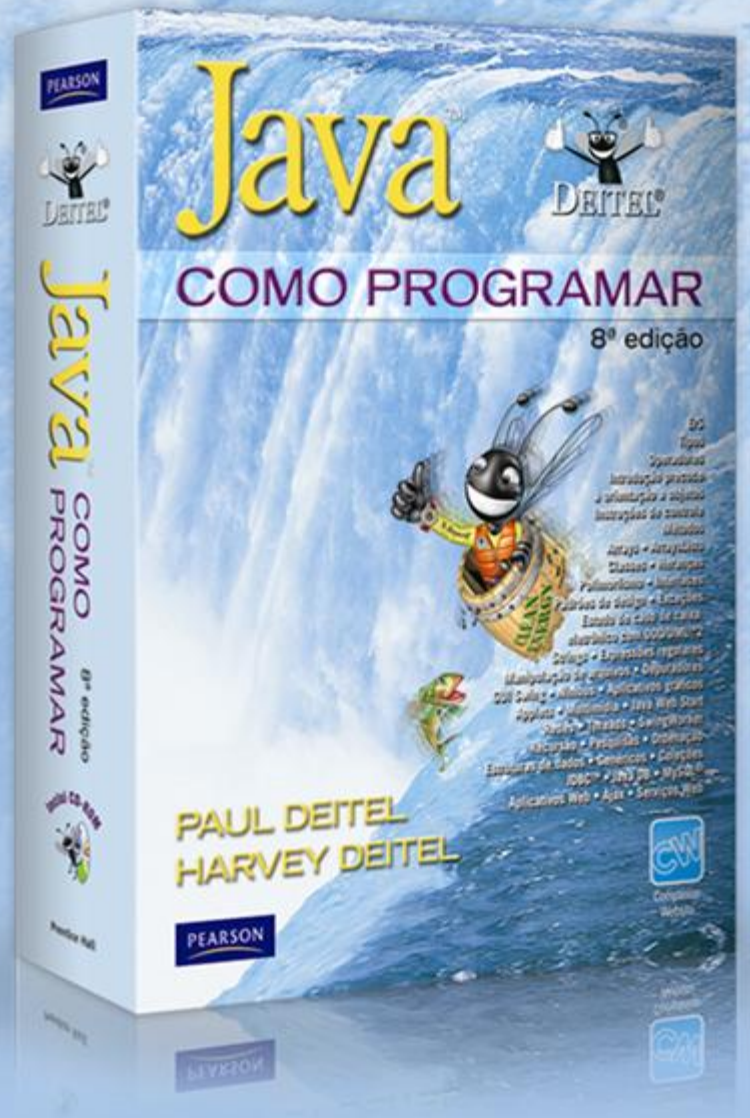
Capítulo 3
Introdução a classes e objetos

Java™ Como Programar, 8/E



(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

OBJETIVOS

Neste capítulo, você aprenderá:

- O que são classes, objetos, métodos e variáveis de instância.
- Como declarar uma classe e utilizá-la para criar um objeto.
- Como declarar métodos em uma classe para implementar os comportamentos da classe.
- Como declarar variáveis de instância em uma classe para implementar os atributos da classe.
- Como chamar os métodos de um objeto para fazer esses métodos realizarem suas tarefas.
- As diferenças entre variáveis de instância de uma classe e variáveis locais de um método.
- Como utilizar um construtor para assegurar que os dados de um objeto sejam inicializados quando o objeto for criado.
- As diferenças entre tipos por referência primitivos.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

3.1 Introdução
3.2 Classes, objetos, métodos e variáveis de instância
3.3 Declarando uma classe com um método e instanciando um objeto de uma classe
3.4 Declarando um método com um parâmetro
3.5 Variáveis de instância, métodos *set* e *get*
3.6 Tipos primitivos *versus* tipos por referência
3.7 Inicializando objetos com construtores
3.8 Números de ponto flutuante e tipo `double`
3.9 (Opcional) Estudo de caso de GUI e imagens gráficas: utilizando caixas de diálogo
3.10 Conclusão

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

3.1 Introdução

- Abordado neste capítulo
  - Classes
  - Objetos
  - Métodos
  - Parâmetros
  - Tipo primitivo `double`

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

3.2 Classes, objetos, métodos e variáveis de instância

- Analogia simples para ajudar a entender classes e seu conteúdo.

  Suponha que você queira guiar um carro e fazê-lo andar mais rápido pisando no pedal acelerador.

  Antes de poder dirigir um carro, alguém tem de projetá-lo.

  Em geral, um carro inicia com os desenhos de engenharia, semelhantes às plantas utilizadas para projetar uma casa.

  Estes incluem o projeto de um pedal acelerador para aumentar a velocidade do carro.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Analogia simples para ajudar a entender classes e seu conteúdo.

  O pedal "oculta" do motorista os complexos mecanismos que realmente fazem o carro ir mais rápido, assim como o pedal de freio "oculta" os mecanismos que diminuem a velocidade do carro e a direção "oculta" os mecanismos que mudam a direção do carro.

  Isso permite que as pessoas com pouco ou nenhum conhecimento de como os motores funcionam dirijam um carro facilmente.

  Antes de poder guiar um carro, ele deve ser construído a partir dos desenhos de engenharia que o descrevem.

  Um carro pronto tem um pedal de acelerador real para fazer o carro andar mais rápido, mas até isso não é suficiente — o carro não acelerará por conta própria, então o motorista deve pressionar o pedal do acelerador.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Para realizar uma tarefa em um programa é necessário um método.
  - O método descreve os mecanismos que realmente realizam suas tarefas.
  - A função oculta de seu usuário as tarefas complexas que ele realiza, assim como o pedal acelerador de um carro oculta do motorista os complexos mecanismos que fazem o carro andar mais rápido.
- Em Java, uma classe abriga um método, assim como os desenhos de engenharia do carro abrigam o projeto de um pedal acelerador.
- Em uma classe, você fornece um ou mais métodos que são projetados para realizar as tarefas da classe.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Você deve criar um objeto de uma classe antes de um programa realizar as tarefas que a classe descreve como fazer.

  Essa é uma razão por que o Java é conhecido como uma linguagem de programação orientada a objetos.
- Ao dirigir um carro, o ato de pressionar o acelerador envia uma mensagem para o carro realizar uma tarefa — fazer o carro andar mais rápido.
- Você **envia mensagens** para um objeto — cada mensagem é implementada como uma **chamada de método** que instrui um método do objeto a realizar sua tarefa.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Um carro tem muitos atributos

  Cor, o número de portas, a capacidade do tanque, a velocidade atual e a quilometragem.
- Atributos são representados como parte do projeto de um carro nos diagramas de engenharia.
- Cada carro mantém seus próprios atributos.

  Cada carro sabe a quantidade de gasolina que há no seu tanque, mas não sabe quanto há no tanque de outros carros.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Um objeto tem atributos que são carregados com o objeto quando ele é utilizado em um programa.
  - Especificados como parte da classe do objeto.
  - Um objeto conta bancária tem um atributo saldo que representa a quantidade de dinheiro na conta.
  - Cada objeto conta bancária sabe o saldo da conta que ele representa, mas não sabe os saldos de outras contas no banco.
- Os atributos são especificados pelas variáveis de instância da classe.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

3.3 Declarando uma classe com um método e instanciando um objeto de uma classe

- Crie uma nova classe (`GradeBook`)
- Use-a para criar um objeto.
- Cada declaração de classe que inicia com a palavra-chave `public` deve ser armazenada em um arquivo que tenha o mesmo nome da classe e terminar com a extensão de nome do arquivo `.java`.
- A palavra-chave `public` tem um modificador de acesso.

  Indica que a classe está "disponível para o público"

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

Erro comum de programação 3.1

Declarar mais de uma classe `public` *no mesmo arquivo é um erro de compilação.*

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- O método `main` é automaticamene chamado pela Java Virtual Machine (JVM) queando você executa um aplicativo.
- Normalmente, você deve chamar métodos explicitamente para instruí-los a realizar suas tarefas.
- Um `public` significa "disponível para o público"

  Pode ser chamado a partir de métodos de outras classes.
- O tipo de retorno especifica o tipo de dados que o método retorna depois de realizar a sua tarefa.
- O tipo de retorno `void` indica que esse método realizará uma tarefa mas *não* retornará (isto é, devolverá) nenhuma informação para seu método chamador ao completar sua tarefa.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- O nome do método é seguido pelo tipo de retorno.
- Por convenção, os nomes de método iniciam com a primeira letra minúscula e as palavras subseqüentes do nome iniciam com uma letra maiúscula.
- Parênteses vazios após um nome de método indicam que o método não requer nenhum parâmetro para realizar sua tarefa.
- Em conjunto, tudo na primeira linha do método é geralmente chamado de cabeçalho do méotdo
- O corpo de todos os métodos é delimitado por chaves de abertura e fechamento.
- O corpo de um método contém uma ou várias instruções que realizam a tarefa do método.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
1  // Figura 3.1: GradeBook.java
2  // Declaração de classe com um método.
3
4  public class GradeBook
5  {
6     // exibe uma mensagem de boas-vindas para o usuário GradeBook
7     public void displayMessage()
8     {
9        System.out.println( "Welcome to the Grade Book!" );
10    } // fim do método displayMessage
11 } // fim da classe GradeBook
```

Executa a tarefa de exibir uma mensagem na tela; o método `displayMessage` deve ser chamado para realizar essa tarefa

Figura 3.1 | Declaração de classe com um método.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Use a classe `GradeBook` em um aplicativo.
- A classe `GradeBook` não é um aplicativo porque não contém `main`.
- Não é possível executar `GradeBook`; receberá uma mensagem de erro como:

```
Exception in thread "main"
  java.lang.NoSuchMethodError: main
```

- Deve-se declarar uma classe separada que contém um método `main` ou colocar um método `main` na classe `GradeBook`.
- Para ajudá-lo a se preparar para programas, utilizamos uma classe separada contendo o método `main` para testar cada nova classe.
- Alguns programadores tratam essa classe como uma *driver class*.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
1  // Figura 3.2: GradeBookTest.java
2  // Criando um objeto GradeBook e chamando seu método displayMessage.
3
4  public class GradeBookTest
5  {
6     // o método main inicia a execução do programa
7     public static void main( String[] args )
8     {
9        // cria um objeto GradeBook e o atribui a myGradeBook
10       GradeBook myGradeBook = new GradeBook();
11
12       // chama método displayMessage de myGradeBook
13       myGradeBook.displayMessage();
14    } // fim de main
15 } // fim da classe GradeBookTest
```

Cria o objeto `GradeBook` e o atribui à variável `myGradeBook`

Invoca o método `displayMessage` no objeto GradeBook que foi atribuído à variável `myGradebook`

```
Welcome to the Grade Book!
```

Figura 3.2 | Criando um objeto `GradeBook` e chamando seu método `displayMessage`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Um método `static` (como `main`) é especial

  Ele pode ser chamado sem primeiro criar um objeto da classe em que o método é declarado.
- Em geral, você não pode chamar um método que pertence à outra classe até criar um objeto dessa classe.
- Declare a uma variável do tipo de classe.

  Cada nova classe que você cria torna-se um novo tipo que pode ser utilizado para declarar variáveis e criar objetos.

  Você pode declarar novos tipos de classe conforme necessário; essa é uma razão por que o Java é conhecido como uma linguagem extensível.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Expressão de criação de instância de classe
  - A palavra-chave new cria um novo objeto da classe especificada à direita da palavra-chave.
  - Utilizada para inicializar uma variável de um tipo de classe.
  - Os parênteses à direita do nome da classe são necessários.
  - Parênteses em combinação com um nome de classe representam uma chamada para um construtor, que é semelhante a um método, mas é utilizado na hora em que um objeto é criado para inicializar os dados do objeto.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Chame um método via a variável de tipo de classe

  Nome variável seguido por um ponto separador (.), o nome do método e parênteses.

  Essa chamada faz com que o método realize sua tarefa.

- Qualquer classe pode conter um método `main`

  A JVM invoca o método `main` somente na classe utilizada para executar o aplicativo.

  Se um aplicativo tiver múltiplas classes que contêm `main`, o método invocado é aquele na classe nomeada no comando `java2`

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Compilando um aplicativo com múltiplas classes

  Compile as classes na Figura 3.1 e Figura 3.2 antes de executar.

  Digite o comando

  ```
  javac GradeBook.java GradeBookTest.java
  ```

  Se o diretório que contém o aplicativo incluir somente os arquivos desse aplicativo, você pode compilar todas as classes no diretório com o comando

  ```
  javac *.java
  ```

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Figura 3.3: Diagrama de classe de UML para a classe `GradeBook`.
- Cada classe é modelada num diagrama de classe como um retângulo com três compartimentos.

  Parte superior: contém o nome de classe centraliazdo horizontalmente em tipo negrito.

  Meio: contém os atributos da classe, que correspondem a variáveis de exemplo (Seção 3.5).

  Parte inferior: contém as operações da classe, que correspondem a métodos.
- Operações são modeladas listando o nome da operação precedido por um modificador de acesso (nesse caso +) e seguido por um conjunto de parêntesis.
- O sinal de adição (+) corresponde à palavra-chave `public`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

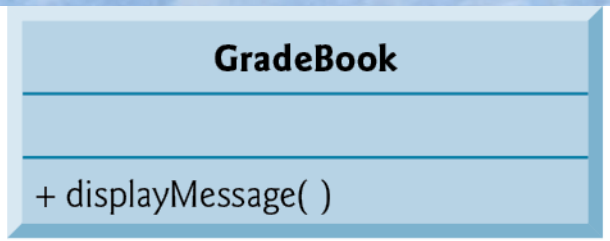


Figura 3.3 | Diagrama de classe UML indicando que a classe `GradeBook` tem uma operação `public displayMessage`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

3.4 Declarando um método com um parâmetro

- Analogia do carro

  O ato de pisar no acelerador envia uma mensagem para o carro realizar uma tarefa — fazer o carro andar mais rápido.

  Quanto mais você pisa no pedal, mais o carro acelera.

  A mensagem para o carro inclui a tarefa a ser realizada e informações adicionais que ajudam o carro a executar a tarefa.

- Parâmetros: Informações adicionais que um método precisa para executar sua tarefa.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Um método pode exigir um ou mais parâmetros que representam informações adicionais necessárias para realizar a tarefa.
  - Definida em uma lista de parâmetros delimitados por vírgula
  - Localizado nos parênteses que se seguem ao nome do método
  - Todo parâmetro deve especificar um tipo e um identificador.
- Uma chamada de método fornece valores — chamados argumentos— para cada um dos parâmetros do método.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
1  // Figura 3.4: GradeBook.java
2  // Declaração de classe com um método que tem um parâmetro.
3
4  public class GradeBook
5  {
6     // exibe uma mensagem de boas-vindas para o usuário de GradeBook
7     public void displayMessage( String courseName )
8     {
9        System.out.printf( "Welcome to the grade book for\n%s!\n",
10          courseName );
11    } // fim do método displayMessage
12 } // fim da classe GradeBook
```

O parâmetro `courseName` fornece as informações adicionais que o método requer para executar sua tarefa

O valor do parâmetro `courseName` é exibido como parte da saída

Figura 3.4 | Declaração de classe com um método que tem um parâmetro.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

```
1  // Figura 3.5: GradeBookTest.java
2  // Cria objeto GradeBook e passa uma String para
3  // seu método displayMessage.
4  import java.util.Scanner; // programa utiliza Scanner
5
6  public class GradeBookTest
7  {
8     // método main inicia a execução de programa
9     public static void main( String[] args )
10    {
11       // cria Scanner para obter entrada a partir da janela de comando
12       Scanner input = new Scanner( System.in );
13
14       // cria um objeto GradeBook e o atribui a myGradeBook
15       GradeBook myGradeBook = new GradeBook();
16
17       // prompt para entrada do nome do curso
18       System.out.println( "Please enter the course name:" );
19       String nameOfCourse = input.nextLine(); // lê uma linha de texto
20       System.out.println(); // gera saída de uma linha em branco
21
```

Lê uma linha de texto

Figura 3.5 | Criando um objeto `GradeBook` e passando uma `String` ao seu método `displayMessage`. (Parte 1 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
22          // chama método displayMessage de myGradeBook
23          // e passa nameOfCourse como um argumento
24          myGradeBook.displayMessage( nameOfCourse );
25       } // fim de main
26    } // fim da classe GradeBookTest
```

```
Please enter the course name:
CS101 Introduction to Java Programming

Welcome to the GradeBook for
CS101 Introduction to Java Programming!
```

Figura 3.5 | Criando um objeto `GradeBook` e passando uma `String` ao seu método `displayMessage`. (Parte 2 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Método `Scanner` nextLine
  - Lê caracteres digitados pelo usuário até que o caractere de nova linha seja encontrado
  - Retorna um `String` contendo os caracteres até, mas não includindo, a nova linha
  - Pressione *Enter* para submeter a string ao programa.
  - Pressionar *Enter* insere um caractere de nova linha no final dos caracteres digitados pelo usuário.
  - O caractere de nova linha é descartado por `nextLine`.
- Método `Scanner` next
  - Lê palavras individuais
  - Lê caracteres até que um caractere de espaço em branco ser encontrado, então retorna uma `String` (o caractere de espaço em branco é descartado).
  - As informações depois do primeiro caractere de espaço podem ser lidas por outras instruções que chamam os métodos de `Scanner` mais adiante no programa.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Mais sobre argumentos e parâmetros

  O número de argumentos em uma chamada de método deve corresponder ao número de parâmetros na lista de parâmetros da declaração do método.

  Os tipos de argumento na chamada do método devem ser "consistentes com" os tipos dos parâmetros correspondentes na declaração do método.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

Erro comum de programação 3.2

Ocorrerá um erro de compilação se o número de argumentos em uma chamada de método não corresponder ao número de parâmetros na declaração do método.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

Erro comum de programação 3.3

Ocorrerá um erro de compilação se o tipo de qualquer argumento em uma chamada de método não for consistente com o tipo do parâmetro correspondente na declaração do método.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- O diagrama da classe UML da Figura 3.6 modela a classe `GradeBook` da Figura 3.4.
- A UML modela um parâmetro listando o nome de parâmetro, seguido por um caractere de dois-pontos e o tipo de parâmetro entre os parênteses que se seguem ao nome da operação.
- O tipo UML `String` corresponde ao tipo Java `String`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

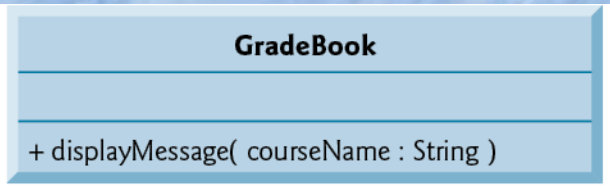


Figura 3.6 | Diagrama de classe UML indicando que a classe `GradeBook` tem uma operação `displayMessage` com um parâmetro `courseName` de tipo UML `String`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Notas sobre declarações `import`
  - As classes `System` e `String` estão no pacote `java.lang`
    - Implicitamente importadas em todo programa Java
    - Pode-se utilizar as classes `java.lang` sem explicitamente importá-las
    - A maioria das outras classes que você utilizará nos programas Java precisa ser importada explicitamente.
  - Classes que são compiladas no mesmo diretório estão no mesmo pacote — conhecido como pacote padrão.
  - As classes do mesmo pacote são importadas implicitamente para os arquivos de código-fonte de outras classes do mesmo pacote.
  - Uma declaração `import` não é requerida se você sempre referenciar uma classe por meio do seu nome de classe completamente qualificado
    - O nome do pacote seguido por um ponto (`.`) e o nome da classe.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

Observação de engenharia de software 3.1

O compilador Java não exigirá as declarações `import` *em um arquivo de código-fonte Java se o nome de classe completamente qualificado for especificado toda vez que um nome de classe for utilizado no código-fonte. A maioria dos programadores Java prefere utilizar declarações* `import`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

3.5 Variáveis de instância, métodos *set* e *get*

- Variáveis locais

  Variáveis declaradas no corpo de um método específico.

  Quando esse método termina, os valores de suas variáveis locais são perdidos.

  Lembrando-se da Seção 3.2, um objeto tem atributos que são carregados com o objeto quando ele é utilizado em um programa. Esses atributos existem antes de um método ser chamado em um objeto e depois de o método completar a execução.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Uma classe normalmente consiste em um ou mais métodos que manipulam os atributos que pertencem a um objeto particular da classe.
  - Os atributos são representados como variáveis em uma declaração de classe.
  - Campos GridBagConstraints.
  - Declarado dentro de uma declaração de classe mas fora do corpo das declarações de método da classe.
- Variável de instância
  - Quando cada objeto de uma classe mantém sua própria cópia de um atributo, o campo é uma variável de instância.
  - Todo objeto (instância) de classe tem uma instância separada da variável na memória.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

```
1  // Figura 3.7: GradeBook.java
2  // classe GradeBook que contém uma variável de instância
3  // courseName e métodos para configurar e obter seu valor.
4
5  public class GradeBook
6  {
7     private String courseName; // nome do curso para esse GradeBook
8
9     // método para configurar o nome do curso
10    public void setCourseName( String name )
11    {
12       courseName = name; // armazena o nome do curso
13    } // fim do método setCourseName
14
15    // método para recuperar o nome do curso
16    public String getCourseName()
17    {
18       return courseName;
19    } // fim do método getCourseName
20
```

Cada objeto GradeBook mantém sua própria cópia da variável de instância courseName

Método que permite ao cliente codificar para alterar o courseName

Método que permite ao cliente codificar para obter o courseName

Figura 3.7 | A classe GradeBook que contém uma variável de instância courseName e métodos para configurar e obter seu valor. (Parte 2 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
21       // exibe uma mensagem de boas-vindas para o usuário GradeBook
22       public void displayMessage()
23       {
24          // chama getCourseName para obter o nome do
25          // o curso que essa GradeBook representa
26          System.out.printf( "Welcome to the GradeBook for\n%s!\n",
27             getCourseName() );
28       } // fim do método displayMessage
29    } // fim da classe GradeBook
```

Nenhum parâmetro requerido; todos os métodos nesta classe já sabem sobre a variável de instância `courseName` e os outros métodos da classe

Boa prática para acessar suas variáveis de instância via métodos *set* ou *get*

Figura 3.7 | A classe `GradeBook` que contém uma variável de instância `courseName` e métodos para configurar e obter seu valor. (Parte 2 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Toda instância (isto é, objeto) da classe contém uma cópia de cada variável de instância.
- As variáveis de instância são geralmente declaradas `private`.

  private é um modificador de acesso.

  `private` são acessíveis somente aos métodos da classe em que eles são declarados
- Declarar uma instância como `private` é conhecido como ocultamento de dados ou ocultamento de informações.
- `private` são encapsuladas (ocultas) no objeto e somente podem ser acessadas por métodos da classe do objeto.

  Isso evita que variáveis de instância sejam modificadas acidentalmente por uma classe em outra parte do programa.

  Os métodos *Set* e *get* são utilizados para acessar variáveis de instância.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

Observação de engenharia de software 3.2

Anteceda cada campo e declaração de método com um modificador de acesso. Geralmente, as variáveis de instância devem ser declaradas `private` *e os métodos* `public`. *(Veremos que é apropriado declarar certos métodos* `private`, *se eles forem acessados apenas por outros métodos da classe.)*

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

Boa prática de programação 3.1

Preferimos listar os campos de uma classe primeiro, de modo que, à medida que você leia o código, também veja os nomes e tipos das variáveis antes que sejam utilizados nos métodos da classe. Você pode listar os campos da classe em qualquer lugar na classe fora de suas declarações de método, mas sua dispersão tende a resultar em um código de difícil leitura.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

Boa prática de programação 3.1

Preferimos listar os campos de uma classe primeiro, de modo que, à medida que você leia o código, também veja os nomes e tipos das variáveis antes que sejam utilizados nos métodos da classe. Você pode listar os campos da classe em qualquer lugar na classe fora de suas declarações de método, mas sua dispersão tende a resultar em um código de difícil leitura.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Quando um método que especifica um tipo de retorno diferente de `void` for chamado e completar sua tarefa, o método retornará um resultado para seu método chamador.
- Os métodos `setCourseName` e `getCourseName` utilizam a variável `courseName` embora ela não seja declarada em nenhum dos métodos.
  - Pode-se usar uma variável de exemplo da classe em cada um dos métodos de classes.
  - Uma exceção a isso são os métodos `static` (Capítulo 8)
- A ordem em que os métodos são declarados em uma classe não determina quando eles são chamados em tempo de execução.
- Um método de uma classe pode chamar outro método da mesma classe usando apenas o nome do método.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Diferentemente das variáveis locais, que não são automaticamente inicializadas, todo campo tem um valor inicial padrão — um valor fornecido pelo Java quando você não especifica o valor inicial do campo.
- Portanto, não é exigido que os campos sejam explicitamente inicializados antes de serem utilizados em um programa — a menos que eles devam ser inicializados com valores diferentes de seus valores padrão.
- O valor padrão de um campo de tipo `String` é `null`

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
1  // Figura 3.8: GradeBookTest.java
2  // Criando e manipulando um objeto GradeBook.
3  import java.util.Scanner; // programa utiliza Scanner
4
5  public class GradeBookTest
6  {
7     // método main inicia a execução de programa
8     public static void main( String[] args )
9     {
10       // cria Scanner para obter entrada a partir da janela de comando
11       Scanner input = new Scanner( System.in );
12
13       // cria um objeto GradeBook e o atribui a myGradeBook
14       GradeBook myGradeBook = new GradeBook();
15
16       // exibe valor inicial de courseName
17       System.out.printf( "Initial course name is: %s\n\n",
18          myGradeBook.getCourseName() );
19
20       // solicita e lê o nome do curso
21       System.out.println( "Please enter the course name:" );
22       String theName = input.nextLine(); // lê uma linha de texto
23       myGradeBook.setCourseName( theName ); // configura o nome do curso
```

Obtém o valor da variável de instância `courseName` do objeto `myGradeBook`

Configura o valor da variável de instância `courseName`

Figura 3.8 | Criando e manipulando um objeto `GradeBook`. (Parte 1 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
24          System.out.println(); // gera saída de uma linha em branco
25          // exibe mensagem de boas-vindas depois de especificar o
26          // nome do curso
27          myGradeBook.displayMessage();
28       } // fim de main
29    } // fim da classe GradeBookTest
```

Exibe a mensagem do GradeBook, incluindo o valor da variável de instância `courseName`

```
Initial course name is: null

Please enter the course name:
CS101 Introduction to Java Programming

Welcome to the GradeBook for
CS101 Introduction to Java Programming!
```

Figura 3.8 | Criando e manipulando um objeto `GradeBook`. (Parte 2 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Métodos *set* e *get*
  - Os campos `private` de uma classe só podem ser manipulados pelos métodos da classe.
  - Um cliente de um objeto chama os métodos `public` da classe para manipular os campos `private` de um objeto da classe.
  - As classes costumam fornecer métodos `public` para permitir aos clientes configurar (*set*, isto é, atribuir valores a) ou obter (*get*, isto é, obter os valores de) variáveis de instância `private`.
  - Os nomes desses métodos não precisam iniciar com *set* ou *get*, mas essa convenção de nomeção é recomendada.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- A Figura 3.9 contém um diagrama de classe UML atualizada da versão da classe `GradeBook` na Figura 3.7.

  Modela a variável de instância `courseName` como um atributo no compartimento do meio da classe.

  A UML representa as variáveis de instância como atributos listando o nome do atributo, seguido por um caractere de dois-pontos e o tipo de atributo.

  Um sinal de subtração (–) corresponde ao modificador de acesso `private`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

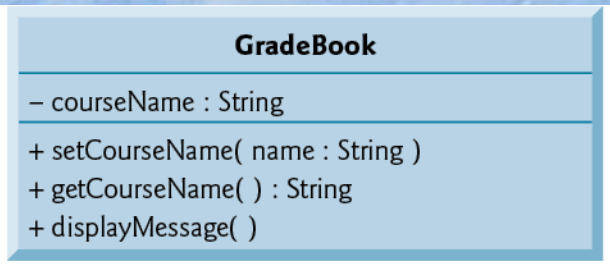


Figura 3.9 | O diagrama de classes UML que indica que a classe `GradeBook` tem um atributo `courseName` privado do tipo UML `String` e três operações públicas — `setCourseName` (com um parâmetro `name` do tipo UML `String`), `getCourseName` (que retorna o tipo UML `String`) e `displayMessage`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

3.6 Tipos primitivos *versus* tipos por referência

- Tipos são divididos em tipos primitivos e tipos por referência.
- Os tipos primitivos são `boolean`, `byte`, `char`, `short`, `int`, `long`, `float` e `double`.
- Todos os tipos não primitivos são tipos por referência.
- Uma variável de tipo primitivo pode armazenar exatamente um valor de seu tipo declarado de cada vez.
- As variáveis de instância de tipo primitivo são inicializadas por padrão — as variáveis dos tipos `byte`, `char`, `short`, `int`, `long`, `float` e `double` são inicializadas como 0, e variáveis do tipo `boolean` são inicializadas como `false`.
- Você pode especificar seu próprio valor inicial para uma variável do tipo primitivo atribuindo à variável um valor na sua declaração.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

Dica de prevenção de erro 3.1

Uma tentativa de utilizar uma variável local não inicializada causa um erro de compilação.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

3.7 Tipos primitivos *versus* tipos por referência

- Os programas utilizam as variáveis de tipos por referência (normalmente chamados referências) para armazenar as localizações de objetos na memória do computador.

  Diz-se que tal variável referencia um objeto no programa
- Os objetos que são referenciados podem todos conter muitas variáveis de instância e métodos.
- As variáveis de instância de tipo por referência são inicializadas por padrão para o valor `null`

  Uma palavra reservada que representa uma "referência a nada."
- Ao utilizar um objeto de outra classe, uma referência ao objeto deve invocar (isto é, chamar) seus métodos.

  Também conhecido como enviar mensagens a um objeto.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

Observação de engenharia de software 3.3

O tipo declarado de uma variável (por exemplo, `int`*,* `double` *ou* `GradeBook`*) indica se a variável é de um tipo primitivo ou tipo por referência. Se o tipo de uma variável não for um dos oito tipos primitivos, então ele é um tipo por referência.*

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

3.8 Inicializando objetos com construtores

- Quando um objeto de uma classe é criado, suas variáveis de instância são inicializadas por padrão.
- Cada classe pode fornecer um construtor que inicializa um objeto de uma classe quando o objeto é criado.
- O Java requer uma chamada de construtor para *todo* objeto que é criado
- A palavra-chave `new` solicita memória do sistema para armazenar um objeto e então chama o construtor da classe correspondente para inicializar o objeto.
- Um construtor *deve* ter o mesmo nome que a classe.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Por padrão, o compilador fornece um construtor padrão sem parâmetros em qualquer classe que não inclui explicitamente um construtor.
  - Variáveis de instância de classe são inicializadas com seus valores padrão
- Podem fornecer seu próprio construtor a fim de especificar uma inicialização personalizada para objetos de sua classe.
- A lista de parâmetros de um construtor especifica os dados que ele exige para realizar sua tarefa.
- Construtores não podem retornar valores, portanto não podem especificar um tipo de retorno.
- Normalmente, os construtores são declarados `public`.
- *Se você declarar qualquer construtor para uma classe, o compilador Java não criará um construtor padrão para essa classe.*

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
1  // Figura 3.10: GradeBook.java
2  // Classe GradeBook com um construtor para inicializar o
3  // nome de um curso.
4  public class GradeBook
5  {
6     private String courseName; // nome do curso para esse GradeBook
7
8     // o construtor inicializa courseName com o argumento String
9     public GradeBook( String name )
10    {
11       courseName = name; // inicializa courseName
12    } // fim do construtor
13
14    // método para configurar o nome do curso
15    public void setCourseName( String name )
16    {
17       courseName = name; // armazena o nome do curso
18    } // fim do método setCourseName
19
```

Construtor que inicializa `courseName` com um argumento

Figura 3.10 | A classe `GradeBook` com um construtor para inicializar o nome do curso. (Parte 1 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
20       // método para recuperar o nome do curso
21       public String getCourseName()
22       {
23          return courseName;
24       } // fim do método getCourseName
25
26       // exibe uma mensagem de boas-vindas para o usuário GradeBook
27       public void displayMessage()
28       {
29          // essa instrução chama getCourseName para obter o
30          // nome do curso que esse GradeBook representa
31          System.out.printf( "Welcome to the GradeBook for\n%s!\n",
32             getCourseName() );
33       } // fim do método displayMessage
34    } // fim da classe GradeBook
```

Figura 3.10 | A classe GradeBook com um construtor para inicializar o nome do curso. (Parte 2 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

```
1  // Figura 3.11: GradeBookTest.java
2  // construtor GradeBook utilizado para especificar o nome
3  // do curso na hora em que cada objeto GradeBook é criado.
4
5  public class GradeBookTest
6  {
7     // método main inicia a execução de programa
8     public static void main( String[] args )
9     {
10       // cria objeto GradeBook
11       GradeBook gradeBook1 = new GradeBook(
12          "CS101 Introduction to Java Programming" );
13       GradeBook gradeBook2 = new GradeBook(
14          "CS102 Data Structures in Java" );
15
16       // exibe valor inicial de courseName para cada GradeBook
17       System.out.printf( "gradeBook1 course name is: %s\n",
18          gradeBook1.getCourseName() );
19       System.out.printf( "gradeBook2 course name is: %s\n",
20          gradeBook2.getCourseName() );
21    } // fim de main
22 } // fim da classe GradeBookTest
```

A expressão de criação de instância de classe inicializa GradeBook e retorna uma referência que é atribuída à variável gradeBook1

A expressão de criação de instância de classe inicializa GradeBook e retorna uma referência que é atribuída à variável gradeBook2

Figura 3.11 | O construtor de GradeBook usado para especificar o nome do curso no momento em que cada objeto GradeBook é criado. (Parte 1 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
gradeBook1 course name is: CS101 Introduction to Java Programming
gradeBook2 course name is: CS102 Data Structures in Java
```

Figura 3.11 | O construtor de `GradeBook` usado para especificar o nome do curso no momento em que cada objeto `GradeBook` é criado. (Parte 2 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

Dica de prevenção de erro 3.2

A menos que a inicialização padrão de variáveis de instância de sua classe seja aceitável, forneça um construtor para assegurar que as variáveis de instância da sua classe sejam adequadamente inicializadas com valores significativos quando cada novo objeto de sua classe for criado.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- O diagrama de classes UML da Fig. 3.12 modela a classe `GradeBook` da Figura 3.10, que tem um construtor que tem um parâmetro `name` do tipo `String`.
- Assim como as operações, a UML modela construtores no terceiro compartimento de uma classe em um diagrama de classe.
- Para distinguir entre um construtor e operações de uma classe, a UML requer que a palavra "constructor" seja colocada entre aspas francesas (« e ») antes do nome do construtor.
- Liste os construtores antes de outras operações no terceiro compartimento.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

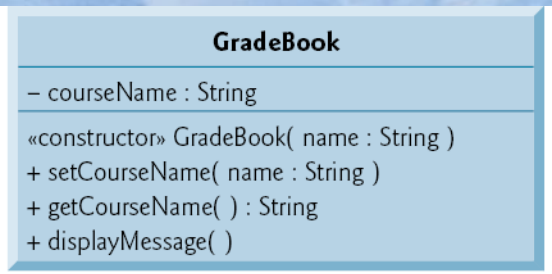


Figura 3.12 | Diagrama de classe UML que indica que a classe `GradeBook` tem um construtor com um parâmetro `name` de tipo UML `String`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

3.9 Números de ponto flutuante e tipo `double`

- Número de ponto flutuante

  Um número com um ponto de fração decimal, como 7,33, 0,0975 ou 1000,12345.

  Tipos primitivos `float` e `double`

  variáveis `double` podem armazenar números com maior magnitude e mais detalhes que variáveis `float`.
- float representa números de ponto flutuante de precisão simples até sete dígitos significativos.
- double representa números de ponto flutuante de precisão dupla que exigem o dobro de memória que `float` e fornecem 15 dígitos significativos — aproximadamente o dobro da precisão de variáveis `float`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- O Java trata literais de ponto flutuante (como 7,33 e 0,0975) como valores `double` por padrão.
- O Apêndice D, "Tipos primitivos", mostra os intervalos de valores de `float`s e `double`s.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

Erro comum de programação 3.4

Utilizar números de ponto flutuante pressupondo que sua representação é precisa pode levar a resultados incorretos.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

```
1  // Figura 3.13: Account.java
2  // classe Account com um construtor para validar e
3  // inicializa a variável de instância balance do tipo double.
4
5  public class Account
6  {
7     private double balance; // variável de instância que armazena o saldo
8
9     // construtor
10    public Account( double initialBalance )
11    {
12       // valida que initialBalance é maior que 0,0;
13       // se não, o saldo é inicializado como o valor padrão 0.0
14       if ( initialBalance > 0.0 )
15          balance = initialBalance;
16    } // fim do construtor Account
17
18    // credita (adiciona) uma quantia à conta
19    public void credit( double amount )
20    {
21       balance = balance + amount; // adiciona quantia ao saldo
22    } // fim do método credit
```

Número de ponto flutuante para o saldo da conta

Parâmetro utilizado para inicializar a variável de instância `balance`

Valida o valor do parâmetro para assegurar que ele é maior que 0

Inicializa `gradeCounter` como 1; indica que a primeira nota está prestes a ser inserida

Figura 3.13 | A classe `account` com um construtor para validar e inicializar a variável de instância `balance` do tipo `double`. (Parte 1 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
23
24       // retorna o saldo da conta
25       public double getBalance()
26       {
27          return balance; // fornece o valor de saldo ao método chamador
28       } // fim do método getBalance
29    } // fim da classe Account
```

Retorna o valor da variável de instância `balance` como `double`

Figura 3.13 | A classe `account` com um construtor para validar e inicializar a variável de instância `balance` do tipo `double`. (Parte 2 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- `System.out.printf`

  Especificador de formato `%.2f`

  `%f` é usado para dar saída de valores do tipo `float` ou `double`.

  `.2` representa o número de casas decimais (`2`) que devem ser enviadas para a saída à direita do ponto decimal — também conhecido como precisão do número.

  Qualquer saída de valor de ponto flutuante com `%.2f` será arredondada para a casa dos centésimos.

- O método `Scanner` `nextDouble` retorna um valor `double` inserido pelo usuário.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
1  // Figura 3.14: AccountTest.Java
2  // Entrada e saída de números de ponto flutuante com objetos Account.
3  import java.util.Scanner;
4
5  public class AccountTest
6  {
7     // método main inicia a execução do aplicativo Java
8     public static void main( String[] args )
9     {
10       Account account1 = new Account( 50.00 ); // cria o objeto Account
11       Account account2 = new Account( -7.53 ); // cria o objeto Account
12
13       // exibe o saldo inicial de cada objeto
14       System.out.printf( "account1 balance: $%.2f \n",
15          account1.getBalance() );
16       System.out.printf( "account2 balance: $%.2f \n\n",
17          account2.getBalance() );
18
19       // cria Scanner para obter entrada a partir da janela de comando
20       Scanner input = new Scanner( System.in );
21       double depositAmount; // quantia de depósito lida a partir do usuário
```

Valores de ponto flutuante com dois dígitos de precisão

Figura 3.14 | Entrada e saída de números de ponto flutuante com objetos `Account`. (Parte 1 de 3.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
22
23          System.out.print( "Enter deposit amount for account1: " ); // prompt
24          depositAmount = input.nextDouble(); // entrada do usuário
25          System.out.printf( "\nadding %.2f to account1 balance\n\n",
26             depositAmount );
27          account1.credit( depositAmount ); // adiciona o saldo de account1
28
29          // exibe os saldos
30          System.out.printf( "account1 balance: $%.2f \n",
31             account1.getBalance() );
32          System.out.printf( "account2 balance: $%.2f \n\n",
33             account2.getBalance() );
34          // prompt
35          System.out.print( "Enter deposit amount for account2: " );
36          depositAmount = input.nextDouble(); // entrada do usuário
37          System.out.printf( "\nadding %.2f to account2 balance\n\n",
38             depositAmount );
39          account2.credit( depositAmount ); // adiciona ao saldo
40                                            // de account2
41          // exibe os saldos
42          System.out.printf( "account1 balance: $%.2f \n",
43             account1.getBalance() );
```

Retorna um valor `double` digitado pelo usuário

Figura 3.14 | Entrada e saída de números de ponto flutuante com objetos `Account`. (Parte 2 de 3.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
44          System.out.printf( "account2 balance: $%.2f \n",
45             account2.getBalance() );
46       } // fim de main
47    } // fim da classe AccountTest
```

```
account1 balance: $50.00
account2 balance: $0.00

Enter deposit amount for account1: 25.53

adding 25.53 to account1 balance

account1 balance: $75.53
account2 balance: $0.00

Enter deposit amount for account2: 123.45

adding 123.45 to account2 balance

account1 balance: $75.53
account2 balance: $123.45
```

Figura 3.14 | Entrada e saída de números de ponto flutuante com objetos `Account`. (Parte 3 de 3.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- O diagrama de classe UML na Fig. 3.15 modela a classe `Account` da Figura 3.13.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

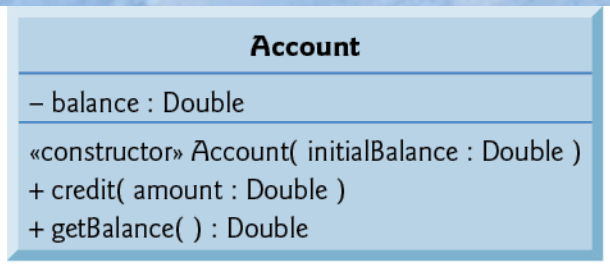


Figura 3.15 | O diagrama de classes UML indicando que a classe `Account` tem um atributo `private balance` do tipo UML `Double`, um construtor (com um parâmetro do tipo UML `Double`) e duas operações `public` — `credit` (com um parâmetro `amount` do tipo UML `Double`) e `getBalance` (retorna o tipo UML `Double`).

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

3.10 (Opcional) Estudo de caso de GUI e imagens gráficas: utilizando caixas de diálogo

- Figura 3.16: Resumo do "Estudo de caso GUI e imagens gráficas" em cada capítulo.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Posição	Título — Exercício(s)
Seção 3.9	Utilizando caixas de diálogo — Entrada e saída básicas com caixas de diálogo
Seção 4.14	Criando desenhos simples — exibindo e desenhando linhas na tela
Seção 5.10	Desenhando retângulos e ovais — Utilizando formas para representar dados
Seção 6.13	Cores e formas preenchidas — Desenhando um alvo e imagens gráficas aleatórias
Seção 7.15	Desenhando arcos — desenhando espirais com arcos
Seção 8.16	Utilizando objetos com elementos gráficos — armazenando formas como objetos
Seção 9.8	Exibindo texto e imagens utilizando rótulos — Fornecendo informações de status
Seção 10.8	Desenhando com polimorfismo — identificando as semelhanças entre as formas
Exercícios 14.17	Expandindo a interface — Utilizando componentes GUI e tratamento de evento
Exercícios 15.31	Adicionando Java 2D — Utilizando a API Java 2D para aprimorar desenhos

Figura 3.16 | Resumo dos estudos de caso de GUI e imagens gráficas em cada capítulo.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Muitos aplicativos utilizam janelas ou caixas de diálogo (também chamadas diálogos) para exibir a saída.
- Em geral, caixas de diálogo são janelas em que os programas exibem mensagens importantes aos usuários.
- A classe JOptionPane fornece caixas de diálogo pré-construídas que permitem aos programas exibir janelas que contém mensagens — essas janelas são chamadas de diálogos de mensagem.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

```
1  // Figura 3.17: Dialog1.java
2  // Imprimindo múltiplas linhas na caixa de diálogo.
3  import javax.swing.JOptionPane; // importa classe JOptionPane
4
5  public class Dialog1
6  {
7     public static void main( String[] args )
8     {
9        // exibe um diálogo com uma mensagem
10       JOptionPane.showMessageDialog( null, "Welcome\nto\nJava" );
11    } // fim de main
12 } // fim da classe Dialog1
```

Importa a classe `JOptionPane` para ser usada neste programa

Exibe o diálogo de mensagem no centro da tela

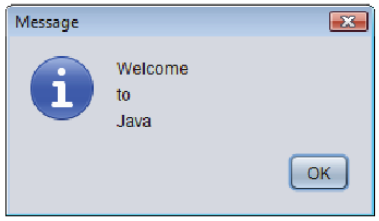


Figura 3.17 | Utilizando `JOptionPane` para exibir múltiplas linhas em uma caixa de diálogo.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- O pacote **javax.swing** contém muitas classes que o ajudam a criar interfaces gráficas com o usuário (GUIs).
- Componentes GUI facilitam a entrada de dados pelo usuário de um programa e apresentação das saídas ao usuário.
- O método `JOptionPane` showMessageDialog exibe uma caixa de diálogo que contém uma mensagem.

  Exige dois argumentos.

  O primeiro ajuda o aplicativo Java a determinar onde posicionar a caixa de diálogo.

    Se o primeiro argumento for `null`, a caixa de diálogo será exibida no centro da tela.

  O segundo argumento é a `String` a ser exibida na caixa de diálogo.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- O método `JOptionPane` `showMessageDialog` é um método static.
- Esses métodos costumam definir tarefas frequentemente utilizadas.
- Em geral, chamado utilizando seu nome de classe seguido por um ponto (`.`) e o nome de método, como em

  NomeDaClasse`.`*nomeDoMétodo*`(` *argumentos* `)`

- Note que não você cria um objeto da classe `JOptionPane` para utilizar seu método `static` `showMessageDialog`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- Um **diálogo de entrada** permite que o usuário insira dados num programa.
- O método `JOptionPane` showInputDialog exibe um diálogo de entrada
  Contém um prompt e um campo (conhecido como text field) em que o usuário pode inserir o texto.
- O método `showInputDialog` (linha 11) retorna uma `String` contendo os caracteres digitados pelo usuário.
- Se você pressionar o butão Cancel ou pressionar a tecla *Esc* , o método retorna `null`.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

- O método `static String` format retura uma `String` formatada.
- O método `format` funciona como o método `System.out.printf`, exceto que `format` retorna a `String` formatada em vez de exibi-la numa janela de comando.

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.


```
1  // Figura 3.18: NameDialog.Java
2  // Entrada básica com uma caixa de diálogo.
3  import javax.swing.JOptionPane;
4
5  public class NameDialog
6  {
7     public static void main( String[] args )
8     {
9        // pede para o usuário inserir seu nome
10       String name =
11          JOptionPane.showInputDialog( "What is your name?" );
12
13       // cria a mensagem
14       String message =
15          String.format( "Welcome, %s, to Java Programming!", name );
16
17       // exibe a mensagem para cumprimentar o usuário pelo nome
18       JOptionPane.showMessageDialog( null, message );
19    } // fim de main
20 } // termina NameDialog
```

Exibe um diálogo de entrada para obter dados do usuário

Cria uma String formatada contendo o nome inserido pelo usuário na caixa de diálogo

Figura 3.18 | Obtendo a entrada de usuário a partir de um diálogo. (Parte 1 de 2)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.

Java™ COMO PROGRAMAR 8ª edição

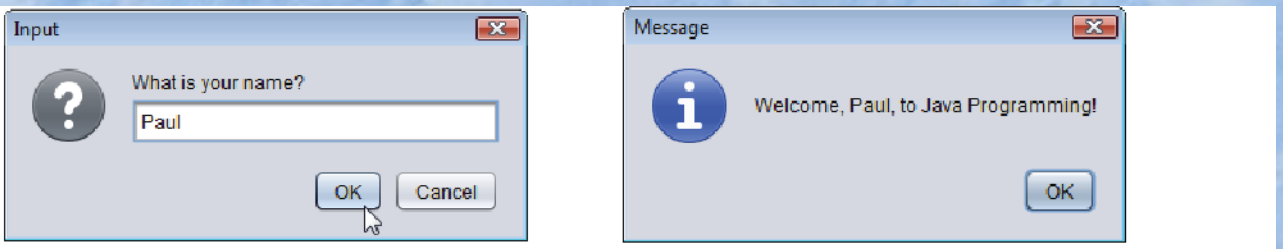


Figura 3.18 | Obtendo a entrada de usuário a partir de um diálogo. (Parte 2 de 2.)

(C) 2010 Pearson Education, Inc. Todos os direitos reservados.