

ORACLE
PRESS



CORE JAVA

Volume I: Fundamentals

THIRTEENTH EDITION

ORACLE

Cay S. Horstmann

Core Java, Volume I: Fundamentals

Table of Contents

Cover

Title

Copyright

Table of Contents

Preface

 To the Reader

 A Tour of This Book

 Conventions

 Sample Code

Acknowledgments

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

2. The Java Programming Environment

 2.1. Installing the Java Development Kit

 2.1.1. Downloading the JDK

Table of Contents

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

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. Assignment

3.5.6. Increment and Decrement Operators

3.5.7. Relational and boolean Operators

3.5.8. The Conditional Operator

3.5.9. Switch Expressions

3.5.10. Bitwise Operators

3.5.11. Parentheses and Operator Hierarchy

3.6. Strings

3.6.1. Concatenation

3.6.2. Splitting Strings

3.6.3. Indexes and Substrings

3.6.4. Strings Are Immutable

Table of Contents

- 3.6.5. Testing Strings for Equality
- 3.6.6. Empty and Null Strings
- 3.6.7. The String API
- 3.6.8. Reading the Online API Documentation
- 3.6.9. Building Strings
- 3.6.10. Text Blocks

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 with switch
- 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 Arguments
- 3.10.6. Array Sorting
- 3.10.7. Multidimensional Arrays
- 3.10.8. Ragged Arrays

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

Table of Contents

4.3. Defining Your Own Classes

- 4.3.1. An Employee Class
- 4.3.2. Use of Multiple Source Files
- 4.3.3. Dissecting 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. Records

- 4.7.1. The Record Concept
- 4.7.2. Constructors: Canonical, Compact, and Custom

4.8. Packages

- 4.8.1. Package Names
- 4.8.2. Class Importation
- 4.8.3. Static Imports
- 4.8.4. Addition of a Class into a Package
- 4.8.5. Package Access

Table of Contents

4.8.6. The Class Path

4.8.7. Setting the Class Path

4.9. JAR Files

4.9.1. Creating JAR files

4.9.2. The Manifest

4.9.3. Executable JAR Files

4.9.4. Multi-Release JAR Files

4.9.5. A Note about Command-Line Options

4.10. Documentation Comments

4.10.1. Comment Insertion

4.10.2. Class Comments

4.10.3. Method Comments

4.10.4. Field Comments

4.10.5. Package Comments

4.10.6. HTML Markup

4.10.7. Links

4.10.8. General Comments

4.10.9. Code Snippets

4.10.10. Comment Extraction

4.11. Class Design Hints

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. Pattern Matching for instanceof

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

Table of Contents

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 Arguments

5.6. Abstract Classes

5.7. Enumeration Classes

5.8. Sealed Classes

5.9. Pattern Matching

5.9.1. Null Handling

5.9.2. Guards

5.9.3. Exhaustiveness

5.9.4. Dominance

5.9.5. Patterns and Constants

5.9.6. Variable Scope and Fallthrough

5.10. Reflection

5.10.1. The Class Class

5.10.2. A Primer on Declaring Exceptions

5.10.3. Resources

5.10.4. Using Reflection to Analyze the Capabilities of Classes

5.10.5. Using Reflection to Analyze Objects at Runtime

5.10.6. Using Reflection to Write Generic Array Code

5.10.7. Invoking Arbitrary Methods and Constructors

5.11. Design Hints for Inheritance

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

Table of Contents

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. Function Types

6.2.5. Method References

6.2.6. Constructor References

6.2.7. Variable Scope

6.2.8. Processing Lambda Expressions

6.2.9. Creating 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 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

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

Table of Contents

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. Should You Use the Java Logging Framework?

7.5.2. Logging 101

7.5.3. The Platform Logging API

7.5.4. Logging Configuration

7.5.5. Log Handlers

7.5.6. Filters and Formatters

7.5.7. A Logging Recipe

7.6. Debugging Tips

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

8.5.5. Generic Record Patterns

8.6. Inheritance Rules for Generic Types

8.7. Wildcard Types

8.7.1. The Wildcard Concept

8.7.2. Supertype Bounds for Wildcards

8.7.3. Unbounded Wildcards

8.7.4. Wildcard Capture

8.8. Restrictions and Limitations

8.8.1. Type Parameters Cannot Be Instantiated with Primitive Types

Table of Contents

- 8.8.2. Runtime Type Inquiry Only Works with Raw Types
- 8.8.3. You Cannot Create Arrays of Parameterized Types
- 8.8.4. Varargs Warnings
- 8.8.5. Generic Varargs Do Not Spread Primitive Arrays
- 8.8.6. You Cannot Instantiate Type Variables
- 8.8.7. You Cannot Construct a Generic Array
- 8.8.8. Type Variables Are Not Valid in Static Contexts of Generic Classes
- 8.8.9. You Cannot Throw or Catch Instances of a Generic Class
- 8.8.10. You Can Defeat Checked Exception Checking
- 8.8.11. Beware of Clashes after Erasure
- 8.8.12. Type Inference in Generic Record Patterns is Limited

8.9. Reflection and Generics

- 8.9.1. The Generic Class 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

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

Table of Contents

9.4.7. Identity Hash Maps

9.5. Copies and Views

9.5.1. Small Collections

9.5.2. Unmodifiable Copies and Views

9.5.3. Subranges

9.5.4. Sets From Boolean-Valued Maps

9.5.5. Reversed Views

9.5.6. Checked Views

9.5.7. Synchronized Views

9.5.8. 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. System Properties

9.7.5. Stacks

9.7.6. Bit Sets

10. Concurrency

10.1. Running Threads

10.2. Thread States

10.2.1. New Threads

10.2.2. Runnable Threads

10.2.3. Blocked and Waiting Threads

10.2.4. Terminated Threads

10.3. Thread Properties

10.3.1. Virtual Threads

10.3.2. Thread Interruption

10.3.3. Daemon Threads

10.3.4. Thread Names and Ids

Table of Contents

10.3.5. Handlers for Uncaught Exceptions

10.3.6. Thread Priorities

10.3.7. Thread Factories and Builders

10.4. Coordinating Tasks

10.4.1. Callables and Futures

10.4.2. Executors

10.4.3. Invoking a Group of Tasks

10.4.4. Thread-Local Variables

10.4.5. The Fork-Join Framework

10.5. Synchronization

10.5.1. An Example of a Race Condition

10.5.2. The Race Condition Explained

10.5.3. Lock Objects

10.5.4. Condition Objects

10.5.5. Deadlocks

10.5.6. The synchronized Keyword

10.5.7. Synchronized Blocks

10.5.8. The Monitor Concept

10.5.9. Volatile Fields

10.5.10. Final Fields

10.5.11. Atomics

10.5.12. On-Demand Initialization

10.5.13. Safe Publication

10.5.14. Sharing with Thread-Local Variables

10.6. Thread-Safe Collections

10.6.1. Blocking Queues

10.6.2. Efficient Maps, Sets, and Queues

10.6.3. Atomic Update of Map Entries

10.6.4. Bulk Operations on Concurrent Hash Maps

10.6.5. Concurrent Set Views

10.6.6. Copy on Write Arrays

10.6.7. Parallel Array Algorithms

10.6.8. Older Thread-Safe Collections

10.7. Asynchronous Computations

10.7.1. Completable Futures

10.7.2. Composing Completable Futures

10.7.3. Long-Running Tasks in User-Interface Callbacks

Table of Contents

10.8. Processes

- 10.8.1. Building a Process
- 10.8.2. Running a Process
- 10.8.3. Process Handles

11. Annotations

11.1. Using Annotations

- 11.1.1. Annotation Elements
- 11.1.2. Multiple and Repeated Annotations
- 11.1.3. Annotating Declarations
- 11.1.4. Annotating Type Uses
- 11.1.5. Making Receivers Explicit

11.2. Defining Annotations

11.3. Annotations in the Java API

- 11.3.1. Annotations for Compilation
- 11.3.2. Meta-Annotations

11.4. Processing Annotations at Runtime

11.5. Source-Level Annotation Processing

- 11.5.1. Annotation Processors
- 11.5.2. The Language Model API
- 11.5.3. Using Annotations to Generate Source Code

11.6. Bytecode Engineering

- 11.6.1. Modifying Class Files
- 11.6.2. Modifying Bytecodes at Load Time

12. The Java Platform Module System

12.1. The Module Concept

12.2. Naming Modules

12.3. The Modular Hello, World! Program

12.4. Requiring Modules

12.5. Exporting Packages

12.6. Modular JARs

12.7. Modules and Reflective Access

12.8. Automatic Modules

12.9. The Unnamed Module

12.10. Command-Line Flags for Migration

12.11. Transitive and Static Requirements

Table of Contents

12.12. Qualified Exporting and Opening

12.13. Service Loading

12.14. Tools for Working with Modules

Appendix

Index