Java Basic

Declarations and Access Control

- Identifiers & JavaBeans
- Legal Identifiers
- Sun's Java Code Conventions
- JavaBeans Standards
- Declare Classes
- Source File Declaration Rules
- Class Declarations and Modifiers
- Concrete Subclass
- Declaring an Interface
- Declaring Interface Constants
- Declare Class Members
- Access Modifiers
- Nonaccess Member Modifiers
- Constructor Declarations
- Variable Declarations
- Declaring Enums

Object Orientation

- Encapsulation
- Inheritance, Is-A, Has-A
- Polymorphism
- Overridden Methods
- Overloaded Methods
- Reference Variable Casting
- Implementing an Interface
- Legal Return Types
- Return Type Declarations
- Returning a Value
- Constructors and Instantiation
- Default Constructor
- Overloaded Constructors

- Statics
- Static Variables and Methods
- Coupling and Cohesion

Assignments

- Stack and Heap—Quick Review
- Literals, Assignments, and Variables
- Literal Values for All Primitive Types
- Assignment Operators
- Casting Primitives
- Using a Variable or Array Element That Is Uninitialized and Unassigned
- Local (Stack, Automatic) Primitives and Objects
- Passing Variables into Methods
- Passing Object Reference Variables
- Does Java Use Pass-By-Value Semantics?
- Passing Primitive Variables
- Array Declaration, Construction, and Initialization
- Declaring an Array
- Constructing an Array
- Initializing an Array
- Initialization Blocks
- Using Wrapper Classes and Boxing
- An Overview of the Wrapper Classes
- Creating Wrapper Objects
- Using Wrapper Conversion Utilities
- Autoboxing
- Overloading
- Garbage Collection
- Overview of Memory Management and Garbage Collection
- Overview of Java's Garbage Collector
- Writing Code That Explicitly Makes Objects Eligible for Garbage Collection

Operators

- Java Operators
- Assignment Operators

- Relational Operators
- instance of Comparison
- Arithmetic Operators
- Conditional Operator
- Logical Operators

Flow Control, Exceptions

- if and switch Statements
- if-else Branching
- switch Statements
- Loops and Iterators
- Using while Loops
- Using do Loops
- Using for Loops
- Using break and continue
- Unlabeled Statements
- Labeled Statements
- Handling Exceptions
- Catching an Exception Using try and catch
- Using finally
- Propagating Uncaught Exceptions
- Defining Exceptions
- Exception Hierarchy
- Handling an Entire Class Hierarchy of Exceptions
- Exception Matching
- Exception Declaration and the Public Interface
- Rethrowing the Same Exception
- Common Exceptions and Errors

Strings, I/O, Formatting, and Parsing

- String, StringBuilder, and String Buffer
- The String Class
- Important Facts About Strings and Memory
- Important Methods in the String Class

- The String Buffer and StringBuilder Classes
- Important Methods in the String Buffer and StringBuilder Classes
- File Navigation and I/O
- Types of Streams
- The Byte-stream I/O hierarchy
- Character Stream Hierarchy
- RandomAccessFile class
- The java.io.Console Class
- Serialization
- Dates, Numbers, and Currency
- Working with Dates, Numbers, and Currencies
- Parsing, Tokenizing, and Formatting
- Locating Data via Pattern Matching
- Tokenizing

Generics and Collections

- Overriding hashCode() and equals()
- Overriding equals()
- Overriding hashCode()
- Collections
- So, What Do You Do with a Collection?
- List Interface
- Set Interface
- Map Interface
- Queue Interface
- Using the Collections Framework
- Array List Basics
- Autoboxing with Collections
- Sorting Collections and Arrays
- Navigating (Searching) Tree Sets and Tree Maps
- Other Navigation Methods
- Backed Collections
- Generic Types
- Generics and Legacy Code
- Mixing Generic and Non-generic Collections

- Polymorphism and Generics

Threads

- Defining, Instantiating, and Starting Threads
- Defining a Thread
- Instantiating a Thread
- Starting a Thread
- Thread States and Transitions
- Thread States
- Preventing Thread Execution
- Sleeping
- Thread Priorities and yield( )
- Synchronizing Code
- Synchronization and Locks
- Thread Deadlock
- Thread Interaction
- Using notifyAll( ) When Many Threads May Be Waiting

Concurrent Patterns in Java

- Introducing Executors, What Is Wrong with the Runnable Pattern?
- Defining the Executor Pattern: A New Pattern to Launch Threads
- Defining the Executor Service Pattern, a First Simple Example
- Comparing the Runnable and the Executor Service Patterns
- Understanding the Waiting Queue of the Executor Service
- Wrapping-up the Executor Service Pattern
- From Runnable to Callable: What Is Wrong with Runnables?
- Defining a New Model for Tasks That Return Objects
- Introducing the Callable Interface to Model Tasks
- Introducing the Future Object to Transmit Objects Between Threads
- Wrapping-up Callables and Futures, Handling Exceptions

Concurrent Collections

- Implementing Concurrency at the API Level
- Hierarchy of Collection and Map, Concurrent Interfaces
- What Does It Mean for an Interface to Be Concurrent?

- Why You Should Avoid Vectors and Stacks
- Understanding Copy On Write Arrays
- Introducing Queue and Deque, and Their Implementations
- Understanding How Queue Works in a Concurrent Environment
- Adding Elements to a Queue That Is Full: How Can It Fail?
- Understanding Error Handling in Queue and Deque
- Introducing Concurrent Maps and Their Implementations
- Atomic Operations Defined by the ConcurrentMap Interface
- Understanding Concurrency for a HashMap
- Understanding the Structure of the ConcurrentHashMap from Java 7
- Introducing the Java 8 ConcurrentHashMap and Its Parallel Methods
- Parallel Search on a Java 8 ConcurrentHashMap
- Parallel Map / Reduce on a Java 8 ConcurrentHashMap
- Parallel ForEach on a Java 8 ConcurrentHashMap
- Creating a Concurrent Set on a Java 8 ConcurrentHashMap
- Introducing Skip Lists to Implement ConcurrentMap
- Understanding How Linked Lists Can Be Improved by Skip Lists
- How to Make a Skip List Concurrent Without Synchronization

Lambda Expressions

- Introduction
- Writing Lambda Expressions
- Functional Interfaces
- Types of Functional Interfaces
- Method reference

Stream API

- Introduction
- Stream API with Collections
- Stream Operations