que precisava de algumas das esquisitisses que fizemos: independência de arquitetura, com processamento em tempo real, confiável e seguro". Java foi apresentado a uma multidão entusiasmada na exposição SunWorld em 1995.

Java foi projetado para ser seguro e portável, beneficiando tanto usuários da Internet como estudantes.

Desde então, Java tem crescido a uma velocidade fenomenal. Programadores escolheram a linguagem porque ela é mais simples do que sua concorrente mais próxima, C++. Além disso, Java tem uma rica *biblioteca* que possibilita escrever programas portáveis capazes de ignorar os sistemas operacionais proprietários – um recurso avidamente perseguido por aqueles que queriam independência em relação a esses sistemas e que foi ferozmente combatido pelos fornecedores. Uma "micro edição" e uma "edição corporativa" da biblioteca Java fez os programadores Java se sentirem em casa

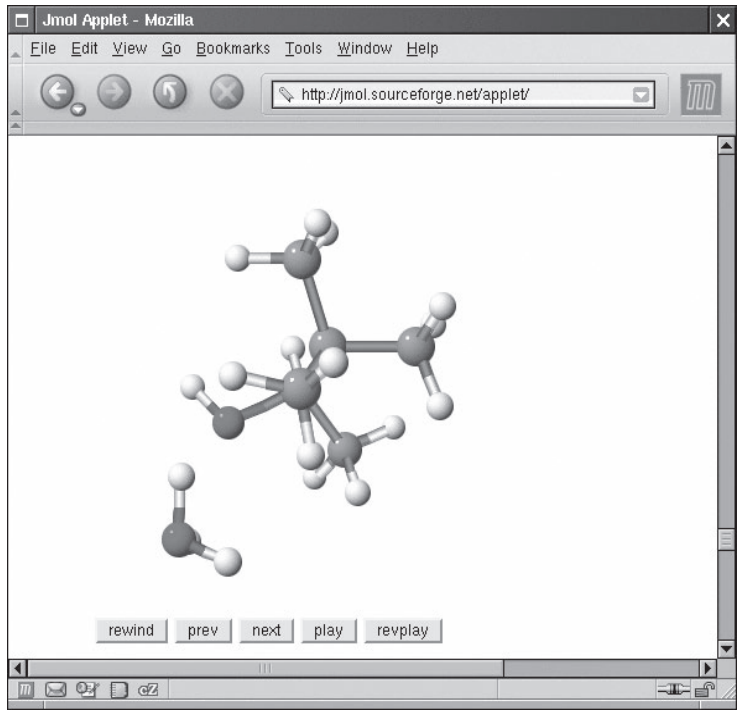


Figura 6 Applet para visualização de moléculas ([1]).

quanto a diferentes tipos de hardware, desde cartões inteligentes e telefones celulares até os maiores servidores de Internet.

Como a linguagem Java foi projetada para a Internet, ela tem dois atributos que a tornam bem adequada para iniciantes: segurança e portabilidade. Se visitar uma página Web que contém código Java (as chamadas *applets* – veja um exemplo na Figura 6), o código começa a executar automaticamente. É importante salientar que você pode confiar que applets são inerentemente seguras. Se uma applet pudesse fazer algo nocivo, como danificar dados ou ler informações pessoais no seu computador, você estaria sob perigo real toda vez que navegasse pela Web – um projetista inescrupuloso poderia disponibilizar uma página Web contendo código perigoso que seria executado na sua máquina assim que você visitasse a página. A linguagem Java tem vários recursos de segurança que garantem que nenhuma applet nociva possa ser executada no seu computador. Como um benefício extra, esses recursos também o ajudam a aprender a linguagem mais rapidamente. A máquina virtual Java pode capturar muitos tipos de erro de iniciantes e informá-los de uma maneira precisa. (Em comparação, muitos erros de iniciantes na linguagem C++ produzem programas que agem de maneira aleatória e confusa.) Outro benefício da linguagem Java é a portabilidade. O mesmo programa Java executará, sem modificação, em Windows, UNIX, Linux ou Macintosh. A portabilidade também é um requisito para applets. Quando você visita uma página Web, o servidor Web que disponibiliza o conteúdo da página não faz idéia do computador que você está utilizando para navegar pela Web. Ele simplesmente retorna o código portável que foi gerado pelo compilador Java. A máquina virtual no seu computador é quem

executa esse código portável. Novamente, há um benefício para o estudante. Você não precisa aprender a escrever programas para diferentes sistemas operacionais.

Atualmente, Java ocupa uma posição sólida como uma das linguagens mais importantes para programação de uso geral e também para ensino de ciência da computação. Mas, embora Java seja uma boa linguagem para iniciantes, ela não é perfeita por três razões.

Como Java não foi especificamente projetado para estudantes, nenhum esforço foi feito para torná-lo realmente simples para escrever programas básicos. Java exige certos requisitos de hardware para escrever até mesmo os programas mais simples. Isso não é um problema para programadores profissionais, mas é uma desvantagem para estudantes iniciantes. À medida que você aprende a programar em Java, haverá momentos em que você terá de se satisfazer com uma explicação preliminar e esperar os detalhes completos em um capítulo posterior.

A linguagem Java foi revisada e estendida várias vezes durante sua vida – veja a Tabela 1. Neste livro, supomos que você instalou a versão Java 5 ou superior.

Java tem uma biblioteca bastante extensa. Focalize o aprendizado naquelas partes da biblioteca que você precisa para seus projetos de programação.

Por fim, não espere aprender tudo sobre Java em um semestre. A linguagem Java em si é relativamente simples, mas contém um amplo conjunto de *pacotes de biblioteca* que são necessários para escrever programas úteis. Há pacotes para gráficos, projetos de interfaces com o usuário, criptografia, rede, som, armazenamento em bancos de dados e muitos outros propósitos. Mesmo programadores Java especialistas não podem esperar conhecer o conteúdo de todos os pacotes – eles utilizam apenas os pacotes necessários para projetos específicos.

Com este livro, você deve esperar aprender boa parte da linguagem Java e dos pacotes mais importantes. Tenha em mente que o objetivo central deste livro não é fazê-lo memorizar as minúcias do Java, mas ensiná-lo como pensar sobre programação.

Tabela 1 Versões de Java

Versão	Ano	Novos recursos importantes
1.0	1996	
1.1	1997	Classes internas
1.2	1998	Swing, coleções
1.3	2000	Melhorias de desempenho
1.4	2002	Assertivas, XML
5	2004	Classes genéricas, laço `for` melhorado, auto-empacotamento, enumerações
6	2006	Melhorias nas bibliotecas

AUTOVERIFICAÇÃO DA APRENDIZAGEM

8. Quais são os dois benefícios mais importantes da linguagem Java?
9. Quanto tempo leva para aprender toda a biblioteca Java?

1.5 Conhecendo seu computador

Reserve algum tempo para se familiarizar com o sistema computacional e o compilador Java que você utilizará para seus trabalhos escolares.

Talvez você esteja cursando sua primeira disciplina de programação ao ler este livro ou talvez esteja trabalhando em um sistema computacional com o qual esteja pouco familiarizado. Invista algum tempo conhecendo o computador. Como os sistemas computacionais variam muito, este livro só pode fornecer um esboço dos passos que você precisa seguir. Utilizar um sistema computacional novo e pouco conhecido pode ser frustrante, especialmente se você estiver sozinho. Procure treinamentos em sua universidade ou peça para um amigo lhe ensinar os princípios básicos do sistema.

Passo 1. Login

Se estiver utilizando seu computador pessoal, provavelmente não precisará se preocupar com esse passo. Mas computadores em um laboratório normalmente não estão abertos para todo mundo. Talvez você precise de um nome ou número de conta e uma senha para ter acesso a esse sistema.

Passo 2. Localize o compilador Java

Sistemas computacionais diferem muito quanto a isto. Em alguns sistemas você precisa abrir uma *janela de shell* (veja Figura 7) e digitar comandos para carregar o compilador. Outros sistemas têm um *ambiente de desenvolvimento integrado* no qual você pode escrever e testar seus programas (veja Figura 8). Muitos laboratórios de universidades possuem material de informação e tutoriais com orientações sobre as ferramentas instaladas no laboratório. Há instruções para vários compiladores populares disponíveis no WileyPLUS (recurso da editora digital).

Passo 3. Entenda arquivos e pastas

Como um programador, você irá escrever, testar e aprimorar programas Java. Seus programas serão mantidos em *arquivos*. Um arquivo é uma coleção de informações reunidas, por exemplo, o texto de um documento redigido com um processador de texto ou as instruções de um programa Java. Arquivos têm nomes, e as regras para nomes válidos diferem entre os sistemas. Alguns sistemas permitem espaços nos nomes dos arquivos; outros não. Alguns fazem distinção entre letras maiúsculas e minúsculas; outros não. A maioria dos compiladores Java exige que arquivos Java terminem em uma *extensão* –`.java`; por exemplo, `Test.java`. Nomes de arquivos Java não podem conter espaços e a distinção entre letras maiúsculas e minúsculas é importante.

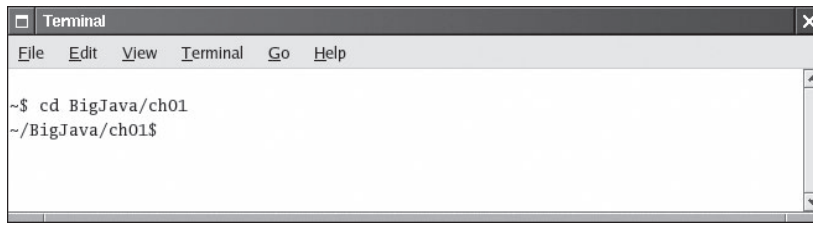


Figura 7 Uma janela de shell.

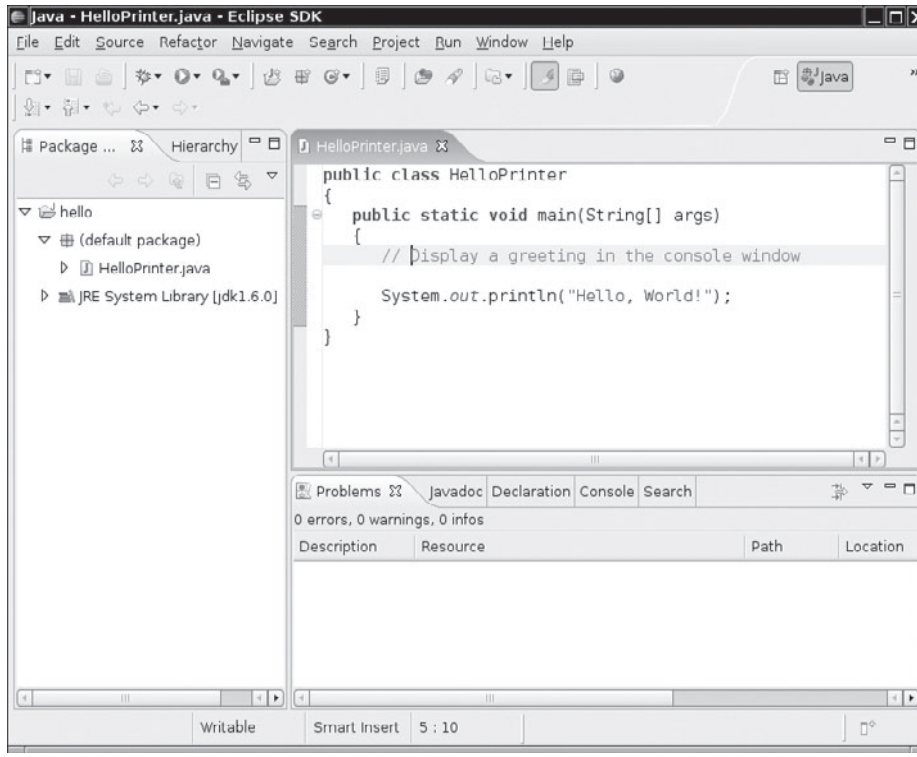


Figura 8 Um ambiente de desenvolvimento integrado.

Os arquivos são armazenados em *pastas* ou *diretórios*. Esses contêineres de arquivos podem estar *aninhados*. Isto é, uma pasta pode conter não apenas arquivos como também outras pastas, que podem conter mais arquivos e pastas (veja Figura 9). Essa hierarquia

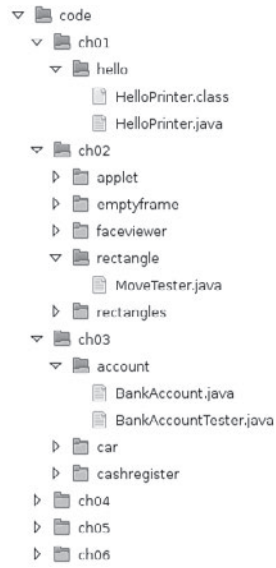


Figura 9
Pastas aninhadas.

pode ser bem ampla, especialmente nos computadores em rede, nos quais alguns arquivos podem estar no seu disco local e outros em algum outro lugar na rede. Embora não precise se preocupar com cada ramificação da hierarquia, você deve se familiarizar com o seu ambiente local. Diferentes sistemas têm diferentes maneiras de exibir arquivos e diretórios. Alguns usam uma exibição gráfica e permitem que você se movimente clicando com o mouse nos ícones de pasta. Em outros sistemas, você precisa inserir comandos para acessar ou inspecionar locais diferentes.

Passo 4. Escreva um programa simples

Na próxima seção, introduziremos um programa bem simples. Você precisará aprender a digitá-lo, executá-lo e corrigir enganos.

Passo 5. Salve seu trabalho

Desenvolva uma estratégia para manter cópias de backup do seu trabalho antes que desastres aconteçam.

Você passará várias horas digitando e melhorando o código de programas Java. Os arquivos de programa resultantes têm valor e você deve tratá-los como trataria outros bens importantes. Uma estratégia conscienciosa de segurança é especialmente importante para os arquivos de computador. Eles são mais frágeis do que documentos em papel ou outros objetos mais tangíveis. É fácil excluir um arquivo acidentalmente e às vezes perdemos arquivos por causa de um mau funcionamento do computador. Só se houver uma cópia, você não precisará redigitar o conteúdo. Como talvez você não consiga lembrar de todo o arquivo, provavelmente gastará quase tanto tempo quanto gastou para digitá-lo e aprimorá-lo. Isso custa tempo e pode fazer com que você perca prazos finais de entrega. Portanto, é crucial aprender a salvaguardar os arquivos e a ter o hábito de fazer isso *antes* de o desastre acontecer. Você pode fazer cópias de segurança, ou *backups*, dos arquivos salvando cópias em um disquete ou CD, em uma outra pasta, na sua rede local ou na Internet.

Autoverificação da aprendizagem

10. Como os projetos de programação são armazenados em um computador?
11. O que fazer para se proteger contra a perda de dados quando você trabalha nos projetos de programação?

Dica de produtividade 1.1

Entenda o sistema de arquivos

Nos últimos anos, o uso de computadores ficou mais fácil para usuários domésticos ou administrativos. Muitos detalhes não-essenciais estão agora escondidos dos usuários casuais. Por exemplo, vários usuários casuais simplesmente armazenam todo o trabalho dentro de uma pasta padrão (como "Home" ou "Meus Documentos") e ignoram os detalhes do sistema de arquivos.

Mas você precisa saber impor uma organização para os dados que cria. Você também precisa ser capaz de localizar e inspecionar os arquivos necessários para traduzir e executar programas Java.

Se você não se sentir confortável com arquivos e pastas, certifique-se de investir algum tempo para aprender esses conceitos. Matricule-se em um curso de curta duração ou faça um tutorial na Web. Há muitos tutoriais gratuitos disponíveis na Internet, mas infelizmente suas localizações mudam com freqüência. Pesquise na Web "tutoriais para arquivos e pastas" e selecione um que vá além dos princípios básicos.

DICA DE PRODUTIVIDADE 1.2

Tenha uma estratégia de backup

A Dica de Produtividade 1.2 discute estratégias para fazer o backup do seu trabalho de programação a fim de assegurar que os dados não sejam comprometidos caso o computador sofra alguma avaria.

1.6 Compilando um programa simples

Agora você está pronto para escrever e executar seu primeiro programa Java. A escolha tradicional do primeiro programa em uma nova linguagem de programação é a de um que exibe uma saudação simples: "Hello, World!". Vamos seguir essa tradição. Eis o programa "Hello, World!" em Java.

ch01/hello/HelloPrinter.java

```
1  public class HelloPrinter
2  {
3     public static void main(String[] args)
4     {
5        // Exibe uma saudação na janela da console
6
7        System.out.println("Hello, World!");
8     }
9  }
```

Saída

```
Hello, World!
```

Java diferencia maiúsculas de minúsculas. Você precisa tomar cuidado e distingui-las corretamente.

Examinaremos esse programa logo a seguir. Por enquanto, você dever criar um novo arquivo de programa e chamá-lo `HelloPrinter.java`. Insira as instruções do programa, compile e execute-o, seguindo o procedimento apropriado para seu compilador.

Java diferencia letras maiúsculas de minúsculas. Você deve inserir letras maiúsculas e minúsculas exatamente como elas aparecem na listagem do programa. Você não pode digitar `MAIN` ou `PrintLn`. Se não for cuidadoso, terá problemas – veja Erro Comum 1.2.

Por outro lado, a organização de Java tem *formato livre*. Você pode utilizar espaços e quebras de linha para separar as palavras e pode colocar tantas palavras quanto possível em uma linha:

```
public class HelloPrinter{public static void main(String[]
args){// Exibe uma saudação na janela de console
System.out.println("Hello, World!");}}
```

Você também pode escrever cada palavra e cada símbolo em uma linha separada:

```
public
class
HelloPrinter
{
public
static
void
main
(
. . .
```

Organize seus programas de modo que possam ser lidos facilmente.

Mas um bom estilo determina que você deve organizar seus programas de uma maneira legível. Forneceremos recomendações para uma boa organização ao longo de todo o livro. O Apêndice A contém um resumo das nossas recomendações.

Quando você executar o programa de teste, a mensagem:

```
Hello, World!
```

aparecerá em algum lugar na tela (veja Figuras 10 e 11). A localização exata depende do seu ambiente de programação.

Agora que vimos o programa em funcionamento, é hora de entender sua constituição. A primeira linha:

```
public class HelloPrinter
```

Classes são os blocos de construção fundamentais dos programas Java.

inicia uma nova *classe*. Classes são um conceito fundamental em Java e você começará a estudá-las no Capítulo 2. Em Java, cada programa consiste em uma ou mais classes.

A palavra-chave `public` indica que a classe pode ser utilizada pelo "público". Mais tarde você encontrará recursos `private`. Nesse ponto, você simplesmente deve considerar:

```
public class NomeDaClasse
{
   . . .
}
```

como uma parte necessária da estrutura exigida para escrever qualquer programa Java. Em Java, cada arquivo-fonte pode conter no máximo uma classe pública, e o nome dessa classe pública deve corresponder ao nome do arquivo que a contém. Por exemplo, a classe `HelloPrinter` *precisa* estar contida em um arquivo `HelloPrinter.java`. É muito importante que os nomes e o *uso de letras maiúsculas e minúsculas* correspondam exatamente. Você pode receber mensagens de erro estranhas se atribuir à classe um nome `HELLOPrinter` ou `helloprinter.java` ao arquivo.

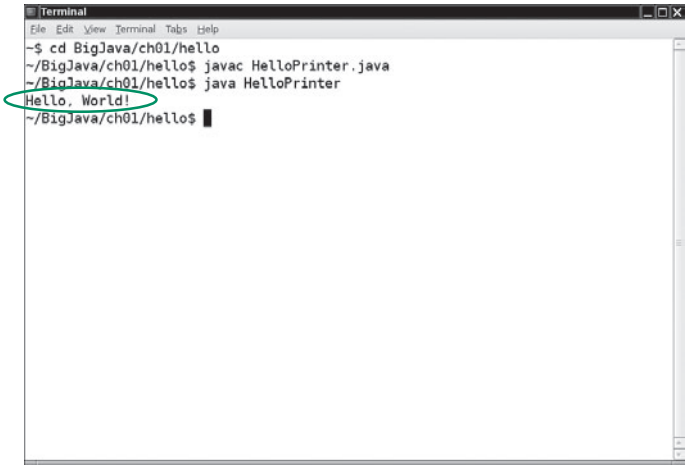


Figura 10 Executando o programa `HelloPrinter` em uma janela da console.

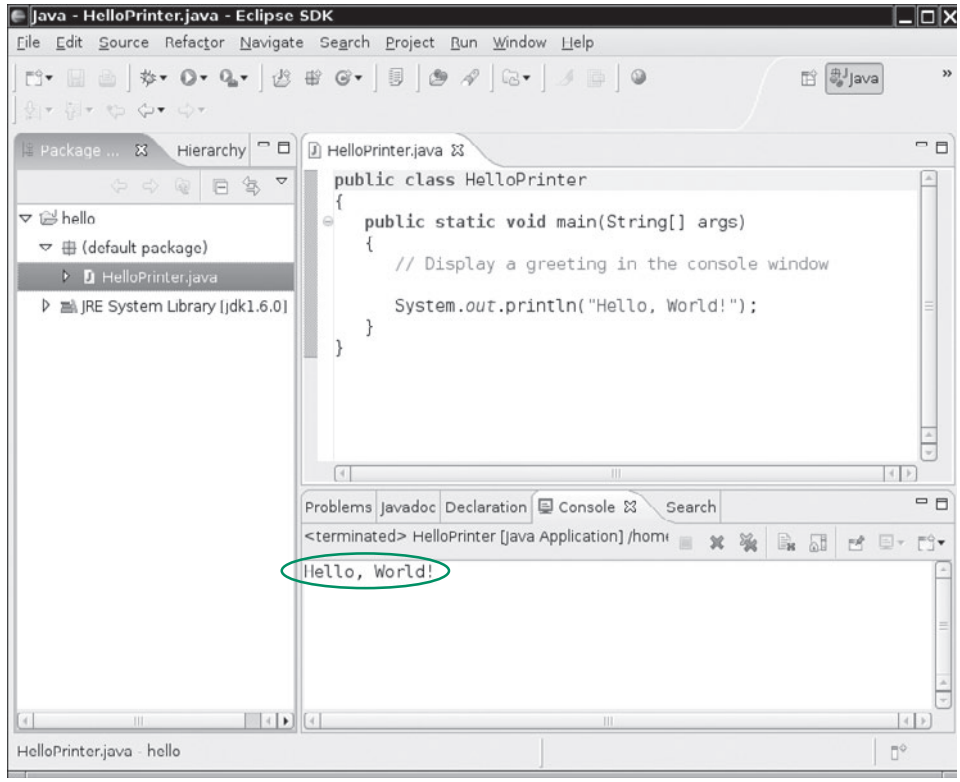


Figura 11
Executando o programa `HelloPrinter` em um ambiente de desenvolvimento integrado.

Toda aplicação Java contém uma classe com um método `main`. Quando a aplicação inicia, as instruções no método `main` são executadas.

A construção

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

define um *método* chamado `main`. Um método contém um conjunto de instruções de programação que descreve como executar uma tarefa particular. Toda aplicação Java deve ter um método `main`. A maioria dos programas Java contém outros métodos além do `main`, e você verá no Capítulo 3 como escrever outros métodos.

Cada classe contém definições de métodos. Cada método contém uma seqüência de instruções.

O *parâmetro* `String[] args` é uma parte obrigatória do método `main`. (Ele contém os *argumentos da linha de comando*, que só discutiremos no Capítulo 11.) A palavra-chave `static` indica que o método `main` não opera sobre um *objeto*. (Como veremos no Capítulo 2, a maioria dos métodos em Java opera sobre objetos, e métodos `static` não são comuns em grandes programas Java. Contudo, `main` sempre dever ser `static`, porque ele começa a executar antes de o programa criar objetos.)

Nesse momento, simplesmente considere

```
public class NomeDaClasse
{
   public static void main(String[] args)
   {
      . . .
   }
}
```

como outra parte do código básico necessário. Nosso primeiro programa tem todas as suas instruções dentro do método `main` de uma classe.

A primeira linha dentro do método `main` é um *comentário*

```
// Exibe uma saudação na janela da console
```

Use comentários para ajudar as pessoas a entender seu programa.

Esse comentário é simplesmente para o benefício de quem está lendo o programa e destina-se a explicar em mais detalhes o que a próxima instrução faz. Qualquer texto incluído entre `//` e o fim da linha é completamente ignorado pelo compilador. Os comentários são utilizados para explicar o programa a outros programadores ou para você mesmo.

As instruções ou os *comandos* no *corpo* do método `main` – isto é, as instruções dentro das chaves (`{}`) – são executadas uma a uma. Cada instrução termina em um ponto-e-vírgula (`;`). Nosso método tem uma única instrução:

```
System.out.println("Hello, World!");
```

Essa instrução imprime uma linha de texto, a saber, “Hello, World!”. Mas há vários locais para os quais um programa pode enviar essa string: uma janela, um arquivo ou um computador em rede no outro lado do mundo. Você precisa especificar que o destino para a string é a *saída padrão do sistema* – isto é, uma janela de console. A janela de console é representada em Java por um objeto chamado `out`. Assim como você precisou colocar o método `main` em uma classe `HelloPrinter`, os projetistas da biblioteca Java precisaram colocar o objeto `out` em uma classe. Eles o colocaram na classe `System`, que contém objetos e métodos úteis para acessar recursos do sistema. Para utilizar o objeto `out` na classe `System`, você deve referenciá-lo como `System.out`.

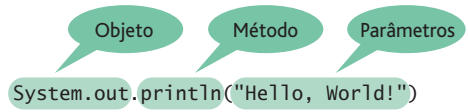


Figura 12 Chamando um método.

Para utilizar um objeto, como `System.out`, especifique o que você quer fazer com ele. Nesse caso, você quer imprimir uma linha de texto. O método `println` executa essa tarefa.

Você não é obrigado a implementar esse método – os programadores que escreveram a biblioteca Java já fizeram isso para nós – mas você precisa *chamar* o método.

Chama-se um método especificando um objeto, o nome do método e os seus parâmetros.

Sempre que você chama um método em Java, você precisa especificar três itens (veja Figura 12):

1. O objeto que você quer utilizar (nesse caso, `System.out`).
2. O nome do método que você quer utilizar (nesse caso, `println`).
3. Um par de parênteses, contendo qualquer outra informação que o método precise (nesse caso, `"Hello, World!"`). O termo técnico para essa informação é um *parâmetro* para o método. Observe que os dois pontos em `System.out.println` têm significados diferentes. O primeiro ponto significa "localize o objeto `out` na classe `System`". O segundo ponto significa "aplique o método `println` a esse objeto".

Uma seqüência de caracteres incluída entre aspas:

```
"Hello, World!"
```

Uma string é uma seqüência de caracteres incluídos entre aspas.

é chamada de *string*. Você deve incluir o conteúdo da string dentro de aspas para que o compilador saiba que você literalmente quer dizer `"Hello, World!"`. Há uma razão para essa exigência. Suponha que você precise imprimir a palavra *main*. Incluindo-a entre aspas, `"main"`, o compilador sabe que você tem em mente a seqüência de caracteres `m a i n`, não o método chamado `main`. A regra é simplesmente incluir todas as strings de texto entre aspas para que o compilador considere-as como texto simples e não tente interpretá-las como instruções de programa.

Você também pode imprimir valores numéricos. Por exemplo, a instrução

```
System.out.println(3 + 4);
```

exibe o número 7.

O método `println` imprime uma string ou um número e então inicia uma nova linha. Por exemplo, a seqüência de instruções

```
System.out.println("Hello");
System.out.println("World!");
```

imprime duas linhas de texto:

```
Hello
World!
```

Há um segundo método, chamado `print`, que você pode utilizar para imprimir um item sem iniciar uma nova linha. Por exemplo, a saída das duas instruções

```
System.out.print("00");
System.out.println(3 + 4);
```

é a única linha

```
007
```

Sintaxe 1.1 Chamada de método

objeto.nomeDoMétodo(*parâmetros*)

Exemplo:

```
System.out.println("Hello, Dave!")
```

Objetivo:

Invocar um método em um objeto e fornecer qualquer parâmetro adicional

Autoverificação da aprendizagem

12. Como você modificaria o programa `HelloPrinter` para imprimir as palavras "Hello" e "World!" em duas linhas?
13. O programa continuaria a funcionar se você omitisse a linha que inicia com `//`?
14. O que o conjunto de instruções a seguir imprime?

```
System.out.print("My lucky number is");
System.out.println(3 + 4 + 5);
```

Erro comum 1.1

Omitindo ponto-e-vírgulas

Em Java, cada instrução deve terminar em um ponto-e-vírgula. Esquecer de digitar um ponto-e-vírgula é um erro comum. Isso confunde o compilador, porque ele usa o ponto-e-vírgula para descobrir onde uma instrução termina e a próxima inicia. O compilador não usa quebras de linha nem fechamento de chaves para reconhecer o fim de uma instrução. Por exemplo, o compilador considera

```
System.out.println("Hello")
System.out.println("World!");
```

uma única instrução, como se você tivesse escrito:

```
System.out.println("Hello") System.out.println("World!");
```

Assim, o compilador não entende essa instrução, porque ele não esperava a palavra `System` após o parêntese de fechamento depois de `"Hello"`. A solução é simples. Verifique todas as instruções para ver se terminam com ponto-e-vírgula, da mesma maneira como você verificaria um ponto final em uma frase qualquer.