

Java SE 25 Programming Complete Ed 1

Contents

I Course Introduction

Course Goals

Audience

Course Structure

1 Introduction to Java

Objectives

What Is Java?

How Java Works

Object-Oriented Principles

Classes and Objects

Classes

Objects

Inheritance

Java APIs

Java Keywords, Reserved Words, and Special Identifiers

Java Naming Conventions

Java Basic Syntax Rules

Defining a Java Class

Accessing Classes Across Packages

Implementing Encapsulation with Access Modifiers

Creating a Java Program with Main Method

Compiling a Java Program

Executing a Java Program

Comments and Documentation

Code Snippets in Javadoc

External Snippets

Summary

Practices for Lesson 1: Overview

2 Primitive Types, Operators, and Flow Control Statements

Objectives

Java Primitives

Declaring and Initializing Primitive Variables

Primitive Declarations and Initializations: Restrictions

Java Operators

Assignment and Arithmetic Operators
Arithmetic Operations and Type Casting
More Mathematical Operations
Binary Number Representation
Bitwise Logical Operators
Equality, Relational, and Conditional Operators
Short-Circuit Evaluation
Flow Control Using if/else Construct
Ternary Operator
Flow Control Using switch Construct
Switch -> No Fall-Through Syntax
Switch Expressions yield a Value
Switch Statements and Expressions Summary
Using JShell (REPL Tool)
JShell Commands
Code Snippets
Summary
Practices for Lesson 2: Overview

3 Text, Date, Time, and Numeric Objects

Objectives
String Initialization
String Operations
String Indexing
Mutable Text Objects
Text Blocks
Spaces, Lines, and Quotes
Wrapper Classes for Primitives
Representing Numbers Using BigDecimal Class
Method Chaining
Local Date and Time API
More Local Date and Time Operations
Instants, Durations, and Periods
Zoned Date and Time
Representing Languages and Countries
Formatting and Parsing Numeric Values
Formatting and Parsing Date and Time Values
Localizable Resources
Formatting Message Patterns
Formatting and Localization: Summary
Summary

Practices for Lesson 3: Overview

4 Classes and Objects

Objectives

UML: Introduction

Modeling Classes

Modeling Interactions and Activities

Designing Classes

Creating Objects

Defining Instance Variables

Defining Instance Methods

Object Creation and Access: Example

Local Variables and Recursive Object Reference

Local Variable Type Inference

Defining Constants

Static Context

Accessing Static Context

Combining Static and Final

Other Static Context Use Cases

IntelliJ IDE: Introduction

Summary

Practices for Lesson 4: Overview

5 Improved Class Design

Objectives

Overload Methods

Variable Number of Arguments

Defining Constructors

Reusing Constructors

Access Modifiers: Summary

Defining Encapsulation

Defining Immutability

Constants and Immutability

Enumerations

Complex Enumerations

Java Memory Allocation

Parameter Passing

Java Memory Cleanup

Summary

Practices for Lesson 5: Overview

6 Implement Inheritance and Use Records

- Objectives
- Extending Classes
- Object Class
- Reusing Parent Class Code Through Inheritance
- Instantiating Classes and Accessing Objects
- Rules of Reference Type Casting
- Verifying Object Type Before Casting the Reference
- Pattern Matching for instanceof
- Reference Code Within the Current or Parent Object
- Defining Subclass Constructors
- Flexible Constructor Bodies
- Class and Object Initialization: Summary
- Overriding Methods and Using Polymorphism
- Reusing Parent Class Logic in Overwritten Method
- Defining Abstract Classes and Methods
- Defining Final Classes and Methods
- Sealed Classes and Interfaces
- Overriding Object Class Operations: toString
- Overriding Object Class Operations: equals
- Override Object Class Operations: hashCode
- Comparing String Objects
- Java Records
- Custom Record Constructors
- Record Patterns
- Pattern Matching for switch
- Factory Methods
- Summary
- Practices for Lesson 6: Overview

7 Interfaces and Generics

- Objectives
- Java Interfaces
- Multiple Inheritance Problem
- Implement Interfaces
- Default, Private, and Static Methods in Interfaces
- Interface Hierarchy
- Default Methods Inheritance
- Interface Is a Type
- Functional Interfaces
- Generics
- Use Generics

Examples of Java Interfaces: `java.lang.Comparable`

Examples of Java Interfaces: `java.util.Comparator`

Examples of Java Interfaces: `java.lang.Cloneable`

Composition Pattern

Summary

Practices for Lesson 7: Overview

8 Arrays and Loops

Objectives

Arrays

Combined Declaration, Creation, and Initialization of Arrays

Multidimensional Arrays

Copying Array Content

Arrays Class

Loops

Processing Arrays by Using Loops

Complex for Loops

Embedded Loops

Break and Continue

Summary

Practices for Lesson 8: Overview

9 Collections

Objectives

Introduction to Java Collection API

Java Collection API Interfaces

Java Collection API Implementation Classes

Java Collection API Interfaces and Implementation Classes

Create List Object

Manage List Contents

Create Set Object

Manage Set Contents

Create Deque Object

Manage Deque Contents

Create HashMap Object

Manage HashMap Contents

Iterate Through Collections

Sequenced Collections

Other Collection Behaviors

Use `java.util.Collections` Class

Access Collections Concurrently

Prevent Collections Corruption
Legacy Collection Classes
Summary
Practices for Lesson 9: Overview

10 Nested Classes and Lambda Expressions

Objectives
Types of Nested Classes
Static Nested Classes
Member Inner Classes
Local Inner Classes
Anonymous Inner Classes
Anonymous Inner Classes and Functional Interfaces
Understand Lambda Expressions
Define Lambda Expression Parameters and Body
Use Method References
Default and Static Methods in Functional Interfaces
Use Default and Static Methods of the Comparator Interface
Use Default and Static Methods of the Predicate Interface
Summary
Practices for Lesson 10: Overview

11 Java Streams API

Objectives
Characteristics of Streams
Create Streams Using Stream API
Stream Pipeline Processing Operations
Using Functional Interfaces
Primitive Variants of Functional Interfaces
Bi-argument Variants of Functional Interfaces
Perform Actions with Stream Pipeline Elements
Perform Filtering of Stream Pipeline Elements
Perform Mapping of Stream Pipeline Elements
Join Streams Using flatMap Operation
Other Intermediate Stream Operations
Custom Intermediate Stream Operations: Gatherers
Short-Circuit Terminal Operations
Process Stream Using count, min, max, sum, average Operations
Aggregate Stream Data using reduce Operation
General Logic of the collect Operation
Using Basic Collectors

- Perform a Conversion of a Collector Result
- Perform Grouping or Partitioning of the Stream Content
- Mapping and Filtering with Respect to Groups or Partitions
- Gatherers
- Parallel Stream Processing
- Parallel Stream Processing Guidelines
- Restrictions on Parallel Stream Processing
- Splitterator
- Splitterator Characteristics
- Summary
- Practices for Lesson 11: Overview

12 Exception Handling, Logging, and Debugging

- Objectives
- Using Java Logging API
- Logging Method Categories
- Guarded Logging
- Log Writing Handling
- Logging Configuration
- Describe Java Exceptions
- Create Custom Exceptions
- Throwing Exceptions
- Catching Exceptions
- Exceptions and the Execution Flow
- Helpful NullPointerExceptions
- Example Throwing an Unchecked Exception
- Example Throwing a Checked Exception
- Handling Exceptions
- Resource Auto-Closure
- Suppressed Exceptions
- Handle Exception Cause
- Java Debugger
- Debugger Actions
- Manipulate Program Data in Debug Mode
- Validate Program Logic Using Assertions
- Normal Program Flow with No Exceptions
- Program Flow Producing a Runtime Exception
- Program Flow Catching Specific Checked Exception
- Program Flow Catching Any Exceptions

Summary

Practices for Lesson 12: Overview

13 Java IO API

Objectives

Java Input-Output Principals

Java Input-Output API

Reading and Writing Binary Data

Basic Binary Data Reading and Writing

Reading and Writing Character Data

Basic Character Data Reading and Writing

Connecting Streams

Standard Input and Output

Using Console

Understand Serialization

Serializable Object Graph

Object Serialization

Serialization of Sensitive Information

Customize Serialization Process

Serialization and Versioning

Working with Filesystems

Constructing Filesystem Paths

Navigating the Filesystem

Analyze Path Properties

Set Path Properties

Create Paths

Create Temporary Files and Folders

Copy and Move Paths

Delete Paths

Handle Zip Archives

Represent Zip Archive as a FileSystem

Access HTTP Resources

Summary

Practices for Lesson 13: Overview

14 Java Concurrency and Multithreading

Objectives

Java Concurrency Concepts

Implement Threads

Thread Life Cycle

Interrupt Thread

- Block Thread
- Make Thread Wait Until Notified
- Common Thread Properties
- Create Executor Service Objects
- Manage Executor Service Life Cycle
- Implementing Executor Service Tasks
- Locking Problems
- Writing Thread-Safe Code
- Ensure Consistent Access to Shared Data
- Nonblocking Atomic Actions
- Ensure Exclusive Object Access Using Intrinsic Locks
- Intrinsic Lock Automation
- Nonblocking Concurrency Automation
- Alternative Locking Mechanisms
- CPU Versus IO Bound Concurrent Tasks
- Virtual Threads API
- Virtual Thread Operations
- Scoped Values
- Scoped Values Operations
- Summary
- Practices for Lesson 14: Overview

15 Modules and Deployment

- Objectives
- Compile, Package, and Execute Nonmodular Java Applications
- Nonmodular Java Characteristics
- What Is a Module?
- Java Modules
- Java Module Categories
- Module Import Declarations
- Define Module Dependencies
- Export Module Content
- Module Example
- Open Module Content
- Open an Entire Module
- Produce and Consume Services
- Services Example
- Multi-Release Module Archives
- Compile and Package a Module
- Execute a Modularized Application
- Migrating Legacy Java Applications Using Automatic module

- Create Custom Runtime Image
- Execute Runtime Image
- Optimize a Custom Runtime Image
- Check Dependencies
- Summary
- Practices for Lesson 15: Overview

A Annotations

- Objectives
- Annotations: Introduction
- Design Annotations
- Apply Annotations
- Dynamically Discover Annotations
- Document the Use of Annotations
- Annotations that Validate Design
- Deprecated Annotation
- Suppress Compiler Warnings
- Varargs and Heap Pollution
- Summary

B Java Database Connectivity

- Objectives
- Java Database Connectivity (JDBC)
- JDBC API Structure
- Manage Database Connections
- Create and Execute Basic SQL Statements
- Create and Execute Prepared SQL Statements
- Create and Execute Callable SQL Statements
- Process Query Results
- Control Transactions
- Discover Metadata
- Customize ResultSet
- Set Up ResultSet Type
- Set Up ResultSet Concurrency and Holdability
- Summary

C Java Security

- Objectives
- Security as Nonfunctional Requirement
- Security Threats
- Denial of Service (DoS) Attack

- Define Security Policies
- Changes in the Security API
- Secure File System and I/O Operations
- Best Practices for Protecting Your Code
- Erroneous Value Guards
- Protect Sensitive Data (Part 1)
- Protect Sensitive Data (Part 2)
- Prevent SQL Injections
- Prevent JavaScript Injections
- Prevent XML Injections
- Discover and Document Security Issues
- Summary

D Advanced Generics

- Objectives
- Compiler Erases Information About Generics
- Generic and Raw Type Compatibility
- Generics and Type Hierarchy
- Wildcard Generics
- Upper Bound Wildcard
- Lower Bound Wildcard
- Collections and Generics Best Practices
- Summary

E Java Applications on Oracle Cloud

- Objectives
- Cloud Application Requirements
- Cloud Application Runtime Infrastructure
- Cloud Java Application Servers
- Package and Deploy Cloud Application
- Optimise deployment with GraalVM
- HTTP Protocol Basics
- REST Service Conventions and Resources
- Configure and Launch REST Service Application Using Helidon SE
- Summary

F Miscellaneous Java Topics

- Objectives
- Builder Design Pattern
- Singleton Design Pattern
- Java Regular Expression API

Regular Expressions: Character Classes
Regular Expressions: Quantifiers
Regular Expressions: Boundaries
File IO Watch Service
Class-File API
Foreign Function and Memory API
Allocating/Deallocating Memory for Native Data
Representing Native Functions
Calling a Native Function
Summary