LỘ TRÌNH JAVA CORE ADVANCED - 30 BUỔI

📋 THÔNG TIN KHÓA HỌC

Thông tin	Chi tiết
Tên khóa học	Java Core Advanced
Điều kiện tiên quyết	☑ Hoàn thành Java Core Pro (42 buổi)
Tổng số buổi	30 buổi
Thời lượng/buổi	2.5 giờ
Tổng thời gian	75 giờ học
Thời gian khóa	8-10 tuần (3 buổi/tuần)
Level	Advanced → Expert
Đầu ra	Production-ready Java Developer

🎯 MỤC TIÊU KHÓA HỌC

Sau khóa học, bạn sẽ:

- ☑ **Hiểu sâu JVM** - Memory management, GC, Performance tuning
- ☑ **Master Concurrency** - ExecutorService, CompletableFuture, Reactive
- ☑ **Advanced Design** - Architecture patterns, Enterprise patterns
- ☑ **Testing Expert** - TDD, Unit testing, Integration testing
- ☑ **Performance Optimization** - Profiling, tuning, best practices
- ☑ **Build & Deploy** - Maven/Gradle advanced, CI/CD ready
- ☑ **Security** - Cryptography, Authentication, Secure coding
- ☑ **Production Skills** - Logging, Monitoring, Debugging

🗂 CẤU TRÚC KHÓA HỌC

Module	Tên Module	Số Buổi
1	JVM Internals & Memory Management	3 buổi
2	Advanced Multithreading & Concurrency	4 buổi
3	Advanced Collections & Data Structures	2 buổi
4	Reflection, Annotations & Metaprogramming	2 buổi
5	Advanced I/O, NIO & Network Programming	3 buổi
6	Testing & TDD (JUnit, Mockito, AssertJ)	3 buổi

Module	Tên Module	Số Buổi
7	Build Tools & Dependency Management	2 buổi
8	Enterprise Design Patterns	3 buổi
9	Performance Optimization & Profiling	2 buổi
10	Security & Cryptography	2 buổi
11	Logging, Monitoring & Debugging	2 buổi
12	Functional Programming Advanced	2 buổi

TỔNG: 30 buổi

CHI TIẾT TỪNG BUỔI HỌC

📦 MODULE 1: JVM INTERNALS & MEMORY MANAGEMENT (3 BUỔI)

BUỔI 1: JVM Architecture Deep Dive

Mục tiêu: Hiểu rõ cách JVM hoạt động

Nội dung:

- **JVM Architecture Components**
  - Class Loader Subsystem
    - Bootstrap, Extension, Application ClassLoaders
    - Class loading process (Load → Link → Initialize)
    - Parent delegation model
    - Custom ClassLoaders
  - Runtime Data Areas
    - Heap (Young Gen, Old Gen)
    - Method Area (Metaspace)
    - Stack (Frame structure)
    - PC Register
    - Native Method Stack
  - Execution Engine
    - Interpreter
    - JIT Compiler (C1, C2)
    - Garbage Collector
- **Bytecode Fundamentals**
  - javap tool
  - Reading bytecode
  - Common bytecode instructions

    - Method invocation types

- **Class File Format**
    - Magic number
    - Constant pool
    - Access flags
    - Methods & Attributes

BUỔI 2: Memory Management & Garbage Collection

Mục tiêu: Master memory management

Nội dung:

- **Heap Memory Structure**
    - Young Generation
        - Eden Space
        - Survivor Spaces (S0, S1)
    - Old Generation (Tenured)
    - Metaspace (non-heap)
- **Garbage Collection Algorithms**
    - **Serial GC** - Single-threaded
    - **Parallel GC** (Throughput) - Multi-threaded
    - **CMS (Concurrent Mark Sweep)** - Low pause
    - **G1 GC (Garbage First)** - Default Java 9+
    - **ZGC** - Ultra-low latency (Java 11+)
    - **Shenandoah** - Low pause alternative
- **GC Tuning**
    - Heap sizing (-Xms, -Xmx)
    - GC selection flags
    - GC logging & analysis
    - Monitoring GC behavior
    - Common GC problems (long pauses, OutOfMemoryError)
- **Memory Leaks**
    - Common causes
    - Detection techniques
    - Heap dump analysis
    - Prevention strategies

BUỔI 3: Performance Tuning & JVM Flags

Mục tiêu: Optimize JVM performance

Nội dung:

- **JVM Tuning Flags**
    - Memory flags (-Xms, -Xmx, -XX:MetaspaceSize)

    - GC flags (-XX:+UseG1GC, -XX:MaxGCPauseMillis)
    - Compilation flags (-XX:CompileThreshold)
    - Debugging flags (-XX:+PrintGCDetails, -XX:+HeapDumpOnOutOfMemoryError)
- **Profiling Tools**
    - **JConsole** - JMX monitoring
    - **VisualVM** - Profiling & analysis
    - **JProfiler** - Commercial profiler
    - **YourKit** - Performance profiler
    - **Java Mission Control (JMC)** - Advanced monitoring
- **Performance Metrics**
    - CPU usage
    - Memory usage
    - Thread analysis
    - Method hotspots
    - Object allocation
- **JVM Crash Analysis**
    - hs_err_pid files
    - Thread dumps
    - Heap dumps
    - Core dumps

🟧 MODULE 2: ADVANCED MULTITHREADING & CONCURRENCY (4 BUỔI)

BUỔI 4: Concurrency Utilities - ExecutorService

Mục tiêu: Master thread pool management

Nội dung:

- **Executor Framework**
    - Executor interface
    - ExecutorService interface
    - ScheduledExecutorService
- **Thread Pools**
    - **Executors factory methods:**
        - newFixedThreadPool(n)
        - newCachedThreadPool()
        - newSingleThreadExecutor()
        - newScheduledThreadPool(n)
    - **ThreadPoolExecutor** - Custom configuration
        - Core pool size vs Max pool size
        - Keep-alive time
        - Work queue types (LinkedBlockingQueue, SynchronousQueue)
        - Rejection policies
- **Callable & Future**
    - Callable vs Runnable

    - Future interface
        - get(), cancel(), isDone()
    - FutureTask
    - ExecutorService.submit()
- **Best Practices**
    - Proper shutdown
    - Exception handling in threads
    - Thread pool sizing
    - Avoiding thread leaks

BUỔI 5: Advanced Synchronization

Mục tiêu: Deep dive concurrency control

Nội dung:

- **java.util.concurrent.locks**
    - **ReentrantLock**
        - lock(), unlock(), tryLock()
        - Fairness policy
        - Condition variables
    - **ReadWriteLock**
        - ReentrantReadWriteLock
        - Read vs Write locks
        - Lock downgrading
    - **StampedLock** (Java 8+)
        - Optimistic reading
        - Read/Write locks
        - Lock conversion
- **Atomic Variables**
    - AtomicInteger, AtomicLong, AtomicBoolean
    - AtomicReference
    - compareAndSet (CAS)
    - Lock-free algorithms
    - ABA problem
- **Concurrent Collections Deep Dive**
    - **ConcurrentHashMap**
        - Internal structure (segments, buckets)
        - Compute methods
        - putIfAbsent(), computeIfAbsent()
    - **CopyOnWriteArrayList**
        - Use cases (read-heavy, infrequent writes)
    - **BlockingQueue implementations**
        - ArrayBlockingQueue
        - LinkedBlockingQueue
        - PriorityBlockingQueue
        - DelayQueue

- **Synchronizers**
  - **CountDownLatch** - Wait for events
  - **CyclicBarrier** - Synchronize threads at barrier
  - **Semaphore** - Permit-based access control
  - **Phaser** (Java 7+) - Flexible barrier
  - **Exchanger** - Thread pair data exchange

BUỔI 6: CompletableFuture & Asynchronous Programming

Mục tiêu: Modern async programming

Nội dung:

- **CompletableFuture Basics**
  - Creating CompletableFuture
    - completedFuture(), supplyAsync(), runAsync()
  - Completing manually (complete(), completeExceptionally())
- **Chaining Operations**
  - **Transformation:**
    - thenApply(), thenApplyAsync()
  - **Consuming:**
    - thenAccept(), thenAcceptAsync()
  - **Running:**
    - thenRun(), thenRunAsync()
  - **Combining:**
    - thenCombine(), thenCompose()
    - allOf(), anyOf()
- **Exception Handling**
  - exceptionally()
  - handle()
  - whenComplete()
- **Advanced Patterns**
  - Parallel execution
  - Sequential composition
  - Timeout handling
  - Custom executors
- **Real-world Use Cases**
  - Async HTTP calls
  - Database queries
  - File processing
  - Microservices communication

BUỔI 7: Fork/Join Framework & Parallel Streams

Mục tiêu: Parallel processing mastery

Nội dung:

- **Fork/Join Framework**
  - ForkJoinPool
  - RecursiveTask
  - RecursiveAction
  - Work-stealing algorithm
  - compute() method
- **Divide & Conquer Problems**
  - Parallel array processing
  - Merge sort parallel
  - Recursive computations
- **Parallel Streams Deep Dive**
  - parallel() vs stream()
  - Common ForkJoinPool
  - Custom ForkJoinPool
  - When to use parallel streams
  - Pitfalls (stateful operations, boxing overhead)
- **Performance Comparison**
  - Sequential vs Parallel
  - Overhead analysis
  - Optimal data size
- **Thread Safety Considerations**
  - Avoiding side effects
  - Reduction operations
  - Collectors thread-safety

MODULE 3: ADVANCED COLLECTIONS & DATA STRUCTURES (2 BUỔI)

BUỔI 8: Custom Collections & Internals

Mục tiêu: Deep understanding of collections

Nội dung:

- **HashMap Internals Deep Dive**
  - Hash function & collision
  - Bucket structure (Node/TreeNode)
  - Load factor & rehashing
  - Tree-ification (Java 8+)
  - Performance characteristics
- **TreeMap Internals**
  - Red-Black tree structure
  - Balancing operations
  - Comparator vs Comparable
- **ArrayList vs LinkedList**
  - Memory layout
  - Performance comparison
  - When to use which

- **Custom Collection Implementation**
  - Implement custom List
  - Implement custom Map
  - Iterator implementation
  - Fail-fast vs Fail-safe
- **Collection Views**
  - Collections.unmodifiableXxx()
  - Collections.synchronizedXxx()
  - SubList, SubMap, SubSet

BUỔI 9: Advanced Data Structures

Mục tiêu: Implement advanced structures

Nội dung:

- **Trees**
  - Binary Search Tree
  - AVL Tree
  - Red-Black Tree (concept)
  - B-Tree (concept)
- **Graphs**
  - Graph representations (adjacency matrix, list)
  - DFS, BFS implementations
  - Shortest path algorithms
- **Heaps**
  - Min Heap, Max Heap
  - Priority Queue internals
  - Heap operations
- **Trie (Prefix Tree)**
  - Implementation
  - Use cases (autocomplete, spell checker)
- **Cache Implementations**
  - LRU Cache (LinkedHashMap)
  - LFU Cache
  - Time-based eviction
- **Bloom Filter**
  - Concept & implementation
  - Use cases

▧ MODULE 4: REFLECTION, ANNOTATIONS & METAPROGRAMMING (2 BUỔI)

BUỔI 10: Reflection API

Mục tiêu: Dynamic code manipulation

Nội dung:

- **Reflection Basics**
  - Class object
  - Getting class information
    - getClass(), .class, Class.forName()
- **Inspecting Classes**
  - Fields (getFields(), getDeclaredFields())
  - Methods (getMethods(), getDeclaredMethods())
  - Constructors
  - Modifiers
  - Annotations
- **Dynamic Invocation**
  - Creating instances (newInstance(), Constructor.newInstance())
  - Invoking methods (Method.invoke())
  - Accessing fields (Field.get(), Field.set())
  - Bypassing access control (setAccessible())
- **Array Reflection**
  - java.lang.reflect.Array
  - Dynamic array creation
- **Use Cases**
  - Dependency injection frameworks
  - ORM frameworks
  - Testing frameworks
  - Serialization
- **Performance Considerations**
  - Reflection overhead
  - Caching reflection objects
  - Alternatives (MethodHandles)

BUỔI 11: Annotations & Annotation Processing

Mục tiêu: Create custom annotations

Nội dung:

- **Annotation Fundamentals**
  - Built-in annotations (@Override, @Deprecated, @SuppressWarnings)
  - Meta-annotations
    - @Retention (SOURCE, CLASS, RUNTIME)
    - @Target (TYPE, METHOD, FIELD, etc.)
    - @Documented
    - @Inherited
    - @Repeatable (Java 8+)
- **Creating Custom Annotations**
  - Annotation declaration
  - Annotation elements

    - Default values
    - Marker annotations
- **Reading Annotations (Runtime)**
    - isAnnotationPresent()
    - getAnnotation()
    - getDeclaredAnnotations()
- **Annotation Processing (Compile-time)**
    - Annotation Processor API
    - javax.annotation.processing
    - Creating annotation processors
    - Code generation
- **Real-world Examples**
    - Validation framework (@NotNull, @Email)
    - ORM mapping (@Entity, @Table, @Column)
    - Dependency injection (@Inject, @Autowired)
    - Testing (@Test, @Before, @After)
- **Hands-on:**
    - Build custom validation framework
    - Build simple DI framework
    - Generate code with annotations

🟧 MODULE 5: ADVANCED I/O, NIO & NETWORK PROGRAMMING (3 BUỔI)

BUỔI 12: NIO (New I/O) Deep Dive

Mục tiêu: Non-blocking I/O mastery

Nội dung:

- **NIO Core Concepts**
    - Buffers
        - ByteBuffer, CharBuffer, IntBuffer, etc.
        - Buffer properties (capacity, position, limit)
        - Buffer operations (flip(), clear(), rewind())
        - Direct vs Heap buffers
    - Channels
        - FileChannel
        - SocketChannel, ServerSocketChannel
        - DatagramChannel
        - Channel transfer (transferTo, transferFrom)
    - Selectors
        - Multiplexing I/O
        - SelectionKey
        - Non-blocking server
- **File Operations with NIO**
    - Path, Paths, Files API

    - Walking file trees
    - Watch service (file monitoring)
    - Memory-mapped files
- **Async I/O (NIO.2 - Java 7)**
    - AsynchronousFileChannel
    - AsynchronousSocketChannel
    - CompletionHandler
- **Performance Comparison**
    - Traditional I/O vs NIO
    - When to use NIO

BUỔI 13: Network Programming - TCP/UDP

Mục tiêu: Socket programming

Nội dung:

- **TCP Networking**
    - Socket & ServerSocket
    - Client-Server architecture
    - Multi-threaded server
    - Connection pooling
- **Protocol Implementation**
    - HTTP client (basic)
    - Custom protocols
    - Message framing
    - Serialization (JSON, Protocol Buffers)
- **UDP Networking**
    - DatagramSocket
    - DatagramPacket
    - UDP vs TCP
    - Use cases
- **Non-blocking Server**
    - NIO Selector-based server
    - Handling multiple connections
    - Reactor pattern
- **Real-world Projects**
    - Chat server/client
    - File transfer
    - HTTP server (basic)

BUỔI 14: HTTP Clients & REST APIs

Mục tiêu: HTTP communication

Nội dung:

- **HttpClient (Java 11+)**
  - Creating HttpClient
  - HttpRequest builder
  - Synchronous vs Asynchronous requests
  - Response handling
- **RESTful API Consumption**
  - GET, POST, PUT, DELETE requests
  - Headers & authentication
  - Request/Response bodies
  - JSON parsing (Jackson, Gson)
- **Advanced Features**
  - WebSocket client
  - HTTP/2 support
  - Connection pooling
  - Timeout & retry
- **Alternative Libraries**
  - Apache HttpClient
  - OkHttp
  - Comparison
- **Hands-on:**
  - Build API client library
  - OAuth authentication
  - Rate limiting

🟧 MODULE 6: TESTING & TDD (3 BUỔI)

BUỔI 15: JUnit 5 Advanced

Mục tiêu: Professional testing

Nội dung:

- **JUnit 5 Architecture**
  - JUnit Platform
  - JUnit Jupiter
  - JUnit Vintage
- **Advanced Annotations**
  - @ParameterizedTest
  - @RepeatedTest
  - @Nested
  - @Tag
  - @TestFactory (dynamic tests)
  - @TestMethodOrder
- **Lifecycle & Extensions**
  - @BeforeAll, @AfterAll
  - @BeforeEach, @AfterEach
  - Extension model

    - Custom extensions
- **Assertions Advanced**
    - assertAll()
    - assertThrows()
    - assertTimeout()
    - Custom assertions
- **Assumptions**
    - assumeTrue(), assumeFalse()
    - Conditional test execution
- **Test Organization**
    - Test suites
    - Parallel execution
    - Test order

BUỔI 16: Mocking with Mockito

Mục tiêu: Unit test isolation

Nội dung:

- **Mockito Fundamentals**
    - Creating mocks
    - mock() vs @Mock
    - Stubbing (when...thenReturn)
- **Verification**
    - verify() method calls
    - Argument matchers
    - Verification modes (times, never, atLeast)
- **Advanced Mocking**
    - Spies (@Spy)
    - Partial mocking
    - Mocking static methods (Mockito 3.4+)
    - Mocking final classes/methods
    - Mocking constructors
- **Argument Captors**
    - @Captor
    - Capturing arguments for assertions
- **BDD Style**
    - given...when...then
    - BDDMockito
- **Best Practices**
    - What to mock vs not mock
    - Avoiding over-mocking
    - Readable tests

BUỔI 17: TDD & Testing Best Practices

Mục tiêu: Test-driven development

Nội dung:

- **TDD Methodology**
  - Red-Green-Refactor cycle
  - Writing tests first
  - Benefits of TDD
- **Test Coverage**
  - JaCoCo setup
  - Code coverage metrics
  - Coverage goals
- **AssertJ**
  - Fluent assertions
  - Readable test code
  - Custom assertions
- **Testing Strategies**
  - Unit tests
  - Integration tests
  - End-to-end tests
  - Test pyramid
- **Database Testing**
  - H2 in-memory database
  - Test data setup
  - @Transactional tests
- **Testing Patterns**
  - Arrange-Act-Assert (AAA)
  - Given-When-Then
  - Test fixtures
  - Test builders
  - Object mothers
- **Hands-on TDD Project**
  - Build feature with TDD
  - Refactoring with tests
  - Legacy code testing

🟫 MODULE 7: BUILD TOOLS & DEPENDENCY MANAGEMENT (2 BUỔI)

BUỔI 18: Maven Advanced

Mục tiêu: Master build automation

Nội dung:

- **Maven Lifecycle**
  - Phases (validate, compile, test, package, install, deploy)
  - Goals
  - Build lifecycle

- **POM Deep Dive**
  - Project coordinates (groupId, artifactId, version)
  - Dependencies
    - Dependency scope (compile, test, provided, runtime)
    - Transitive dependencies
    - Dependency exclusions
    - Dependency management
  - Properties
  - Profiles
- **Plugins**
  - Compiler plugin
  - Surefire (test execution)
  - JaCoCo (code coverage)
  - Assembly plugin (packaging)
  - Shade plugin (uber jar)
  - Release plugin
- **Multi-module Projects**
  - Parent POM
  - Module structure
  - Inter-module dependencies
- **Repository Management**
  - Local repository
  - Central repository
  - Custom repositories
  - Nexus / Artifactory
- **Best Practices**
  - Version management
  - Property usage
  - Profile strategies

BUỔI 19: Gradle & Modern Build Tools

Mục tiêu: Alternative build systems

Nội dung:

- **Gradle Fundamentals**
  - Groovy DSL vs Kotlin DSL
  - Build script structure
  - Tasks
  - Dependencies
- **Gradle vs Maven**
  - Performance comparison
  - Flexibility
  - Incremental builds
- **Advanced Gradle**
  - Custom tasks

    - Plugins
    - Multi-project builds
    - Dependency configurations
- **CI/CD Integration**
    - GitHub Actions
    - Jenkins integration
    - GitLab CI
- **Artifact Publishing**
    - Maven Central
    - GitHub Packages
- **Build Optimization**
    - Caching
    - Parallel execution
    - Build scans

MODULE 8: ENTERPRISE DESIGN PATTERNS (3 BUỔI)

BUỔI 20: Architectural Patterns

Mục tiêu: System design patterns

Nội dung:

- **Layered Architecture**
    - Presentation layer
    - Business logic layer
    - Data access layer
    - Separation of concerns
- **Repository Pattern**
    - Abstract data access
    - Generic repository
    - Unit of Work
- **Service Layer Pattern**
    - Business logic encapsulation
    - Transaction boundaries
    - Service composition
- **Dependency Injection**
    - Constructor injection
    - Setter injection
    - DI containers (concept)
- **Domain-Driven Design (DDD) Basics**
    - Entities
    - Value Objects
    - Aggregates
    - Domain Services
    - Repositories
- **Event-Driven Architecture**

    - Event sourcing
    - CQRS (Command Query Responsibility Segregation)
    - Event bus pattern

BUỔI 21: Integration Patterns

Mục tiêu: System integration

Nội dung:

- **Messaging Patterns**
    - Point-to-Point
    - Publish-Subscribe
    - Request-Reply
    - Message routing
- **API Design Patterns**
    - RESTful principles
    - API versioning
    - Pagination
    - Filtering & sorting
    - HATEOAS
- **Circuit Breaker**
    - Fault tolerance
    - Fallback strategies
    - Resilience4j (concept)
- **Bulkhead Pattern**
    - Resource isolation
    - Thread pool separation
- **Retry Pattern**
    - Exponential backoff
    - Retry policies
- **Saga Pattern**
    - Distributed transactions
    - Compensation

BUỔI 22: Concurrency Patterns

Mục tiêu: Concurrent design patterns

Nội dung:

- **Producer-Consumer**
    - BlockingQueue implementation
    - Multiple producers/consumers
- **Thread Pool Pattern**
    - Worker threads
    - Task queue

    - Graceful shutdown
- **Read-Write Lock Pattern**
    - Optimistic vs Pessimistic locking
    - StampedLock usage
- **Double-Checked Locking**
    - Lazy initialization
    - Thread-safe singleton
- **Immutable Object Pattern**
    - Thread safety through immutability
    - Builder pattern for immutables
- **Monitor Object Pattern**
    - Synchronized access
    - Condition variables
- **Active Object Pattern**
    - Decoupling method execution
    - Asynchronous method invocation

MODULE 9: PERFORMANCE OPTIMIZATION & PROFILING (2 BUỔI)

BUỔI 23: Performance Analysis

Mục tiêu: Identify bottlenecks

Nội dung:

- **Profiling Techniques**
    - CPU profiling
    - Memory profiling
    - Thread profiling
    - I/O profiling
- **Profiling Tools**
    - VisualVM hands-on
    - JProfiler
    - YourKit
    - Java Flight Recorder
- **Benchmarking**
    - JMH (Java Microbenchmark Harness)
    - Writing benchmarks
    - Avoiding pitfalls
    - Interpreting results
- **Memory Analysis**
    - Heap dump analysis
    - Memory leak detection
    - Object retention analysis
    - GC log analysis
- **Thread Analysis**
    - Thread dumps

    - Deadlock detection
    - Thread contention

BUỔI 24: Optimization Techniques

Mục tiêu: Write performant code

Nội dung:

- **Code-level Optimizations**
    - Algorithm optimization
    - Data structure selection
    - Loop optimization
    - String handling
    - Boxing/Unboxing avoidance
- **Memory Optimization**
    - Object pooling
    - Flyweight pattern
    - Primitive collections
    - Memory-efficient data structures
- **Caching Strategies**
    - In-memory caching
    - Cache eviction policies
    - Cache coherence
    - Caffeine library
- **Lazy Initialization**
    - On-demand loading
    - Lazy collections
- **JVM Tuning**
    - Heap sizing
    - GC tuning
    - JIT compilation
- **Database Optimization**
    - Connection pooling
    - Batch operations
    - Query optimization
    - Index usage
- **Concurrency Optimization**
    - Lock-free algorithms
    - CAS operations
    - Minimizing lock contention

🔒 MODULE 10: SECURITY & CRYPTOGRAPHY (2 BUỔI)

BUỔI 25: Java Security Fundamentals

Mục tiêu: Secure coding

Nội dung:

- **Security Manager**
  - Security policies
  - Permissions
  - Code signing
- **Cryptography Basics**
  - Encryption vs Hashing
  - Symmetric vs Asymmetric
- **Java Cryptography Architecture (JCA)**
  - MessageDigest (MD5, SHA-256)
  - Cipher (AES, RSA)
  - Key generation
  - KeyStore
- **Hashing**
  - Password hashing
  - BCrypt, SCrypt
  - Salt & pepper
- **Encryption**
  - AES encryption/decryption
  - RSA encryption/decryption
  - Key management
- **Digital Signatures**
  - Signature creation
  - Signature verification
  - Certificates
- **SSL/TLS**
  - HTTPS connections
  - Certificate validation
  - SSLContext

BUỔI 26: Secure Coding Practices

Mục tiêu: Prevention of vulnerabilities

Nội dung:

- **OWASP Top 10**
  - Injection attacks
  - Broken authentication
  - Sensitive data exposure
  - XXE attacks
  - Security misconfiguration
- **Input Validation**
  - Sanitization
  - Whitelist vs Blacklist
  - Regex validation

- **SQL Injection Prevention**
  - PreparedStatement
  - Parameterized queries
  - ORM best practices
- **XSS Prevention**
  - Output encoding
  - Content Security Policy
- **Authentication & Authorization**
  - Password storage
  - Session management
  - JWT tokens
  - OAuth 2.0 concepts
- **Secure Random**
  - SecureRandom vs Random
  - Cryptographically secure randomness
- **Best Practices**
  - Principle of least privilege
  - Defense in depth
  - Fail securely
  - Logging sensitive data

📝 MODULE 11: LOGGING, MONITORING & DEBUGGING (2 BUỔI)

BUỔI 27: Logging Frameworks

Mục tiêu: Professional logging

Nội dung:

- **Logging Fundamentals**
  - Why logging matters
  - Log levels (TRACE, DEBUG, INFO, WARN, ERROR)
  - Structured logging
- **SLF4J (Simple Logging Facade)**
  - Abstraction layer
  - Logger creation
  - Parameterized logging
- **Logback**
  - Configuration (XML)
  - Appenders (Console, File, RollingFile)
  - Layouts & Patterns
  - Filters
  - Async logging
- **Log4j 2**
  - Architecture
  - Configuration
  - Performance

- **Best Practices**
  - What to log
  - Log levels usage
  - Performance impact
  - Avoiding sensitive data
  - Correlation IDs
  - MDC (Mapped Diagnostic Context)
- **Centralized Logging**
  - ELK Stack concept (Elasticsearch, Logstash, Kibana)
  - Log aggregation
  - Structured JSON logs

BUỔI 28: Debugging & Monitoring

Mục tiêu: Production troubleshooting

Nội dung:

- **Debugging Techniques**
  - Debugger advanced features
  - Conditional breakpoints
  - Expression evaluation
  - Remote debugging
- **JMX (Java Management Extensions)**
  - MBeans
  - MBeanServer
  - JConsole
  - Exposing metrics
- **Application Monitoring**
  - Metrics (Micrometer)
  - Health checks
  - Counters, Gauges, Timers
- **Distributed Tracing**
  - Trace context
  - Span concept
  - OpenTelemetry (concept)
- **Alerts & Notifications**
  - Threshold-based alerts
  - Anomaly detection
- **Production Debugging**
  - Thread dumps analysis
  - Heap dumps on-demand
  - Flight Recorder
  - Live debugging considerations

MODULE 12: FUNCTIONAL PROGRAMMING ADVANCED (2 BUỔI)

BUỔI 29: Functional Patterns & Techniques

Mục tiêu: Functional programming mastery

Nội dung:

- **Immutability Deep Dive**
  - Persistent data structures
  - Copy-on-write
  - Immutables library
- **Higher-Order Functions**
  - Functions as first-class citizens
  - Currying
  - Partial application
  - Function composition
- **Monads (Concept)**
  - Optional as monad
  - Stream as monad
  - Try monad pattern
- **Lazy Evaluation**
  - Supplier for laziness
  - Lazy sequences
  - Infinite streams
- **Memoization**
  - Caching function results
  - Implementation patterns
- **Tail Recursion**
  - Tail call optimization (JVM limitations)
  - Trampolining

BUỔI 30: Reactive Programming Introduction

Mục tiêu: Reactive concepts

Nội dung:

- **Reactive Principles**
  - Asynchronous
  - Non-blocking
  - Backpressure
  - Responsive
- **Reactive Streams**
  - Publisher
  - Subscriber
  - Subscription
  - Processor
- **Project Reactor (Basics)**
  - Mono

    - Flux
    - Operators
- **RxJava (Basics)**
    - Observable
    - Observer
    - Schedulers
- **Use Cases**
    - Event-driven systems
    - High-throughput systems
    - Microservices
- **When to Use Reactive**
    - Benefits
    - Trade-offs
    - Complexity considerations

🎯 FINAL PROJECT

Project: Distributed Task Scheduler System

Mô tả: Xây dựng hệ thống lập lịch và thực thi tasks phân tán, production-ready

Core Features:

1. **Task Management**
    - Create, schedule, cancel tasks
    - Cron expressions support
    - One-time vs recurring tasks
2. **Distributed Execution**
    - Multiple worker nodes
    - Task distribution
    - Load balancing
    - Failover handling
3. **Monitoring & Logging**
    - Task execution logs
    - Performance metrics
    - Health checks
    - Admin dashboard (console)
4. **Persistence**
    - Database for tasks & history
    - Transaction management
5. **Concurrency**
    - Thread pool management
    - Concurrent task execution
    - Thread-safe state management

Technical Requirements:

- ☑ Advanced Multithreading (ExecutorService, CompletableFuture)

- ☑ NIO for network communication
- ☑ Custom Annotations for task definition
- ☑ Reflection for task discovery
- ☑ Comprehensive testing (JUnit 5, Mockito)
- ☑ Logging (SLF4J + Logback)
- ☑ JMX monitoring
- ☑ Performance optimization
- ☑ Security (authentication, encryption)
- ☑ Maven multi-module project
- ☑ Design patterns (5+)

Timeline: 3 tuần

📊 PHÂN BỔ BÀI TẬP

Tổng số bài tập: 450 bài

Module	Số Bài	Loại Bài
JVM & Memory	40	Analysis, tuning exercises
Concurrency	80	Threading problems, patterns
Collections	30	Implementation, optimization
Reflection & Annotations	35	Framework building
NIO & Network	45	Socket programming, protocols
Testing	60	TDD katas, test cases
Build Tools	20	Configuration, plugins
Design Patterns	45	Pattern implementation
Performance	30	Profiling, optimization
Security	25	Secure coding
Logging & Monitoring	20	Setup, best practices
Functional	20	FP exercises

Phân bổ theo độ khó:

- **Medium:** 200 bài (44%)
- **Hard:** 200 bài (44%)
- **Expert:** 50 bài (12%)

📚 TÀI LIỆU HỌC TẬP

Sách Bắt Buộc:

1. **"Effective Java" 3rd Edition** - Joshua Bloch ★★★
2. **"Java Concurrency in Practice"** - Brian Goetz ★★★
3. **"Java Performance: The Definitive Guide"** - Scott Oaks

Sách Tham Khảo: 4. **"Clean Architecture"** - Robert C. Martin 5. **"Release It!"** - Michael T. Nygard 6. **"Test Driven Development"** - Kent Beck

Online Resources:

- Oracle Java Documentation (Advanced)
- Baeldung (Advanced tutorials)
- InfoQ Java Articles
- DZone Java Zone

🎓 KẾT QUẢ SAU KHÓA HỌC

Bạn sẽ có khả năng:

☑ **Architect complex systems** - Design production-ready applications
☑ **Optimize performance** - Profile & tune Java applications
☑ **Write production code** - Testing, logging, monitoring
☑ **Handle concurrency** - Complex multithreading scenarios
☑ **Troubleshoot issues** - Debug production problems
☑ **Secure applications** - Implement security best practices
☑ **Build efficiently** - Master build tools & CI/CD

Career Paths:

- **Senior Java Developer**
- **Java Architect**
- **Performance Engineer**
- **DevOps Engineer**

Next Steps:

1. **Spring Ecosystem** (Spring Boot, Spring Security, Spring Data)
2. **Microservices Architecture**
3. **Cloud Platforms** (AWS, Azure, GCP)
4. **Container Orchestration** (Docker, Kubernetes)
5. **Big Data** (Hadoop, Spark)

GOOD LUCK! Master Java at the deepest level! 🚀

From Core to Expert - The Complete Journey