

JAVA Developer's Guide

by Jamie Jaworski

C O N T E N T S

Introduction

Chapter1 *The Java Phenomenon*

- What Is Java?
- The Origins of Java
- Why Program in Java?
- Java and HotJava
- Summary

Chapter2 *Java Overview*

- Getting the JDK
- A Quick Tour of the JDK
- The Java Language
 - ◆ Java Is Familiar and Simple
 - ◆ Java Is Object-Oriented
 - ◆ Java Is Safer and More Reliable
 - ◆ Java Is Secure
 - ◆ Java Is Multithreaded
 - ◆ Java Is Interpreted and Portable
 - ◆ Java Is the Programming Language of the Web
- The Java API
- Summary

Chapter3 *Using the Java Developer's Kit*

- Overview
- The Compiler
 - ◆ Using Classes from Other Packages
 - ◆ Setting CLASSPATH
 - ◆ Changing the Root Directory
 - ◆ Generating Debugging Tables
 - ◆ Code Optimization
 - ◆ Suppressing Warnings
 - ◆ Using Verbose Mode
- The Interpreter
 - ◆ Changing CLASSPATH
 - ◆ Checking for Source Code Changes
 - ◆ Verifying Your Code
 - ◆ Controlling the Garbage Collector
 - ◆ Changing Properties
 - ◆ Setting Memory and Stack Limits
 - ◆ Debugging Options
- The Debugger
- The Disassembler
- The Applet Viewer
- Automating Software Documentation
- Header File Generation

- Running the Demo Programs
- Summary

Chapter 4 **First Programs: Hello World! to BlackJack**

- Hello World!
 - ◆ Comments
 - ◆ Java Program Structure
 - ◆ The package Statement
 - ◆ The import Statement
 - ◆ Classes and Methods
 - ◆ The System Class
- I Can Read!
 - ◆ Overview of ICanReadApp
 - ◆ Declaring Variables and Creating Objects
 - ◆ Identifiers and Keywords
 - ◆ Using System.in
- Type This!
 - ◆ Overview of TypeThisApp
 - ◆ The Primitive Java Data Types
 - ◆ Literal Values
- BlackJack
 - ◆ Overview of BlackJackApp
 - ◆ Arrays
 - ◆ Statements
- Summary

Chapter 5 **Classes and Objects**

- Object-Oriented Programming Concepts
 - ◆ It's an Object-Oriented World
 - ◆ Composition and Reuse
 - ◆ Classes
 - ◆ Classification and Inheritance
 - ◆ Multiple Inheritance
 - ◆ Messages, Methods, and Object Interaction
 - ◆ Encapsulation
 - ◆ Polymorphism
 - ◆ Dynamic Binding
- Java Classes
 - ◆ Class Syntax
 - ◆ The Point Class
 - ◆ Class Modifiers
 - ◆ Extending Superclasses
 - ◆ Adding Body to Classes
 - ◆ The CGrid Class
 - ◆ The CObject Class
 - ◆ The PrintCGrid Class
 - ◆ The BorderedPrintCGrid Class
 - ◆ The CGPoint Class
 - ◆ The CGBox Class
 - ◆ The CGText Class
 - ◆ The KeyboardInput Class
 - ◆ The CDrawApp Program
 - ◆ Running CDrawApp
 - ◆ CDrawApp's Implementation of Object-Oriented Concepts
- Summary

Chapter 6 **Interfaces**

- The Purpose of Java Interfaces
- The Benefits of Interfaces
- Declaring Interfaces

- Implementing Interfaces
- The CDrawApp Interface Example
 - ◆ The CGTextEdit Interface
 - ◆ Updating the CGText Class
 - ◆ The CGTextPoint Class
 - ◆ The CGTextBox Class
 - ◆ Updating the CDraw Class
 - ◆ Running the Example
 - ◆ Example Summary
- Using Interfaces as Abstract Types
- Interface Constants
- Extending Interfaces
- Combining Interfaces
- Summary

Chapter 7 **Exceptions**

- Eliminating Software Errors
- Error Processing and Exceptions
- Throwing Exceptions
- Declaring Exceptions
- Declare or Catch?
- Using the try Statement
- Catching Exceptions
- Nested Exception Handling
- Rethrowing Exceptions
 - ◆ Analysis of NestedExceptionTest
- Summary

Chapter 8 **Multithreading**

- Understanding Multithreading
- How Java Supports Multithreading
 - ◆ Creating Subclasses of Thread
 - ◆ Implementing Runnable
- Thread States
- Thread Priority and Scheduling
- Synchronization
- Daemon Threads
- Thread Groups
- Summary

Chapter 9 **Using the Debugger**

- Overview of the Debugger
- An Extended Example
- Debugging Multithreaded Programs
- Summary

Chapter 10 **Automating Software Documentation**

- How javadoc Works
- Using javadoc
- Placing Doc Comments
- Using javadoc Tags
- Embedding Standard HTML
- Summary

Chapter 11 **Language Summary**

- The package Statement
- The import Statement
- Comments

- Identifiers
- Reserved Words
- Primitive Data Types and Literal Values
- Class Declarations
- Variable Declarations
- Constructor Declarations
- Access Method Declarations
- Static Initializers
- Interfaces
- Blocks and Block Bodies
- Local Variable Declarations
- Statements
 - ◆ Empty Statement
 - ◆ Block Statement
 - ◆ Method Invocation
 - ◆ Allocation Statements
 - ◆ Assignment Statements
 - ◆ The if Statement
 - ◆ Statement Labels
 - ◆ The switch Statement
 - ◆ The break Statement
 - ◆ The for Statement
 - ◆ The while Statement
 - ◆ The do Statement
 - ◆ The continue Statement
 - ◆ The synchronized Statement
 - ◆ The try Statement
 - ◆ The return Statement
- Operators
- Summary

Chapter 12 *Portable Software and the java.lang Package*

- The Object and Class Classes
 - ◆ Object
 - ◆ Class
 - ◆ A Touch of Class
- The ClassLoader, SecurityManager, and Runtime Classes
 - ◆ ClassLoader
 - ◆ SecurityManager
 - ◆ Runtime
- The System Class
 - ◆ Property-Related Methods
 - ◆ Security Manager-Related Methods
 - ◆ Runtime-Related Methods
 - ◆ Odds and Ends
 - ◆ Time and Properties
- Wrapped Classes
 - ◆ The Boolean Class
 - ◆ The Character Class
 - ◆ The Integer and Long Classes
 - ◆ The Double and Float Classes
 - ◆ The Number Class
 - ◆ All Wrapped Up
- The Math Class
- The String and StringBuffer Classes
 - ◆ String Literals
 - ◆ The + Operator and StringBuffer
 - ◆ String Constructors
 - ◆ String Access Methods
 - ◆ The StringBuffer Class
- Threads and Processes
 - ◆ Runnable
 - ◆ Thread

- ◆ ThreadGroup
- ◆ Process
- ◆ Hello Again
- The Compiler Class
- Exceptions and Errors
 - ◆ The Throwable Class
 - ◆ The Error Class
 - ◆ The Exception Class
- Summary

Chapter 13 *Stream-Based Input/Output and the `java.io` Package*

- Streams
- The `java.io` Class Hierarchy
- The `InputStream` Class
 - ◆ The `read()` Method
 - ◆ The `available()` Method
 - ◆ The `close()` Method
 - ◆ Markable Streams
 - ◆ The `skip()` Method
- The `OutputStream` Class
 - ◆ The `write()` Method
 - ◆ The `flush()` Method
 - ◆ The `close()` Method
- Byte Array I/O
 - ◆ The `ByteArrayInputStream` Class
 - ◆ The `ByteArrayOutputStream` Class
 - ◆ The `ByteArrayIOApp` Program
 - ◆ The `StringBufferInputStream` Class
- File I/O
 - ◆ The `File` Class
 - ◆ The `FileDescriptor` Class
 - ◆ The `FileInputStream` Class
 - ◆ The `FileOutputStream` Class
 - ◆ The `FileIOApp` Program
- The `SequenceInputStream` Class
 - ◆ The `SequenceIOApp` Program
- Filtered I/O
 - ◆ The `FilterInputStream` Class
 - ◆ The `FilterOutputStream` Class
 - ◆ Buffered I/O
 - ◆ `PushbackInputStream`
 - ◆ The `LineNumberInputStream` Class
 - ◆ Data I/O
 - ◆ The `PrintStream` Class
 - ◆ Piped I/O
- The `RandomAccessFile` Class
 - ◆ The `RandomIOApp` Program
- The `StreamTokenizer` Class
 - ◆ The `StreamTokenApp` Program
- Summary

Chapter 14 *Useful Tools in the `java.util` Package*

- The `Date` Class
 - ◆ `DateApp`
- The `Random` Class
 - ◆ `RandomApp`
- The `Enumeration` Interface
- The `Vector` Class
 - ◆ `VectorApp`
- The `Stack` Class
 - ◆ `StackApp`

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts



Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research



With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips



Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.