



Core Java[®] SE 9

for the Impatient

Second Edition

Cay S. Horstmann



Core Java[®] SE 9 for the Impatient

Second Edition

Core Java SE 9 for the Impatient

Table of Contents

Cover

Title Page

Copyright Page

Contents

Preface

Acknowledgments

About the Author

1 FUNDAMENTAL PROGRAMMING STRUCTURES

1.1 Our First Program

1.1.1 Dissecting the Hello, World Program

1.1.2 Compiling and Running a Java Program

1.1.3 Method Calls

1.1.4 JShell

1.2 Primitive Types

1.2.1 Signed Integer Types

1.2.2 Floating-Point Types

1.2.3 The char Type

1.2.4 The boolean Type

1.3 Variables

1.3.1 Variable Declarations

1.3.2 Names

1.3.3 Initialization

1.3.4 Constants

Table of Contents

1.4 Arithmetic Operations

- 1.4.1 Assignment
- 1.4.2 Basic Arithmetic
- 1.4.3 Mathematical Methods
- 1.4.4 Number Type Conversions
- 1.4.5 Relational and Logical Operators
- 1.4.6 Big Numbers

1.5 Strings

- 1.5.1 Concatenation
- 1.5.2 Substrings
- 1.5.3 String Comparison
- 1.5.4 Converting Between Numbers and Strings
- 1.5.5 The String API
- 1.5.6 Code Points and Code Units

1.6 Input and Output

- 1.6.1 Reading Input
- 1.6.2 Formatted Output

1.7 Control Flow

- 1.7.1 Branches
- 1.7.2 Loops
- 1.7.3 Breaking and Continuing
- 1.7.4 Local Variable Scope

1.8 Arrays and Array Lists

- 1.8.1 Working with Arrays
- 1.8.2 Array Construction
- 1.8.3 Array Lists
- 1.8.4 Wrapper Classes for Primitive Types
- 1.8.5 The Enhanced Loop
- 1.8.6 Copying Arrays and Array Lists

Table of Contents

1.8.7 Array Algorithms

1.8.8 Command-Line Arguments

1.8.9 Multidimensional Arrays

1.9 Functional Decomposition

1.9.1 Declaring and Calling Static Methods

1.9.2 Array Parameters and Return Values

1.9.3 Variable Arguments

Exercises

2 OBJECT-ORIENTED PROGRAMMING

2.1 Working with Objects

2.1.1 Accessor and Mutator Methods

2.1.2 Object References

2.2 Implementing Classes

2.2.1 Instance Variables

2.2.2 Method Headers

2.2.3 Method Bodies

2.2.4 Instance Method Invocations

2.2.5 The this Reference

2.2.6 Call by Value

2.3 Object Construction

2.3.1 Implementing Constructors

2.3.2 Overloading

2.3.3 Calling One Constructor from Another

2.3.4 Default Initialization

2.3.5 Instance Variable Initialization

2.3.6 Final Instance Variables

2.3.7 The Constructor with No Arguments

2.4 Static Variables and Methods

2.4.1 Static Variables

Table of Contents

2.4.2 Static Constants

2.4.3 Static Initialization Blocks

2.4.4 Static Methods

2.4.5 Factory Methods

2.5 Packages

2.5.1 Package Declarations

2.5.2 The jar Command

2.5.3 The Class Path

2.5.4 Package Access

2.5.5 Importing Classes

2.5.6 Static Imports

2.6 Nested Classes

2.6.1 Static Nested Classes

2.6.2 Inner Classes

2.6.3 Special Syntax Rules for Inner Classes

2.7 Documentation Comments

2.7.1 Comment Insertion

2.7.2 Class Comments

2.7.3 Method Comments

2.7.4 Variable Comments

2.7.5 General Comments

2.7.6 Links

2.7.7 Package, Module, and Overview Comments

2.7.8 Comment Extraction

Exercises

3 INTERFACES AND LAMBDA EXPRESSIONS

3.1 Interfaces

3.1.1 Declaring an Interface

3.1.2 Implementing an Interface

Table of Contents

- 3.1.3 Converting to an Interface Type
- 3.1.4 Casts and the instance of Operator
- 3.1.5 Extending Interfaces
- 3.1.6 Implementing Multiple Interfaces
- 3.1.7 Constants
- 3.2 Static, Default, and Private Methods**
 - 3.2.1 Static Methods
 - 3.2.2 Default Methods
 - 3.2.3 Resolving Default Method Conflicts
 - 3.2.4 Private Methods
- 3.3 Examples of Interfaces**
 - 3.3.1 The Comparable Interface
 - 3.3.2 The Comparator Interface
 - 3.3.3 The Runnable Interface
 - 3.3.4 User Interface Callbacks
- 3.4 Lambda Expressions**
 - 3.4.1 The Syntax of Lambda Expressions
 - 3.4.2 Functional Interfaces
- 3.5 Method and Constructor References**
 - 3.5.1 Method References
 - 3.5.2 Constructor References
- 3.6 Processing Lambda Expressions**
 - 3.6.1 Implementing Deferred Execution
 - 3.6.2 Choosing a Functional Interface
 - 3.6.3 Implementing Your Own Functional Interfaces
- 3.7 Lambda Expressions and Variable Scope**
 - 3.7.1 Scope of a Lambda Expression
 - 3.7.2 Accessing Variables from the Enclosing Scope
- 3.8 Higher-Order Functions**

Table of Contents

3.8.1 Methods that Return Functions

3.8.2 Methods That Modify Functions

3.8.3 Comparator Methods

3.9 Local and Anonymous Classes

3.9.1 Local Classes

3.9.2 Anonymous Classes

Exercises

4 INHERITANCE AND REFLECTION

4.1 Extending a Class

4.1.1 Super- and Subclasses

4.1.2 Defining and Inheriting Subclass Methods

4.1.3 Method Overriding

4.1.4 Subclass Construction

4.1.5 Superclass Assignments

4.1.6 Casts

4.1.7 Final Methods and Classes

4.1.8 Abstract Methods and Classes

4.1.9 Protected Access

4.1.10 Anonymous Subclasses

4.1.11 Inheritance and Default Methods

4.1.12 Method Expressions with

4.2 Object: The Cosmic Superclass

4.2.1 The toString Method

4.2.2 The equals Method

4.2.3 The hashCode Method

4.2.4 Cloning Objects

4.3 Enumerations

4.3.1 Methods of Enumerations

4.3.2 Constructors, Methods, and Fields

Table of Contents

4.3.3 Bodies of Instances

4.3.4 Static Members

4.3.5 Switching on an Enumeration

4.4 Runtime Type Information and Resources

4.4.1 The Class

4.4.2 Loading Resources

4.4.3 Class Loaders

4.4.4 The Context Class Loader

4.4.5 Service Loaders

4.5 Reflection

4.5.1 Enumerating Class Members

4.5.2 Inspecting Objects

4.5.3 Invoking Methods

4.5.4 Constructing Objects

4.5.5 JavaBeans

4.5.6 Working with Arrays

4.5.7 Proxies

Exercises

5 EXCEPTIONS, ASSERTIONS, AND LOGGING

5.1 Exception Handling

5.1.1 Throwing Exceptions

5.1.2 The Exception Hierarchy

5.1.3 Declaring Checked Exceptions

5.1.4 Catching Exceptions

5.1.5 The Try-with-Resources Statement

5.1.6 The finally Clause

5.1.7 Rethrowing and Chaining Exceptions

5.1.8 Uncaught Exceptions and the Stack Trace

5.1.9 The Objects.requireNonNull Method

Table of Contents

5.2 Assertions

5.2.1 Using Assertions

5.2.2 Enabling and Disabling Assertions

5.3 Logging

5.3.1 Using Loggers

5.3.2 Loggers

5.3.3 Logging Levels

5.3.4 Other Logging Methods

5.3.5 Logging Configuration

5.3.6 Log Handlers

5.3.7 Filters and Formatters

Exercises

6 GENERIC PROGRAMMING

6.1 Generic Classes

6.2 Generic Methods

6.3 Type Bounds

6.4 Type Variance and Wildcards

6.4.1 Subtype Wildcards

6.4.2 Supertype Wildcards

6.4.3 Wildcards with Type Variables

6.4.4 Unbounded Wildcards

6.4.5 Wildcard Capture

6.5 Generics in the Java Virtual Machine

6.5.1 Type Erasure

6.5.2 Cast Insertion

6.5.3 Bridge Methods

6.6 Restrictions on Generics

6.6.1 No Primitive Type Arguments

Table of Contents

- 6.6.2 At Runtime, All Types Are Raw
- 6.6.3 You Cannot Instantiate Type Variables
- 6.6.4 You Cannot Construct Arrays of Parameterized Types
- 6.6.5 Class Type Variables Are Not Valid in Static Contexts
- 6.6.6 Methods May Not Clash after Erasure
- 6.6.7 Exceptions and Generics

6.7 Reflection and Generics

- 6.7.1 The Class<T> Class
- 6.7.2 Generic Type Information in the Virtual Machine

Exercises

7 COLLECTIONS

- 7.1 An Overview of the Collections Framework
- 7.2 Iterators
- 7.3 Sets
- 7.4 Maps
- 7.5 Other Collections
 - 7.5.1 Properties
 - 7.5.2 Bit Sets
 - 7.5.3 Enumeration Sets and Maps
 - 7.5.4 Stacks, Queues, Deques, and Priority Queues
 - 7.5.5 Weak Hash Maps

7.6 Views

- 7.6.1 Small Collections
- 7.6.2 Ranges
- 7.6.3 Unmodifiable Views

Exercises

8 STREAMS

- 8.1 From Iterating to Stream Operations

Table of Contents

8.2 Stream Creation

8.3 The filter, map, and flatMap Methods

8.4 Extracting Substreams and Combining Streams

8.5 Other Stream Transformations

8.6 Simple Reductions

8.7 The Optional Type

8.7.1 How to Work with Optional Values

8.7.2 How Not to Work with Optional Values

8.7.3 Creating Optional Values

8.7.4 Composing Optional Value Functions with

8.7.5 Turning an Optional Into a Stream

8.8 Collecting Results

8.9 Collecting into Maps

8.10 Grouping and Partitioning

8.11 Downstream Collectors

8.12 Reduction Operations

8.13 Primitive Type Streams

8.14 Parallel Streams

Exercises

9 PROCESSING INPUT AND OUTPUT

9.1 Input/Output Streams, Readers, and Writers

9.1.1 Obtaining Streams

9.1.2 Reading Bytes

9.1.3 Writing Bytes

9.1.4 Character Encodings

9.1.5 Text Input

9.1.6 Text Output

9.1.7 Reading and Writing Binary Data

Table of Contents

9.1.8 Random-Access Files

9.1.9 Memory-Mapped Files

9.1.10 File Locking

9.2 Paths, Files, and Directories

9.2.1 Paths

9.2.2 Creating Files and Directories

9.2.3 Copying, Moving, and Deleting Files

9.2.4 Visiting Directory Entries

9.2.5 ZIP File Systems

9.3 HTTP Connections

9.3.1 The URLConnection and HttpURLConnection Classes

9.3.2 The HTTP Client API

9.4 Regular Expressions

9.4.1 The Regular Expression Syntax

9.4.2 Finding One Match

9.4.3 Finding All Matches

9.4.4 Groups

9.4.5 Splitting along Delimiters

9.4.6 Replacing Matches

9.4.7 Flags

9.5 Serialization

9.5.1 The Serializable Interface

9.5.2 Transient Instance Variables

9.5.3 The readObject and writeObject Methods

9.5.4 The readResolve and writeReplace Methods

9.5.5 Versioning

Exercises

10 CONCURRENT PROGRAMMING

10.1 Concurrent Tasks

Table of Contents

10.1.1 Running Tasks

10.1.2 Futures

10.2 Asynchronous Computations

10.2.1 Completable Futures

10.2.2 Composing Completable Futures

10.2.3 Long-Running Tasks in User-Interface Callbacks

10.3 Thread Safety

10.3.1 Visibility

10.3.2 Race Conditions

10.3.3 Strategies for Safe Concurrency

10.3.4 Immutable Classes

10.4 Parallel Algorithms

10.4.1 Parallel Streams

10.4.2 Parallel Array Operations

10.5 Threadsafe Data Structures

10.5.1 Concurrent Hash Maps

10.5.2 Blocking Queues

10.5.3 Other Threadsafe Data Structures

10.6 Atomic Counters and Accumulators

10.7 Locks and Conditions

10.7.1 Locks

10.7.2 The synchronized Keyword

10.7.3 Waiting on Conditions

10.8 Threads

10.8.1 Starting a Thread

10.8.2 Thread Interruption

10.8.3 Thread-Local Variables

10.8.4 Miscellaneous Thread Properties

10.9 Processes

Table of Contents

10.9.1 Building a Process

10.9.2 Running a Process

10.9.3 Process Handles

Exercises

11.4 Processing Annotations at Runtime

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 Standard Annotations

11.3.1 Annotations for Compilation

11.3.2 Annotations for Managing Resources

11.3.3 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

Exercises

12 THE DATE AND TIME API

12.1 The Time Line

12.2 Local Dates

12.3 Date Adjusters

12.4 Local Time

Table of Contents

- 12.5 Zoned Time
- 12.6 Formatting and Parsing
- 12.7 Interoperating with Legacy Code
- Exercises

13 INTERNATIONALIZATION

- 13.1 Locales
 - 13.1.1 Specifying a Locale
 - 13.1.2 The Default Locale
 - 13.1.3 Display Names
- 13.2 Number Formats
- 13.3 Currencies
- 13.4 Date and Time Formatting
- 13.5 Collation and Normalization
- 13.6 Message Formatting
- 13.7 Resource Bundles
 - 13.7.1 Organizing Resource Bundles
 - 13.7.2 Bundle Classes
- 13.8 Character Encodings
- 13.9 Preferences
- Exercises

14 COMPILING AND SCRIPTING

- 14.1 The Compiler API
 - 14.1.1 Invoking the Compiler
 - 14.1.2 Launching a Compilation Task
 - 14.1.3 Reading Source Files from Memory
 - 14.1.4 Writing Byte Codes to Memory
 - 14.1.5 Capturing Diagnostics
- 14.2 The Scripting API

Table of Contents

14.2.1 Getting a Scripting Engine

14.2.2 Bindings

14.2.3 Redirecting Input and Output

14.2.4 Calling Scripting Functions and Methods

14.2.5 Compiling a Script

14.3 The Nashorn Scripting Engine

14.3.1 Running Nashorn from the Command Line

14.3.2 Invoking Getters, Setters, and Overloaded Methods

14.3.3 Constructing Java Objects

14.3.4 Strings in JavaScript and Java

14.3.5 Numbers

14.3.6 Working with Arrays

14.3.7 Lists and Maps

14.3.8 Lambdas

14.3.9 Extending Java Classes and Implementing Java Interfaces

14.3.10 Exceptions

14.4 Shell Scripting with Nashorn

14.4.1 Executing Shell Commands

14.4.2 String Interpolation

14.4.3 Script Inputs

Exercises

15 THE JAVA PLATFORM MODULE SYSTEM

15.1 The Module Concept

15.2 Naming Modules

15.3 The Modular Hello, World! Program

15.4 Requiring Modules

15.5 Exporting Packages

15.6 Modules and Reflective Access

Table of Contents

15.7 Modular JARs

15.8 Automatic Modules and the Unnamed Module

15.9 Command-Line Flags for Migration

15.10 Transitive and Static Requirements

15.11 Qualified Exporting and Opening

15.12 Service Loading

15.13 Tools for Working with Modules

Exercises

Index