# Contents at a Glance

**Introduction**

**Part I: Getting Started**
* Hour 1: Becoming a Programmer 3
* 2 Writing Your First Program 13
* 3 Vacationing in Java 25
* 4 Understanding How Java Programs Work 39

**Part II: Learning the Basics of Programming**
* 5 Storing and Changing Information in a Program 49
* 6 Using Strings to Communicate 65
* 7 Using Conditional Tests to Make Decisions 79
* 8 Repeating an Action with Loops 95

**Part III: Working with Information in New Ways**
* 9 Storing Information with Arrays 107
* 10 Creating Your First Object 121
* 11 Describing What Your Object Is Like 137
* 12 Making the Most of Existing Objects 155

**Part IV: Programming a Graphical User Interface**
* 13 Building a Simple User Interface 169
* 14 Laying Out a User Interface 187
* 15 Responding to User Input 201
* 16 Building a Complex User Interface 219

**Part V: Moving into Advanced Topics**
* 17 Creating Interactive Web Programs 235
* 18 Handling Errors in a Program 249
* 19 Creating a Threaded Program 265
* 20 Reading and Writing Files 283

**Part VI: Writing Internet Applications**
* 21 Reading and Writing XML Data 299
* 22 Creating Web Services with JAX-WS 313
* 23 Creating Java2D Graphics 327
* 24 Writing Android Apps 343

**Part VII: Appendixes**
* A Using the NetBeans Integrated Development Environment 373
* B Where to Go from Here: Java Resources 381
* C This Book’s Website 387
* D Setting Up an Android Development Environment 389

Index 397
### Table of Contents

INTRODUCTION 1

**PART I: Getting Started**

HOUR 1: Becoming a Programmer
- Choosing a Language 4
- Telling the Computer What to Do 5
- How Programs Work 7
- When Programs Don’t Work 8
- Choosing a Java Programming Tool 8
- Installing a Java Development Tool 9

HOUR 2: Writing Your First Program
- What You Need to Write Programs 13
- Creating the Saluton Program 14
- Beginning the Program 14
- Storing Information in a Variable 17
- Saving the Finished Product 18
- Compiling the Program into a Class File 19
- Fixing Errors 19
- Running a Java Program 20

HOUR 3: Vacationing in Java
- First Stop: Oracle 25
- Going to School with Java 27
- Lunch in JavaWorld 29
- Watching the Skies at NASA 31
- Getting Down to Business 32
- Stopping by Java Boutique for Directions 33
- Running Java on Your Phone 35

HOUR 4: Understanding How Java Programs Work
- Creating an Application 39
- Sending Arguments to Applications 41
- Creating an Applet 42

PART II: Learning the Basics of Programming

HOUR 5: Storing and Changing Information in a Program
- Statements and Expressions 49
- Assigning Variable Types 50
- Naming Your Variables 54
- Storing Information in Variables 54
- All About Operators 55
- Using Expressions 59

HOUR 6: Using Strings to Communicate
- Storing Text in Strings 65
- Displaying Strings in Programs 66
- Using Special Characters in Strings 67
- Pasting Strings Together 68
- Using Other Variables with Strings 68
- Advanced String Handling 70
- Presenting Credits 72

HOUR 7: Using Conditional Tests to Make Decisions
- if Statements 79
- if-else Statements 83
- switch Statements 84
- The Conditional Operator 86
- Watching the Clock 87

HOUR 8: Repeating an Action with Loops
- for Loops 95
- while Loops 98
- do-while Loops 99
- Exiting a Loop 100
- Naming a Loop 101
- Testing Your Computer Speed 102
PART III: Working with Information in New Ways

HOUR 9: Storing Information with Arrays
Creating Arrays ........................................ 108
Using Arrays ........................................ 109
Multidimensional Arrays .................................. 111
Sorting an Array ........................................ 111
Counting Characters in Strings .................... 113

HOUR 10: Creating Your First Object
How Object-Oriented Programming Works 121
Objects in Action ........................................ 122
What Objects Are ........................................ 124
Understanding Inheritance .......................... 125
Building an Inheritance Hierarchy ................ 125
Converting Objects and Simple Variables ......... 127
Creating an Object .................................... 132

HOUR 11: Describing What Your Object Is Like
Creating Variables ..................................... 137
Creating Class Variables .............................. 139
Creating Behavior with Methods .................. 140
Putting One Class Inside Another ................. 146
Using the this Keyword ................................ 147
Using Class Methods and Variables ................ 148

HOUR 12: Making the Most of Existing Objects
The Power of Inheritance .............................. 155
Establishing Inheritance ................................ 157
Working with Existing Objects ..................... 159
Storing Objects of the Same Class in Vectors 160
Creating a Subclass ..................................... 164

PART IV: Programming a Graphical User Interface

HOUR 13: Building a Simple User Interface
Swing and the Abstract Windowing Toolkit 169
Using Components ..................................... 170
Creating Your Own Component ...................... 180

HOUR 14: Laying Out a User Interface
Using Layout Managers ............................... 187
Laying Out an Application ............................. 192

HOUR 15: Responding to User Input
Getting Your Programs to Listen ................... 201
Setting Up Components to Be Heard ............... 202
Handling User Events ................................ 202
Completing a Graphical Application ............... 207

HOUR 16: Building a Complex User Interface
Scroll Panes ............................................ 219
Sliders .................................................. 222
Change Listeners ...................................... 223
Using Image Icons and Toolbars .................. 227

PART V: Moving into Advanced Topics

HOUR 17: Creating Interactive Web Programs
Standard Applet Methods ............................. 235
Putting an Applet on a Web Page .................. 238
Creating an Applet ..................................... 239
Sending Parameters from a Web Page ............... 242
Handling Parameters in an Applet ................ 243
Using the Object Tag .................................. 245

HOUR 18: Handling Errors in a Program
Exceptions ............................................. 249
Throwing Exceptions ................................. 256
Throwing and Catching Exceptions ................ 258

HOUR 19: Creating a Threaded Program
Threads ................................................. 265
Working with Threads ................................. 270
Starting with init() .................................... 272
Catching Errors as You Set Up URLs ............... 272
Handling Screen Updates in the paint() Method .. 273
Starting the Thread .................................... 274
Handling Mouse Clicks ............................... 276
Displaying Revolving Links ......................... 276
About the Author

Rogers Cadenhead is a writer, computer programmer, and web developer who has written more than 20 books on Internet-related topics, including *Sams Teach Yourself Java in 21 Days*. He maintains the Drudge Retort and other websites that receive more than 20 million visits a year. This book’s official website is at www.java24hours.com.

Dedication

With this edition of the book, I’d like to break from tradition and cheat my family and friends out of praise, because frankly it’s going to their heads. I dedicate this book to James Gosling, Mike Sheridan, Kim Polese, Bill Joy, and the others who launched the first version of this amazing programming language back in 1995. A language I was once surprised to see running on a web page is now running apps on millions of Android phones around the world—a testimonial to the visionary work you did at the late Sun Microsystems. Long may the purple reign!

Acknowledgments

To the folks at Sams—especially Mark Taber, Songlin Qiu, Tonya Simpson, Charlotte Kughen, and Boris Minkin. No author can produce a book like this on his own. Their excellent work will give me plenty to take credit for later.

To my wife, Mary, and my sons, Max, Eli, and Sam. Although our family has not fulfilled my dream of becoming death-defying high-wire trapeze acrobats, I’m the world’s proudest husband and father in a household of acrophobics.

Reader Acknowledgments

I’d also like to thank readers who have sent helpful comments about corrections, typos, and suggested improvements to the book. The list includes Brian Converse, Philip B. Copp III, Wallace Edwards, M.B. Ellis, Kevin Foad, Adam Grigsby, Mark Hardy, Kelly Hoke, Donovan Kelorii, Russel Loski, Jason Saredy, Mike Savage, Peter Schrier, Gene Wines, Jim Yates, and others who shall remain nameless because they helped me improve the book before I started this list.
We Want to Hear from You!

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we’re doing right, what we could do better, what areas you’d like to see us publish in, and any other words of wisdom you’re willing to pass our way.

You can email or write me directly to let me know what you did or didn’t like about this book—as well as what we can do to make our books stronger.

*Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.*

When you write, please be sure to include this book’s title and author as well as your name and phone or email address. I will carefully review your comments and share them with the author and editors who worked on the book.

E-mail: feedback@samspublishing.com

Mail: Mark Taber  
Executive Editor  
Sams Publishing  
800 East 96th Street  
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
Introduction

As the author of computer books, I spend a lot of time lurking in the computer section of bookstores, observing the behavior of readers while I’m pretending to read the latest issue of In Touch Weekly magazine.

Because of my research, I’ve learned that if you have picked up this book and turned to the introduction, I have only 12 more seconds before you put it down and head to the coffee bar for a double-tall-decaf-skim-with-two-shots-of-vanilla-hold-the-whip latte.

So I’ll keep this brief: Computer programming with Java is easier than it looks. I’m not supposed to tell you that because thousands of programmers have used their Java skills to get high-paying jobs in software development, web application programming, and mobile app creation. The last thing any programmer wants is for the boss to know that anyone who has persistence and a little free time can learn this language, the most popular programming language in use today. By working your way through each of the one-hour tutorials in Sams Teach Yourself Java in 24 Hours, you’ll be able to learn Java programming quickly.

Anyone can learn how to write computer programs—even if they can’t program a DVR. Java is one of the best programming languages to learn because it’s a useful, powerful, modern technology that’s embraced by thousands of programmers around the world.

This book is aimed at nonprogrammers, new programmers who hated learning the subject, and experienced programmers who want to quickly get up to speed with Java. It uses Java 7, the version of the language just released.

Java is an enormously popular programming language because of the things it makes possible. You can create programs that feature a graphical user interface, design software that makes the most of the Internet, read XML data, create a game that runs on an Android cell phone, and more.
This book teaches Java programming from the ground up. It introduces the concepts in English instead of jargon with step-by-step examples of working programs you will create. Spend 24 hours with this book and you’ll be writing your own Java programs, confident in your ability to use the language and learn more about it. You also will have skills that are becoming increasingly important—such as network computing, graphical user interface design, and object-oriented programming.

These terms might not mean much to you now. In fact, they’re probably the kind of thing that makes programming seem intimidating and difficult. However, if you can use a computer to balance your checkbook, or create a photo album on Facebook, you can write computer programs by reading *Sams Teach Yourself Java in 24 Hours*.

At this point, if you would rather have coffee than Java, please reshelved this book with the front cover facing outward on an endcap near a lot of the store’s foot traffic.
Before you venture further into Java programming, it’s worthwhile to learn more about the language and see what programmers are doing with it today. Though Java has outgrown its origins as a language focused on web browser programs, you can still find some interesting examples of how Java is used on the Web.

During this hour, we take a look at sites that feature Java programs and talk about the history and development of the language.

To go on this vacation, you need a web browser that has been set up to run Java programs.

Load your browser of choice, put on your best batik shirt, and get ready to take a vacation. You won’t be leaving your house, and you won’t experience the simpler pleasures of tourism, such as reckless cab drivers, exotic food, exotic locals, exotic locals with food, and so on. Look on the bright side though: no traveler’s check hassles, no passports, and no Montezuma’s revenge.

First Stop: Oracle

The Java vacation begins at www.java.com, a site created by Oracle, the company that owns the Java language.

A Java program that runs as part of a web page is called an applet. Applets are placed on pages like other elements of a page. A markup language called HTML defines where the program should be displayed, how big it is, and what the program does when it runs. Java also enhances the Web in two other ways: Desktop programs written in Java can be launched from a web browser, and Java servlets are run by web servers to deliver web applications.
Oracle’s Java division leads the development of the Java language and related software. The Java in Action section of Java.com showcases how Java is being used on websites, Android phones, and other platforms. Millions of devices run programs written with Java. Figure 3.1 shows RuneScape, a massively multiplayer online game powered by Java. You can play the game for free by using any web browser to visit www.runescape.com.

Java.com provides a place to learn about how Java is being used. Oracle also offers a more technically oriented website for Java programmers at http://www.oracle.com/technetwork/java. This site is the place to find the latest released versions of NetBeans and the Java Development Kit along with other programming resources.

A Brief History of Java

Bill Joy, one of the executives at Sun Microsystems when the company created Java, called the language “the end result of 15 years of work to produce a better, more reliable way to write computer programs.” Java’s creation was a little more complicated than that.

Java was developed in 1990 by James Gosling as a language that would serve as the brains for smart appliances (interactive TVs, omniscient ovens, SkyNet military satellites that enslave mankind, and so on). Gosling was unhappy with the results he was getting by writing programs with a programming language called C++. In a burst of inspiration, he holed up in his office and wrote a new language to better suit his needs.
Gosling named his new language Oak after a tree he could see from his office window. The language was part of his company’s strategy to make a fortune when interactive TV became a multimillion-dollar industry. That still hasn’t happened today (though Netflix, TiVo, and others are making a game attempt), but something completely different took place for Gosling’s new language. Just as Oak was about to be scrapped, the Web became popular.

In a fortuitous circumstance, many qualities that made Gosling’s language good on its appliance project made it suitable for adaptation to the Web. His team devised a way for programs to be run safely from web pages and a catchy new name was chosen to accompany the language’s new purpose: Java.

Although Java can be used for many other things, the Web provided the showcase it needed. When the language rose to prominence, you had to be in solitary confinement or a long-term orbital mission to avoid hearing about it.

There have been eight major releases of the Java language:

- **Fall 1995:** Java 1.0—The original release
- **Spring 1997:** Java 1.1—An upgrade that improved support for graphical user interfaces
- **Summer 1998:** Java 2 version 1.2—A huge expansion, making the language a general-purpose programming language
- **Fall 2000:** Java 2 version 1.3—A release for enhanced multimedia
- **Spring 2002:** Java 2 version 1.4—An upgrade of Internet support, XML capabilities, and text processing
- **Spring 2004:** Java 2 version 5—A release offering greater reliability and automatic data conversion
- **Winter 2006:** Java 6—A upgrade with a built-in database and web services support
- **Summer 2011:** Java 7—The current release, which adds new core language improvements, memory management improvements, and the Nimbus graphical user interface

**NOTE**

You might have heard that Java is an acronym that stands for Just Another Vague Acronym. You also might have heard that it was named for the Gosling’s love of coffee. The story behind Java’s naming contains no secret messages or declarations of liquid love. Java was chosen as the name for the same reason that comedian Jerry Seinfeld likes to say the word salsa: It sounds cool.

---

**Going to School with Java**

The Web includes numerous resources for educators and schoolchildren. Because Java programs can offer a more interactive experience than standard web pages, some programmers have used the language to write learning programs for the Internet.
For one such example, visit http://www.cs.ubc.ca/~van/sssjava to access a ski jump simulator created by Michiel van de Panne, a computer science professor at the University of British Columbia. The program uses Java to demonstrate physics-based animation as a skier tries several different slopes and jumps. The motion of the skier is controlled by moving a mouse one of eight directions, each of which affects the success of a jump. Figure 3.2 shows one run of the program right before my virtual skier met a gruesome end.

Numerous educational programs are available for many different operating systems, but one thing that makes this program stand out is its availability. The simulator is run directly from a web page. No special installation is needed, and, unlike most desktop software, it isn’t limited to a particular operating system. You can run Java programs on any computer that has a Java Virtual Machine (JVM).

The JVM loaded by a browser is the same one used to run the Saluton program during Hour 2, “Writing Your First Program.” A browser’s JVM only can run Java programs that are set up to run on web pages and cannot handle programs set up to run elsewhere, such as in a file folder.

The first browsers to support Java included a built-in JVM. Today, browsers support Java by relying on the Java Plug-in, a JVM that works as a browser enhancement.
A Java program, such as the ski-jump simulator, does not have to be written for a specific operating system. Because operating systems like Windows also are called platforms, this advantage is called platform independence. Java was created to work on multiple systems. Originally, Java’s developers believed it needed to be multiplatform because it would be used on a variety of appliances and other electronic devices.

Users can run the programs you write with Java on a variety of systems without requiring any extra work from you. Under the right circumstances, Java can remove the need to create specific versions of a program for different operating systems and devices.

**Lunch in JavaWorld**

After working up an appetite on the slopes, take a lunch break with *JavaWorld*, an online magazine for Java programmers. Visit www.javaworld.com.

*JavaWorld* offers how-to articles, news stories, and research centers on hot areas of Java development. One of the advantages of the publication’s web format is that it can display functional Java programs in conjunction with articles. Figure 3.3 shows a Java poetry magnet board that accompanies a tutorial explaining how it is written.

![A JavaWorld how-to article on how to create a poetry magnet board includes a working example of the program.](image)

**NOTE**

*JavaWorld* occasionally moves things around, but at the time of this writing, you can go directly to the poetry magnet board tutorial at www.cadenhead.org/poetry. If that page is unavailable, use the site’s search engine to look for the word “poetry.”

*JavaWorld* publishes articles and commentary about the language and its development. One issue that has been hotly debated since Java’s introduction is whether the language is secure.
Security is important because of the way Java programs work when they are placed on a web page. The Java programs you have tried during this hour were downloaded to your computer. When the program was finished downloading, it ran on your computer.

Unless you know a whole lot of people, most web pages you visit are published by strangers. In terms of security, running their programs isn’t a lot different than letting the general public come over and borrow your computer. If the Java language did not have safeguards to prevent abuse, its programs could introduce viruses onto your system, delete files, play the collected works of Justin Bieber, and do other unspeakable things. Java includes several different kinds of security to make sure that its programs are safe when run from web pages.

The main security is provided by restrictions on Java programs running over the Web:

- No program can open, read, write, or delete files on the user’s system.
- No program can run other programs on the user’s system.
- All windows created by the program are identified clearly as Java windows.
- Programs cannot make connections to websites other than the one from which they came.
- All programs are verified to make sure that nothing was modified after they were compiled.

Although there are no guarantees, the language has been proven to have enough safeguards to be usable over the Web.

The Java language also offers a more flexible security policy for programs that run in a browser. You can designate some companies and programmers as trusted developers, which enables their Java programs to run in your browser without the restrictions that normally would be in place.

This system of trust is established through the use of signed applets that have digital signatures, files that clearly identify the author of a Java program. These signatures are created in collaboration with independent verification groups such as VeriSign.

If you ever have authorized a program to run in a browser such as Internet Explorer or Google Chrome, you have worked with a similar system of trust and identity verification.
Applets can still be useful today, but over the years other technology, such as Flash, Silverlight, and HTML5, have been employed for web page–based programs. Java is more commonly encountered on mobile apps, server programs, and desktop software.

**Watching the Skies at NASA**

The first afternoon stop on the Java tour is a trip to NASA, a U.S. government agency that makes extensive use of Java. One of the most popular examples is SkyWatch, an applet that helps stargazers keep an eye out for orbiting satellites. Load it in your browser by visiting www.cadenhead.org/nasa; you are forwarded automatically to NASA’s SkyWatch site.

SkyWatch superimposes the current location and path of eight different satellites—which you can add or drop from view—over a globe of the world. The applet running in Figure 3.4 shows the SEASAT-1 satellite making a patch from the Bootes constellation to the Hercules constellation.

![NASA’s SkyWatch applet monitors the location and path of orbiting satellites, a boon to metal bird-watchers.](image)

The applet redraws the position of each tracked satellite as it runs. This kind of real-time update is possible because the Java language is multithreaded. *Multithreading* is a way for the computer to do more than one thing at the same time. One part of a program takes care of one task, another part takes care of a different task, and the two parts can pay no attention to each other. Each part of a program in this example is called a *thread*. 
In a program such as SkyWatch, each satellite could run in its own thread. If you use an operating system such as Windows 7, you’re using a type of this behavior when you run more than one program at the same time. If you’re at work playing Desktop Tower Defense in one window while running a company sales report in another window and making a long-distance call to a friend, congratulate yourself—you’re multithreading!

**Getting Down to Business**

At this point in your travels, you might have the impression that Java is primarily of use to space buffs, atrocious poets, and terrible skiers. The next stop on our trip shows an example of Java getting down to business.

Direct your web browser to the JTicker website at www.jticker.com.

The publisher of JTicker, a company called Stock Applets, develops Java programs that display business news headlines and stock quotes for use on other websites. Figure 3.5 shows a demo of its scrolling stock ticker.

Unlike other stock analysis programs that require the installation of software on the computers of each employee who needs access, the use of Java enables customers of Stock Applets to make the programs available to anyone with a web browser. All employees have to do is access the company’s website.

![FIGURE 3.5](Image)

Java programs from Stock Applets report stock market prices.

You can think of a program like this stock ticker applet in several different ways. One is to think of a program as an object—something that exists in
the world, takes up space, and has certain things it can do. Object-oriented programming (OOP), which Java uses (read more in Hour 10, “Creating Your First Object”), is a way of creating computer programs as a group of objects. Each object handles a specific job and knows how to speak to other objects. For example, a stock ticker program could be set up as the following group of objects:

- A quote object, which represents an individual stock quote
- A portfolio object, which holds a set of quotes for specific stocks
- A ticker object, which displays a portfolio
- An Internet object, a user object, and many others

Under that model, the stock ticker software is a collection of all the objects necessary to get work done.

OOP is a powerful way to create programs, and it makes the programs you write more useful. Consider the stock software. If the programmer wants to use the quote capabilities of that program in some other software, the quote object can be used with the new program. No changes need to be made.

**Stopping by Java Boutique for Directions**

This world tour of Java programs is being led by a professional who is well-versed in the hazards and highlights of web-based travel. You’ll be venturing out on your own trips soon, so it’s worthwhile to stop at one of the best guides for the tourist who wants to see Java: Java Boutique at http://javaboutique.internet.com.

Java Boutique features a directory of Java programs and programming resources related to the language. One of the best uses of the site for programmers is to see what programs are available that offer source code. In case you’re unfamiliar with the term, source code is another name for the text files that are used to create computer programs. The `Saluton.java` file you developed during Hour 2 is an example of source code.

The Source Code link on the Java Boutique’s home page lists the programs in the site’s directory that include their source code.
One of the programs whose source code is available is Aleksey Udovydchenko’s Absolute, a space videogame in which you control a ship and blast your way through an asteroid field (see Figure 3.6). The game features scrolling animation, graphics, keyboard control, and sound. To learn more and play the game, visit http://javaboutique.internet.com/Absolute.

The entire Absolute program was written in just more than 700 lines of code. That’s an extremely small number, considering everything the program does. Java includes an extensive library of classes you can use in your own programs. Udovydchenko employs a class called Image to display graphics such as asteroids and an AudioClip class to play sounds such as laser fire and explosions.

One goal of Java’s design was to make it easier to learn than C++, the language Gosling was having fits with on his smart-appliance project. Much of Java is based on C++, so programmers who have learned to use that language find it easier to learn Java. However, some of the elements of C++ that are the hardest to learn and use correctly are not present in Java.

For people learning programming for the first time, Java is easier to learn than C++. Some languages are created to make it easier for experienced programmers to harness the capabilities of the computer in their programs.
These languages include shortcuts and other features that programming veterans easily understand.

Java does not use some of these features, preferring to make the language as simple as an object-oriented programming language can be. Java was created to be easy to learn, easy to debug, and easy to use. Java includes numerous enhancements that make it a worthy competitor to other languages.

**Running Java on Your Phone**

The last stop on your whirlwind tour of Java is the nearest Google Android cell phone. Every single program that runs on Android has been programmed with Java. These mobile programs, which extend the functionality of the phones, are called apps. One of the most popular apps is a game called Angry Birds, shown in Figure 3.7.

You can learn more about this game, if you’re not already familiar with it, by visiting www.angrybirds.com. (But don’t do it! The game will obliterate any hope you had of being productive for the rest of the day, week, or even month—depending on how much you hate fortified pigs.)

Android ends the trip around Java because it’s becoming an incredibly popular place for the language to be used. After you learn Java, you can apply your skills developing your own apps using the Android Software Development Kit (SDK), a free programming toolkit that runs on Windows, MacOS, and Linux.
More than 250,000 apps have been created for Android phones and other devices that run the mobile operating system. You learn more about it in Hour 24, “Writing Android Apps.”

**Summary**

Now that the hour-long vacation is over, it’s time to put away your luggage and get ready for a return to actual Java programming.

During the next 21 hours, you will master the basic building blocks of the Java language, learn how to create your own objects to accomplish tasks in object-oriented programming, design graphical user interfaces, and much more.

Unless you’ve stopped reading this book to play Angry Birds.
Q&A

Q. Why are Java applets no longer popular?
A. When the Java language was introduced in the mid-'90s, most people were learning the language to write applets. Java was the only way to create interactive programs that ran in a web browser.

Over the years, alternatives emerged. Macromedia Flash, Microsoft Silverlight, and the new web publishing HTML5 standard all offer ways to put programs on web pages.

Applets were hampered by poor loading time and slow support for new versions of Java by browser developers. A Java plug-in was introduced that could run the current version of Java in browsers, but by that time Java had outgrown its origins and was a sophisticated general-purpose programming language.

Q. What's a Chris Steak House, and why does Ruth have one?
A. Ruth’s Chris Steak House, the chain of more than 120 upscale steak restaurants across the United States and a handful of other countries, has an odd two-first-name name that reveals its humble origins and the stubborn streak of its founder.

The chain was founded in 1965 as a solitary New Orleans restaurant owned by Ruth Fertel, a single mother of two sons. Fertel saw a classified ad offering a restaurant for sale and took out a $22,000 home mortgage to buy it (equivalent to around $150,000 in present dollars).

She reached a deal to keep the name Chris Steak House with original owner Chris Matulich, but later had to relocate after a kitchen fire.

Fertel’s contract did not permit her to use the Chris Steak House name anywhere but the original location, so she renamed it Ruth’s Chris Steak House. Though she had no restaurant or culinary expertise, the business was so successful that she began offering it as a franchise within 12 years. She disregarded several suggestions over the years to change the name to broaden its appeal.

“I’ve always hated the name,” she once told a reporter for Fortune magazine, “but we’ve always managed to work around it.”

Fertel, who died in 2002, was born on Feb. 5, 1927—the same day that Matulich opened the steakhouse.

Workshop

If your mind hasn’t taken a vacation by this point, test your knowledge of this hour with the following questions.
Quiz

1. How did object-oriented programming get its name?
   A. Programs are considered to be a group of objects working together.
   B. People often object because it’s hard to master.
   C. Its parents named it.

2. Which of the following isn’t a part of Java’s security?
   A. Web programs cannot run programs on the user’s computer.
   B. The identity of a program’s author is always verified.
   C. Java windows are labeled as Java windows.

3. What is a program’s capability to handle more than one task called?
   A. Schizophrenia
   B. Multiculturalism
   C. Multithreading

Answers

1. A. It’s also abbreviated as OOP

2. B. Programmers can use digital signatures and an identity-verification company such as VeriSign in Java, but it isn’t required.

3. C. This also is called multitasking, but the term multithreading is used in conjunction with Java because a separately running part of a program is called a thread.

Activities

Before unpacking your luggage, you can explore the topics of this hour more fully with the following activities:

- Use the Java Boutique site at http://javaboutique.internet.com to find out what card games have been developed using the language.

- Visit Oracle’s website for Java users, www.java.com, and click the “Do I Have Java?” link. Follow the instructions to see whether Java’s present on your computer. Download and install the most up-to-date version, if prompted to do so.

Solutions for the activities in this book are presented on the book’s website at www.java24hours.com.
INDEX

NUMERICS

2D graphics, 330
   arcs, 332-333, 341
   circles, 332
   ellipses, 332
   lines, 330
PiePanel application, 333
   PiePanel.java source code, 338
   PieSlice class, 335-336
rectangles, 331

SYMBOLS

< > (angle brackets), 238
; (semicolon), 17, 22, 102
!= (inequality operator), 81
$ (dollar sign), 54
% operator, 56
' (single quotation mark), 51, 67
/ (backslash), 67
" (double quotation mark), 51
> (greater than operator), 81
\n (newline character), 180
_ (underscore) character, 53
* (multiplication operator), 56
+ (plus sign)
   addition operator (+), 56
   concatenation operator, 68-69
+= operator, 69
-- (decrement operator), 56
- (minus sign), 56
/ (division operator), 56
/ (forward slash) character, 284
// (double slashes), 17
// (two slash characters), 258
= (equal sign), 52, 54
== (equality operator), 81
? (question mark), 86-87
@Override annotation, 314
@WebMethod annotation, 315

A

Absolute program, 34
Abstract Windowing Toolkit. See AWT
access control
   definition of, 138
   methods, 142
   variables, 138
      default, 139
      private variables, 139
      protected variables, 139
accessor methods, 142
ActionListener interface, 202, 271
actionPerformed() method, 202-203, 212, 276
activities
   Hour 1, 12
   Hour 2, 24
   Hour 3, 38
   Hour 4, 48
   Hour 5, 64
   Hour 6, 77
   Hour 7, 94
   Hour 8, 106
   Hour 9, 119
   Hour 10, 136
   Hour 11, 154
   Hour 12, 168
   Hour 13, 186
   Hour 14, 200
   Hour 15, 218
   Hour 16, 233
   Hour 17, 248
   Hour 18, 264
   Hour 19, 281
   Hour 20, 297, 311
   Hour 21, 326, 342
Activity class, 346
Add Library dialog box, 303
Add Repository dialog box, 391
add() method, 157
add(Component) method, 228
addActionListener() method, 202
addChangeListener() method, 223
adding
  emulators, 350
  plug-ins, Eclipse, 392
addItemListener() method, 204, 206
addition operator (+), 56
addKeyListener() method, 204
addOneToField() method, 212
addSlice() method, 335
Aggregator application, 307, 309
Agile Java Development with Spring, Hibernate and Eclipse, 381
ALIGN attribute (APPLET tag), 239
Android
  applications
    configuring AVDs, 350-351
    creating, 345-349
    Debug Configurations, 351-352
design, 366
design, 355-358
    interface design, 359-362
    manifest files, 358-359
    navigating, 346-348
    running, 352-354
  writing Java code, 362-368
Java on phones, running, 35
Java Virtual Machines, 389-390
resources, 358
SDKs, 390
Android Virtual Devices. See AVDs
AndroidManifest.xml file, 347, 357-360
Angry Birds application, 35
Annotations, applying, 314-315
apostrophes (‘), 51
Apple iPhones, 343
APPLET tag (HTML), 238-239
  ALIGN attribute, 239
  CODE attribute, 239
  CODEBASE attribute, 239, 247
  HEIGHT attribute, 239
  WIDTH attribute, 239
applets, 25, 42, 235
  class files, 236
  compared to applications, 236
definition of, 25, 39, 235
displaying
  drawString() method, 240
  paint() method, 236-237
  repaint() method, 236
event handling, 201
  actionPerformed() method, 202
  check boxes, 204
  combo boxes, 204
  event listeners, 201-202
  keyboard events, 206
events, 236
HTML markup, 238-239
initializing, 237-238
Java Boutique, 33-35
JTicker website, 32-33
LinkRotator, 273
methods, 235-236
  destroy(), 238
  init(), 237-238
  paint(), 236-237
  repaint(), 236
  start(), 238
  stop(), 238
object tags, applying, 245-246
parameters
  naming, 243
  passing, 243
receiving, 243
ShowWeight applet example, 244
WeightScale applet example, 243-245
real-world examples, Visible Human Project website, 27
Revolve, 270
class declaration, 271
error handling, 272
event handling, 276
initializing, 272
screen updates, 273
threads, 274-275
variables, 271
RootApplet, 43-44
SalutonApplet
displaying, 240
HTML markup, 241
source code listing, 240
saving, 7
security, digital signatures, 30
starting, 238
stopping, 238
structure, 43
threads, 270
  class declarations, 271
  error handling, 272
  event handling, 276
  initializing, 272
  running, 274-275
  screen updates, 273
  starting, 274
  stopping, 275
  variables, 271
WeightScale source code, 243-245
windows, sizing, 239
appletviewers, 44
applications. See also applets
  Aggregator, 307, 309
applications

Android
  configuring AVDs, 350-351
  creating, 345-349
  Debug Configurations, 351-352
design, 355-358
interface design, 359-362
manifest files, 358-359
navigating, 346-348
overview of, 343-345
running, 352-354
writing code, 362-368

Angry Birds, 35
applets, creating, 42-44
arguments, 46
autodialers, 123
Benchmark, 103-104
Calculator, 251
Clock
  output, 90
  source code, 89-90

ClockFrame, 183
colors, 313, 327
  RGB values, 329
  setting, 329

compared to applets, 236
compiling, 19, 40
Configurator.java, 294-295
Console, 289
creating, 39-42
Credits, code listing, 72
Crisis, 188-189
definition of, 39
deploying, 394
Fonts, 313, 327
formatting, 192, 196-197

Game
  output, 82
  source code, 82

ID3Reader, 286-288
Java Boutique, 33-35
KeyViewer.java, 205-206
LeaderActivity, 362-368
LottoMadness, 192-193, 196-197
  applet version, 216
  event listeners, 208
  LottoEvent.java class, 209-211
  methods, 212-213
  source code listing, 213-215

multithreading, 31
Name
  output, 113
  source code, 112

NetBeans
  running, 378
  troubleshooting, 378-380
NewCalculator, 252
NewRoot, 41, 130
Nines, 97
NumberDivider, 254-255
Organizing block statements, 81-83
PageCatalog, 258-261
PieFrame, 338-339
PiePanel, 333
  PiePanel.java source code, 338
  PieSlice class, 335-336
PlanetWeight, 60-61
PrimeFinder, 268-269
properties.xml, 301
PropertyFileCreator.java, 300
ReadConsole, 289
Root
  compiling, 40
  source code, 39
  running, 7
Saluton
  class declarations, 15
  class statements, 16
  compiling, 19
  greeting variable, 17-18
  line-by-line breakdown, 18
  main() block, 16
  running, 20
  saving, 18
  source code, 15
  troubleshooting, 19-20
  writing, 14-15
SalutonApplet, 241-242
  saving, 7
SpaceRemover, 110
SquareRootClient, 320-323
SquareRootServer, 315
SquareRootServerImpl, 316
SquareRootServerPublisher, 318
stock analysis, 32-33
StringLister.java, 162-163
strings, viewing, 66-67
Tool, 228, 230
troubleshooting, 8

Variable
  char variables, 51
  code listing, 52
  floating-point variables, 51
  int statement, 50
  integer variables, 51
  string variables, 51

Virus, 148
  class constructor, 143
  getSeconds() method, 142
  setSeconds() method, 142
  showVirusCount(), 144
  tauntUser() method, 143

VirusLab
  output, 150
  source code, 149-150
WeatherStation, 304-307
Wheel of Fortune, 113
character arrays, 115
integer arrays, 115
applications

letterCount array, 115

nested loops, 115

output, 114

source code, 113

writing, 13

Wheel of Fortune application, 113

output, 114

source code, 113

Arrays class, applying, 112-113

/assets, 347

assigning variables

types, 50

values, 54-55

asterisk (*), 56

attributes, 122, 137

ALIGN, 239

CODE, 239

CODEBASE, 239, 247

HEIGHT, 239

HTML, 238

inheritance, 125-126

SRC, 238

WIDTH, 239

autoboxing, 131

autodialers, 123

AVDs (Android Virtual devices), 350-351

AWT (Abstract Windows Toolkit), 169

Insets class, 191-192

app_name string resource, 349

Arc2D class, 332-333

arcs, drawing, 332-333, 341

arguments, 46

applications, 41

methods, 142-143

ArrayIndexOutOfBoundsException, 250

arrayoutofbounds errors, 109

arrays, 109, 111

declaring, 108

definition of, 107

elements, 108

initial values, 108

multidimensional, 111

sample application, 110

sorting, 111-113

upper limits, checking, 109

B

backslash (/), escape code, 67

backspaces, escape code, 67

BASIC (Beginner's All Symbolic Instruction Code), 4, 10

behavior

hierarchy, 125-126

inheritance, 125

Bell, Joshua, 6

Benchmark application, 103-104

benchmarks, 102

BlackBerrys, 343

blank spaces in source code, 22

block statements, 49, 81-83

blocks, 16-17

books, Java-related, 381

Boole, George, 53

Boolean variables, 53

BorderLayout manager, 190-191

borders, Insets class, 191-192

braces ({}), 16-17, 49, 82, 92

brackets ([[]), 82, 92

break statement, 84, 92, 100

breaking loops, 100-101

Browser JAR/Folder dialog box, 303

browsers

Java Plug-in, 28

downloading, 242

buffered input streams, 288-290

Console application, 289

creating, 288

ReadConsole application, 289

reading, 288

bugs, 8. See also debugging

Builder class, 304

buttons, creating, 174-176

bytecode, 284

bytes, 284, 296

C

C++, 5, 10

CableModem class, 133

Cadenhead, Rogers, 381

Cafe au Lait website, 383

Calculator application, 251

calling web services, 323

cannot resolve symbol (error message), 20

career opportunities, 385

carriage returns, escape code, 67
classes

choice lists, event handling, 204. See combo boxes
choosing programming languages, interpreted languages, 7
Chrome browser, 44. See also Google; interfaces
circles, drawing, 332
classes, 122
  Activity, 346
  Applet methods, 156-157
  Arc2D, 332-333
  ArrayIndexOutOfBoundsException
    Exception, 250
  Arrays, applying, 112-113
  CableModem, 133
  Color, 329
  Console, 289-290
  declaring, 15-16
  documentation, 382
  DslModem, 133
  Ellipse2D, 332
  encapsulation, 142
  Exception, 250
  file, 133, 284-285
  FileInputStream, 290
  FileOutputStream, 290
  Graphics, 237
  Graphics2D, 330
    arcs, 332-333, 341
    circles, 332
    ellipses, 332
    lines, 330
    rectangles, 331
  hierarchy, 155, 167
  inheritance, 125-126, 135
    155-158
  inner classes, 146-147
  Insets, 191-192
  JApplet, 155-156, 235
    inheritance, 156-157
    methods, 157
    subclasses, 157
  JButton, 174
  JCheckBox, 177-178
  JComboBox, 178-179
  JFrame, 171
  JLabel, 176-177
  JPanel, 180
  JScrollPane, 219
  JSlider, 222
  JTextField, 179
  JTextArea, 179
  LottoEvent, 209, 211
  methods, 144
  Modem, 124, 132
  ModemTester, 133-134
  nesting, 146
  NetBeans, 376-377
  objects
    looping, 162-163
    storing, 160-162
  PieSlice, 335-336
  Point, 164
  Point3D, 164
  code listing, 164
  creating, 164-165
  testing, 165-166
  private, 135
  R, 363
  ReadConsole, 289
  Rectangle2D, 331
  Revolve, 271
  statement, 15-16, 124
  subclasses, 126, 133, 157-159, 164-165
  superclasses, 126
  testing, 165-166
  Thread, 265
  variables
    creating, 139-140
    values, 140
  Virus, 137
case
  changing strings, 71, 75
  sensitivity, variable names, 54
  statements, 84
casting, 127
  definition of, 127
  destinations, 127
  objects, 132
  sources, 127
  variables, 127-128
catch statement, 272, 280
catching
  Calculator application, 251-252
  DivideNumbers sample application, 254
  errors, 272
  exceptions, 249-255
  NewCalculator application, 253
  NumberDivider sample application, 254-255
  PageCatalog sample application, 258-261
  SumNumbers sample application, 251, 261
  try-catch blocks, 250-255, 261
  try-catch-finally blocks, 255
cell phones, 343. See also Android
CENTER tag (HTML), 238
close listeners, 223
  ColorSlide sample application, 227
  registering objects as, 223-224
changing string case, 71, 75
char variables, declaring, 51, 65
characters
  definition of, 51, 65
  special, escape codes, 67-68
  strings, counting, 113-115
charts, pie, 121
check boxes
  creating, 177-178
  event handling, 204
checkAuthor() method, 148
clearAllFields() method, 212
clients, 320-322
Clock application
  output, 90
  source code, 89-90
ClockFrame application, 183
clocks, 87. See also Clock application
close() method, 291
closing streams, 291
code
  annotations, formatting, 314-315
  writing Android applications, 362-368
CODE attribute (APPLET tag), 239
code listings
  Benchmark application, 103
  CableMode class, 133
  Calculator application, 251
  Clock application, 88-89
  ColorSliders application, 224
  Commodity program, 85-86
  Console application, 289
  Credits application, 72-73
  Crisis application, 188-189
  DslModem class, 133
  Game program, 82
  HomePage.java, 259
  ID3Reader application, 286-288
  KeyViewer.java, 205-206
  LinkRotator applet, 276-279
  LottoEvent.java class, 209, 211
  LottoMadness application, 193-195, 213-215
  MailWriter application, 220
  Modem class, 132
  ModemTester class, 133-134
  Name application, 112
  NewCalculator application, 252
  NewRoot application, 41, 130
  Nines application, 97-98
  NumberDivider application, 254
  PageCatalog application, 260
  PiePanel.java source code, 336-338
  PlanetWeight application, 60
  Point3D class, 164
  PointTester.java program, 165-166
  PrimeFinder application, 268-269
  Root application, 40
  RootApplet application, 43
  Saluton application, 15, 18
  SalutonApplet
    HTML file, 241
    source code, 240
  ShowWeight applet, 244
  SpaceRemover.java application, 110
  StringLister.java, 162-163
  Tool application, 229
  Variable application, 52
  Virus application, 149
  VirusLab application, 149-151
  Wheel of Fortune application
    output, 114
    source code, 114
  WriteMail application, 221
CODEBASE attribute
  APPLET tag, 239, 247
  OBJECT tag, 247
Color class, 328-329
colors, 313, 327
  Color class, 328
  displaying RGB values, 329
  Font class, 327-328
  RGB values, 329
  setting, 329
ColorSliders application, 227
combo boxes
  creating, 178-179
  event handling, 204
commands, javac, 40. See also methods
comments, 17, 22, 304
comparing strings, 70
  equal/not equal comparisons, 81
  less/greater than comparisons, 80-81
compiled languages, performance, 10
compilers
  definition of, 7
  javac, error messages, 20
compiling applications, 19, 40
complex for loops, 102
components, 170, 219
  arranging, 185
  buttons, creating, 174-176
  change listeners, 223
    ColorSliders sample application, 227
  registering objects as, 223-224
check boxes
  creating, 177-178
  event handling, 204
ClockFrame application, 183
  combo boxes, 178-179, 204
  creating, 180-183
  disabling, 206-207
  enabling, 206-207
  frames, 170-171
    adding components to, 174
    creating, 171, 174
    sizing, 172
  image icons, 227-228
    creating, 227
    Tool sample application, 228-230
  labels, 176-177
subject

Defining 403

Configurator.java application, 294-295
configuring
AVDs (Android Virtual Devices), 350-351
Debug Configurations, 351-352
phones, 394-395
ConfigWriter.java application, 291
Console application, 289
constants, 55
constructor methods, 143
arguments, 144
declaring, 143
inheritance, 144
containers, 170, 180
continue statement, 100
contracts, WSDL (Web Service Description Language), 318
controlling access.
debugging
Android applications, 357, 366
definition of, 8
OOP applications, 123
phones, 395
declaring
arrays, 108, 111
classes
class statement, 15-16
subclasses, 157-159, 164-165
methods, 141
class methods, 144
constructors, 143
public methods, 142
variables, 17, 50
Boolean, 53
char, 65
char variables, 51
class variables, 139-140
floating-point, 51
integers, 50
long, 52
object variables, 137-138
short, 52
strings, 51, 66
decrement operator (––), 56
decrementing variables, 56-58
default statement, 84
default.properties file, 348
defining
classes, inner classes, 146-147
services, 313
deleting files, 285

deploying
  Android applications, 354
  applications, 394

Deployment Target Selection Mode, 352

design
  Android, 355-358
  interfaces, 359-362

destinations (casting), 127

destroy() method, 238

detecting errors in Android applications, 357

determining string lengths, 70-71

development history of Java, 27

Development settings, 354

dialects, 302

dialog boxes, Add Repository, 391

digital signatures, 30

disabling components, 206-207

displaying
  applets
    drawString() method, 240
    paint() method, 236-237
    repaint() method, 236
  colors, 329
  pie graphs, 339
  revolving links, 279
  strings
    println() method, 66-67
    special characters, 67-68
  text areas, 179
  variable contents, 18
  web services, 323

displaySpeed() method, 124-125

division, 59

division operator (/), 56

do-while loops, 99-101

dockable toolbars, 228

docking toolbars, 230

documentation, 9, 232, 382

dollar sign ($), 54

double quotation mark ("), 51

double slashes (//), 17

draw() method, 330
drawing
  arcs, 332-333, 341
  circles, 332
  ellipses, 332
  lines/shapes, 329-330
  pie graphs, 333
    PiePanel.java source code, 338
    PieSlice class, 335-336
  rectangles, 331
drawRoundRect() method, 331-332
drawString() method, 141, 240

DslModem class, 133

EarthWeb's Java directory, 385

Eclipse
  Android plug-ins, 344. See also Android
  installing, 390
  plug-ins, 392
  projects, creating, 355

editing
  NetBeans, 376-377
  string resources, 348
  XML, 349

editors, text, 13

educational applications, 27

elements, 108
  comment, 304
  forecastday, 305
  initial values, 108

Ellipse2D class, 332

efficiently, 358

ellipses, drawing, 332

else statements, 83

employment opportunities, 385

emulators (Android), configuring, 350-351

enabling components, 206-207

encapsulation, 142

endless loops, 105

Endpoint Interfaces, 317
  annotations, 314-315
  creating, 313

equal sign (=), 52, 54

equality operator (==), 81

equals() method, 70, 156

error handling, 249
  catching exceptions, 249-250
  multiple exceptions, 253-255
  PageCatalog sample application, 258-261
  try-catch blocks, 250-255, 261
  try-catch-finally blocks, 255
  creating exceptions, 262
  ignoring exceptions, 258
  memory errors, 262
  stack overflows, 262
  throwing exceptions, 250, 256-258
  PageCatalog sample application, 258-261
  throw statements, 256
  try-catch statements, 272

errors
  Android applications, 357
  arrayoutofbounds, 109
  bugs, 8
  cannot resolve symbol message, 20
  exceptions, 109, 117
  handling. See error handling
  javac error messages, 20
  logic errors, 8
  NetBeans, 379
Saluton program, troubleshooting, 19-20
syntax errors, 8
escape codes, 67-68
evaluating expressions, operator precedence, 59
Evans, Ben, 383
event handling, 201
    actionPerformed() method, 202, 276
check boxes, 204
combo boxes, 204
event listeners, 201-202
    ActionListener interface, 202
    LottoMadness application, 208-211
keyboard events, 206
event listeners, 201-202
    ActionListener interface, 202
    actionPerformed() method, 202
    adding, 201
    LottoMadness application, 208-209, 211
EventListener interfaces, 201-202
Everlong.mp3 file, 287-288
Exception class, 250
exceptions, 109, 117
    ArrayIndexOutOfBoundsException
    Exception, 250
catching, 249-250
    multiple exceptions, 253-255
    PageCatalog sample application, 258-261
try-catch blocks, 250-255, 261
try-catch-finally blocks, 255
creating, 262
ignoring, 258
NumberFormatException, 253-254
throwing, 250, 256-258
    PageCatalog sample application, 258-261
    throw statements, 256
executing. See starting existing objects, 159-160
exists() method, 284
exiting loops, 100-101
expressions, 49-50, 55, 59-61. See also operators
    advantages, 60
    operator precedence, 58-59
extends statement, 132, 157
expressions (file), .class, 22

F

File class, 284-285
File.pathSeparator, 284
FileInputStream class, 290-292
FileOutputStream class, 290
files
    checking existence of, 284
    creating, 284
    deleting, 285
File class, 284-285
file extensions, .class, 22
    finding size of, 285
    manifest, Android applications, 358-359
    reading
        ID3Reader application, 286-288
        streams, 285-286
        renaming, 285
        writing to, 290-291
    XML
        creating, 299-302
        reading, 302-307
        RSS syndication feeds, 307-309
    fill() method, 330
    fillRect() method, 329-331
    fillRoundRect() method, 331
    finding strings within strings, 71-72

Fisher, Timothy R., 381
float statement, 51
floating-point variables, declaring, 51
FlowLayout manager, 176, 187
folders, viewing, 356. See also files
Font class, applying, 327-328
fonts, 327
for loops, 95-97
    complex for loops, 102
counter variables, 96
empty sections, 102
exiting, 100-101
sample application, 97
syntax, 96-97
vectors, 162-163
forecastday element, 305
formatting. See also configuring; design
    annotations, 314-315
applications, 39, 192, 196-197
    Android, 345-352
    creating applets, 42-44
    sending arguments to, 41-42
classes, NetBeans, 376-377
Color class, 328
components, 180-183
Font class, 327-328
interfaces
    annotations, 314-315
    Endpoint Interfaces, 313
threads, 266
variables, 137-140
web service clients, 320-322
XML files, 299-302
formfeeds, escape codes, 67
forward slash (/) character, 284
frames, 170
    adding components to, 174
    creating, 170-171
    SalutonFrame.java example, 174
    sizing, 172
Gosling, James, 4, 26, 303, 344, 373
graphics, 330
arcs, 332-333, 341
circles, 332
color, 313, 327
RGB values, 329
setting, 329
ellipses, 332
fonts, 313, 327
Graphics class, 237
icons, 227-228
creating, 227
Tool sample application, 228-229
lines, drawing, 330
PiePanel application, 333
PiePanel.java source code, 338
PieSlice class, 335-336
rectangles, drawing, 331
Graphics class, 237
Graphics2D class, 330
arcs, 332-333, 341
circles, 332
ellipses, 332
lines, 330
rectangles, 331
graphs, pie, 333, 339
PiePanel.java source code, 338
PieSlice class, 335-336
greater than operator, 81
greeting variables
declaring, 17
displaying contents of, 18
GridLayout manager, 189-190
GridLayout() method, 197
GUIs (graphical user interfaces), 170, 219
AWT (Abstract Windowing Toolkit), 169
buttons, creating, 174-176
change listeners, 223
ColorSliders sample application, 227
registering objects as, 223-224
check boxes
creating, 177-178
event handling, 204
ClockFrame application, 183
combo boxes
creating, 178-179
event handling, 204
enabling/disabling components, 206-207
event handling, 201
event listeners, 201-202
ActionListener interface, 202
actionPerformed() method, 202
adding, 201
frames, 170
adding components to, 174
creating, 170-171
SalutonFrame.java example, 174
sizing, 172
image icons, 227-228
creating, 227
Tool sample application, 228, 230
Insets, 191-192
labels, creating, 176-177
layout managers, 187
BorderLayout, 190-191
FlowLayout, 187
GridLayout, 189-190
LottoMadness sample application, 192-197
panels, creating, 180
scroll panes, 219
adding components to, 220
creating, 219-220
MailWriter sample application, 221
interfaces

sliders, 222-223
WriteMail sample application, 222
Swing, 169
text
areas, 179
fields, 176-177
write-protecting, 198
toolbars, 227
creating, 228
dockable toolbars, 228
Tool sample application, 228, 230
windows, 170-172, 174

H

handling errors. See error handling
Harold, Elliotte, 303, 383
HEIGHT attribute (APPLET tag), 239
"Hello world!", 20
Hemrajani, Anil, 381
hierarchies, Java classes, 155
history of Java, 26-27
HomePage.java listing, 259
horizontal sliders
creating, 222
labels, 223
HTML (Hypertext Markup Language), 238
angle brackets (< >), 238
APPLET, 238-239
CENTER, 238
P 238
hyphen (-), subtraction operator, 56

I

I/O (input/output)
stream, 283-284, 299
buffered input streams, 288-290
byte streams, 284
closing, 291
defined, 283-284
reading data from, 285-288
writing data to, 290-291
IceRocket, 383
icons, 227-228
creating, 227
Tool sample application, 228, 230
ID3Reader application, 286-288
IDEs (integrated development environments), 344, 373
if statements, 79-81, 83, 92
blocks, 81-83
equal/not equal comparisons, 81
less than/greater than comparisons, 80-81
if-else statements, 83
ignoring exceptions, 258
ImageIcon constructor, 227
ImageIcon() method, 227
implementing Service Implementation Beans, 316-317
import statement, 237
incrementing variables, 56-58
indexOf() method, 71-72
inequality operator (!=), 81
infinite loops, 105
InformIT, 384

init() block statements, 43
init() method, 237-238, 272
initializing
applets, 237-238, 272
definition of, 105
inner classes, 146-147
input/output. See I/O
Insets class, 191-192
installing
Android
plug-ins, 391-393
SDKs, 390
Eclipse, 390
NetBeans, 373
programming tools, 9
int statement, 50
integers
arrays, creating, 108
variable types, 50
integrated development environments. See IDEs
Intel, 343
Intent() method, 365
interfaces, 227. See also GUIs
ActionListener, 202, 271
AWT (Abstract Windowing Toolkit), 169
buttons, 174, 176
ChangeListener, 223
check boxes, 177-178
combo boxes, 178-179
components, 170, 180-183
defined, 201
design, Android applications, 359-362
Endpoint Interfaces
annotations, 314-315
creating, 313
EventListener, 201-202
frames, 170-173
interfaces

overriding, 157
setBackgroundColor(), 157
setLayout(), 157
subclasses, 157
JAR (Java Applet Ratings Service), 34
JARS (Java Review Service), 384
Java 7 Developer Blog, 383
Java Boutique website, 33-35
Java Development Kits. See JDKs
Java Development Tools. See JDTs
Java EE 6 Tutorial, The Basic Concepts, 381
Java Enterprise Edition. See JEE
Java Mobile Edition. See JME
Java Phrasebook, 381
Java Plug-in, 28, 242
Java Review Service, 384
Java Standard Edition. See JSE
Java Virtual Machines. See JVMs
Java website, 382
Javac
commands, 40
compilers, error messages, 20
JavaWorld website, 29-30
javax.xml.ws, 317
JAX-WS library packages, 322
JButton objects, 174
JCheckBox class, 177-178
JComboBox class, 178-179
JDKs (Java Development Kits), 8, 320
applications
Saluton program, 14-15
writing, 13
installing, 9
JDTs (Java Development Tools), 390
JEE (Java Enterprise Edition), 373
Jendrock, Eric, 381
JFrame class, 171
JLabel class, 176-177
JME (Java Mobile Edition), 373
job opportunities, 385
Joy, Bill, 26
JPanel class, 180
JScrollPane class, 219
JScrollPane() method, 219
JSE (Java Standard Edition), 373
JSlider class, 222
JSlider() method, 222
JTextArea class, 179-180
JTextField class, 176-177
JTicker website, 32-33
JToolBar() method, 228
JVMs (Java Virtual Machines), 20, 28

K
keyboards, event handling, 206
KeyListener interface, 204-206
KeyViewer.java application, 205-206
keywords, this, 147-148

L
Label() method, 176
labels
creating, 176-177
sliders, 223
languages
OOP. See OOP
selecting, 4-5
layout managers, 187
FlowLayout, 187
GridLayout, 189-190
LottoMadness sample application, 192-197
LeaderActivity application, 362-368
length variable, 109, 117
length() method, 70, 285
lengths of strings, determining, 70-71
LGPL (GNU Lesser General Public License), 303
libraries, XOM, 303. See also XOM
Line2D class, 330
lines, drawing, 329-330
LinkRotator applet, 273
links
  revolving, displaying, 279
  variables with strings, 68-69
listeners, 201-202
  ActionListener interface, 202
  actionPerformed() method, 202
  adding, 201
  change listeners, 223
    ColorSliders sample application, 227
    registering objects as, 223-224
  LottoMadness application, 208-211
listFiles() method, 285
listings. See also code listings
  Aggregator application, 307-309
  ClockFrame application, 183
  ClockPanel application, 181
  Configurator.java application, 294-295
  ConfigWriter.java application, 291
  HomePage.java application, 259
  LeaderActivity application, 362-368
  NumberDivider application, 254-255
  PageCatalog application, 260
  PieFrame application, 338-339
  Playback application, 175
  properties.xml application, 301
PropertyFileCreator.java application, 300
SalutonFrame.java application, 173
SquareRootClient application, 320-323
SquareRootServer application, 315
SquareRootServerImpl. application, 316
SquareRootServerPublisher application, 318
WeatherStation.java application, 305-307
Web Service Description Language Contract application, 319
lists, choice lists, 204
load() method, 292
loading applets, 43
Log.i() methods, 364
logic errors, 8
long variable type, 52
loops
  Benchmark application, 103-104
definition of, 95
  do-while, 99
  exiting, 100-101
  for, 95-97
complex for loops, 102
counter variables, 96
empty sections, 102
sample application, 97
syntax, 96-97
vectors, 162-163
infinite loops, 105
naming, 101
nesting, 101
while, 98-99
LottoEvent.java class, 209, 211
LottoEvent.java class, 209, 211
methods
  actionPerformed(), 212
  addOneToField(), 212
  clearAllFields(), 212
  getActionCommand(), 212
  itemStateChanged(), 212
  matchedOne(), 212
  numberGone(), 212
  source code listing, 213, 215
LottoMadness() method, 197
lowercase, changing strings to, 71
magazines, JavaWorld, 29-30
MailWriter application, 221
main() blocks, Saluton program, 16
MalformedURLException errors, 258, 273
managers. See layout managers
managing resources, 356-358
manifest files, Android applications, 358-359
matchedOne() method, 212
memory errors, 262
messages, SOAP, 322
methods, 137, 140, 236
  accessor, 142
  actionPerformed(), 202-203, 212, 276
  add(), 157
  add(Component), 228
  addActionListener(), 202
  addChangeListener(), 223
  addItemListener(), 204
  numberGone(), 212
  addSlice(), 335
applets, 235
arguments, 142-143
checkAuthor(), 148
class methods, declaring, 144
clearAllFields(), 212
close(), 291
constructors, 143
arguments, 144
declaring, 143
inheritance, 144
createNewFile(), 285
currentThread(), 275
declaring, 141
definition of, 70
destroy(), 238
displaySpeed(), 124-125
draw(), 330
drawRoundRect(), 332
drawString(), 141, 240
equals(), 70, 156
effects(), 284
fill(), 330
fillRect(), 329, 331
fillRoundRect(), 331
get(int), 304
getActionCommand(), 203, 212
getAttribute(), 304-305
getFirstChildElement(), 304
getFirstChildElement(), 304
getId(), 364
getInsets(), 192
getKeyChar(), 205
getCode(), 205
getKeyName(), 284
getParameter(), 243
getPort(), 322
getProperty(), 293
getSeconds(), 142
ggetSource(), 203, 223
getSquareRoot(), 315, 320
getStateChange(), 204
getTime(), 315
getURL(), 272
getValue(), 304-305
getValuesAdjusting(), 224
getVirusCount(), 149
GridLayout(), 197
ImageIcon(), 227
indexOf(), 71-72
init(), 237-238, 272
init() blocks, 43
Intent(), 203, 212
JScrollPane(), 219
JSlider(), 222
JToolBar(), 228
Label(), 176
length(), 70, 285
load(), 292
Log.i(), 364
LottoMadness(), 197
main() blocks, 16
matchedOne(), 212
numberGone(), 212
overriding, 157-158
pack(), 172
paint(), 43, 157-158, 236-237
parseInt(), 130, 152
println(), 61, 66-67, 141
public, 142
read(), 285
readLine(), 290
renameTo(), 285
repaint(), 236, 273
return values, 75, 141
run(), 267, 274-275
setBackground(), 157
setColor(), 273
setContentView(), 363
setDefaultCloseOperation(), 172
setEditable(), 179, 198
setEnabled(), 206
setLayout(), 157, 188
setLayoutManager(), 175
setProperty(), 293
setSeconds(), 142
setSize(), 172
setText(), 217
setTitle(), 171
showDocument(), 276
showVirusCount(), 144
skip(), 286
sleep(), 266
sort(), 112
start(), 238, 274
StateChanged(), 223
stop(), 238, 270, 275
storeToXML(), 300
substring(), 287
System.out.println(), 127, 376
tauntUser(), 143
TextArea(), 180
toCharArray(), 110
toUpperCase(), 71
toLowerCase(), 110
toUpperCase(), 71, 75
variable scope, 145-146
void keyPressed(), 204
void keyReleased(), 204
void keyTyped(), 205
write(), 290

mfl arrays, 111
minus sign (-)
decrement operator (—), 56
subtraction operator, 56
Modem class, 124, 132
Modem objects, 123
modems
CableModem class, 133
DslModem class, 133
Modem class, 132
ModemTester class, 133-134
operators

NetBeans Field Guide, 373
NetBeansProjects, 375
Netscape Navigator, downloading Java Plug-ins, 242
New Android Project Wizard, 345, 349, 355
New File Wizard, 14
New Project button, 375
New Project Wizard, 375
new statements, 108, 143
NewCalculator application, 252
newline characters, 180
   escape codes, 67
NewRoot application, 130
   source code, 41
news aggregators, 307. See also RSS syndication feeds
newSuffix variable, 129
Nines application, 97
nu.xom package, 304
NumberDivider application, 254-255
NumberFormatException, 253, 256
numberGone() method, 212
numbers, displaying sequence of prime numbers, 268-269
numeric variable types, 52
Nvidia, 343

Oak language, 27
OBJECT tag (HTML), CODEBASE attribute, 247
object-oriented programming, See OOP
objects, 137. See also classes
   attributes, 122, 137
   behavior, 122
   casting, 132
   classes, 122
   converting, 127-131
   creating, 124-125, 132-134
   existing, 159-160
   inheritance, 125-126, 155-157
   Modem, 123
   Monitor, 123
   PieChart, 122
   referencing, 147-148
   storing, 160-163
   tags, 245-246
   variables, 137-139
      private, 139
      protected, 139
   onCreate() method, 363
   onCREATE() method, 363
   online communities, Stack Overflow, 384
OOP (object-oriented programming), 33, 121-122, 170. See also classes
   advantages of, 122-123
   applications, debugging, 123
   autoboxing/unboxing, 131
   encapsulation, 142
   inheritance, 125-126, 135, 155-157
   objects
      casting, 132
      creating, 124-125, 132, 134
   objects. See objects
      overview, 33, 121
Open Handset Alliance, 343

operators

addition (+), 56
concatenation (+), 68-69
decrement (– –), 56
division (/), 56
equality (==), 81
greater than (>), 81
inequality (!=), 81
modulus (%), 56
multiplication (*), 56
precedence, 58-59
subtraction (-), 56
ternary (?), 86-87
Oracle, 25
Oracle Technology Network for Java Developers, 382
order of precedence, operators, 58-59
organizing
applications, block statements, 81-83
resources, 356-358
output. See I/O (input/output)
@Override annotation, 314
overriding methods, 157-158

P
P tag (HTML), 238
pack() method, 172
Package Explorer, applying, 348
packages, 139
Android SDKs, installing, 394
javax.xml.ws package, 317
JAX-WS library, 322
PageCatalog application, 258-261
pageTitle array, 271
paint() method, 236-237, 273
block statements, 43
overriding, 157-158
panels, creating, 180
PARAM tag (HTML), 242
NAME attribute, 243
VALUE attribute, 243
parameters
handling
ShowWeight applet, 244
WeightScale applet, 243-245
naming, 243
passing to applets, 243
receiving in applets, 243
values, assigning, 243
parseInt() method, 130, 152
passing
arguments
to applications, 41
to methods, 142-143
parameters to applets, 243
pasting
into strings, 69
strings together, 68
percent sign (%), modulus operator, 56
performance, interpreted languages, 10
phones. See also Android
configuring, 394-395
running Java on, 35
pie charts, 121
pie graphs, creating, 333
PiePanel.java source code, 338
PieSlice class, 335-336
viewing, 339
PieChart object, 122
PieFrame application, 338-339
PiePanel application, 333
PiePanel.java source code, 338
PieSlice class, 335-336
PieSlice class, 335-336
pipe (|) characters, 254
PlanetWeight application code listing, 60-61
platform independence, 29
Playback.java, 175
plug-ins
Android, 344, 391-393
definition of, 242
Java Plug-in, 242
plus signs (+)
addition operator, 56
concatenation operator, 68-69
increment operator (++), 56
Point class, 164
Point3D class, 164
creating, 164-165
testing, 165-166
postfixing, 57
precedence, operators, 58-59
preferences, configuring Android, 393
prefixing, 57
prime numbers, displaying sequence of, 268-269
PrimeFinder application, 268-269
printing strings
println() method, 66-67
special character, 67-68
println() method, 61, 66-67, 141
private classes, 135
private variables, 139
program listings. See code listings
programming
Android, 389
configuring phones, 394-395
Eclipse, 390
plug-ins, 391-393
SDKs, 390
languages, selecting, 4-5
OOP (object-oriented programming). See also OOP
advantages of, 122-123
casting, 129
creating objects, 124, 132-134
overview of, 121
Saluton program
creating, 14-15
running, 20
tools
installing, 9
selecting, 8-9
programs. See applications; software
proguard.cfg file, 348
Project Location text field, 375
Project Properties dialog box, 303
Project Selection dialog box, 352
projects
Android applications, navigating, 346-348
creating, 355
NetBeans, 374-375
properties
configuration, reading/writing, 292-295
customizing, 361
Properties object, 293, 299
Properties.xml application, 301
PropertyFileCreator.java application, 300
protected variables, 139
public methods, 142
public statements, 124
publishing web services, 317-318

Q

QName, 321
question mark (?), 86-87
quizzes
Hour 1, 11
Hour 2, 23
Hour 3, 37
Hour 4, 47
Hour 5, 63
Hour 6, 76
Hour 7, 93-94
Hour 8, 105-106
Hour 9, 118
Hour 10, 135-136
Hour 11, 153
Hour 12, 167-168
Hour 13, 185-186
Hour 14, 199
Hour 15, 217-218
Hour 16, 232-233

reduced rectangles, drawing

Hour 17, 247
Hour 18, 263
Hour 19, 280
Hour 20, 296-297, 310-311
Hour 21, 341-342

quotation marks
double ("), 51
escape codes, 67
single (’), 51

R

R class, 363
R.java file, 363
read() method, 285
ReadConsole application, 289
reading
configuration properties, 292-295
files, 285
ID3Reader application, 286-288
read() method, 285
skip() method, 286
RSS syndication feeds, 307, 309
streams, buffered input streams, 288
XML files, 302-307
readLine() method, 290
real-world Java projects
JavaWorld website, 29-30
Visible Human Project website, 27, 29
receiving parameters to applets, 243
recommended reading, 381
Rectangle2D class, 331
rectangles, drawing, 331
Red, Green Blue (RGB) color system, 329
Reference Chooser dialog box, 361
replacing objects, this statement, 147-148
registering objects as change listeners, 223-224
renameTo() method, 285
renaming files, 285
repaint() method, 236, 273
/res folder, 347, 357
resources, 381. See also websites
Android, 358
folders, viewing, 356
Java-related books, 381
job opportunities, 385
managing, 356-358
naming, 349
strings, editing, 348
restricting access, 138. See also access control
return values (methods), 75, 141
Revolve applet, 270
class declaration, 271
error handling, 272
event handling, 276
methods
actionPerformed(), 276
init(), 272
r(), 274-275
start(), 274
stop(), 275
screen updates, 273
threads
running, 274-275
starting, 274
stopping, 275
variables, 271
Revolve class, creating, 271
revolving links, displaying, 279
RGB values (red, green, blue), 329
Root application, 40
RootApplet applet, 43-44
rounded rectangles, drawing, 331
RSS syndication feeds, reading

Sams Teach Yourself Java 2 in 21 Days, 381
Sams Teach Yourself Java 2 in 24 Hours website, 387-388
Sams Teach Yourself Java in 24 Hours website, 383
Samsung, 343
saving
applications, 7
Saluton programs, 18
scope (variables), 145-146
screens, updating, 273
scroll panes, 219
adding components to, 220
creating, 219-220
MailWriter sample application, 221
WriteMail sample application, 222
SDKs (Software Development Kits), 343, 390
searching strings, 71-72
searchKeywords variable, 69
security, 30
digital signatures, 30
trusted developers, 30
selecting
languages, 4-5
programming tools, 8-9
semicolumn (\;), 17, 22, 102
Service Implementation Bean, 316-317
services
clients, creating, 320-322
defining, 313
publishing, 317-318
SquareRootServer, 313
setBackground() method, 157
setColor() method, 273
setContentView() method, 363
setDefaultCloseOperation() method, 172
setEditable() method, 179, 198
setEnabled() method, 206
setLayout() method, 157, 188
setLayoutManager() method, 175
setProperty() method, 293
setSeconds() method, 142
setSize() method, 172
setText() method, 217
setTitle() method, 171
shapes
arcs, 332-333, 341
circles, 332
drawing, 329-330
ellipses, 332
lines, 330
PiePanel application, 333
PiePanel.java source code, 338
PieSlice class, 335-336
rectangles, 331
short variable type, 52
showDocument() method, 276
showVirusCount() method, 144
signatures (digital), 30
single quotation marks (\'), escape code, 67
sizing applet windows, 239
skip() method, 286
SkyWatch, 31-32
slashes (/\), 17
sleep() method, 266
sliders
creating, 222-223
labels, 223
slowing down threads, 266
SOAP messages, 322
software
Absolute program, 34
overview, 5-6
strings, viewing, 66-67
troubleshooting, 8
Software Development Kits. See SDKs
sort() method, 112
strings

continuing, 100
default, 84
definition of, 5
different, 84
expressions, 50, 59-61extends, 132, 157float, 51if, 79-80, 83, 92blocks, 81-83equal/not equal comparisons, 81less/greater than comparisons, 80-81if-else, 83import, 237init(), 43int, 50loops
definition of, 95

static statement, 140, 144

stop() method, 238, 270, 275
stopping
applets, 238
threads, 275
storeToXML() method, 300
storing
looping, 162-163
objects, 160-162
variables, 54-55
streams, 283-284, 299
buffered input streams, 288-290
  Console application, 289
  creating, 288
  ReadConsole application, 289
  reading, 288
byte streams, 284
  closing, 291
defined, 283-284
  reading data from, 285
  ID3Reader application, 286-288
  read() method, 285
  skip() method, 286
writing to, 290-291

String data type, 17
StringLister.java source code, 162-163
strings, 65-66
  adding to, 69
  arrays, 108. See also arrays
  changing case of, 71, 75
  characters, counting, 113-115
  comparing, 70
equal/not equal comparisons, 81
  less/greater than comparisons, 80-81
concatenating, 68
definition of, 51, 66
determining length of, 70-71
displaying
println() method, 66-67
special characters, 67-68
finding within other strings, 71-72
resources, editing, 348
variables, 51
declaring, 66
linking, 68-69
strings.xml file, 349
Stroustrup, Bjarne, 5
subclasses, 126
creating, 133, 157-159, 164-165
substring() method, 287
subtraction operator (-), 56
Sun website, 25-26, 382
super statement, 165
class declarations, 158-159
superclasses, 126
Swing, 169, 219
buttons, creating, 174-176
change listeners, 223
ColorSliders sample application, 224-227
registering objects as, 223-224
check boxes
creating, 177-178
event handling, 204
combo boxes
creating, 178-179
event handling, 204
documentation, 232
enabling/disabling components, 206-207
event listeners, 201-202
ActionListener interface, 202
actionPerformed() method, 202
adding, 201

LottoMadness application, 208-211
image icons, 227-228
creating, 227
Tool sample application, 228, 230
JApplet class, 235
labels, creating, 176-177
layout managers, 187
BorderLayout, 190-191
FlowLayout, 187
GridLayout, 189-190
LottoMadness sample application, 192-197
panels, creating, 180
scroll panes, 219
adding components to, 220
creating, 219-220
MailWriter sample application, 221
WriteMail sample application, 222
sliders
creating, 222-223
labels, 223
text
areas, 179
fields, 176-177
write protecting, 198
toolbars, 227
creating, 228
dockable toolbars, 228
Tool sample application, 228-230
switch statements, 84-86
syndication feeds, reading RSS, 307-309
syntax errors, 8
System.out.println() method, 127, 376

T-Mobile G1s, 343
tabs, escape code, 67
tags
angle brackets (< >), 238
APPLET, 238-239
ALIGN attribute, 239
CODE attribute, 239
CODEBASE attribute, 239, 247
HEIGHT attribute, 239
WIDTH attribute, 239
CENTER, 238
HTML, 238, 242-243
objects
applying, 245-246
CODEBASE attribute, 247
P, 238
PARAM, 242
NAME attribute, 243
VALUE attribute, 243
tauntUser() method, 143
ternary operator (?), 86-87
testing
computer speed, 103-104
Points3D class, 165-166
SalutonApplet program, 241-242
SquareRootServerPublisher application, 318
text
areas, 179
Color class, 328
editors, 13
fields
creating, 176-177
write protecting, 198
Font class, 327-328
pasting into strings, 69
TextArea() constructor method, 180
this keyword, 147-148
variables

tools
appletviewer, 44
programming
installing, 9
selecting, 8-9
toUpperCase() method, 71, 75
travel Java Boutique, 33-35
troubleshooting
Android applications, 357
exceptions, 249-253. See also exceptions
NetBeans, 378, 380
Saluton program, 19-20
software, 8
trusted developers, 30
try-catch blocks, 250-255, 261, 273
Calculator application, 251-252
DivideNumbers sample application, 254
NewCalculator application, 253
NumberDivider sample application, 254-255
SumNumbers sample application, 251, 261
try-catch statement, 272
try-catch-finally blocks, 255
TryPoints.java listing, 165
Twitter, 385
two slash characters (//), 258
type values (variables), casting, 127
types
Boolean, 53
byte, 52
char, 51
long, 52
short, 52
variables, 50

U
Udovydchenko, Aleksey, 34
unboxing, 131
underscore (_) characters, 53, 54
University of British Columbia, 28
updating screens, 273
upper limits of arrays, checking, 109
uppercase, changing strings to, 71, 75
user events, 201
ActionListener interface, 202
combo boxes, 204
components, 206
handling, 202-203
keyboard events, 204-206
LottoMadness application, 207-208, 212-213

V
validity, 302
van de Panne, Michiel, 28
Variable application
code listing, 52
int statement, 50
variables
characters, 51
floating-point, 51
integers, 51
strings, 51
variables
access control, 138
arrays, 109, 111
declaring, 108
definition of, 107
elements, 108
initial values, 108
multidimensional, 111
variables

sample application, 110
  sorting, 111-113
assigning values, 54-55
casting, 127-128
converting to objects, 129-131
counter variables
  definition of, 96
  initializing, 96
data types, String, 17
declaring, 17, 50
  class variables, 139-140
  object variables, 137-138
  definition of, 49
displaying contents of, 18
  initializing, definition of, 105
length, 117
naming conventions, 54, 62
newSuffix, 129
private, 139
protected, 139
referencing, this statement, 147-148
Revolve applet, 271
Revolve program, 271
scope, 145-146
searchKeywords, 69
strings, 66
  changing case, 71, 75
  comparing, 70
  concatenating, 68
  declaring, 66
determining length, 70-71
displaying, 66-67
escape codes, 67-68
linking, 68-69
types
  assigning, 50
  Boolean, 53
  char, 51, 65
  floating-point, 51
  integers, 50
  long, 52
  short, 52
  strings, 51
  values
    assigning, 55
    decrementing, 56-58
    incrementing, 56-58
    starting values, 55
vectors, objects
  looping, 162-163
  storing, 160-162
Verburg, Martijn, 383
VeriSign website, 30
vertical sliders, creating, 223
viewing
  Android projects, 347
  appletviewers, 44
  pie graphs, 339
  resources, 356
  revolving links, 279
  strings, 66-67
text areas, 179
web services, 323
Virus application, 148
  class constructor, 143
  methods
    getSeconds(), 142
    setSeconds(), 142
    tauntUser(), 143
    showVirusCount(), 144
Virus class, 137
VirusLab application
  output, 150
  source code, 149-150
Visual Basic, 4
void keyPressed() method, 204
void keyReleased() method, 204
void keyTyped() method, 205
void statement, 141

W
WeatherStation.application, 304-307
Web Service Description Language,
  See WSDL
web services
  clients, creating, 320-322
  publishing, 317-318
  SquareRootServer, 313
Web Tools Platform. See WTP
weblogs, 383
@WebMethod annotation, 315
websites
  Cafe au Lait, 383
  Gamelan, 385
  InformIT, 382
  JARS (Java Review Service), 384
  Java Boutique, 33-35
  JTicker, 32-33
  JavaWorld, 29-30
  Liberty BASIC interpreter, 6
  Sams Publishing, 382
  Sams Teach Yourself Java 2 in 24 Hours, 387-388
  Sams Teach Yourself Java in 24 Hours, 383
  Sun, 25-26, 382
  VeriSign, 30
  Workbench, 383
WeightScale applets, source code, 243-245
well-formed data (XML formatting), 302
Wheel of Fortune application, 113
  character arrays, 115
  integer arrays, 115
  letterCount array, 115
  nested loops, 115
  output, 114
  source code, 113
while loops, 98-101
widgets, customizing properties, 361
WIDTH attribute (APPLET tag), 239
windows, 170-172, 174
   Debug Configurations, 351
wizards
   New Android Project Wizard, 345, 349
   New File Wizard, 14
   New Project Wizard, 375
Workbench website, 383
write protecting text fields, 198
write() method, 290
WriteMail application, 222
writing
   applications, 13, 39
   creating applets, 42-44
   Saluton programs, 14-15
   sending arguments to, 41-42
code, Android applications, 362-368
Color class, 328
configuration properties, 292-295
Font class, 327-328
streams, 290-291
WSDL (Web Service Description Language), 318-320
WTP (Web Tools Platform), 390

X-Y

XML (Extensible Markup Language)
editing, 349
files
   creating, 299-302
   reading, 302-307
RSS syndication feeds, 307-309
XOM (XML Object Model), 303

Z

Zamenhof, Ludwig, 20