

C O R E
JAVA

Volume I—Fundamentals

ELEVENTH EDITION



CAY S. HORSTMANN

Contents

Preface

Acknowledgments

Chapter 1: An Introduction to Java

- 1.1 Java as a Programming Platform
- 1.2 The Java “White Paper” Buzzwords
 - 1.2.1 Simple
 - 1.2.2 Object-Oriented
 - 1.2.3 Distributed
 - 1.2.4 Robust
 - 1.2.5 Secure
 - 1.2.6 Architecture-Neutral
 - 1.2.7 Portable
 - 1.2.8 Interpreted
 - 1.2.9 High-Performance
 - 1.2.10 Multithreaded
 - 1.2.11 Dynamic
- 1.3 Java Applets and the Internet
- 1.4 A Short History of Java
- 1.5 Common Misconceptions about Java

Chapter 2: The Java Programming Environment

- 2.1 Installing the Java Development Kit
 - 2.1.1 Downloading the JDK
 - 2.1.2 Setting up the JDK
 - 2.1.3 Installing Source Files and Documentation
- 2.2 Using the Command-Line Tools
- 2.3 Using an Integrated Development Environment
- 2.4 JShell

Chapter 3: Fundamental Programming Structures in Java

- 3.1 A Simple Java Program
- 3.2 Comments
- 3.3 Data Types
 - 3.3.1 Integer Types
 - 3.3.2 Floating-Point Types
 - 3.3.3 The char Type
 - 3.3.4 Unicode and the char Type
 - 3.3.5 The boolean Type
- 3.4 Variables and Constants
 - 3.4.1 Declaring Variables
 - 3.4.2 Initializing Variables
 - 3.4.3 Constants
 - 3.4.4 Enumerated Types
- 3.5 Operators
 - 3.5.1 Arithmetic Operators
 - 3.5.2 Mathematical Functions and Constants
 - 3.5.3 Conversions between Numeric Types
 - 3.5.4 Casts
 - 3.5.5 Combining Assignment with Operators
 - 3.5.6 Increment and Decrement Operators
 - 3.5.7 Relational and boolean Operators
 - 3.5.8 Bitwise Operators
 - 3.5.9 Parentheses and Operator Hierarchy
- 3.6 Strings
 - 3.6.1 Substrings
 - 3.6.2 Concatenation
 - 3.6.3 Strings Are Immutable
 - 3.6.4 Testing Strings for Equality
 - 3.6.5 Empty and Null Strings
 - 3.6.6 Code Points and Code Units
 - 3.6.7 The String API
 - 3.6.8 Reading the Online API Documentation
 - 3.6.9 Building Strings
- 3.7 Input and Output
 - 3.7.1 Reading Input
 - 3.7.2 Formatting Output
 - 3.7.3 File Input and Output
- 3.8 Control Flow
 - 3.8.1 Block Scope
 - 3.8.2 Conditional Statements

3.8.3	Loops
3.8.4	Determinate Loops
3.8.5	Multiple Selections—The switch Statement
3.8.6	Statements That Break Control Flow
3.9	Big Numbers
3.10	Arrays
3.10.1	Declaring Arrays
3.10.2	Accessing Array Elements
3.10.3	The “for each” Loop
3.10.4	Array Copying
3.10.5	Command-Line Parameters
3.10.6	Array Sorting
3.10.7	Multidimensional Arrays
3.10.8	Ragged Arrays

Chapter 4: Objects and Classes

4.1	Introduction to Object-Oriented Programming
4.1.1	Classes
4.1.2	Objects
4.1.3	Identifying Classes
4.1.4	Relationships between Classes
4.2	Using Predefined Classes
4.2.1	Objects and Object Variables
4.2.2	The LocalDate Class of the Java Library
4.2.3	Mutator and Accessor Methods
4.3	Defining Your Own Classes
4.3.1	An Employee Class
4.3.2	Use of Multiple Source Files
4.3.3	Dissociating the Employee Class
4.3.4	First Steps with Constructors
4.3.5	Declaring Local Variables with var
4.3.6	Working with null References
4.3.7	Implicit and Explicit Parameters
4.3.8	Benefits of Encapsulation
4.3.9	Class-Based Access Privileges
4.3.10	Private Methods
4.3.11	Final Instance Fields
4.4	Static Fields and Methods
4.4.1	Static Fields
4.4.2	Static Constants
4.4.3	Static Methods
4.4.4	Factory Methods
4.4.5	The main Method
4.5	Method Parameters
4.6	Object Construction
4.6.1	Overloading
4.6.2	Default Field Initialization
4.6.3	The Constructor with No Arguments
4.6.4	Explicit Field Initialization
4.6.5	Parameter Names
4.6.6	Calling Another Constructor
4.6.7	Initialization Blocks
4.6.8	Object Destruction and the finalize Method
4.7	Packages
4.7.1	Package Names
4.7.2	Class Importation
4.7.3	Static Imports
4.7.4	Addition of a Class into a Package
4.7.5	Package Access
4.7.6	The Class Path
4.7.7	Setting the Class Path
4.8	JAR Files
4.8.1	Creating JAR Files
4.8.2	The Manifest
4.8.3	Executable JAR Files
4.8.4	Multi-Release JAR Files
4.8.5	A Note about Command-Line Options
4.9	Documentation Comments
4.9.1	Comment Insertion
4.9.2	Class Comments
4.9.3	Method Comments
4.9.4	Field Comments
4.9.5	General Comments
4.9.6	Package Comments
4.9.7	Comment Extraction
4.10	Class Design Hints

Chapter 5: Inheritance

5.1	Classes, Superclasses, and Subclasses
5.1.1	Defining Subclasses
5.1.2	Overriding Methods
5.1.3	Subclass Constructors
5.1.4	Inheritance Hierarchies
5.1.5	Polymorphism
5.1.6	Understanding Method Calls
5.1.7	Preventing Inheritance: Final Classes and Methods
5.1.8	Casting
5.1.9	Abstract Classes
5.1.10	Protected Access
5.2	Object: The Cosmic Superclass
5.2.1	Variables of Type Object
5.2.2	The equals Method
5.2.3	Equality Testing and Inheritance
5.2.4	The hashCode Method
5.2.5	The toString Method
5.3	Generic Array Lists
5.3.1	Declaring Array Lists
5.3.2	Accessing Array List Elements
5.3.3	Compatibility between Typed and Raw Array Lists
5.4	Object Wrappers and Autoboxing
5.5	Methods with a Variable Number of Parameters
5.6	Enumeration Classes
5.7	Reflection
5.7.1	The Class Class
5.7.2	A Primer on Declaring Exceptions
5.7.3	Resources
5.7.4	Using Reflection to Analyze the Capabilities of Classes
5.7.5	Using Reflection to Analyze Objects at Runtime
5.7.6	Using Reflection to Write Generic Array Code
5.7.7	Invoking Arbitrary Methods and Constructors
5.8	Design Hints for Inheritance

Chapter 6: Interfaces, Lambda Expressions, and Inner Classes

6.1	Interfaces
6.1.1	The Interface Concept
6.1.2	Properties of Interfaces
6.1.3	Interfaces and Abstract Classes
6.1.4	Static and Private Methods
6.1.5	Default Methods
6.1.6	Resolving Default Method Conflicts
6.1.7	Interfaces and Callbacks
6.1.8	The Comparator Interface
6.1.9	Object Cloning

- 6.2 Lambda Expressions
 - 6.2.1 Why Lambdas?
 - 6.2.2 The Syntax of Lambda Expressions
 - 6.2.3 Functional Interfaces
 - 6.2.4 Method References
 - 6.2.5 Constructor References
 - 6.2.6 Variable Scope
 - 6.2.7 Processing Lambda Expressions
 - 6.2.8 More about Comparators
- 6.3 Inner Classes
 - 6.3.1 Use of an Inner Class to Access Object State
 - 6.3.2 Special Syntax Rules for Inner Classes
 - 6.3.3 Are Inner Classes Useful? Actually Necessary? Secure?
 - 6.3.4 Local Inner Classes
 - 6.3.5 Accessing Variables from Outer Methods
 - 6.3.6 Anonymous Inner Classes
 - 6.3.7 Static Inner Classes
- 6.4 Service Loaders
- 6.5 Proxies
 - 6.5.1 When to Use Proxies
 - 6.5.2 Creating Proxy Objects
 - 6.5.3 Properties of Proxy Classes

Chapter 7: Exceptions, Assertions, and Logging

- 7.1 Dealing with Errors
 - 7.1.1 The Classification of Exceptions
 - 7.1.2 Declaring Checked Exceptions
 - 7.1.3 How to Throw an Exception
 - 7.1.4 Creating Exception Classes
- 7.2 Catching Exceptions
 - 7.2.1 Catching an Exception
 - 7.2.2 Catching Multiple Exceptions
 - 7.2.3 Rethrowing and Chaining Exceptions
 - 7.2.4 The finally Clause
 - 7.2.5 The try-with-Resources Statement
 - 7.2.6 Analyzing Stack Trace Elements
- 7.3 Tips for Using Exceptions
- 7.4 Using Assertions

 - 7.4.1 The Assertion Concept
 - 7.4.2 Assertion Enabling and Disabling
 - 7.4.3 Using Assertions for Parameter Checking
 - 7.4.4 Using Assertions for Documenting Assumptions
- 7.5 Logging
 - 7.5.1 Basic Logging
 - 7.5.2 Advanced Logging
 - 7.5.3 Changing the Log Manager Configuration
 - 7.5.4 Localization
 - 7.5.5 Handlers
 - 7.5.6 Filters
 - 7.5.7 Formatters
 - 7.5.8 A Logging Recipe
- 7.6 Debugging Tips

Chapter 8: Generic Programming

- 8.1 Why Generic Programming?
 - 8.1.1 The Advantage of Type Parameters
 - 8.1.2 Who Wants to Be a Generic Programmer?
- 8.2 Defining a Simple Generic Class
- 8.3 Generic Methods
- 8.4 Bounds for Type Variables
- 8.5 Generic Code and the Virtual Machine
 - 8.5.1 Type Erasure
 - 8.5.2 Translating Generic Expressions
 - 8.5.3 Translating Generic Methods
 - 8.5.4 Calling Legacy Code
- 8.6 Restrictions and Limitations
 - 8.6.1 Type Parameters Cannot Be Instantiated with Primitive Types
 - 8.6.2 Runtime Type Inquiry Only Works with Raw Types
 - 8.6.3 You Cannot Create Arrays of Parameterized Types
 - 8.6.4 Varargs Warnings
 - 8.6.5 You Cannot Instantiate Type Variables
 - 8.6.6 You Cannot Construct a Generic Array

- 8.6.7 Type Variables Are Not Valid in Static Contexts of Generic Classes
- 8.6.8 You Cannot Throw or Catch Instances of a Generic Class
- 8.6.9 You Can Defeat Checked Exception Checking
- 8.6.10 Beware of Clashes after Erasure
- 8.7 Inheritance Rules for Generic Types
- 8.8 Wildcard Types
- 8.8.1 The Wildcard Concept
- 8.8.2 Supertype Bounds for Wildcards
- 8.8.3 Unbounded Wildcards
- 8.8.4 Wildcard Capture
- 8.9 Reflection and Generics
- 8.9.1 The `GenericClass` Class
- 8.9.2 Using `Class<T>` Parameters for Type Matching
- 8.9.3 Generic Type Information in the Virtual Machine
- 8.9.4 Type Literals

Chapter 9: Collections

- 9.1 The Java Collections Framework
- 9.1.1 Separating Collection Interfaces and Implementation
- 9.1.2 The `Collection` Interface
- 9.1.3 Iterators
- 9.1.4 Generic Utility Methods
- 9.2 Interfaces in the Collections Framework
- 9.3 Concrete Collections
- 9.3.1 Linked Lists
- 9.3.2 Array Lists
- 9.3.3 Hash Sets
- 9.3.4 Tree Sets
- 9.3.5 Queues and Deques
- 9.3.6 Priority Queues
- 9.4 Maps
- 9.4.1 Basic Map Operations
- 9.4.2 Updating Map Entries
- 9.4.3 Map Views
- 9.4.4 Weak Hash Maps
- 9.4.5 Linked Hash Sets and Maps
- 9.4.6 Enumeration Sets and Maps
- 9.4.7 Identity Hash Maps
- 9.5 Views and Wrappers
- 9.5.1 Small Collections
- 9.5.2 Subranges
- 9.5.3 Unmodifiable Views
- 9.5.4 Synchronized Views
- 9.5.5 Checked Views
- 9.5.6 A Note on Optional Operations
- 9.6 Algorithms
- 9.6.1 Why Generic Algorithms?
- 9.6.2 Sorting and Shuffling
- 9.6.3 Binary Search
- 9.6.4 Simple Algorithms
- 9.6.5 Bulk Operations
- 9.6.6 Converting between Collections and Arrays
- 9.6.7 Writing Your Own Algorithms
- 9.7 Legacy Collections
- 9.7.1 The `Hashtable` Class
- 9.7.2 Enumerations
- 9.7.3 Property Maps
- 9.7.4 Stacks
- 9.7.5 Bit Sets

Chapter 10: Graphical User Interface Programming

- 10.1 A History of Java User Interface Toolkits
 - 10.2 Displaying Frames
 - 10.2.1 Creating a Frame
 - 10.2.2 Frame Properties
 - 10.3 Displaying Information in a Component
 - 10.3.1 Working with 2D Shapes
 - 10.3.2 Using Color
 - 10.3.3 Using Fonts
-

- 10.3.4 Displaying Images
- 10.4 Event Handling
 - 10.4.1 Basic Event Handling Concepts
 - 10.4.2 Example: Handling a Button Click
 - 10.4.3 Specifying Listeners Concisely
 - 10.4.4 Adapter Classes
 - 10.4.5 Actions
 - 10.4.6 Mouse Events
 - 10.4.7 The AWT Event Hierarchy
- 10.5 The Preferences API

Chapter 11: User Interface Components with Swing

- 11.1 Swing and the Model-View-Controller Design Pattern
- 11.2 Introduction to Layout Management
 - 11.2.1 Layout Managers
 - 11.2.2 Border Layout
 - 11.2.3 Grid Layout
 - 11.3 Text Input
 - 11.3.1 Text Fields
 - 11.3.2 Labels and Labeling Components
 - 11.3.3 Password Fields
 - 11.3.4 Text Areas
 - 11.3.5 Scroll Panes
 - 11.4 Choice Components
 - 11.4.1 Checkboxes
 - 11.4.2 Radio Buttons
 - 11.4.3 Borders
 - 11.4.4 Combo Boxes
 - 11.4.5 Sliders
 - 11.5 Menus
 - 11.5.1 Menu Building
 - 11.5.2 Icons in Menu Items
 - 11.5.3 Checkbox and Radio Button Menu Items
 - 11.5.4 Pop-Up Menus
 - 11.5.5 Keyboard Mnemonics and Accelerators
 - 11.5.6 Enabling and Disabling Menu Items
 - 11.5.7 Toolbars
 - 11.5.8 Tooltips
 - 11.6 Sophisticated Layout Management
 - 11.6.1 The Grid Bag Layout
 - 11.6.1.1 The `gridx`, `gridy`, `gridwidth`, and `gridheight` Parameters
 - 11.6.1.2 Weight Fields
 - 11.6.1.3 The `fill` and `anchor` Parameters
 - 11.6.1.4 Padding
 - 11.6.1.5 Alternative Method to Specify the `gridx`, `gridy`, `gridwidth`, and `gridheight` Parameters
 - 11.6.1.6 A Grid Bag Layout Recipe
 - 11.6.1.7 A Helper Class to Tame the Grid Bag Constraints
 - 11.6.2 Custom Layout Managers
- 11.7 Dialog Boxes
 - 11.7.1 Option Dialogs
 - 11.7.2 Creating Dialogs
 - 11.7.3 Data Exchange
 - 11.7.4 File Dialogs

Chapter 12: Concurrency

- 12.1 What Are Threads?
- 12.2 Thread States
 - 12.2.1 New Threads
 - 12.2.2 Runnable Threads
 - 12.2.3 Blocked and Waiting Threads
 - 12.2.4 Terminated Threads
- 12.3 Thread Properties
 - 12.3.1 Interrupting Threads
 - 12.3.2 Daemon Threads
 - 12.3.3 Thread Names
 - 12.3.4 Handlers for Uncaught Exceptions
 - 12.3.5 Thread Priorities
- 12.4 Synchronization

- 12.4.1 An Example of a Race Condition
- 12.4.2 The Race Condition Explained
- 12.4.3 Lock Objects
- 12.4.4 Condition Objects
- 12.4.5 The synchronized Keyword
- 12.4.6 Synchronized Blocks
- 12.4.7 The Monitor Concept
- 12.4.8 Volatile Fields
- 12.4.9 Final Variables
- 12.4.10 Atomics
- 12.4.11 Deadlocks
- 12.4.12 Thread-Local Variables
- 12.4.13 Why the stop and suspend Methods Are Deprecated
- 12.5 Thread-Safe Collections
 - 12.5.1 Blocking Queues
 - 12.5.2 Efficient Maps, Sets, and Queues
 - 12.5.3 Atomic Update of Map Entries
 - 12.5.4 Bulk Operations on Concurrent Hash Maps
 - 12.5.5 Concurrent Set Views
 - 12.5.6 Copy on Write Arrays
 - 12.5.7 Parallel Array Algorithms
 - 12.5.8 Older Thread-Safe Collections
- 12.6 Tasks and Thread Pools
 - 12.6.1 Callables and Futures
 - 12.6.2 Executors
 - 12.6.3 Controlling Groups of Tasks
 - 12.6.4 The Fork-Join Framework
- 12.7 Asynchronous Computations
 - 12.7.1 Completable Futures
 - 12.7.2 Composing Completable Futures
 - 12.7.3 Long-Running Tasks in User Interface Callbacks
- 12.8 Processes
 - 12.8.1 Building a Process
 - 12.8.2 Running a Process
 - 12.8.3 Process Handles

Appendix

Index
