

Contents

1 Introduction to Java

Course Goals	1-2
Audience	1-3
Course Schedule	1-4
Course Practices	1-5
Lesson Objectives	1-6
What Is Java?	1-7
How Java Works?	1-8
Classes	1-9
Objects	1-10
Inheritance	1-11
Java APIs	1-12
Java Keywords, Reserved Words, and a Special Identifier	1-13
Java Naming Conventions	1-14
Java Basic Syntax Rules	1-16
Define Java Class	1-17
Access Classes Across Packages	1-18
Use Access Modifiers	1-19
Create Main Application Class	1-20
Compile Java Program	1-21
Execute Java Program	1-22
Comments and Documentation	1-23
Summary	1-25
Practices	1-26

2 Primitive Types, Operators, and Flow Control Statements

Objectives	2-2
Declare and Initialize Primitive Variables	2-4
Restrictions on Primitive Declarations and Initializations	2-5
Java Operators	2-6
Assignment and Arithmetic Operators	2-7
Arithmetic Operations and Type Casting	2-8
More Mathematical Operations	2-9
Binary Number Representation	2-10
Bitwise Operators	2-11

Equality, Relational, and Conditional Operators 2-12
Short-Circuit Evaluation 2-13
Flow Control Using if/else Construct 2-14
Ternary Operator 2-15
Flow Control Using switch Construct 2-16
JShell 2-17
Summary 2-18
Practices 2-19

3 Text, Date, Time, and Numeric Objects

Objectives 3-2
String Initialization 3-3
String Operations 3-4
String Indexing 3-5
StringBuilder: Introduction 3-6
Wrapper Classes for Primitives 3-7
Representing Numbers Using BigDecimal Class 3-8
Method Chaining 3-9
Local Date and Time 3-10
More Local Date and Time Operations 3-11
Instants, Durations, and Periods 3-13
Zoned Date and Time 3-14
Represent Languages and Countries 3-15
Format and Parse Numeric Values 3-17
Format and Parse Date and Time Values 3-18
Localizable Resources 3-21
Format Message Patterns 3-22
Formatting and Localization: Example 3-23
Summary 3-24
Practices 3-25

4 Classes and Objects

Objectives 4-2
UML: Introduction 4-3
Modeling Classes 4-4
Modeling Interactions and Activities 4-6
Designing Classes 4-7
Creating Objects 4-8
Define Instance Variables 4-9
Define Instance Methods 4-10
Object Creation and Access: Example 4-11

Local Variables and Recursive Object Reference 4-12

Local Variable Type Inference 4-14

Define Constants 4-15

Static Context 4-16

Accessing Static Context 4-17

Combining Static and Final 4-18

Other Static Context Use Cases 4-19

NetBeans IDE: Introduction 4-21

Summary 4-22

Practices 4-23

5 Improved Class Design

Objectives 5-2

Overload Methods 5-3

Variable Number of Arguments 5-5

Define Constructors 5-6

Reuse Constructors 5-7

Access Modifiers Summary 5-8

Define Encapsulation 5-9

Define Immutability 5-10

Constants and Immutability 5-11

Enumerations 5-12

Complex Enumerations 5-13

Java Memory Allocation 5-14

Parameter Passing 5-15

Java Memory Cleanup 5-16

Summary 5-17

Practices 5-18

6 Inheritance

Objectives 6-2

Extend Classes 6-3

Object Class 6-4

Reuse Parent Class Code Through Inheritance 6-6

Instantiating Classes and Accessing Objects 6-7

Rules of Reference Type Casting 6-8

Verify Object Type Before Casting the Reference 6-9

Reference Code Within the Current or Parent Object 6-10

Define Subclass Constructors 6-11

Class and Object Initialization Summary 6-12

Override Methods and Use Polymorphism 6-14

- Reuse Parent Class Logic in Overwritten Method 6-16
- Define Abstract Classes and Methods 6-17
- Define Final Classes and Methods 6-19
- Override Object Class Operations: toString 6-20
- Override Object Class Operations: equals 6-21
- Override Object Class Operations: hashCode 6-22
- Compare String Objects 6-23
- Factory Method Pattern 6-24
- Summary 6-25
- Practices 6-26

7 Interfaces

- Objectives 7-2
- Java Interfaces 7-3
- Multiple Inheritance Problem 7-4
- Implement Interfaces 7-5
- Default, Private, and Static Methods in Interfaces 7-6
- Interface Hierarchy 7-7
- Interface Is a Type 7-8
- Functional Interfaces 7-9
- Generics 7-10
- Use Generics 7-11
- Examples of Java Interfaces: java.lang.Comparable 7-13
- Examples of Java Interfaces: java.util.Comparator 7-14
- Examples of Java Interfaces: java.lang.Cloneable 7-15
- Composition Pattern 7-16
- Summary 7-17
- Practices 7-18

8 Arrays and Loops

- Objectives 8-2
- Arrays 8-3
- Combined Declaration, Creation, and Initialization of Arrays 8-4
- Multidimensional Arrays 8-5
- Copying Array Content 8-6
- Arrays Class 8-7
- Loops 8-8
- Processing Arrays by Using Loops 8-9
- Complex for Loops 8-10
- Embedded Loops 8-11
- Break and Continue 8-12

Summary 8-13

Practices 8-14

9 Collections

Objectives 9-2

Introduction to Java Collection API 9-3

Java Collection API Interfaces and Implementation Classes 9-4

Create List Object 9-5

Manage List Contents 9-6

Create Set Object 9-7

Manage Set Contents 9-8

Create Deque Object 9-9

Manage Deque Contents 9-10

Create HashMap Object 9-11

Manage HashMap Contents 9-12

Iterate through Collections 9-13

Other Collection Behaviors 9-14

Use java.util.Collections Class 9-15

Access Collections Concurrently 9-16

Prevent Collections Corruption 9-17

Legacy Collection Classes 9-19

Summary 9-20

Practices 9-21

10 Nested Classes and Lambda Expressions

Objectives 10-2

Types of Nested Classes 10-3

Static Nested Classes 10-6

Member Inner Classes 10-7

Local Inner Classes 10-9

Anonymous Inner Classes 10-10

Anonymous Inner Classes and Functional Interfaces 10-12

Understand Lambda Expressions 10-13

Define Lambda Expression Parameters and Body 10-14

Use Method References 10-15

Default and Static Methods in Functional Interfaces 10-16

Use Default and Static Methods of the Comparator Interface 10-17

Use Default and Static Methods of the Predicate Interface 10-18

Summary 10-19

Practices 10-20

11 Java Streams API

- Objectives 11-2
- Characteristics of Streams 11-3
- Create Streams Using Stream API 11-4
- Stream Pipeline Processing Operations 11-5
- Using Functional Interfaces 11-6
- Primitive Variants of Functional Interfaces 11-7
- Bi-argument Variants of Functional Interfaces 11-9
- Perform Actions with Stream Pipeline Elements 11-10
- Perform Filtering of Stream Pipeline Elements 11-11
- Perform Mapping of Stream Pipeline Elements 11-12
- Join Streams using flatMap Operation 11-13
- Other Intermediate Stream Operations 11-14
- Short-Circuit Terminal Operations 11-15
- Process Stream Using count, min, max, sum, average Operations 11-16
- Aggregate Stream Data using reduce Operation 11-17
- General Logic of the collect Operation 11-19
- Using Basic Collectors 11-20
- Perform a Conversion of a Collector Result 11-21
- Perform Grouping or Partitioning of the Stream Content 11-22
- Mapping and Filtering with Respect to Groups or Partitions 11-23
- Parallel Stream Processing 11-25
- Parallel Stream Processing Guidelines 11-26
- Restrictions on Parallel Stream Processing 11-27
- Summary 11-29
- Practices 11-30

12 Handle Exceptions and Fix Bugs

- Objectives 12-2
- Using Java Logging API 12-3
- Logging Method Categories 12-4
- Guarded Logging 12-6
- Log Writing Handling 12-7
- Logging Configuration 12-9
- Describe Java Exceptions 12-10
- Create Custom Exceptions 12-11
- Throwing Exceptions 12-12
- Catching Exceptions 12-13
- Exceptions and the Execution Flow 12-14
- Example Throwing an Unchecked Exception 12-15

Example Throwing a Checked Exception	12-16
Handling Exceptions	12-17
Resource Auto-Closure	12-18
Suppressed Exceptions	12-19
Handle Exception Cause	12-20
Java Debugger	12-21
Debugger Actions	12-22
Manipulate Program Data in Debug Mode	12-23
Validate Program Logic Using Assertions	12-24
Normal Program Flow with No Exceptions	12-25
Program Flow Producing a Runtime Exception	12-27
Program Flow Catching Specific Checked Exception	12-29
Program Flow Catching Any Exceptions	12-31
Summary	12-33
Practices	12-34

13 Java IO API

Objectives	13-2
Java Input-Output Principals	13-3
Java Input-Output API	13-4
Reading and Writing Binary Data	13-5
Basic Binary Data Reading and Writing	13-6
Reading and Writing Character Data	13-8
Basic Character Data Reading and Writing	13-9
Connecting Streams	13-10
Standard Input and Output	13-11
Using Console	13-12
Understand Serialization	13-14
Serializable Object Graph	13-15
Object Serialization	13-16
Serialization of Sensitive Information	13-17
Customize Serialization Process	13-18
Serialization and Versioning	13-19
Working with Filesystems	13-20
Constructing Filesystem Paths	13-22
Navigating the Filesystem	13-24
Analyse Path Properties	13-25
Set Path Properties	13-26
Create Paths	13-28
Create Temporary Files and Folders	13-29
Copy and Move Paths	13-30

- Delete Paths 13-31
- Handle Zip Archives 13-32
- Represent Zip Archive as a FileSystem 13-33
- Access HTTP Resources 13-35
- Summary 13-36
- Practices 13-37

14 Java Concurrency and Multithreading

- Objectives 14-2
- Java Concurrency Concepts 14-3
- Implement Threads 14-5
- Thread Life Cycle 14-6
- Interrupt Thread 14-7
- Block Thread 14-8
- Make Thread Wait Until Notified 14-9
- Common Thread Properties 14-11
- Create Executor Service Objects 14-12
- Manage Executor Service Life Cycle 14-16
- Implementing Executor Service Tasks 14-18
- Locking Problems 14-20
- Writing Thread-Safe Code 14-21
- Ensure Consistent Access to Shared Data 14-23
- Non-Blocking Atomic Actions 14-24
- Ensure Exclusive Object Access Using Intrinsic Locks 14-25
- Intrinsic Lock Automation 14-26
- Non-Blocking Concurrency Automation 14-28
- Alternative Locking Mechanisms 14-29
- Summary 14-30
- Practices 14-31

15 Java Modules

- Objectives 15-2
- Compile, Package, and Execute Non-Modular Java Applications 15-3
- What Is a Module? 15-5
- Java Platform Module System (JPMS) 15-7
- JPMS Module Categories 15-8
- Define Module Dependencies 15-9
- Export Module Content 15-10
- Modules Example 15-11
- Open Module Content 15-12
- Open an Entire Module 15-13

Produce and Consume Services	15-14
Services Example	15-15
Multi-Release Module Archives	15-16
Compile and Package a Module	15-17
Execute a Modularized Application	15-18
Migrating Legacy Java Applications Using Automatic Modules	15-19
Create Custom Runtime Image	15-20
Execute Runtime Image	15-22
Optimize a Custom Runtime Image	15-23
Summary	15-25
Practices	15-26