used often in a variety of applications, it might be helpful to store it as a stored procedure in the DBMS. Then, you can call the stored procedure from any of your applications when you need to execute the SQL statement. Because stored procedures are already in the DBMS, they usually execute faster than SQL statements that are submitted from applications outside the DBMS.

We won't go into the details of stored procedures in this book, but we will point you in the right direction if you want to learn more. Each DBMS has its own syntax for creating a stored procedure in SQL, so you will have to consult your DBMS documentation to determine the format. Once you have properly written a stored procedure in SQL, you simply submit it to the DBMS using the `Statement` class's `execute` method. To execute a stored procedure, you must create a `CallableStatement` object. `CallableStatement` is an interface in the `java.sql` package. To create a `CallableStatement` object, you call the `Connection` class's `prepareCall` statement.

16.13 Common Errors to Avoid

- **Using the `==` operator instead of the `=` operator in an SQL statement.** The equal-to operator in SQL is one `=` sign, instead of two.
- **Using double quotes around strings instead of single quotes.** String literals in SQL are enclosed in single quotes instead of double quotes.
- **Using `&&` and `||` in an SQL statement.** The logical AND and logical OR operators in SQL are the words `AND` and `OR`, not the `&&` and `||` symbols.
- **Not using the correct `WHERE` clause in an `UPDATE` statement.** Be careful that you do not leave out the `WHERE` clause and the conditional expression when using an `UPDATE` statement. You could change the contents of every row in the table!
- **Not using the correct `WHERE` clause in a `DELETE` statement.** Be careful that you do not leave out the `WHERE` clause and the conditional expression when using a `DELETE` statement. You could delete every row in the table!
- **Not using the correct `WHERE` clause when joining data.** When joining data from multiple tables, be sure to use a `WHERE` clause to specify search criteria that link the appropriate columns. Failure to do so will result in a large set of unrelated data.

Review Questions and Exercises

Multiple Choice and True/False

1. This is the technology that makes it possible for a Java application to communicate with a DBMS.
   a. DBMSC
   b. JDBC
   c. JDBMS
   d. JDSQL

2. This is a standard language for working with database management systems.
   a. Java
   b. COBOL
   c. SQL
   d. BASIC
3. The data that is stored in a table is organized in _________.
   a. rows
   b. files
   c. folders
   d. pages
4. The data that is stored in a row is divided into _________.
   a. sections
   b. bytes
   c. columns
   d. tables
5. This is a column that holds a unique value for each row, and can be used to identify specific rows.
   a. ID column
   b. public key
   c. designator column
   d. primary key
6. This type of SQL statement is used to retrieve rows from a table.
   a. `RETRIEVE`
   b. `GET`
   c. `SELECT`
   d. `READ`
7. This contains the results of an SQL `SELECT` statement.
   a. select set
   b. result set
   c. SQL set
   d. collection set
8. This clause allows you to specify search criteria with the `SELECT` statement.
   a. `SEARCH`
   b. `WHERE`
   c. `AS`
   d. `CRITERIA`
9. This is a Java class that is designed to communicate with a specific DBMS.
   a. JDBC driver
   b. DBMS Superclass
   b. DBMS Subclass
   d. Stream converter
10. This is a string listing the protocol that should be used to access a database, the name of the database, and potentially other items.
    a. JDBC driver
    b. JDBC locator
    c. Database URL
    d. Database specifier

11. This method is specified in the `Statement` interface, and should be used to execute a `SELECT` statement.
    a. `execute`
    b. `executeUpdate`
    c. `executeQuery`
    d. `executeSelect`
12. This method is specified in the `Statement` interface, and should be used to execute an `UPDATE` statement.
    a. `execute`
    b. `executeUpdate`
    c. `executeQuery`
    d. `executeSelect`
13. This method is specified in the `Statement` interface, and should be used to execute an `INSERT` statement.
    a. `execute`
    b. `executeUpdate`
    c. `executeQuery`
    d. `executeSelect`
14. This SQL statement is used to insert rows into a table.
    a. `INSERT`
    b. `ADD`
    c. `CREATE`
    d. `UPDATE`
15. This SQL statement is used to remove rows from a table.
    a. `REMOVE`
    b. `ERASE`
    c. `PURGE`
    d. `DELETE`
16. This SQL statement is used to delete an entire table.
    a. `REMOVE`
    b. `DROP`
    c. `PURGE`
    d. `DELETE`
17. This is a column in one table that references a primary key in another table.
    a. secondary key
    b. fake key
    c. foreign key
    d. duplicate key
18. **True/False:** Java comes with its own built-in DBMS.
19. **True/False:** A Java programmer that uses a DBMS to store data does not need to know about the physical structure of the data.
20. **True/False:** You use SQL instead of Java to write entire applications, including the user interface.
21. **True/False:** In SQL, the not-equal-to operator is `!=`, which is the same as in Java.
22. **True/False:** When a `ResultSet` object is initially created, its cursor is pointing at the first row in the result set.

23. **True/False:** In a transaction, it is permissible for only some of the database updates to be made.
24. **True/False:** The term *rollback* refers to undoing changes to a database.

Find the Error

1. Find the error in the following SQL statement.

```
SELECT * FROM Coffee WHERE Description = "French Roast Dark"
```

2. Find the error in the following SQL statement.

```
SELECT * FROM Coffee WHERE ProdNum != '14-001'
```

3. Find the error in the following Java code. Assume that conn references a valid `Connection` object.

```
// This code has an error!!!
String sql = "SELECT * FROM Coffee";
Statement stmt = conn.createStatement();
ResultSet result = stmt.execute(sql);
```

Algorithm Workbench

1. What SQL data types correspond with the following Java types?
   - `int`
   - `float`
   - `String`
   - `double`
2. Look at the following SQL statement.

```
SELECT Name FROM Employee
```

   What is the name of the table from which this statement is retrieving data?

   What is the name of the column that is being retrieved?

For questions 3 through 12, assume that a database has a table named `Stock`, *with the following columns:*

Column Name	Type
`TradingSymbol`	`CHAR(10)`
`CompanyName`	`CHAR(25)`
`NumShares`	`INT`
`PurchasePrice`	`DOUBLE`
`SellingPrice`	`DOUBLE`

3. Write a `SELECT` statement that will return all of the columns from every row in table.
4. Write a `SELECT` statement that will return the `TradingSymbol` column from every row in table.
5. Write a `SELECT` statement that will return the `TradingSymbol` column and the `NumShares` column from every row in table.
6. Write a `SELECT` statement that will return the `TradingSymbol` column only from the rows where `PurchasePrice` is greater than 25.00.

7. Write a `SELECT` statement that will return all of the columns from the rows where `TradingSymbol` starts with "SU".
8. Write a `SELECT` statement that will return the `TradingSymbol` column only from the rows where `SellingPrice` is greater than `PurchasePrice`, and `NumShares` is greater than 100.
9. Write a `SELECT` statement that will return the `TradingSymbol` column and the `NumShares` column only from the rows where `SellingPrice` is greater than `PurchasePrice`, and `NumShares` is greater than 100. The results should be sorted by the `NumShares` column, in ascending order.
10. Write an SQL statement that will insert a new row into the `Stock` table. The row should have the following column values:

    `TradingSymbol`: XYZ
    `CompanyName`: "XYZ Company"
    `NumShares`: 150
    `PurchasePrice`: 12.55
    `SellingPrice`: 22.47
11. Write an SQL statement that does the following: For each row in the `Stock` table, if the `TradingSymbol` column is "XYZ", change it to "ABC".
12. Write an SQL statement that will delete rows in the `Stock` table where the number of shares is less than 10.
13. Assume that the following declaration exists.

    ```
    final String DB_URL = "jdbc:derby:CoffeeDB";
    ```

    The string referenced by `DB_URL` is a database URL. Write a statement that uses this string to get a connection to the database.
14. Assuming that `conn` references a valid `Connection` object, write code to create a `Statement` object. (Do not be concerned about result set scrolling or concurrency.)
15. Look at the following declaration.

    ```
    String sql = "SELECT * FROM Coffee WHERE Price > 10.00";
    ```

    Assume also that `stmt` references a valid `Statement` object. Write code that executes the SQL statement referenced by the `sql` variable.
16. Assume that the following code is used to retrieve data from the `CoffeeDB` database's `Coffee` table. Write the code that should appear inside the loop to display the contents of the result set.

    ```
    String sql = "SELECT * FROM Coffee";
    Connection conn = DriverManager.getConnection(DB_URL);
    Statement stmt = conn.createStatement();
    ResultSet result = stmt.executeQuery(sql);
    while (result.next())
    {
       // Finish this code!!
    }
    stmt.close();
    conn.close();
    ```

17. Write an SQL statement to create a table named `Car`. The `Car` table should have the columns to hold a car's manufacturer, year model, and a 20-character vehicle ID number.
18. Write an SQL statement to delete the `Car` table you created in Algorithm Workbench 17.

Short Answer

1. If you are writing an application to store the customer and inventory records for a large business, why would you not want to use traditional text or binary files?
2. You hear a fellow classmate say the following: "JDBC is a standard language for working with database management systems. It was invented at IBM." Are these statements correct, or is he confusing JDBC with something else?
3. When we speak of database organization, we speak of such things as rows, tables, and columns. Describe how the data in a database is organized into these conceptual units.
4. What is a primary key?
5. What is a result set?
6. What are the relational operators in SQL for the following comparisons?
   Greater-than
   Less-than
   Greater-than or equal-to
   Less-than or equal-to
   Equal-to
   Not equal-to
7. What is the number of the first row in a table? What is the number of the first column in a table?
8. What is metadata? What is result set metadata? When is result set metadata useful?
9. What is a foreign key?

Programming Challenges

MyProgrammingLab™ *Visit www.myprogramminglab.com to complete many of these Programming Challenges online and get instant feedback.*

1. Customer Inserter

Write an application that connects to the `CoffeeDB` database, and allows the user to insert a new row into the `Customer` table.

2. Customer Updater

Write an application that connects to the `CoffeeDB` database, and allows the user to select a customer, then change any of that customer's information. (You should not attempt to change the customer number, because it is referenced by the `UnpaidOrder` table.)

3. Unpaid Order Sum

Write an application that connects to the `CoffeeDB` database, then calculates and displays the total amount owed in unpaid orders. This will be the sum of each row's `Cost` column.

4. Unpaid Order Lookup

Write an application that connects to the `CoffeeDB` database and displays a `JList` component. The `JList` component should display a list of customers with unpaid orders. When the user clicks on a customer, the application should display a summary of all the unpaid orders for that customer.

5. Population Database

Make sure you have downloaded the book's source code from the companion Web site at `www.pearsonhighered.com/gaddis`. In this chapter's source code folder you will find a program named `CreateCityDB.java`. Compile and run the program. The program will create a Java DB database named `CityDB`. The `CityDB` database will have a table named `City`, with the following columns:

Column Name	Data Type
`CityName` *Primary key*	`CHAR (50)`
`Population`	`DOUBLE`

The `CityName` column stores the name of a city and the `Population` column stores the population of that city. After you run the `CreateCityDB.java` program, the `City` table will contain 20 rows with various cities and their populations.

Next, write a program that connects to the `CityDB` database, and allows the user to select any of the following operations:

- Sort the list of cities by population, in ascending order.
- Sort the list of cities by population, in descending order.
- Sort the list of cities by name.
- Get the total population of all the cities.
- Get the average population of all the cities.
- Get the highest population.
- Get the lowest population.

6. `Personnel` Database Creator

Write an application that creates a database named `Personnel`. The database should have a table named `Employee`, with columns for employee ID, name, position, and hourly pay rate. The employee ID should be the primary key. Insert at least five sample rows of data into the `Employee` table.

7. Employee Inserter

Write a GUI application that allows the user to add new employees to the `Personnel` database you created in Programming Challenge 6.

8. Employee Updater

Write a GUI application that allows the user to look up an employee in the `Personnel` database you created in Programming Challenge 6. The user should be able to change any of the employee's information except employee ID, which is the primary key.

9. `PhoneBook` Database

Write an application that creates a database named `PhoneBook`. The database should have a table named `Entries`, with columns for a person's name and phone number.

Next, write an application that lets the user add rows to the `Entries` table, look up a person's phone number, change a person's phone number, and delete specified rows.

Java™ Quick Reference

Primitive Data Types

Data Type	Description
`boolean`	Boolean (`true` or `false`)
`char`	Character
`int`	Integer
`short`	Short integer
`long`	Long integer
`float`	Single precision floating point
`double`	Double precision floating point

Opening a File for Output:

```
import java.io.*;

PrintWriter outputFile = new
     PrintWriter(filename);
```

Opening a File for Input:

```
import java.io.*;
import java.util.Scanner;

File myFile = new File(filename);
Scanner inputFile = new Scanner(myFile);
```

Forms of the `if` *Statement*

Simple `if` statement:

```
if (expression)
   statement;
```

Example:

```
if (x < y)
   x++;
```

`if/else` statement:

```
if (expression)
   statement;
else
   statement;
```

Example:

```
if (x < y)
   x++;
else
   x--;
```

`if/else if` statement:

```
if (expression)
   statement;
else if (expression)
   statement;
else
   statement;
```

Example:

```
if (x < y)
   x++;
else if (x < z)
   x--;
else
   y++;
```

To conditionally-execute more than one statement, enclose the statements in braces:

Form:

```
if (expression)
{
   statement;
   statement;
}
```

Example:

```
if (x < y)
{
   x++;
   z = x;
}
```

Web Sites

For the Gaddis Series:

`www.pearsonhighered.com/gaddis`

For Pearson Computing:

`www.pearsonhighered.com/cs`

Format of a Class with a Static `main` *Method*

```
public class ClassName
{
   public static void main(String[] args)
   {
      statements;
   }
}
```

Commonly Used Operators

Assignment Operators

`=`	Assignment
`+=`	Combined addition/assignment
`-=`	Combined subtraction/assignment
`*=`	Combined multiplication/assignment
`/=`	Combined division/assignment
`%=`	Combined modulus (remainder)/assignment

Arithmetic Operators

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`%`	Modulus (remainder)

Relational Operators

`<`	Less than
`<=`	Less than or equal to
`>`	Greater than
`>=`	Greater than or equal to
`==`	Equal to
`!=`	Not equal to

Logical Operators

`&&`	AND
`||`	OR
`!`	NOT

Increment/Decrement

`++`	Increment
`--`	Decrement

The `while` *Loop*

Form:

```
while (expression)
   statement;

while (expression)
{
   statement;
   statement;
}
```

Example:

```
while (x < 100)
   System.out.print(x++);

while (x < 100)
{
   System.out.print(x);
   x++;
}
```

The `do-while` *Loop*

Form:

```
do
   statement;
while (expression);

do
{
   statement;
   statement;
} while (expression);
```

Example:

```
do
   System.out.print(x++);
while (x < 100);

do
{
   System.out.print(x);
   x++;
}  while (x < 100);
```

Java™ Quick Reference (continued)

The `for` *Loop*

Form:

```
for (Initialization; Test; Update)
   statement;

for (Initialization; Test; Update)
{
     statement;
     statement;
}
```

Example:

```
for (int count = 0; count < 10; count++)
    System.out.print(count);

for (int count = 0; count < 10; count++)
{
    System.out.print("The value of count is ");
    System.out.println(count);
}
```

The `switch/case` *Statement*

Form:

```
switch (Expression)
{
   case Constant:
      statement(s);
      break;
   case Constant:
      statement(s);
      break;
   default :
      statement(s);
}
```

Example:

```
switch (choice)
{
    case 0 :
        System.out.println("You selected 0.");
        break;
    case 1 :
        System.out.println("You selected 1.");
        break;
    default :
        System.out.println("You did not select 0 or 1.");
}
```

To create a `Scanner` *object for reading keyboard input:*

```
Scanner keyboard = new Scanner(System.in);
```

For the `Scanner` *class, use this* `import` *statement:*

```
import java.util.Scanner;
```

`Scanner` *Class Methods for Reading Input*

Method	Use this method to. . .
`byte nextByte()`	Read a `byte`
`double nextDouble()`	Read a `double`
`float nextFloat()`	Read a `float`
`int nextInt()`	Read an `int`
`String nextLine()`	Read a `String`
`long nextLong()`	Read a `long`
`short nextShort()`	Read a `short`

Example Code using the `Scanner` Class to Read Keyboard Input:

```
// Create a Scanner object.
Scanner keyboard =
            new Scanner(System.in);

// Read a String from the keyboard.
String str;
str = keyboard.nextLine();

// Read an int from the keyboard.
int number;
number = keyboard.nextInt();

// Read a double from the keyboard.
double val;
val = keyboard.nextDouble();
```

Using `JOptionPane` *to Display a Message Dialog:*

```
JOptionPane.showMessageDialog(null,
                       "Hello World");
```

Using `JOptionPane` *to Display an Input Dialog:*

```
String name;
name = JOptionPane.showInputDialog("Enter " +
                          "your name.");
```

For `JOptionPane` use the following **`import`** statement:

```
import javax.swing.JOptionPane;
```

Wrapper Class Conversion Methods

`byte Byte.parseByte(String s)`
Converts a string to a `byte`.

`double Double.parseDouble(String s)`
Converts a string to a `double`.

`float Float.parseFloat(String s)`
Converts a string to a `float`.

`int Integer.parseInt(String s)`
Converts a string to an `int`.

`long Long.parseLong(String s)`
Converts a string to a `long`.

`short Short.parseShort(String s)`
Converts a string to a `short`.

Index

A
abstract classes and methods, 668–675
Abstract Windowing Toolkit (AWT)
  See also specific classes
  applets created with, 931–935
  classes, list of, 931
  class hierarchy, 843–844
  defined, 755, 756–757
  portability, 931–935
access
  package, 654–655
  sequential file, 735
  specification in UML diagrams, 345
accessor method, 344
access specifiers
  private, 332
  protected, 649–655
  public, 28, 332
accumulators, 221, 223, 431
action command, 783, 786
`ActionEvent` class
  `getActionCommand()`, 783–786
  `getSource()`, 783, 786–787
`ActionEvent` object, 782–783
action events, 772
`ActionListener` interface, 771–778
action listeners, handling events with, 758, 771–778
`actionPerformed()`, 771–772
actual parameters, 285
adapter classes, 961–966
`addActionListener()`, 777, 780–784, 785, 968
adding/inserting items in `ArrayList`, 477, 481–482
addition operator, 55
`addListSelectionListener()`, 847
address, 4
aggregation
  description of, 519–529
  security issues, 527–529
  in UML diagrams, 527
algorithms, 6
  array, 430–437
  binary search, 470–472
  binary search, recursive, 999–1002
  selection sort, 467–470
  sequential search, 451–453
ALU (arithmetic and logic unit), 3
AND (`&&`), 135, 136–138, 1039–1040
anonymous object, 765, 777
Apache Derby, 1016
API (application programmer interface), 33
  standard packages, 387
appearance, 898–900
`append()`, 586–587
appending data to files, 240–241, 735
`Applet` class, 931
  See also `AudioClip` interface
applets
  audio, playing, 971–974
  AWT, creating with, 931–935
  defined, 8–9, 911–912
  differences between GUI and, 923
  events, handling in, 926–930
  restrictions on, 913
  running, 924–926
  running with `appletviewer`, 925–926
  security, 913, 925
  Swing, creating with, 922–930
`appletviewer`, 925–926
application programmer interface. *See* API
applications
  defined, 911
  Java, 8–9
  playing audio in, 975
  software, 5
arcs, drawing, 937, 944–946
arguments, 34
  arrays as, 426–430
  command-line, 472–473
  formatting string, 172–173
  object references/arguments to methods, passing, 289–292
  objects as, passing, 364–366, 504–507
  other names for, 285
  parameter data type compatibility and, 285–286
  passing, by value, 288–289
  passing, to a method, 283–293
  passing multiple, 286–287
  variable length argument lists, 474–476
arithmetic and logic unit (ALU), 3
arithmetic operators
  `+` (addition), 55
  associativity of, 58
  `/` (division), 55, 56
  `%` (modulus), 55, 56
  `*` (multiplication), 55
  – (negation), 55
  precedence of operations, 57–59
  `+` (string concatenation), 40–41
  – (subtraction), 55
`ArrayList` class
  adding/inserting items, 477, 481–482
  autoboxing and unboxing, 604–605
  capacity, 483–484
  constructor, 483, 484
  defined, 476
  diamond operator for type inference, 484
  enhanced `for` loop with, 478–479
  `get()`, 477–478
  object, creating and using, 477–478
  removing items, 480–481
  set/replacing items, 482–483
  `size()`, 477
  `toString()`, 479–480
arrays
  accessing elements of, 409–410
  algorithms, 430–437, 467–472
  as arguments, 426–430

arrays (*continued*)
  averaging values in numeric, 432
  binary search algorithm, 470–472
  binary search algorithm, recursive, 999–1002
  bounds checking, 413–414
  comparing, 430–431
  content, inputting/outputting, 410–413
  copying, 424–426
  declaration notation, 416
  defined, 407
  elements, displaying, 417–419, 461
  enhanced for loop, 420–421
  errors, off-by-one, 414, 420
  files and, 442–443
  finding highest/lowest values in numeric, 432–433
  initialization, 415–416, 458–459
  invalid subscripts with, 413–414
  length field in, 420, 448, 459–460
  length of, 419–420
  of objects, 437–441, 448–451
  one-dimensional, 454
  partially filled, 441–442
  passing, to methods, 426–430, 463–465
  ragged, 465
  reference variables, reassigning, 423–424
  returning, from methods, 443–445
  selection sort algorithm, 467–470
  sequential search algorithm, 451–453
  size, user specified, 421–423
  size declarator, 408–409
  String, 445–448
  subscripts, 409, 413–414, 453
  summing values in numeric, 431–432, 461–463
  with three or more dimensions, 466
  two-dimensional, 454–465
assignment operators, 52
  combined, 63–64
assignment statements, 39, 52–53
associativity, 141–142
  of arithmetic operators, 58
@exception, 728
@param, 292–293
@return, 300–301
attributes, 20
audio, playing
  in an applet, 971–974
  in an application, 975
AudioClip interface
  getAudioClip(), 972
  getCodeBase(), 972
  getDocumentBase(), 972
  loop(), 972
  play(), 971, 972
  stop(), 972
autoboxing, 604–605
auto commit, 1099
AWT. *See* Abstract Windowing Toolkit

B
background color, 778–782
backslashes, 36, 37, 38, 241
backspace, 37
backups, external hard drives for, 5
bankAccount, 378–384
base case, 989
base class, 620
BASIC, 7, 8
Bell Laboratories, 7
binary digit, 4
binary files
  appending data to, 735
  DataInputStream class, 732–734
  DataOutputStream class, 730–732
  defined, 234, 729–730
  FileInputStream class, 732–734
  FileOutputStream class, 730–732, 735
  raw format, 730
  reading data from, 732–734
  writing data to, 730–732
binary numbers, 6
binary operators, 55
binary search algorithm, 470–472
  recursive, 999–1002
binding
  defined, 377
  dynamic, 664–666
  late, 665
bit, 4
blank lines, 28
block comments, 79
block of statements, 117, 201
body
  class, 29
  document, 915
  loop, 198
  method, 30, 275
boolean data type, 50
boolean expressions
  logical operators and, 135, 136
  relational operators and, 112
boolean value, returning, 304
BorderFactory class, list of methods, 816
BorderLayout manager, 788, 791–798
borders
  compound, 816
  defined, 815
  empty, 816–817
  etched, 816
  line, 816, 817
  lowered bevel, 816
  matte, 816
  methods for creating, 816
  objects, 815
  raised bevel, 816
  setBorder(), 815, 852
  titled, 816, 817
bounds checking, 413–414
braces, 29–31, 201
Brandi's Bagel House application, 818–830
breaks, creating text, 918–921
break statements, 153–154, 233
browsers, 913, 931
buffers
  file, 235
  keyboard, 90
buildMenuBar(), 887
buildTextMenu(), 887
Button class, 931
ButtonGroup class, 805
buttons, 756, 765
  See also JButton class; JButton component; Radio buttons
  displaying images in, 868–874
byte, 4
byte code, 8
byte data type, 45, 46
Byte.parseByte(), 96
Byte wrapper class, 96, 603

C
calling methods, 73, 276–280, 299–300
  hierarchical, 281–282
  superclass constructor, 632–639
call stack, 718
Car class, 393
case conversion, character, 567–569
case sensitive, 29, 43
case statement, 153–154
cast operators, 66–67, 286
catch block, 702
catch clause, 702
  using multiple, 708–716
  using one, 718–721
CDs (compact discs), 5
CellPhone class example, 356–360
central processing unit (CPU), 7, 12–13
  fetch/decode/execute cycle, 3–4
  parts/organization of, 3
  role of, 3

chains of inheritance, 655–660
Character class
  case conversion, 567–569
  description of, 562–569
  isDigit(), 562
  isLetter(), 562,
  isLetterOrDigit(), 562
  isLowerCase(), 562
  isSpaceChar(), 562
  isUpperCase(), 562
  isWhiteSpace(), 562
  toLowerCase(), 568
  toUpperCase(), 568
characters
  comparing, 118–119
  conversion, 164
  literals, 50–51
  reading, 88
  Unicode, 51–52, 118–119
  wrapping, 891
charAt(), 74, 586
char data type, 50–52, 562
CheckBox class, 931
check boxes, 756, 804, 810–815
  See also JCheckBox class
check box menu item, 880
  See also JCheckBoxMenuItem class
checked exceptions, 722–723
Cho Han game example, 366–376
circles, drawing, 944
  concentric, 994–996
classes
  See also specific classes
  accessor(), 344
  access specifiers, 332
  adapter, 961–966
  aggregation, 519–529
  base, 620
  body of, 29
  collaborations, 546–550
  constructors and, 352–356
  data hiding, 344–345
  definition, 28
  derived, 620
  finding, 388–391
  header, 28
  hierarchies of, 661, 843–844
  inner, 771
  instance fields and, 347–351,
    384–385, 497–498
  instance methods and, 334,
    347–351, 497–498
  instance of, 347–351
  interfaces, 675–686
  in Java API, 325–327
  layout of members, 346
  mutator method, 344
  names, 29, 44
  objects created from, 324–325
  peer, 757
  private access specifier, 332
  protected members, 649–654
  public access specifier, 332
  responsibilities, identifying,
    391–394
  static, 497–503
  type variables, 71–72, 328
  writing, step-by-step instructions,
    330–346
.class file extension, 28, 764
close(), 731, 733
COBOL, 7
code reuse, 274
collaboration, class, 546–550
color
  background and foreground,
    setting, 778–782
  changing background color of
    JFrame content pane, 782
  constants, list of, 778
  getColor(), 937
ColorCheckBoxWindow, 811–814
color chooser dialog box, 879
ColorWindow class, 778–782
columns, 1019–1022
combined assignment operators,
  63–64
combo boxes, 756, 862–868
  See also JComboBox class
command-line arguments, 472–473
commas
  in numeric literals, 47, 48
  separated value, 606
  separators, 168–170
comments
  block, 79
  defined, 77
  documentation, 79–81, 282,
    292–293, 300–301
  multi-line, 78–79
  single-line, 77–78
  slash marks, use of, 28, 31
commit, 1099
compact discs (CDs), 5
comparing arrays, 430–431
comparing string objects, 143–149
compiler, role of, 12–13
compiling programs, 14–15
components
  See also type of
  adding, to a window, 765–770
  borders, 815–817
  buttons, 756, 765, 868–874
  check boxes, 756, 810–815
  combo boxes, 756, 862–868
  defined, 755
  heavyweight, 757
  labels, 756, 765, 868–874
  lightweight, 757
  lists, 756, 846–862
  panels, 766–770
  radio buttons, 756, 804–810
  sliders, 756, 893–898
  text areas, 889–893
  text fields, 756, 765, 844–845
compound borders, 816
compound operators, 63
computer systems
  hardware, 2–5
  software, 5–6
concat(), 581
concatenation operator, 40–41
conditional expression, 150–152
conditional loops, 211
conditionally executed, 110, 117–118
conditional operator, 150–152
console
  output, 33
  window, 33
constants
  color, 778
  enum, 535
  EXIT_ON_CLOSE, 760–761
  HIDE_ON_CLOSE, 761
  MAX_VALUE, 604
  MIN_VALUE, 604
  named, 69–70
constructors, 175
  copy, 518–519
  default, 354–355, 638–639
  no-arg, 355, 638–639
  object creation and, 352–356
  overloading, 376–384, 534
  StringBuilder class, 585–586
  String class, 356
  StringTokenizer class, 596
  superclass, 629–631, 632–639
  in UML diagrams, 354
containers, GUI, 758
  adding layout manager to, 788
  nesting panels, 795–798
content panes, 766–770, 782
continue statements, 233
control characters, 36
control unit, 3
control variable, loop, 200
conversion
  character, 164
  data type, 65–68, 95–98, 285–286
copy (copying)
  arrays, 424–426
  constructors, 518–519
  deep, 527–529
  devices for, 5

copy (copying) (*continued*)
  objects, 516–518
  reference, 424, 516
  shallow, 527–529
count-controlled loops, 211, 218–220
counter variables, 212
countTokens(), 597
C programming language, 7, 8
C++ programming language, 7, 8
C# programming language, 7
CPU. *See* central processing unit
CRC cards, 549–550
createCompoundBorder(), 816
createEmptyBorder(), 816
createEtchedBorder(), 816
createLineBorder(), 816
createLoweredBevelBorder(), 816
createMatteBorder(), 816
createRaisedBevelBorder(), 816
createTitledBorder(), 816
.csv file, 606
currency symbols, 48
cursor navigation methods, 1063
Customer class, 392

D

data
  appending, to files, 240–241, 735
  hiding, 20, 344–345
  metadata, ResultSet, 1063–1066
  reading, from files, 241–242, 732–734
  stale, 345
  storage, 4–5
  writing, to files, 234–240, 730–732
database example (CoffeeDB)
  columns, getting, 1026–1027
  connecting to, 1016–1018
  CREATE and DROP TABLE statements, 1057–1060
  creating, 1016
  DELETE statement, JDBC, 1053–1056
  DELETE statement, SQL, 1048, 1053
  displaying tables, 1067–1077
  INSERT statement, JDBC, 1046–1047
  INSERT statement, SQL, 1044–1046
  joining data from multiple tables, 1080–1081
  keys, primary, 1021–1022
  LIKE operator, 1038–1039
  logical operators (and, or), 1039–1040
  mathematical functions, 1041–1043
  ORDER BY clause, 1040–1041
  passing SQL statements to DBMS, 1024–1034
  relational database, 1077–1099
  ResultSet object, 1024–1030
  rows, getting, 1025–1026
  SELECT statement, 1022–1023, 1030–1034, 1038
  tables, rows, and columns, 1019–1022
  UPDATE statement, JDBC, 1049–1052
  UPDATE statement, SQL, 1048–1049
  WHERE clause, 1034–1040, 1049, 1053, 1081
database management systems (DBMS)
  defined, 1013–1014
  Java DB, 1016
  JDBC (Java Database Connectivity), 1014–1015
  JDBC, creating a, 1060–1061
  packages, 5, 1015
  password-protected, 1018
  ResultSet, metadata, 1063–1066
  ResultSet, navigation methods, 1063
  ResultSet, scrollable, 1062–1063
  Structured Query Language (SQL), 1015
  transactions, 1099–1101
DataInputStream class, 732–734
DataOutputStream class, 730–732
data types
  boolean, 50
  byte, 45, 46
  char, 50–52
  conversion between, 65–68, 95–98, 285–286
  double, 45, 47
  enumerated, 535–543
  float, 45, 47
  floating-point, 47–49
  int, 45, 46–47
  long, 45, 46
  mixing, in expressions, 67–68
  numeric data, 45, 603–605
  primitive, 44–52, 65–68
  ranking, 65–66
  short, 45, 46
  SQL, 1021
  in UML diagrams, 345–346
DB2, 1015
debugging GUI applications, 831–835
DecimalFormat class, 174–181
decimal notation, 49
decision structures
  conditional operator, 150–152
  defined, 110
  if-else-if statement, 129–134
  if-else statement, 119–121
  if statement, 109–119
  logical operators
  nested if statements, 122–129
  switch statement, 152–160
declarations
  array notation, 416
  variable, 39, 149–150
decorations, window, 760
decrement operator (++), 193–196
  postfix versus prefix modes, 196
deep copy, 527–529
default constructor, 354–355, 638–639
default error message, retrieving, 705–708
default exception handler, 700
default statement, 153–154, 155
definition, class, 28
delete(), 588–589
deleteCharAt(), 588–589
DELETE statement
  JDBC, 1053–1056
  SQL, 1048, 1053
delimiters, 236, 599–600
depth of recursion, 987
derived class, 620
deserialization, 742
diagnostic messages, 831–832
dialog boxes, 92–98
diamond operator for type inference, 484
dice with objects, example, 360–364
digital versatile discs (DVDs), 5
Dimension class, 886–887
direct recursion, 992
disk drives, 4–5
divide-and-conquer approach, 273–273
division
  integer, 57
  operator, 55, 56
  remainder of, 56
doClick(), 810, 814–815
documentation comments, 79–81, 282, 292–293, 300–301, 728
document body, 915
document head, 914
document structure tags, 914–916
dot, 34
double data type, 45, 47
Double.parseDouble(), 96
Double wrapper class, 96, 603
do-while loops, 208–210
drawArc(), 937, 944–946
DrawBoxes, 961–966
drawCircles(), 996

drawing
- arcs, 937, 944–946
- circles, 944
- concentric circles, 994–996
- lines, 937, 940
- ovals, 937, 942–944
- on panels, 951–956
- polygons, 938, 946
- rectangles, 937, 940–942
- strings, 938, 948–950
- XY coordinate system, 936

drawLine(), 937, 940
drawOval(), 937, 942–944
drawPolygon(), 938, 946
drawRect(), 937, 940–942
drawString(), 938, 948–950
DriverManager.getConnection(), 1016–1018
DVDs (digital versatile discs), 5
dynamic binding, 664–666

E

editor, text, 12
else, trailing, 132–133
EmbeddedMain class, 763–765
empty borders, 816–817
encapsulation, 20
endsWith(), 570–573
enhanced for loop, 420–421, 478–479
E notation, 49
enter key, 88
entity relationship diagrams, 1079
enum constants, 535
enumerated data types, 535–543
- switching on, 542–543

EOFException class, 701
equals(), 144–146, 513–516, 662
Error class, 722
error messages
- cannot resolve symbol, 76
- default error message, retrieving, 705–708

errors
- common, to avoid, 99–100, 181–182, 259, 310–311, 394–395, 485, 550, 610, 687, 746, 836, 900–901, 976, 1007, 1101
- compiler, 297, 298
- exception handler used for recovering from, 711–714
- logical, 17
- off-by-one, 414, 420
- syntax, 12
- using trailing else to catch, 132

escape sequences, 36–37
event-driven programming, 758
event listener, handling, 771–778
events
- action, 772
- in applets, 926–930
- defined, 771
- firing, 771
- handling, 771–778
- item, 811
- listener, 758, 771–778
- mouse, 957–967
- object, 771
- source, 771

Excel spreadsheet, 606
Exception class, description of, 700–701
exception handler
- defined, 700
- recovering from errors using, 711–714

exceptions
- catch clause, using one, 718–721
- catch clauses, using multiple, 708–716
- checked, 722–723
- classes, 700–701
- creating your own classes for, 725–728
- default error message, retrieving, 705–708
- default handler, 700
- defined, 237, 699, 700
- EOFException class, 701
- Error class, 722
- Exception class, 700–701
- FileNotFoundException class, 701
- finally clause, 716–717
- handling, 699–723
- handling multiple, 708–716
- IllegalArgumentException class, 724
- InputMismatchException class, 720
- IOException class, 237, 239, 241, 245, 310, 701, 720
- MalformedURLException class, 975
- methods and throwing, 310
- NumberFormatException class, 715–716, 719–720
- polymorphism and, 708
- RuntimeException class, 701, 722, 724
- stack trace, 718–719
- throwing, 310, 723–729
- throws clause, 237, 245
- try statement, 701–705, 715–716
- unchecked, 722–723

executable files, 12
exists(), 249–252
EXIT_ON_CLOSE constant, 760–761
explicit import statement, 386
expressions
- initialization, 211–212, 215–218
- regular, 601
- return statement, 298
- test, 211–212
- update, 211–212, 215, 217–218

extending classes from JPanel class, 818–830
extending the JFrame class, 761–763

F

factorial(), 989–992
fetch/decode/execute cycle, 3–4
Fibonacci series, 996–998
fields
- instance, 347–351, 384–385, 497–498
- interface, 680
- length, 420, 448, 459–460
- object, 323
- static, 498–500
- text, 756, 765
- text, read-only, 844–845

file chooser dialog box, 877–878
- *See also* JFileChooser class

File class, 234
- exists(), 249–252

FileInputStream class, 732–734
FileNotFoundException class, 701
FileOutputStream class, 730–732, 735
files
- appending data to, 240–241, 735
- arrays and, 442–443
- binary, 234, 729–735
- buffers and, 235
- closing, 234
- comma separated value format, 606
- detecting end of, 245–246
- exceptions, 237
- existence, checking for, 249–252
- FileWriter class, 234, 240–241
- input/output, 234–252
- location of, specifying, 241
- names, 14
- opening, 234
- pointer, 738–741
- PrintWriter class, 234–240
- random access, 735–745
- reading, 234
- reading data from, 241–242, 732–734
- reading lines from, using nextLine(), 242–245
- reading primitive values from, 246–248
- read position of, 243

files (*continued*)
- Scanner class, 234, 241–248
- sequential access to, 735
- source, 12
- text, 234
- throws clause, 237, 245
- writing data to, 234–240, 730–732

FileWriter class, 234, 240–241
fillArc(), 937, 944–946
fillOval(), 937, 940–944
fillPolygon(), 938, 946–948
fillRect(), 937
finalize(), 546
final key word, 69–70
finally block, 717
finally clause, 716–717
flags, 118, 168–172
flash memory, 5
float data type, 45, 47
floating-point data types, 47–49
floating-point literals, 48–49
Float.parseFloat(), 96
Float wrapper class, 96, 603
floppy disk drives, 5
FlowLayout manager, 788–791
fonts, 892–893
- setFont(), 889, 938

foreground color, 778–782
foreign key, 1078
for loops
- ArrayList class and, 478–479
- count-controlled, 211, 218–220
- counter variable, 212
- defined, 211
- enhanced, 420–421, 478–479
- header, 211
- initialization expression, 211–212, 215–218
- pretest, 214
- test expression, 211–212
- update expression, 211–212, 215, 217–218
- user-controlled, 216–217

formal parameters, 285
formatting
- pattern, 175
- string arguments, 172–173
- System.out.printf(), 162–173
- text formatting tags, 916–918

FORTRAN, 7
Frame class, 931
frames, 758
- *See also* JFrame class

functional decomposition, 306

G

garbage collection, 544–546
- finalize(), 546

generalization and specialization, 619–620
get(), 477–478
getActionCommand(), 783–786
getArea(), 341–344
getAudioClip(), 972
getChars(), 577, 578–580, 586
getCodeBase(), 972
getColor(), 937
getColumnCount(), 1064
getColumnDisplaySize(), 1064
getColumnName(), 1064
getColumnTypeName(), 1064
getContentPanel(), 782
getDelay(), 968
getDocumentBase(), 972
getDouble(), 1027
getInt(), 1027
getLength(), 338–341
getSelectedFile(), 878
getSelectedIndex(), 848, 858, 863–867
getSelectedItem(), 863–867
getSelectedValue(), 848, 858
getSource(), 783, 786–787
getString(), 1027
getTableName(), 1064
getters, 344
getText(), 776, 890
getValue(), 895–898
getWidth(), 338–341
getX(), 958
getY(), 958
Gosling, James, 8
graphical user interface (GUI)
- appearance, 898–900
- components, 755–756
- components, adding to a window, 765–770
- creating windows, 758–787
- creation of, 756–757
- debugging, 831–835
- defined, 755
- differences between applets and, 923
- event-driven programming, 758
- layout managers, 787–803
- main(), 763–765

Graphics class
- drawArc(), 937, 944–946
- drawCircles(), 996
- drawLine(), 937, 940
- drawOval(), 937, 942–944
- drawPolygon(), 938, 946
- drawRect(), 937, 940–942
- drawString(), 938, 948–950
- fillArc(), 937, 944–946
- fillOval(), 937, 940–944
- fillPolygon(), 938, 946–948
- fillRect(), 937
- getColor(), 937
- objects, 936–950
- setColor(), 937
- setFont(), 938

greatest common divisor (GCD), finding, 998–999
Green Team, 8
GridLayout manager, 788, 798–803
GUI. *See* graphical user interface

H

hard drives, 4–5
hardware
- central processing unit (CPU), 3–4, 7, 12–13
- components of, 2
- defined, 2
- input devices, 5
- memory, 4
- output devices, 5
- secondary storage, 4–5

hash code, 662
hasMoreTokens(), 597
headers
- class, 28
- loop, 198, 211
- method, 29–30, 237, 245, 275–276
- while loop, 198

heavyweight components, 757
hexadecimal numbers, 662–663
HIDE_ON_CLOSE constant, 761
hiding, data, 20, 344–345
hierarchical method calls, 281–282
hierarchies, class, 661, 843–844
HotJava, 8
HTML. *See* Hypertext Markup Language
hypertext, defined, 913
Hypertext Markup Language (HTML), 8, 79
- breaks, creating, 918–921
- defined, 913
- document body, 915
- document head, 914
- document structure tags, 914–916
- limitations of, 911
- links, inserting, 921–922
- text formatting tags, 916–918

Hypertext Markup Language (HTML), tags
- <a href=. . .>, 921–922
- <applet></applet>, 924
- <b></b>, 918
- <body></body>, 915
- <br />, 918–921
- <center></center>, 916–917

  - `<head></head>`, 914
  - `<h1></h1>` through `<h6></h6>`, 916
  - `<hr />`, 919–920
  - `<html></html>`, 914
  - `<i></i>`, 918
  - `<p />`, 919
  - `<title></title>`, 914

I

- IDEs (integrated development environments), 15
- identifiers, 9, 11, 42–43
- `if-else-if` statements
  - compared to nested decision structure, 133–134
  - description of, 129–134
  - trailing `else` clause, 132–133
- `if-else` statements, 119–121
- `if` statements
  - braces used with, 118
  - characters, comparing, 118–119
  - description of, 109–119
  - flags, 118
  - flowcharting, 110–111
  - multiple statements, 117–118
  - nested, 122–129
  - programming style and, 116–118
  - relational operators to form conditions, 111–112
- `IllegalArgumentException` class, 724
- `ImageIcon` class, 868–870
  - constructor, 869
- images, displaying, 868–874
- immutable objects, 290, 528
- implementing an interface, 676–680, 772
- `import` statement
  - explicit, 386
  - `Scanner` class and, 87–88
  - wildcard, 387
- increment operator (++), 193–196
  - postfix versus prefix modes, 196
- indentation, 82
- `indexOf()`, 574, 586
- indirect recursion, 992
- infinite loops, 200–201
- inheritance
  - base class, 620
  - chains of, 655–660
  - defined, 619
  - derived class, 620
  - does not work in reverse, 631
  - extend `JFrame` class, 761–763
  - generalization and specialization, 619–620
  - "is a" relationship, 620–628, 666–667
  - subclasses and, 620
  - superclass, 620
  - superclass constructors, 629–631, 632–639
  - superclass methods, overriding, 640–645
  - UML diagrams, 628–629
- initialization
  - array, 415–416
  - array, two-dimensional, 458–459
  - variable, 52–53
- inner class, 771
- input
  - devices, 5
  - dialogs, displaying, 93
  - file, 234–252
  - keyboard, 84–91
  - validation, 204–207
- `InputMismatchException` class, 720
- `insert()`, 587–588
- inserting
  - items in `ArrayList`, 477, 481–482
  - links, 921–922
- `INSERT` statement
  - JDBC, 1046–1047
  - SQL, 1044–1046
- instance
  - class, 71, 347–351
  - fields, 347–351, 384–385, 497–498
  - methods, 334, 347–351, 497–498
  - variables, 351
- `instanceof` operator, 667
- `int` data type, 39, 45, 46–47
- `Integer.parseInt()`, 96
- integers
  - data types, 45, 46–47
  - division, 57
  - literals, 42, 47
  - mixed, 67–68
- `IntegerWrapper` class, 96, 603
- integrated development environments (IDEs), 15
- interfaces
  - *See also* graphical user interface
  - defined, 675
  - fields, 680
  - implementing, 676–680, 772
  - key word, 675
  - multiple, 680
  - polymorphism and, 681–686
  - serializing, 745
  - UML diagram, 680–681
- Internet Explorer, 79, 931
- `IOException` class, 237, 239, 241, 245, 310, 701, 720
- "is a" relationship, 620–628, 666–667
- `isDigit()`, 562
- `isLetter()`, 562,
- `isLetterOrDigit()`, 562
- `isLowerCase()`, 562
- `isRunning()`, 968
- `isSelected()`, 809–810, 811–814
- *`isSpaceChar()`*, 562
- `isUpperCase()`, 562
- `isWhiteSpace()`, 562
- item event, 811
- item listener, 811
- iterations, loop, 199–200

J

- `JApplet` class, 923
- Java
  - applets, 8–9, 911–913, 922–935
  - applications, 8–9
  - compiler, 12–13
  - defined, 7
  - editions, 14
  - history of, 8
  - parts of a simple program, 27–31
  - portability, 13
  - security, 9
  - 7 edition, 484, 719–721
  - virtual machine, 12–13, 544, 665, 701, 702
- `java.applet`, 387
- `java.awt`, 387, 758
  - class hierarchy, 843–844
- `java.awt.event`, 772, 776, 874, 957, 958, 965
- `java` command, 14–15, 28, 473, 831
- Java Database Connectivity. *See* JDBC
- Java DB, 1016
- Java Development Kit (JDK), 14
- `javadoc`, documentation and, 79–81, 282
- Java Enterprise Edition (EE), 14
- `.java` file extension, 12, 28
- Java Foundation Classes (JFC), 755, 756
- `java.io`, 387, 701, 742, 878
- `java.lang`, 387
- Java Micro Edition (ME), 14
- `java.net`, 387, 975
- JavaScript, 7
- `java.security`, 387
- `java.sql`, 387, 1018
- Java Standard Edition, (SE) 14
- `java.text`, 387
- `java.util`, 387, 595
- Java Virtual Machine (JVM), 12–13, 544, 665, 701, 702
- `javax.swing`, 387, 758
  - class hierarchy, 843–844
- `javax.swing.event`, 847

JButton class, 765
  addActionListener(), 777, 780–784, 785
  constructor, 769, 870
  displaying images, 870–874
  setActionCommand(), 786
  setIcon(), 869
JButton component, action events and, 773
JCheckBox class
  constructor, 810–811
  doClick(), 814–815
  events, responding to, 811
  isSelected(), 811–814
JCheckBoxMenuItem class, 881
JColorChooser class, 879
JComboBox class
  constructor, 862
  defined, 862
  editable, 867–868
  events, responding to, 863
  getSelectedIndex(), 863–867
  getSelectedItem(), 863–867
  retrieving selected item, 863–867
  setEditable(), 867–868
JComponent class, 889
JDBC (Java Database Connectivity), 1014–1015
  DELETE statement, JDBC, 1053–1056
  INSERT statement, 1046–1047
  UPDATE statement, 1049–1052
JDK (Java Development Kit), 14
JFC (Java Foundation Classes), 755, 756
JFileChooser class
  constructor, 877
  defined, 877
  getSelectedFile(), 878
  showOpenDialog(), 877–878
  showSaveDialog(), 878
JFrame class
  content panes and panels, 766–770, 782
  EXIT_ON_CLOSE constant, 760–761
  getContentPane(), 782
  HIDE_ON_CLOSE constant, 761
  inheritance to extend, 761–763
  instance, 760
  paint(), overriding, 938, 950
  repaint(), 950–951
  setDefaultCloseOperation(), 760
  setLayout(), 788
  setSize(), 760
  setTitle(), 760
  setVisible(), 761
JLabel class, 765
  constructor, 869
  displaying images, 868–874
  setIcon(), 869
JList class
  adding items to, 857
  addListSelectionListener(), 847
  border, placing around, 852
  constructor, 846
  defined, 846
  events, responding to, 847
  getSelectedIndex(), 848, 858
  getSelectedValue(), 848, 858
  multiple interval selection mode, 858–862
  multiple selection lists, 857–862
  retrieving selected item, 848–851
  scroll bar, adding to, 852–857
  selection modes, 846–847
  setBorder(), 852
  setListData(), 857
  setVisibleRowCount(), 852–853
  single interval selection mode, 857–858
JMenuBar class, 881
JMenu class, 881
JMenuItem class, 881
JOptionPane class, 758
  dialog boxes displayed using, 92–98
  showInputDialog(), 93
  showMessageDialog(), 92–93
JPanel class
  content panes and panels, 766–770
  extending classes from, 818–830
  nesting, 795–798
  paintComponent(), 951–956
  setBorder(), 814, 815, 852
JRadioButton class
  constructor, 804
  doClick(), 810
  events, responding to, 805–809
  grouping, 805
  isSelected(), 809–810
  radio buttons, creating, 756, 804–810
JRadioButtonMenuItem class, 881
JScrollPane class, 857, 890–891
  constructor, 853
JSlider class
  constructor, 894
  defined, 893
  events, responding to, 895
  getValue(), 895–898
  tick mark spacing, 894–895
JTable class, 1067–1077
JTextArea class
  constructor, 889–890
  defined, 889
  getText(), 890
  scroll bars, 890–891
  setLineWrap(), 891
  setText(), 890
  setWrapStyleWord(), 891
JTextField class, 765
  constructor, 769
  getText(), 776
  read-only text fields, 844–845
  setEditable(), 844–845
JVM. *See* Java Virtual Machine

K
keyboard
  buffer, 90
  input from, 84–91
  mnemonics, 874–876
KeyEvent class, 874–875
keys
  foreign, 1078
  primary, 1021–1022
key words, 9, 10, 42
  super, 632–638, 648

L
Label class, 931
labels, 756, 765
  See also JLabel class
  displaying images in, 868–874
languages, programming
  common elements, 9–15
  description of, 6–9
lastIndexOf(), 574, 575–577, 586
late binding, 665
layout managers
  adding, to a container, 788
  BorderLayout, 788, 791–798
  defined, 787–788
  FlowLayout, 788–791
  GridLayout, 788, 798–803
leading whitespace, 582
leading zeros, 170–171
left-justifying numbers, 171–172
length(), 73, 74, 586
length field, array, 420, 448, 459–460
lexicographical comparison, 147
lifetime, 296
lightweight components, 757
LIKE operator, 1038–1039
line borders, 816, 817
lines
  blank, 28
  drawing, 937, 940
  program, 33
  reading, from files using nextLine(), 242–245
  wrapping, 891
links, inserting, 921–922

Linux, 5
listener
  action/event, 758, 771–778
  item, 811
  mouse, 957
  mouse motion, 957
lists
  See also JList class
  defined, 756, 846
ListSelectionListener interface, 847
ListSelectionModel class, 847
literals
  character, 50–51
  floating-point, 48–49
  how to use, 39–44
  integer, 42, 47
  string, 30, 42
local variables, 76, 295–297
logical errors, 17
logical operators
  && (AND), 135, 136–138, 1039–1040
  associativity of, 141–142
  boolean expressions using, 136
  ! (NOT), 135, 141
  numeric ranges with, 142–143
  || (OR), 135, 139–140, 1039–1040
  precedence of, 141–142
long data type, 45, 46
Long.parseLong(), 96
Long wrapper class, 96, 603
look and feel (appearance), 898–900
loop(), 972
loops
  body, 198
  break statement in, 233
  choice of, 233
  conditional, 211
  continue statement in, 233
  control variable, 200
  count-controlled, 211, 218–220
  defined, 197
  do-while, 208–210
  enhanced for, 420–421, 478–479
  for, 211–220
  header, 198, 211
  infinite, 200–201
  iterations, 199–200
  nested, 225–233
  posttest, 208
  pretest, 200, 214
  running totals and, 220–223
  sentinel values and, 223–225
  user-controlled, 210, 216–217
  while, 197–210
lowercase, converting characters to, 562

M
machine language, 6–7
Mac OS, 5
main()
  body of, 30
  in GUI class, 763–765
  header, 30
MalformedURLException class, 975
markup language, 914
Math class
  .PI named constant, 70
  .pow, 62
  .sqrt, 62
mathematical functions, SQL, 1041–1043
MAX_VALUE constant, 604
memory
  flash, 5
  random-access, 4, 12
  secondary, 4–5
  storing characters in, 52
menus
  bar, 880
  components of, 880–881
  constructing, 881–888
  defined, 880
  items, 880
  sub, 881
message(), 987
message dialogs, displaying, 92–93
metadata, ResultSet, 1063–1066
methods
  See also specific methods
  abstract, 668–675
  accessor, 344
  arguments, passing, 283–293, 504–507
  binding, 377
  body, 275
  calling, 73, 276–280
  defined, 20, 273
  defining a void, 275–276
  detail section, 80
  divide-and-conquer approach, 273–274
  documentation comments, 282, 292–293
  headers, 29–30, 237, 245, 275–276
  hierarchical calls, 281–282
  how to use, 273–274, 301–304
  instance, 334, 347–351, 497–498
  javadoc comments with, 282
  local variables, 76, 295–297
  modifiers, 276
  mutator, 344
  name, 276
  object references/arguments to methods, passing, 289–292, 504–507
  overloading, 376–384
  overriding, 640–645
  parameter variables and, 283, 285–286
  passing arrays to, 426–430, 463–465
  private, 332
  problem solving with, 306–311
  public, 276, 332
  returning a boolean value, 304
  returning arrays from, 443–445
  returning objects from, 507–509
  return statement, 298
  return type and, 298
  signature of, 377
  static, 276, 501–503
  summary section, 80
  synchronized, 594
  throw exceptions, 310
  value-returning, 273, 274, 275, 297–301
  variable-length argument lists and, 474–476
  void, 273, 274–276
Microsoft, 7, 931, 1015
minimum field width, 166–168
minus sign (negation), 55
MIN_VALUE constant, 604
mnemonics, 874–876
modifiers, method, 276
modulus operator, 55, 56
mouse
  adapter classes, 961–966
  events, 957–967
  listener, 957
  motion listener, 957
MouseAdapter class, 961–966
mouseClicked(), 957
mouseDragged(), 958
mouseEntered(), 957
MouseEvent class, 957–958
mouseExited(), 957
MouseListener interface, methods, 957
MouseMotionAdapter class, 965
MouseMotionListener interface, methods, 958
mouseMoved(), 958
mousePressed(), 957
mouseReleased(), 957
multi-catch, 720–721
multi-line comments, 78–79
multiplication operator, 55
multithreaded application, 594
mutator method, 344
MySQL, 1015

N

named constants, 69–70
names
  class, 29, 44
  identifiers, 42–43
  method, 276
  variable, 43
narrowing conversion, 66
navigation methods, 1063
Navigator, 8
negation operator, 55
nested if statements
  description of, 122–129
  flowcharting, 122
  multiple, 126–129
nested loops, 225–233
Netscape Navigator, 8
new key word, 84, 407, 410, 415, 446, 458, 509, 586, 777
nextByte(), 86, 246
nextDouble(), 86, 246, 254
nextFloat(), 86, 246, 254
nextInt(), 86, 246, 254
nextInt(int n), 254
nextLine(), 86, 242–245
nextLong(), 86, 246, 254
nextShort(), 86, 246
nextToken(), 597
no-arg constructor, 355, 638–639
NOT (!) operator, 135, 141
null references, avoiding, 529–532
null statements, 117
NumberFormatException class, 715–716, 719–720
numbers, converting strings to, 95–98
numeric data, data types for, 45
numeric data types, wrapper classes for, 603–605
numeric ranges, checking with logical operators, 142–143

O

Oak, 8
Object class
  description of, 661–663
  equals(), 662
  toString(), 662
ObjectInputStream class, 742
object-oriented design
  class collaboration, 546–550
  classes, finding, 388–391
  class responsibilities, identifying, 391–394
  CRC cards, 549–550
  problem domain, 388
object-oriented programming (OOP)
  attributes in, 20
  data hiding in, 20, 344–345
  defined, 19–21
  encapsulation in, 20
  methods in, 20
ObjectOutputStream class, 742
objects
  aggregation, 519–529
  anonymous, 765, 777
  as arguments/references, passing, 289–292, 364–366, 504–507
  ArrayList, creating and using, 477–478
  arrays of, 437–441, 448–451
  comparing string, 143–149
  constructors, 352–356, 518–519
  copy constructors, 518–519
  copying, 516–518
  created from classes, 70–71
  creating, with DecimalFormat, 174–180
  creating string, 72–73
  default constructor, 354–355
  defined, 323
  deserialization, 742
  examples of, 324
  fields, 323
  graphic, 936–950
  immutable, 290, 528
  out, 33–34
  primitive variables versus, 327–330
  returning, from a method, 507–509
  returning reference to, 304–306
  serialization, 741–745
  state, 509
  timer, 967–971
off-by-one errors, 414, 420
one-dimensional arrays, 454
OOP. *See* object-oriented programming
operands, 52
operating systems, 5
operators
  = (assignment), 52
  += (assignment), 64
  -= (assignment), 64
  *= (assignment), 64
  /= (assignment), 64
  %= (assignment), 64
  binary, 55
  cast, 66–67, 286
  combined/compound assignment, 63–64
  conditional, 150–152
  – (decrement), 193–196
  defined, 9, 11
  <> (diamond), 484
  ++ (increment), 193–196
  instanceof, 667
  LIKE, 1038–1039
  ternary, 55, 150
  unary, 55
operators, arithmetic
  + (addition), 55
  associativity of, 58
  / (division), 55, 56
  % (modulus), 55, 56
  * (multiplication), 55
  – (negation), 55
  precedence of operations, 57–59
  + (string concatenation), 40–41
  – (subtraction), 55
operators, logical
  && (AND), 135, 136–138, 1039–1040
  associativity of, 141–142
  boolean expressions using, 136
  ! (NOT), 135, 141
  numeric ranges with, 142–143
  || (OR), 135, 139–140, 1039–1040
  precedence of, 141–142
operators, relational
  == (equal to), 111–112, 430–431, 1034
  > (greater than), 111–112, 1034
  >= (greater than or equal to), 111–112, 1034
  < (less than), 111–112, 1034
  <= (less than or equal to), 111–112, 1034
  <> (not equal to), 1034
  != (not equal to), 111–112
optical devices, 5
Oracle/Sun Microsystems, 14, 1015
ORDER BY clause, 1040–1041
ordinal value, 537
OR (||) operator, 135, 139–140, 1039–1040
out object, 33–34
output
  devices, 5, 33
  file, 234–252
ovals, drawing, 937, 942–944
overloading
  methods and constructors, 376–384, 534
  overriding versus, 645–648
overriding
  overloading versus, 645–648
  paint(), 938, 950
  preventing, 648
  superclass methods, 640–645

P
package access, 654–655
packages, import statement and, 386–387
paint(), overriding, 938, 950
paintComponent(), 951–956
Panel class, 931
panels
  See also JPanel class
  description of, 766–770
  drawing on, 951–956
  nesting, 795–798
parameter variables (parameters), 283
  @param, 292–293
  data type compatibility, 285–286
  initializing local variables with, 296–297
  list, 286
  other names for, 285
  scope, 286
  in UML diagrams, 345–346
  vararg, 474–476
parentheses, 31, 59, 151, 276
parse methods, 96, 603
partially filled arrays, 441–442
Pascal, 7
passed by value, 288–289
password-protected database, 1018
peer classes, 757
percentages, calculating, 59–62
Perl, 7
PHP, 7
PI constant, 70
pixel, 760
play(), 971, 972
plug-ins, 931
polygons, drawing, 938, 946
polymorphism
  defined, 664
  description of, 663–668
  dynamic binding, 664–666
  exceptions and, 708
  interfaces, 681–686
portability, 13, 931–935
postfix mode, 194–196
posttest loops, 208
precedence, operator, 57–59, 141–142
precision, 165–166
preferred size, 886
prefix mode, 194–196
pretest loops, 200, 214
primary keys, 1021–1022
priming read, 205, 225
primitive data types, 44–52
  conversion between, 65–68
  versus objects, 327–330
primitive type variables, 71
primitive values from files, reading, 246–248
primitive variables versus objects, 327–330
print(), 35–37
  patterns with nested loops, 228–233
  PrintWriter class and, 235, 236
printf(), 162–173
println(), 33–37
  PrintWriter class and, 235–236
PrintWriter class
  print(), 235, 236
  println(), 235–236
  writing data to files using, 234–240
private access specifier, 332
problem domain, 388
problem solving
  extending classes from JPanel class, 818–830
  methods and, 306–311
  recursion and, 988–992
procedural programming, 19–20
procedure, 19
  stored, 1100–1101
programmer-defined names, 9, 11
programmers, role of, 2
programming
  common elements, 9–15
  defensive, 543
  event-driven, 758
  languages, 6–9
  object-oriented, 19–21
  process, 16–19
  style, 82–83, 116, 202
programs
  compiling running, 14–15
  defined, 6–7
  design with count-controlled loops, 211, 218–220
  design with while loop, 202–203
  parts of simple, 27–31
protected access specifier, 649–655
protected members, 649–654
pseudocode, 19
public access specifier, 28, 29, 332
public modifier, 276
punctuation, 9, 11
Python, 7

Q
quotation marks, 31, 37, 40, 41

R
radio button menu item, 880–881
  See also JRadioButtonMenuItem class
radio buttons
  See also JRadioButton class
  description of, 756, 804–810
ragged arrays, 465
RAM (random-access memory), 4, 12
RandomAccessFile class
  constructor, 736
  file pointer, 738–741
  format, 736–737
  reading and writing with, 737–738
  seek(), 738–739
random access files, 735–745
random-access memory (RAM), 4, 12
Random class
  how to use, 253–258
  nextDouble(), 254
  nextFloat(), 254
  nextInt(), 254
  nextInt(int n), 254
  nextLong(), 254
random numbers
  applications, 253
  generating, 253–258
range, numeric, 142–143
rangeSum(), 993–994
raw binary format, 730
readBoolean(), 733
readByte(), 733
readChar(), 733
readDouble(), 733
readFloat(), 733
reading
  data from files, 241–242, 732–734
  lines from files using nextLine(), 242–245
  primitive values from files, 246–248
  RandomAccessFile class and, 737–738
  strings, 734–735
reading(), 733
readLong(), 733
readObject(), 742
read-only text fields, 844–845
read position, 243
readShort(), 733
readUTF(), 735
Rectangle class, writing example, 330–346
rectangles, drawing, 937, 940–942
recursion
  base case and, 989
  binarySearch(), 999–1002
  concentric circles, drawing, 994–996
  defined, 985–988
  depth, 987
  direct, 992
  factorial(), 989–992

recursion (*continued*)
  Fibonacci series, 996–998
  greatest common divisor, finding, 998–999
  indirect, 992
  methods, 987–988
  problem solving with, 988–992
  recursive case, 989
  summing array elements with, 993–994
  Towers of Hanoi, 1002–1007
recursiveBinarySearch(), 999–1002
recursive case, 989
reference copy, 424, 516
reference variables, 72, 174–175
  objects and, 328–329
  reassigning array, 423–424
  this, 532–534
  uninitialized, 354
referential integrity, 1079
regionMatches(), 571, 573
registering event listeners, 773
regular expression, 601
relational database, 1077–1099
relational operators
  == (equal to), 111–112, 430–431, 1034
  > (greater than), 111–112, 1034
  >= (greater than or equal to), 111–112, 1034
  < (less than), 111–112, 1034
  <= (less than or equal to), 111–112, 1034
  <> (not equal to), 1034
  != (not equal to), 111–112
relationships
  "is a," 620–628, 666–667
  whole-part, 523
removing items from ArrayList, 480–481
repaint(), 950–951
replace(), 581–582, 588
replacing items in ArrayList, 482–483
reserved words, 9, 10
responsibilities, identifying class, 391–394, 591–592
ResultSet object, 1024–1030
  metadata, 1063–1066
  navigation methods, 1063
  scrollable, 1062–1063
return
  arrays from methods, 443–445
  @return, 300–301
  boolean value, 304
  objects from methods, 507–509
  statements, 298
  type, 298
rollback, 1099
rows, 1019–1022
Ruby, 7
running programs, 14–15
running totals, 220–223
RuntimeException class, 701, 723, 724

S
Scanner class
  characters, reading, 88
  close(), 242
  import statement, 87–88
  mixing calls, 88–91
  nextByte(), 86, 246
  nextDouble(), 86, 246
  nextFloat(), 86, 246
  nextInt(), 86, 246
  nextLine(), 86, 242–245
  nextLong(), 86, 246
  nextShort(), 86, 246
  reading files with, 234, 241–248
  reading keyboard input, 84–91
scientific notation, 49
scope
  coming into, 150
  instance fields, 384–385
  leaving, 150
  parameter variable, 286
  variable, 75–77, 149–150
scrollable ResultSet, 1062–1063
scroll bar
  adding to, 852–857
  JTextArea class and, 890–891
SDK (Software Development Kit), 14
search algorithm
  binary search, 470–472
  binary search, recursive, 999–1002
  sequential search, 451–453
secondary memory, 4–5
security
  aggregate classes and, 527–529
  applet, 913, 925
  Java, 9
seek(), 738–739
selection sort algorithm, 467–470
SELECT statement, 1022–1023, 1030–1034, 1038
self-documenting program, 43
semicolons, 30–31, 117, 276
sentinel values, 223–225
separator bar (menus), 881
sequence structure, 110
sequential file access, 735
sequential search algorithm, 451–453
Serializable interface, 745
serialization, 741–745
ServiceQuote class, 393–394
set(), ArrayList, 482–483
setActionCommand(), 786
setBorder(), 814, 815, 852
setCharAt(), 588–589
setDefaultCloseOperation(), 760
setDelay(), 968
setEditable(), 844–845, 867–868
setFont(), 889, 938
setIcon(), 869
setLayout(), 788
setLength(), 332–336
setLineWrap(), 891
setListData(), 857
setLookAndFeel(), 899
setPreferredSize(), 886–887
setSelectionMode(), 847
setSize(), 760
setters, 344
setText(), 890
setTitle(), 760
setToolTipText(), 876
setVisible(), 761
setVisibleRowCount(), 852–853
setWidth(), 337–338
setWrapStyleWord(), 891
shadowing, 385, 533–534
shallow copy, 527–529
shapes, drawing. *See* drawing
short-circuit evaluation, 136–137
short data type, 45, 46
Short.parseShort(), 96
Short wrapper class, 96, 603
showDialog(), 879
showInputDialog(), 93
showMessageDialog(), 92–93
showOpenDialog(), 877–878
showSaveDialog(), 878
signature, 377
simple Java program, 27–31
single-line comments, 77–78
size, array
  declarator, 408–409
  user specified, 421–423
slash marks, 28, 31
sliders, 756, 893–898
  See also JSlider class
software
  application, 5
  categories, 5
  defined, 2
  engineering, 18–19
Software Development Kit (SDK), 14
sorting algorithms, selection sort, 467–470
source code, 12
source file, 12
spaces, displaying, 36
specialization, 619–620

splash screens, 830–831
split(), 600–602
spreadsheets, 5
SQLException, 1018
SQL (Structured Query Language), 1015
  data types, 1021
  DELETE statement, SQL, 1048, 1053
  INSERT statement, 1044–1046
  mathematical functions, 1041–1043
  passing SQL statements to DBMS, 1024–1034
  relational operators, 1034
  SELECT statement, 1022–1023
  stored procedures, 1100–1101
  UPDATE and DELETE statements, 1048–1049
stack trace, 718–719
stale data, avoiding, 345
standard input device, 84
standard output device, 33
star seven device (*7), 8
start(), 968
startsWith(), *570–573*
state, object, 509
statements, 11
  assignment, 39
static class members, 497–503
static fields, 498–500
static methods, 276, 501–503
static modifier, 276
Stock class, 510–513
stop(), 968, 972
storage devices, types of, 4–5
stored procedures, 1100–1101
StringBuffer class, 594
StringBuilder class
  append(), 586–587
  charAt(), 586
  constructors, 585–586
  defined, 584
  delete(), 588–589
  deleteCharAt(), 588–589
  getChars(), 586
  indexOf(), 586
  insert(), 587–588
  lastIndexOf(), 586
  length(), 586
  replace(), 588
  setCharAt(), 588–589
  substring(), 586
  toString(), 589
String class
  arguments, formatting, 172–173
  arrays of, 445–448
  charAt(), 74
  compareTo(), 146–148
  compareToIgnoreCase(), 148
  concat(), 581
  constructor, 356
  defined, 70
  endsWith(), 570–573
  equals(), 144–146
  getChars(), 577, 578–580
  how to use, 70–75
  indexOf(), 574
  lastIndexOf(), 574, 575–577
  length(), 73, 74
  objects, comparing, 143–149
  objects, creating, 72–73
  objects of, passed as argument, 289–292
  reading, 734–735
  regionMatches(), 571, 573
  replace(), 581–582
  returning a String object from a method, 507–509
  split(), 600–602
  startsWith(), 570–573
  substring(), 577–578
  substrings, extracting, 577–580
  substrings, searching for, *570–577*
  toCharArray(), 578–580
  toLowerCase(), 74
  toUpperCase(), 74
  trim(), 581, 582
  valueOf(), 582–583
  variable declaration, 71
  writing, 734–735
strings
  arguments, formatting, 172–173
  concatenation operator, 40–41
  converting, to numbers, 95–98
  defined, 70
  drawing, in graphics, 938, 948–950
  literals, 30, 42
  tokenizing, 595–602
StringTokenizer class
  constructors, 596
  countTokens(), 597
  defined, 595
  extracting tokens, 596–599
  hasMoreTokens(), 597
  multiple delimiters, 599–600
  nextToken(), 597
  trimming string, 600
strongly typed language, 48, 65
Structured Query Language. *See* SQL
style, programming, 82–83, 116, 202
subclass, 620
submenu, 881
subscripts, 409, 453
substring(), 577–578, 586
substrings
  extracting, 577–580
  searching for, 570–577
subtraction operator, 55
Sun Microsystems, 7, 8, 14
SunWorld, 8
superclass, 620
  constructors, 629–631, 632–639
  methods, overriding, 640–645
super key word, 632–638, 648
Swing class, 755, 756–757
  applets, creating with, 922–930
  class hierarchy, 843–844
  creating windows, 758–787
switches, 4
switch statements
  break, 153–155
  case, 152–153
  default, 153–155
  description of, 152–160
  enumerated data types, 542–543
symbols
  \ (backslash), 36, 38, 241
  { (brace, left/opening brace), 29–30, 31
  } (brace, right/closing), 29, 31
  /***/ (comments, documentation), 79–81
  /**/ (comments, multi-line), 78–79
  // (comments, single-line), 77–78
  currency, 43, 48
  -- (decrement), 193–196
  <> (diamond), 484
  / (forward slash), 38, 241
  ++ (increment), 193–196
  ( ) (parentheses), 31, 59, 151, 276
  % (percent), 59
  " " (quotations, double), 31, 40, 41
  ' ' (quotations, single), 50
  ; (semicolons), 30–31, 117, 276
  // (slash, double), 31, 77–78
symbols, escape sequences
  \\ (backslash, double), 37, 241
  \b (backspace), 37
  \t (horizontal tab), 37
  \n (new line), 36–37
  \" (quote, double), 37
  \' (quote, single), 37
  \r (return), 37
symbols, operators
  = (assignment), 52
  += (assignment), 64
  -= (assignment), 64
  *= (assignment), 64
  /= (assignment), 64
  %= (assignment), 64
symbols operators (arithmetic)
  + (addition), 55
  associativity of, 58

symbols operators (arithmetic) (*continued*)
/ (division), 55, 56
% (modulus), 55, 56
* (multiplication), 55
– (negation), 55
precedence of operations, 57–59
+ (string concatenation), 40–41
– (subtraction), 55
symbols, operators (logical)
&& (AND), 135, 136–138, 1039–1040
associativity of, 141–142
boolean expressions using, 136
! (NOT), 135, 141
numeric ranges with, 142–143
|| (OR), 135, 139–140, 1039–1040
precedence of, 141–142
symbols, operators (relational)
== (equal to), 111–112, 1034
> (greater than), 111–112, 1034
>= (greater than or equal to), 111–112, 1034
< (less than), 111–112, 1034
<= (less than or equal to), 111–112, 1034
<> (not equal to), 1034
!= (not equal to), 111–112
synchronized methods, 594
syntax, 9
syntax errors, 12
System class
exit method, 94–95, 761
out object, 33–34
System.exit(), 94–95, 761
System.out.printf(), 162–173
System.out.println(), 831–835

T
tables, 1019–1022
displaying, 1067–1077
tags
@exception, 728
@param, 292–293
@return, 300–301
document structure, 914–916
text formatting, 916–918
telephone numbers, formatting and unformatting example, 589–593
TempConverterWindow class, 898–900
events, handling in applets, 926–930
ternary operators, 55, 150
TestScoreReader class example, 606–610
text areas, 889–893
See also JTextArea class
text breaks, creating, 918–921
text editor, 12
TextField class, 931
text fields
See also JTextField class
defined, 756, 765
read-only, 844–845
text files, 234
text formatting tags, 916–918
this reference variable, 532
calling overloaded constructor with, 534
overcoming shadowing with, 533–534
thread, 95
three-dimensional arrays, 466
throwing exceptions, 310, 723–729
throws clause, 237, 245, 724
throw statement, 723–729
tick mark spacing, 894–895
Timer class
addActionListener(), 968
constructor, 967
getDelay(), 968
isRunning(), 968
objects, 967–971
setDelay(), 968
start(), 968
stop(), 968
time sharing, 5
titled borders, 816, 817
toBinaryString(), 603–604
toCharArray(), 578–580
toHexString(), 603–604
tokenizing strings, 595–602
tokens, 595
toLowerCase(), 74, 568
toOctalString(), 603–604
tool tips, 874, 876
toString(), 479–480, 509–513, 589, 603, 662
toUpperCase(), 74, 568
toURL(), 975
Towers of Hanoi, 1002–1007
trailing else clause, 132–133
trailing whitespace, 582
transactions, 1099–1101
trim(), 581, 582
truncated, 57, 66
truth tables
AND operator, 136
NOT operator, 141
OR operator, 139
try block, 702
try statement
catch block, 702
catch clause, 702
description of, 701–705
exception handling, 715–716
finally block, 717
finally clause, 716–717
try block, 702
two-dimensional arrays
declaring, 454–458
defined, 454
displaying elements in, 461
initializing, 458–459
length field in, 459–460
passing, to methods, 463–465
ragged, 465
summing columns in, 462–463
summing elements in, 461–462
summing rows in, 462
typefaces, 892–893

U
UIManager class, 899
unary operators, 55
unboxing, 604–605
unchecked exceptions, 722–723
Unicode, 51–52, 118–119
Unicode Text Format (UTF)-8 encoding, 734–735
Unified Modeling Language (UML) diagrams
abstract classes and methods, 674
access specification, 345
aggregation, 527
BankAccount class, 378
Car class, 393
chains of inheritance, 660
class design, 331
constructors, 354
Customer class, 392
data type notation, 345–346
inheritance, 628–629
interfaces, 680–681
parameter variable notation, 345–346
protected members, 654
ServiceQuote class, 394
Stock class, 510
uniform resource identifier (URI), 921, 975
uninitialized reference variable, 354
UNIX, 5
UPDATE statement
JDBC, 1049–1052
SQL, 1048–1049
uppercase, converting characters to, 562
URI (uniform resource identifier), 921, 975
URI class, 975
USB drives, 5
user-controlled loops, 210, 216–217
UTF (Unicode Text Format)-8 encoding, 734–735

V
validation, input, 204–207
valueChanged(), 847
valueOf(), 582–583
value-returning method, 273, 274, 275, 297–301
  boolean value, returning, 304
vararg parameter, 474–476
variable length argument lists, 474–476
variables
  accumulators, 221, 223, 431
  assignment, 52–53
  class type, 71–72, 328
  counter, 212
  declaration, 39, 149–150
  defined, 11–12
  formatter, 174–175
  holding one value at a time, 53–54
  how to use, 39–44
  initialization, 52–53
  instance, 351
  lifetime, 296
  local, 76, 295–297
  names, 43
  parameter, 283, 285–286
  primitive, 327–330
  primitive type, 71
  reference, 72, 174–175, 328–329, 354, 423–424, 532–534
  scope of, 75–77, 149–150, 286
  shadowing, 385, 533–534
  this, 532–534
vertex, 946
Visual Basic, 7
void methods, 273, 274–276

W
Web page, 911–912
Web *server*, 911–912
WHERE clause, 1034–1040, 1049, 1053, 1081
while loop
  body of, 198
  braces, 201
  control variable, 200
  defined, 197
  do-, 208–210
  header of, 198
  infinite, 200–201
  for input validation, 204–207
  iteration, 199–200
  as pretest loop, 200
  program design with, 202–203
  programming style, 202
whitespace, 80, 582
whole-part relationship, 523
widening conversion, 66
wildcard import statement, 387
window decorations, 760
windows, creating, 758–787
Windows, operating system, 5
word processing, 5
word wrapping, 891
wrapper classes
  autoboxing, 604–605
  Byte, 96, 603
  Character, 562–569
  defined, 561
  Double, 96, 603
  Float, 96, 603
  Integer, 96, 603
  Long, 96, 603
  MAX_VALUE constants, 604
  MIN_VALUE constants, 604
  numeric data types, 603–605
  parse methods, 96, 603
  Short, 96, 603
  substrings, 570–580
  toBinaryString(), 603–604
  toHexString(), 603–604
  toOctalString(), 603–604
  toString(), 603
  unboxing, 604–605
writeBoolean(), 731
writeByte(), 731
writeChar(), 731
writeDouble(), 731
writeFloat(), 731
writing
  classes, step-by-step instructions, 330–346
  data to files, 234–240, 730–732
  RandomAccessFile class and, 737–738
  strings, 734–735
writeInt(), 731
writeLong(), 731
writeObject(), 742
writeShort(), 731
writeUTF(), 731

X
XY coordinate system, 936

Z
zeros, padding numbers with leading, 170–171

Credits

Figure 1-7 jGRASP screenshots used by permission of James H. Cross II.

Figures 2-10 © 1995, 2011 Oracle and/or its affiliates. All rights reserved.

Figures 2-2, 2-14, 2-15, 2-16, 2-17, 2-18, 2-19, 3-4, 3-5, 3-11, 3-12, 3-13, 3-16, 3-17, 3-18, 3-19, 3-20, 3-21, 4-5, 4-12, 4-13, 4-14, 4-15, 4-16, 4-20, 5-5, 5-6, 5-17, 5-19, 6-18, 6-28, 7-11, 8-2, 8-5, 9-1, 9-2, 9-3, 9-4, 9-6, 10-3, 10-4, 10-7, 10-12, 11-2, 11-3, 11-6, 11-7, 11-8, 11-9, 11-10, 12-1, 12-2, 12-3, 12-5, 12-6, 12-9, 12-10, 12-11, 12-13, 12-14, 12-15, 12-17, 12-18, 12-20, 12-22, 12-23, 12-26, 12-27, 12-28, 12-29, 12-34, 12-35, 12-36, 12-37, 12-39, 13-2 through 13-32, 14-14, 14-18, 14-19, 14-21, 14-23, 14-25, 14-26, 14-27, 14-28, 14-29, 14-30, 14-31, 14-32, 15-4, 15-6, 16-17, 16-18, 16-20, 16-21, 16-24, 16-25, 16-26 Screenshots © 2011 by Oracle Corporation. Reprinted with permission.

Figure 9-8 Used with permission from Microsoft.

Figures 14-2 through 14-13, 14-15, 14-17 Screenshots © 2011 by Microsoft Corporation. Reprinted with permission. MICROSOFT AND/OR ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THE INFORMATION CONTAINED IN THE DOCUMENTS AND RELATED GRAPHICS PUBLISHED AS PART OF THE SERVICES FOR ANY PURPOSE. ALL SUCH DOCUMENTS AND RELATED GRAPHICS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. MICROSOFT AND/OR ITS RESPECTIVE SUPPLIERS HEREBY DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS INFORMATION, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL MICROSOFT AND/OR ITS RESPECTIVE SUPPLIERS BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF INFORMATION AVAILABLE FROM THE SERVICES.

THE DOCUMENTS AND RELATED GRAPHICS CONTAINED HEREIN COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN. MICROSOFT AND/OR ITS RESPECTIVE SUPPLIERS MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED HEREIN AT ANY TIME. PARTIAL SCREEN SHOTS MAY BE VIEWED IN FULL WITHIN THE SOFTWARE VERSION SPECIFIED.