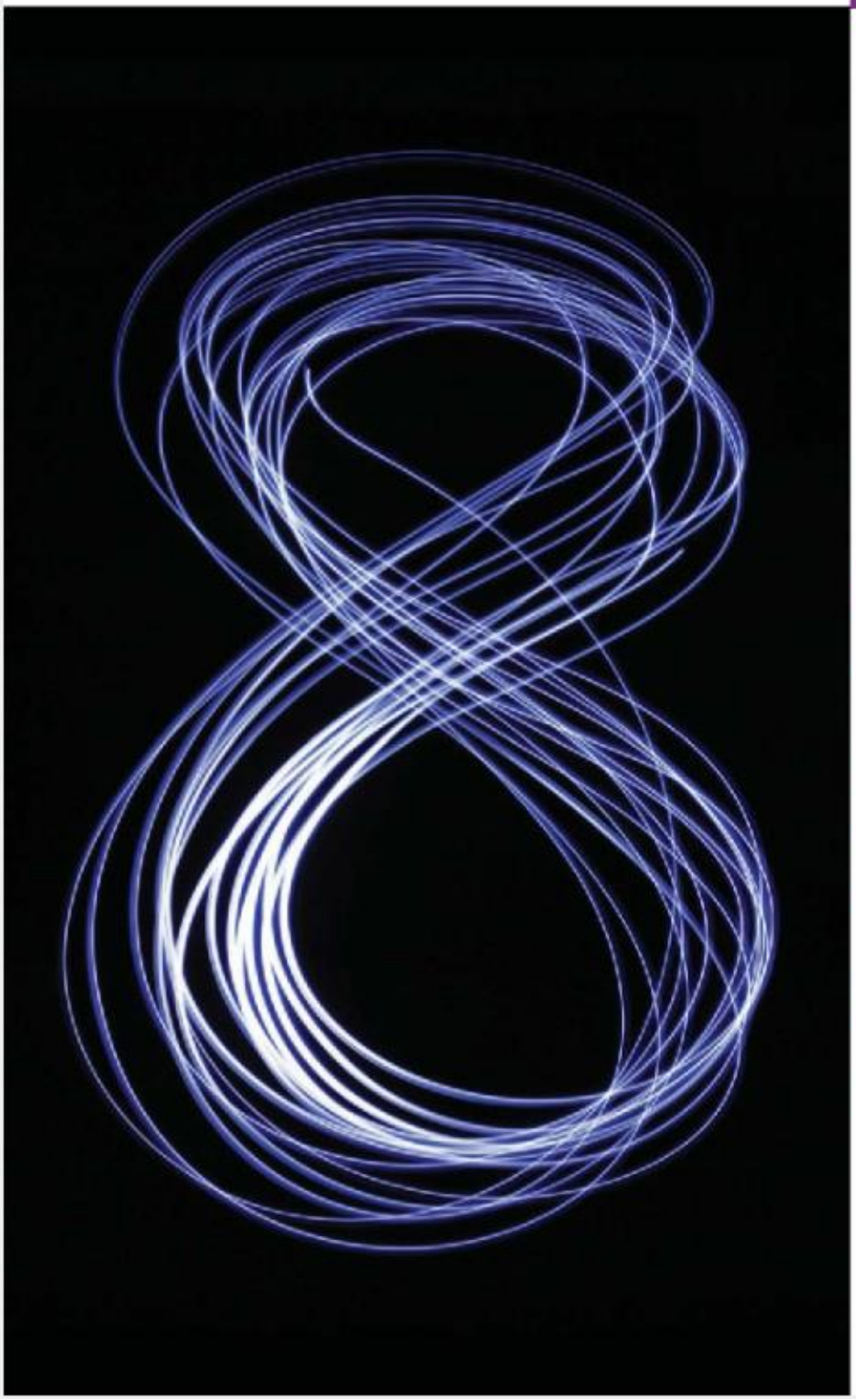
Introdução a computadores, internet e Java

1



O homem ainda é o computador mais extraordinário.
— John F. Kennedy

Bom design é bom negócio.
— Thomas J. Watson, fundador da IBM

Objetivos

Neste capítulo, você irá:

- Conhecer os empolgantes desenvolvimentos no campo da informática.
- Aprender os conceitos básicos de hardware, software e redes de computadores.
- Entender a hierarquia de dados.
- Entender os diferentes tipos de linguagem de programação.
- Entender a importância do Java e de outros tipos de linguagem de programação.
- Entender programação orientada a objetos.
- Aprender a importância da internet e da web.
- Conhecer um ambiente de desenvolvimento de programa Java típico.
- Fazer test-drive de aplicativos Java.
- Observar algumas das principais tecnologias de software recentes.
- Ver como se manter atualizado com as tecnologias da informação.

Sumário

1.1 Introdução

Bem-vindo ao Java, uma das linguagens de programação mais utilizadas no mundo. Você já conhece as tarefas poderosas que os computadores executam. Usando este manual, você escreverá instruções que fazem com que os computadores realizem essas tarefas. O **software** (isto é, as instruções que você escreve) controla o **hardware** (isto é, os computadores).

Você aprenderá a *programação orientada a objetos* — atualmente a metodologia-chave de programação. Você vai criar e trabalhar com muitos *objetos de software*.

Para muitas organizações, a linguagem preferida a fim de atender às necessidades de programação corporativa é o Java. Ele também é amplamente utilizado para implementar aplicativos e softwares baseados na internet para dispositivos que se comunicam por uma rede.

A Forrester Research prevê que mais de dois bilhões de PCs estarão em uso até 2015.[1] De acordo com a Oracle, 97% dos desktops corporativos, 89% dos desktops PC, 3 bilhões de dispositivos (Figura 1.1) e 100% de todos os players Blu-ray Disc™ executam o Java, e há mais de 9 milhões de desenvolvedores Java.[2]

De acordo com um estudo realizado pela Gartner, os dispositivos móveis continuarão a ultrapassar os PCs como os dispositivos de computação principais dos usuários; estima-se que 1,96 bilhão de smartphones e 388 milhões de tablets serão distribuídos em 2015 — 8,7 vezes o número de PCs.[3] Em 2018, o mercado de aplicativos móveis deverá alcançar US$ 92 bilhões.[4] Isso está criando oportunidades profissionais significativas para pessoas que programam aplicativos móveis, muitos dos quais são programados em Java (veja a Seção 1.6.3).

[1] http://www.worldometers.info/computers.
[2] http://www.oracle.com/technetwork/articles/java/javaone12review-1863742.html.
[3] http://www.gartner.com/newsroom/id/2645115.
[4] https://www.abiresearch.com/press/tablets-will-generate-35-of-this-years-25-billion-.

Dispositivos		
Blu-ray Disc™	Caixas automáticos	Canetas inteligentes
Cartões de crédito	Consoles de jogos	Celulares
Cartões inteligentes	Dispositivos médicos	Copiadoras
Decodificadores de TV (*set-top boxes*)	Estações de pagamento de estacionamento	Desktops (computadores de mesa)
e-Readers	Medidores inteligentes	Impressoras
Eletrodomésticos	Robôs	Interruptores de luz
Imagens por ressonância magnética (MRIs)	Scanners de tomografia computadorizada	Roteadores
Passes de transporte	Sintonizadores de TV a cabo	Sistemas de diagnóstico veicular
Sistemas de aviação	Sistemas de segurança residencial	Sistemas de informação e entretenimento para automóveis
Smartphones	Tablets	Sistemas de navegação GPS
Terminais lotéricos	Termostatos	Televisões

Figura 1.1 | Alguns dispositivos que usam Java.

Java Standard Edition

O Java evoluiu tão rapidamente que esta décima edição do *Java: como programar* — baseada no **Java Standard Edition 7 (Java SE 7)** e no **Java Standard Edition 8 (Java SE 8)** — foi publicada apenas 17 anos após a primeira edição. O Java Standard Edition contém os recursos necessários para desenvolver aplicativos de desktop e servidor. O livro pode ser usado com o Java SE 7 *ou* o Java SE 8 (lançado logo depois que esta obra foi publicada originalmente em inglês). Todos os recursos Java SE 8 serão discutidos em seções modulares, fáceis de incluir ou omitir ao longo da leitura.

Antes do Java SE 8, a linguagem suportava três paradigmas de programação — *programação procedural*, *programação orientada a objetos* e *programação genérica*. O Java SE 8 acrescenta a *programação funcional*. No Capítulo 17, mostraremos como usar a programação funcional para escrever programas de forma mais rápida e concisa, com menos bugs e que são mais fáceis de *paralelizar* (isto é, executar múltiplos cálculos ao mesmo tempo) a fim de tirar proveito das atuais arquiteturas de hardware multiprocessadas com o intuito de melhorar o desempenho do aplicativo.

Java Enterprise Edition

O Java é utilizado para um espectro de aplicações tão amplo que ele tem duas outras versões. O **Java Enterprise Edition (Java EE)** é adequado para desenvolver aplicativos em rede distribuída e em grande escala e também aplicativos baseados na web. No passado, a maioria dos aplicativos de computador era executada em computadores "independentes" (que não estavam conectados em rede). Já os aplicativos de hoje podem ser escritos para que se comuniquem entre os computadores no mundo pela internet e web. Mais adiante neste livro discutiremos como elaborar esses aplicativos baseados na web com o Java.

Java Micro Edition

O **Java Micro Edition (Java ME)** — um subconjunto do Java SE — é voltado para o desenvolvimento de aplicativos para dispositivos embarcados com recursos limitados, como *smartwatches*, MP3 players, decodificadores de TV (*set-top boxes*), medidores inteligentes (para monitorar o uso de energia elétrica) e muitos outros.

1.2 Hardware e software

Os computadores podem executar cálculos e tomar decisões lógicas incrivelmente mais rápido que os seres humanos. Muitos dos computadores pessoais de hoje em dia podem realizar bilhões de cálculos em um segundo — mais do que um ser humano é capaz durante a vida. *Supercomputadores* já realizam *milhares de trilhões* (*quatrilhões*) de instruções por segundo! O supercomputador Tianhe-2 do departamento de tecnologia de defesa da Universidade Nacional da China pode executar mais de 33 quatrilhões de cálculos por segundo (33,86 *petaflops*)![1] Para colocar isso em perspectiva, *o supercomputador Tianhe-2 pode executar em um segundo cerca de 3 milhões de cálculos para cada pessoa no planeta!* E os "limites máximos" de supercomputação estão aumentando rapidamente.

[1] http://www.top500.org/.

Os computadores processam dados sob o controle de conjuntos de instruções chamados **programas de computador**. Esses programas de software orientam o computador por meio de ações ordenadas especificadas por pessoas chamadas **programadores** de computador. Neste livro, você aprenderá uma metodologia de programação-chave que melhora a produtividade do programador, reduzindo, assim, os custos de desenvolvimento de softwares — a *programação orientada a objetos*.

Um computador é composto por vários dispositivos chamados hardware (por exemplo, teclado, tela, mouse, unidades de disco, memória, unidades de DVD e unidades de processamento). Os custos da computação estão *caindo drasticamente*, por conta dos rápidos avanços nas tecnologias de hardware e software. Os computadores que ocupavam grandes salas e custavam milhões de dólares há décadas agora são gravados em chips de silício menores que uma unha, ao custo de apenas alguns poucos dólares. Ironicamente, o silício é um dos materiais mais abundantes na Terra — é um componente da areia comum. A tecnologia do chip de silício deixou a computação tão econômica que os computadores se tornaram um produto de consumo popular.

1.2.1 Lei de Moore

Todos os anos, você provavelmente espera para pagar pelo menos um pouco mais pela maioria dos produtos e serviços. Aconteceu o oposto no caso das áreas de informática e comunicações, especialmente no que diz respeito ao hardware que suporta essas tecnologias. Por muitas décadas, os custos de hardware caíram rapidamente.

A cada um ou dois anos, as capacidades dos computadores praticamente *dobram*. Essa tendência notável muitas vezes é chamada **lei de Moore**, cujo nome vem da pessoa que a identificou na década de 1960, Gordon Moore, cofundador da Intel, a atual maior fabricante de processadores de computadores e sistemas incorporados. A lei de Moore e observações relacionadas se aplicam especialmente à quantidade de memória que os computadores têm para os programas, a quantidade de armazenamento secundário (como armazenamento em disco) que eles têm a fim de manter programas e dados em relação a períodos mais longos de tempo, e suas velocidades de processador — as velocidades em que eles *executam* os programas (isto é, fazem seu trabalho).

Crescimento semelhante ocorreu na área de comunicações — os custos despencaram bruscamente enquanto a demanda enorme por *largura de banda* das comunicações (isto é, a capacidade de transmitir informações) atraiu concorrência intensa. Não conhecemos nenhuma outra área em que a tecnologia melhora e os custos caem de maneira tão rápida. Essa melhoria fenomenal está estimulando verdadeiramente a *revolução da informação*.

1.2.2 Organização do computador

Independentemente das diferenças na aparência *física*, os computadores podem ser visualizados como divididos em várias **unidades lógicas** ou seções lógicas (Figura 1.2).

Unidade lógica	Descrição
Unidade de entrada	Essa seção de "recebimento" obtém informações (dados e programas de computador) de **dispositivos de entrada** e as coloca à disposição de outras unidades para serem processadas. A maioria das entradas do usuário é inserida nos computadores por meio de teclados, telas sensíveis ao toque e dispositivos do tipo mouse. Outras formas de entrada incluem recebimento de comandos de voz, digitalização de imagens e códigos de barras, leitura de dispositivos de armazenamento secundário (como discos rígidos, unidades de DVD, Blu-ray Disc™ e unidades flash USB — também chamadas de "pen drives" ou "cartões de memória"), recebimento de vídeos de uma webcam e fazer seu computador receber informações da internet (como ao transmitir vídeos a partir do YouTube® ou baixar e-books da Amazon). Novas formas de entrada incluem dados de posição a partir de dispositivo GPS e informações de orientação de um *acelerômetro* (dispositivo que responde à aceleração para cima/baixo, para esquerda/direita e para a frente/trás) em um controlador de smartphone ou jogo (como o Microsoft® Kinect® e Xbox®, Wii™ Remote e Sony® PlayStation® Move).
Unidade de saída	Essa seção de "entrega" pega as informações que o computador processou e as coloca em vários **dispositivos de saída** para torná-las disponíveis a fim de serem utilizadas fora do computador. A maioria das informações que são produzidas pelos atuais computadores é exibida em telas (incluindo as sensíveis ao toque), impressa em papel ("a abordagem verde" desencoraja isso), reproduzida como áudio ou vídeo em PCs e tocadores de mídia (como iPods, da Apple) e telas gigantes em estádios, transmitida pela internet ou usada para controlar outros dispositivos, como robôs e eletrodomésticos "inteligentes". As informações também são comumente produzidas para dispositivos de armazenamento secundário, como discos rígidos, unidades de DVD e unidades de flash USB. Uma forma popular recente de saída é a onda dos smartphones.

continua

continuação

Unidade lógica	Descrição
Unidade de memória	Essa seção de "armazenamento" de acesso rápido e de relativa baixa capacidade retém as informações que foram inseridas por meio da unidade de entrada, tornando-as imediatamente disponíveis para processamento, quando necessário. A unidade de memória também retém informações processadas até que elas possam ser colocadas em dispositivos de saída pela unidade de saída. As informações na unidade de memória são *voláteis* — em geral, são perdidas quando o computador é desligado. A unidade de memória é com frequência chamada **memória**, **memória principal** ou **RAM** (Random Access Memory). A memória principal dos computadores desktop e notebook tem até 128 GB de RAM. GB representa gigabytes; um gigabyte é aproximadamente um bilhão de bytes. Um **byte** tem oito bits. Um bit é um `0` ou um `1`.
Unidade de aritmética e lógica (ALU)	Essa seção de "produção" executa *cálculos* como adição, subtração, multiplicação e divisão. Também contém os mecanismos de *decisão* que permitem ao computador, por exemplo, comparar dois itens da unidade de memória para determinar se são iguais ou não. Nos sistemas atuais, a ALU é implementada como parte da próxima unidade lógica, a CPU.
Unidade de processamento central (CPU)	Essa seção "administrativa"coordena e supervisiona a operação das outras seções. A CPU diz à unidade de entrada quando as informações devem ser lidas e transferidas para a unidade de memória, informa à ALU quando as informações da unidade de memória devem ser utilizadas em cálculos e instrui a unidade de saída sobre quando enviar as informações da unidade de memória para certos dispositivos de saída. Muitos computadores de hoje têm múltiplas CPUs e, portanto, podem realizar muitas operações simultaneamente. Um **processador de múltiplos núcleos** (ou **multi-core**) implementa o multiprocessamento em um único chip de circuito integrado — por exemplo, um *processador de dois núcleos* (ou *dual-core*) tem duas CPUs e um *processador de quatro núcleos* (ou *quad-core*) tem quatro. Computadores desktop de hoje têm processadores que podem executar bilhões de instruções por segundo.
Unidade de armazenamento secundária	Essa é a seção de "armazenamento" de longo prazo e de alta capacidade. Programas ou dados que não são utilizados ativamente pelas outras unidades, em geral, são colocados em dispositivos de armazenamento secundário (por exemplo, *discos rígidos*) até que sejam necessários, talvez horas, dias, meses ou até mesmo anos mais tarde. As informações nos dispositivos de armazenamento secundário são *persistentes* — elas são preservadas mesmo quando a energia do computador é desligada. As informações no armazenamento secundário levam muito mais tempo para serem acessadas do que aquelas na memória principal, mas seu custo por unidade é bem mais baixo. Exemplos de dispositivos de armazenamento secundário incluem discos rígidos, unidades de DVD e unidades de flash USB, algumas das quais podem armazenar mais de 2 TB (TB significa terabytes; um terabyte é aproximadamente um trilhão de bytes). Discos rígidos típicos em computadores desktop e notebook armazenam 2 TB, e alguns desktops podem armazenar até 4 TB.

Figura 1.2 | Unidades lógicas de um computador.

1.3 Hierarquia de dados

Os itens de dados processados pelos computadores formam uma **hierarquia de dados** que torna-se maior e mais complexa em estrutura à medida que passamos dos itens de dados simples (chamados "bits") para aqueles mais ricos, como caracteres e campos. A Figura 1.3 ilustra uma parte da hierarquia de dados.

Bits

O menor item de dados em um computador pode assumir o valor `0` ou o valor `1`. É chamado **bit** (abreviação de "dígito binário" — um dígito que pode assumir um de *dois* valores). Notavelmente, as funções impressionantes desempenhadas pelos computadores envolvem apenas as manipulações mais simples de `0`s e `1`s — *examinar, configurar* e *inverter o valor de um bit* (de `1` para `0` ou de `0` para `1`).

Caracteres

É tedioso para as pessoas trabalhar com dados na forma de baixo nível de bits. Em vez disso, elas preferem trabalhar com *dígitos decimais* (0–9), *letras* (A–Z e a–z) e *símbolos especiais* (por exemplo, $, @, %, &, *, (,), –, +, ", :, ? e /). Dígitos, letras e símbolos especiais são conhecidos como **caracteres**. O **conjunto de caracteres** do computador é o conjunto de todos os caracteres utilizados para escrever programas e representar itens de dados. Computadores processam apenas `1`s e `0`s, assim um conjunto de caracteres

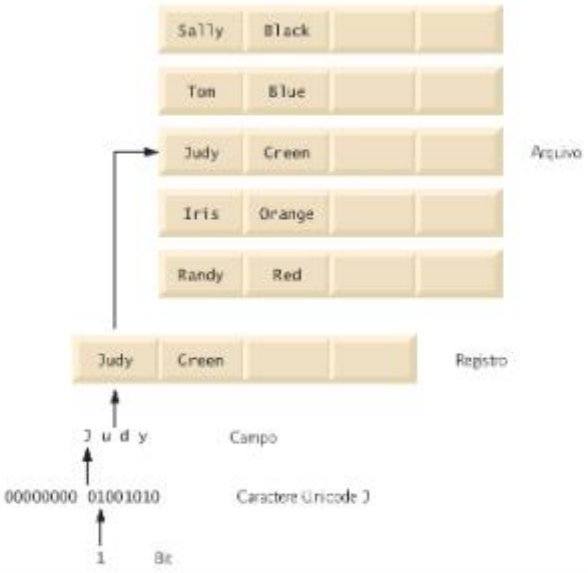


Figura 1.3 | Hierarquia de dados.

de um computador representa cada caractere como um padrão de `1`s e `0`s. O Java usa caracteres **Unicode®** que são compostos por um, dois ou quatro bytes (8, 16 ou 32 bits). O Unicode contém caracteres para muitos idiomas no mundo. Consulte o Apêndice H, na Sala Virtual (em inglês), para obter mais informações sobre o Unicode. Consulte o Apêndice B para informações adicionais sobre o conjunto de caracteres **ASCII (American Standard Code for Information Interchange)** — um subconjunto popular do Unicode que representa letras maiúsculas e minúsculas, dígitos e vários caracteres especiais comuns.

Campos

Assim como caracteres são compostos de bits, **campos** são compostos de caracteres ou bytes. Um campo é um grupo de caracteres ou bytes que transmitem um significado. Por exemplo, um campo que consiste em letras maiúsculas e minúsculas pode ser usado para representar o nome de uma pessoa, e um campo que consiste em dígitos decimais pode representar a idade de uma pessoa.

Registros

Vários campos relacionados podem ser usados para compor um **registro** (implementado como uma `class` no Java). Por exemplo, em um sistema de folha de pagamento, o registro para um funcionário poderia consistir nos seguintes campos (possíveis tipos para esses campos são mostrados entre parênteses):

- Número de identificação de funcionário (um número inteiro).
- Nome (uma string de caracteres).
- Endereço (uma string de caracteres).
- Remuneração por hora (um número com um ponto decimal).
- Rendimentos anuais até a presente data (um número com um ponto decimal).
- Quantidade de impostos retidos (um número com um ponto decimal).

Portanto, um registro é um grupo de campos relacionados. No exemplo anterior, todos os campos pertencem ao *mesmo* funcionário. Uma empresa pode ter muitos funcionários e um registro de folha de pagamento para cada um.

Arquivos

Um **arquivo** é um grupo de registros relacionados. [*Observação*: de maneira mais geral, um arquivo contém dados arbitrários em formatos arbitrários. Em alguns sistemas operacionais, um arquivo é visto simplesmente como uma *sequência de bytes* — qualquer organização dos bytes em um arquivo, como os dados em registros, é uma visualização criada pelo programador de aplicativo. Você verá como fazer isso no Capítulo 15.] Não é incomum uma organização ter muitos arquivos, que contêm alguns bilhões, ou mesmo trilhões, de caracteres de informações.

Banco de dados

Um **banco de dados** é uma coleção de dados organizados para fácil acesso e manipulação. O modelo mais popular é o *banco de dados relacional*, em que os dados são armazenados em *tabelas* simples. Uma tabela inclui *registros* e *campos*. Por exemplo, uma tabela de alunos pode incluir os campos nome, sobrenome, especialização, ano, número de identificação do aluno e rendimento acadêmico médio. Os dados para cada aluno são um registro, e as informações individuais em cada registro são os campos. Você pode *pesquisar, classificar* e manipular os dados com base no relacionamento com várias tabelas ou bancos de dados. Por exemplo, uma universidade pode usar os dados do banco de dados dos alunos em combinação com os dados dos bancos de dados dos cursos, habitação no *campus*, planos de refeição etc. Discutiremos bancos de dados no Capítulo 24.

Big data

A quantidade de dados que está sendo produzida no mundo é enorme e está crescendo rapidamente. De acordo com a IBM, cerca de 2,5 quintilhões de bytes (2,5 *exabytes*) de dados são criados por dia e 90% dos dados do mundo foram criados apenas nos últimos dois anos![6] De acordo com um estudo da Digital Universe, o suprimento global de dados alcançou 2,8 *zettabytes* (igual a 2,8 trilhões de gigabytes) em 2012.[7] A Figura 1.4 mostra algumas medições comuns de bytes. Aplicações de **big data** lidam com essas quantidades enormes de dados e esse campo está crescendo depressa, criando muitas oportunidades para desenvolvedores de software. Segundo uma pesquisa realizada pelo Gartner Group, mais de 4 milhões de empregos em TI no mundo todo suportarão *big data* até 2015.[8]

Unidade	Bytes	O que é aproximadamente
1 kilobyte (KB)	1024 bytes	10^3 (1024 bytes, exatamente)
1 megabyte (MB)	1024 kilobytes	10^6 (1.000.000 bytes)
1 gigabyte (GB)	1024 megabytes	10^9 (1.000.000.000 bytes)
1 terabyte (TB)	1024 gigabytes	10^{12} (1.000.000.000.000 bytes)
1 petabyte (PB)	1024 terabytes	10^{15} (1.000.000.000.000.000 bytes)
1 exabyte (EB)	1024 petabytes	10^{18} (1.000.000.000.000.000.000 bytes)
1 zettabyte (ZB)	21024 exabytes	10^{21} (1.000.000.000.000.000.000.000 bytes)

Figura 1.4 | Medições de bytes.

1.4 Linguagens de máquina, assembly e de alto nível

Os programadores escrevem instruções em várias linguagens de programação, algumas diretamente compreensíveis por computadores, outras requerendo passos intermediários de *tradução*. Centenas dessas linguagens estão em uso atualmente. Elas podem ser divididas em três tipos gerais:

1. Linguagens de máquina
2. Linguagens assembly
3. Linguagens de alto nível

Linguagens de máquina

Qualquer computador só pode entender diretamente sua própria **linguagem de máquina**, definida pelo seu projeto de hardware. As linguagens de máquina consistem geralmente em strings de números (em última instância, reduzidas a 1s e 0s) que instruem os computadores a realizar suas operações mais elementares uma de cada vez. As linguagens de máquina são *dependentes de máquina* (uma determinada linguagem de máquina pode ser utilizada apenas em um tipo de computador). Elas são complicadas para seres humanos. Por exemplo, eis uma seção de um primeiro programa de folha de pagamento em linguagem de máquina que adiciona o pagamento de horas extras à base de pagamentos e armazena o resultado como salário bruto:

```
+1300042774
+1400593419
+1200274027
```

[6] http://www-01.ibm.com/software/data/bigdata/.
[7] http://www.guardian.co.uk/news/datablog/2012/dec/19/big-data-study-digital-universe-global-volume.
[8] http://tech.fortune.cnn.com/2013/09/04/big-data-employment-boom/.

Linguagens assembly e assemblers

A programação em linguagem de máquina era simplesmente muito lenta e tediosa para a maioria dos programadores. Em vez de utilizar strings de números que os computadores poderiam entender de maneira direta, os programadores começaram a usar abreviações em inglês para representar operações elementares. Essas abreviações formaram a base de **linguagens assembly**. *Programas tradutores* chamados **assemblers** foram desenvolvidos para converter os primeiros programas de linguagem assembly em linguagem de máquina a velocidades de computador. A seção a seguir de um programa de folha de pagamento em linguagem assembly soma os ganhos em horas extras ao salário de base e armazena o resultado no salário bruto:

```
load        basepay
add         overpay
store       grosspay
```

Embora tal código seja mais claro para humanos, ele é incompreensível para computadores até ser traduzido em linguagem de máquina.

Linguagens de alto nível e compiladores

Com o advento das linguagens assembly, o uso de computadores aumentou rapidamente, mas os programadores ainda tinham de usar inúmeras instruções para realizar até mesmo as tarefas mais simples. A fim de acelerar o processo de programação, foram desenvolvidas **linguagens de alto nível** em que instruções únicas poderiam ser escritas para realizar tarefas substanciais. Os programas tradutores chamados **compiladores** convertem os programas de linguagem de alto nível em linguagem de máquina. Linguagens de alto nível permitem aos programadores escrever instruções que se pareçam com o inglês cotidiano e contenham notações matemáticas comumente utilizadas. Um programa de folha de pagamentos escrito em linguagem de alto nível poderia conter uma *única* instrução como:

```
grossPay = basePay + overTimePay
```

Do ponto de vista do programador, as linguagens de alto nível são preferíveis às de máquina e às assembly. O Java é uma das linguagens de programação de alto nível mais amplamente usadas.

Interpretadores

Compilar um programa grande de linguagem de alto nível em linguagem de máquina pode levar tempo considerável de computador. Os programas *interpretadores*, desenvolvidos para executar diretamente programas de linguagem de alto nível, evitam o tempo de espera da compilação, embora sejam mais lentos do que programas compilados. Discutiremos mais sobre interpretadores na Seção 1.9, na qual você aprenderá que o Java utiliza uma combinação de desempenho afinado e inteligente de compilação e interpretação para executar os programas.