LỘ TRÌNH JAVA CORE PRO - 42 BUỔI

📋 THÔNG TIN KHÓA HỌC

Thông tin	Chi tiết
Tổng số buổi	42 buổi
Thời lượng/buổi	2 giờ
Tổng thời gian	84 giờ học
Thời gian khóa	10-14 tuần (3-4 buổi/tuần)
Yêu cầu đầu vào	Không cần kinh nghiệm lập trình
Đầu ra	Java Core Pro + Modern Java (Lambda, Stream API)

🗂 CẤU TRÚC KHÓA HỌC

Module	Tên Module	Số Buổi
1	Java Fundamentals & Syntax	5 buổi
2	String Processing & Regular Expressions	3 buổi
3	Random & Date/Time API	2 buổi
4	Object-Oriented Programming (OOP)	6 buổi
5	Design Patterns Cơ Bản	4 buổi
6	SOLID Principles & Clean Code	2 buổi
7	Exception Handling	2 buổi
8	Lambda Expressions & Functional Programming	3 buổi
9	Stream API	2 buổi
10	Collections Framework	4 buổi
11	Input/Output & File Processing	2 buổi
12	SQL & Database Connectivity (JDBC)	4 buổi
13	Multithreading Basics	2 buổi
14	Generics	1 buổi

TỔNG: 42 buổi

CHI TIẾT TỪNG BUỔI HỌC

▧ MODULE 1: JAVA FUNDAMENTALS & SYNTAX (5 BUỔI)

BUỔI 1: Setup & Java Basics

Mục tiêu: Cài đặt môi trường, hiểu JVM, viết chương trình đầu tiên

Nội dung:

- **Environment Setup**
  - Cài đặt JDK 17 LTS hoặc 21 LTS
  - Cài đặt IntelliJ IDEA Community
  - Git installation & GitHub setup
- **Java Architecture**
  - JVM vs JRE vs JDK
  - Write Once, Run Anywhere
  - Compile & Run process
  - Bytecode
- **First Program**
  - Hello World
  - Structure của Java program
  - main() method
  - System.out.println()
- **Variables & Data Types**
  - Primitive types: byte, short, int, long, float, double, char, boolean
  - Reference types preview
  - Variable declaration & initialization
  - Naming conventions
  - Literals & constants (final)

BUỔI 2: Operators & Control Flow

Mục tiêu: Làm chủ operators và điều khiển luồng

Nội dung:

- **Operators**
  - Arithmetic: +, -, *, /, %
  - Relational: ==, !=, >, <, >=, <=
  - Logical: &&, ||, !
  - Assignment: =, +=, -=, *=, /=
  - Increment/Decrement: ++, --
  - Ternary: ? :
  - Operator precedence
- **Input/Output**

  - Scanner class
  - nextInt(), nextDouble(), nextLine()
  - System.out formatting
- **Conditional Statements**
  - if, if-else, if-else-if
  - Nested if
  - switch-case (traditional)
  - Enhanced switch (Java 14+)
  - Switch expressions

BUỔI 3: Loops & Patterns

Mục tiêu: Master loops và solving pattern problems

Nội dung:

- **Loops**
  - for loop
  - while loop
  - do-while loop
  - Enhanced for loop (preview)
  - Nested loops
  - break statement
  - continue statement
  - Labeled break/continue
- **Loop Patterns**
  - Iteration patterns
  - Counter-controlled loops
  - Sentinel-controlled loops
  - Nested loop patterns

BUỔI 4: Methods

Mục tiêu: Functions và code reusability

Nội dung:

- **Method Basics**
  - Method declaration & definition
  - Return type
  - Parameters vs Arguments
  - void methods
  - Calling methods
- **Method Features**
  - Method overloading
  - Variable scope (local vs instance preview)
  - Pass by value

  - Varargs (variable arguments)
  - Recursive methods
- **Method Design**
  - Single responsibility
  - Naming conventions
  - When to create methods

BUỔI 5: Arrays

Mục tiêu: Array manipulation mastery

Nội dung:

- **Array Basics**
  - Declaration & initialization
  - Accessing elements
  - Array length
  - Default values
- **1D Arrays**
  - Traversal (for, for-each)
  - Input/output arrays
  - Common operations
- **Multi-dimensional Arrays**
  - 2D arrays
  - Matrix representation
  - Nested loops với 2D arrays
- **Arrays Class**
  - Arrays.toString()
  - Arrays.sort()
  - Arrays.copyOf()
  - Arrays.fill()
  - Arrays.equals()
  - Arrays.binarySearch()

▧ MODULE 2: STRING PROCESSING & REGEX (3 BUỔI)

BUỔI 6: String Fundamentals

Mục tiêu: Master String class

Nội dung:

- **String Basics**
  - String immutability
  - String literal vs new String()
  - String pool internals
- **String Methods - Part 1**

    - length(), isEmpty(), isBlank()
    - charAt(int index)
    - indexOf(), lastIndexOf()
    - substring()
    - toLowerCase(), toUpperCase()
    - trim(), strip(), stripLeading(), stripTrailing()
- **String Comparison**
    - equals() vs ==
    - equalsIgnoreCase()
    - compareTo(), compareToIgnoreCase()
    - contentEquals()
- **String Search**
    - contains()
    - startsWith(), endsWith()
    - matches() (preview)

BUỔI 7: String Manipulation & StringBuilder

Mục tiêu: Advanced String operations

Nội dung:

- **String Methods - Part 2**
    - concat() vs + operator
    - replace(), replaceAll(), replaceFirst()
    - split()
    - join() (Java 8+)
    - repeat() (Java 11+)
    - String.format()
    - formatted() (Java 15+)
- **StringBuilder & StringBuffer**
    - Mutable strings
    - append(), insert(), delete()
    - reverse()
    - capacity() vs length()
    - StringBuilder vs StringBuffer
    - Performance comparison
- **String Performance**
    - String concatenation trong loops (bad)
    - StringBuilder trong loops (good)
    - Memory implications

BUỔI 8: Regular Expressions (Regex)

Mục tiêu: Pattern matching mastery

Nội dung:

- **Regex Basics**
  - What is regex?
  - Pattern class
  - Matcher class
  - matches(), find(), group()
- **Regex Syntax**
  - **Character Classes**
    - [abc], [^abc], [a-z], [0-9]
    - \d (digit), \D (non-digit)
    - \w (word char), \W (non-word)
    - \s (whitespace), \S (non-whitespace)
    - . (any character)
  - **Quantifiers**
    - - (0 or more)
    - - (1 or more)
    - ? (0 or 1)
    - {n}, {n,}, {n,m}
  - **Anchors**
    - ^ (start), $ (end)
    - \b (word boundary)
  - **Groups**
    - () capturing groups
    - (?😃 non-capturing groups
    - | (OR operator)
  - **Special Characters**
    - Escape với \
- **String Methods với Regex**
  - matches(String regex)
  - replaceAll(String regex, String replacement)
  - replaceFirst(String regex, String replacement)
  - split(String regex)

▧ MODULE 3: RANDOM & DATE/TIME API (2 BUỔI)

BUỔI 9: Random Numbers & DateTime Basics

Mục tiêu: Random generation và date/time fundamentals

Nội dung:

- **Random Numbers**
  - **java.util.Random**
    - nextInt(), nextLong(), nextDouble(), nextBoolean()
    - nextInt(bound)
    - Random trong range [min, max]
    - Random với seed
  - **Math.random()**
  - **ThreadLocalRandom**
    - Better cho concurrent apps
    - current().nextInt()
- **Date/Time API (java.time) - Part 1**
  - Legacy API overview (Date, Calendar - avoid)
  - **LocalDate**
    - now(), of(), parse()
    - getYear(), getMonth(), getDayOfMonth()
    - getDayOfWeek()
    - plusDays(), minusMonths(), withYear()
    - isAfter(), isBefore(), isEqual()
    - isLeapYear()
  - **LocalTime**
    - now(), of()
    - getHour(), getMinute(), getSecond(), getNano()
    - plusHours(), minusMinutes()
  - **LocalDateTime**
    - Combining date and time
    - toLocalDate(), toLocalTime()

BUỔI 10: Advanced Date/Time & Formatting

Mục tiêu: Complete date/time operations

Nội dung:

- **ZonedDateTime**
  - Timezone aware
  - ZoneId.of()
  - now(ZoneId)
  - Timezone conversions
- **Period & Duration**
  - **Period** (date-based)
    - between()
    - getDays(), getMonths(), getYears()
    - plus(), minus()
  - **Duration** (time-based)
    - between()

        - toHours(), toMinutes(), toSeconds()
- **DateTimeFormatter**
    - Predefined formatters
        - ISO_LOCAL_DATE, ISO_LOCAL_TIME, ISO_LOCAL_DATE_TIME
    - Custom patterns
        - "yyyy-MM-dd"
        - "dd/MM/yyyy HH:mm:ss"
        - "MMM dd, yyyy"
    - format() method
    - parse() method
- **Instant**
    - Timestamp
    - Machine time
    - Conversions

▧ MODULE 4: OBJECT-ORIENTED PROGRAMMING (6 BUỔI)

BUỔI 11: Classes & Objects

Mục tiêu: OOP foundation

Nội dung:

- **OOP Overview**
    - Procedural vs OOP
    - Real-world modeling
    - 4 Pillars: Encapsulation, Inheritance, Polymorphism, Abstraction
- **Classes & Objects**
    - Class declaration
    - Fields (instance variables)
    - Methods (instance methods)
    - Object creation (new keyword)
    - Reference variables
    - Multiple objects
    - null reference
- **Constructors**
    - Default constructor
    - Parameterized constructor
    - Constructor overloading
    - this keyword
    - Constructor chaining (this())
    - Copy constructor
- **Access Modifiers**
    - private, public, protected, default
    - Getters & Setters
    - Encapsulation

BUỔI 12: Static Members & toString()

Mục tiêu: Class-level members

Nội dung:

- **Static Members**
  - Static variables (class variables)
  - Static methods
  - Static blocks
  - Static import
  - When to use static
  - Instance vs Static
- **Object Class Methods**
  - toString()
  - equals() (preview)
  - hashCode() (preview)

BUỔI 13: Inheritance

Mục tiêu: Code reuse through inheritance

Nội dung:

- **Inheritance Basics**
  - extends keyword
  - is-a relationship
  - Parent class (superclass)
  - Child class (subclass)
  - Single inheritance
- **Constructor in Inheritance**
  - super() call
  - Implicit super()
  - Constructor chaining
  - Initialization order
- **Method Overriding**
  - @Override annotation
  - Rules for overriding
  - super.method() call
  - Overriding vs Overloading
- **final Keyword**
  - final variables
  - final methods (cannot override)
  - final classes (cannot extend)
- **protected Access Modifier**

BUỔI 14: Polymorphism

Mục tiêu: Runtime flexibility

Nội dung:

- **Polymorphism Types**
  - Compile-time (overloading)
  - Runtime (overriding)
- **Upcasting & Downcasting**
  - Parent ref = new Child()
  - Upcasting (implicit, safe)
  - Downcasting (explicit, risky)
  - instanceof operator
  - Pattern matching (Java 16+)
- **Dynamic Method Dispatch**
  - Method resolution at runtime
  - Virtual method invocation
- **Polymorphic Collections**
  - Array of parent type
  - ArrayList
  - Processing heterogeneous collections

BUỔI 15: Abstract Classes

Mục tiêu: Abstraction với abstract classes

Nội dung:

- **Abstraction Concept**
  - Hide implementation details
  - Show essential features
  - Real-world analogies
- **Abstract Classes**
  - abstract keyword
  - Abstract methods (no body)
  - Concrete methods (with body)
  - Cannot instantiate
  - Can have constructor
  - Can have fields
  - Can have static members
- **When to Use**
  - Common base với shared code
  - Partial implementation
  - Template Method pattern

BUỔI 16: Interfaces

Mục tiêu: 100% abstraction & multiple inheritance

Nội dung:

- **Interface Basics**
  - interface keyword
  - Abstract methods (implicit public abstract)
  - implements keyword
  - Multiple interface implementation
  - Interface extending interface
- **Interface Fields**
  - public static final (implicit)
  - Constants only
- **Modern Interface Features (Java 8-9)**
  - Default methods (Java 8)
  - Static methods (Java 8)
  - Private methods (Java 9)
- **Abstract Class vs Interface**
  - When to use which
  - Comparison table
  - Design decisions

▧ MODULE 5: DESIGN PATTERNS CƠ BẢN (4 BUỔI)

BUỔI 17: Singleton & Factory

Mục tiêu: Creational patterns

Nội dung:

- **Design Patterns Introduction**
  - Gang of Four (GoF)
  - Categories: Creational, Structural, Behavioral
- **Singleton Pattern**
  - One instance only
  - Implementations:
    - Eager initialization
    - Lazy initialization
    - Thread-safe (synchronized)
    - Double-checked locking
    - Bill Pugh (recommended)
- **Factory Method Pattern**
  - Create objects without specifying exact class
  - Product interface
  - Concrete products
  - Factory method
- **Simple Factory**

BUỔI 18: Builder & Prototype

Mục tiêu: Complex object creation

Nội dung:

- **Builder Pattern**
  - Telescoping constructor problem
  - Builder class (static nested)
  - Method chaining
  - Fluent interface
  - When to use
- **Prototype Pattern**
  - Clone objects
  - Shallow vs Deep copy
  - Cloneable interface
  - clone() method
  - When to use

BUỔI 19: Adapter & Decorator

Mục tiêu: Structural patterns

Nội dung:

- **Adapter Pattern**
  - Convert incompatible interfaces
  - Object Adapter (preferred)
  - Target, Adaptee, Adapter
  - Real-world examples
- **Decorator Pattern**
  - Add functionality dynamically
  - Component, Decorator, Concrete Decorators
  - Wrapper pattern
  - Alternative to subclassing

BUỔI 20: Strategy & Observer

Mục tiêu: Behavioral patterns

Nội dung:

- **Strategy Pattern**
  - Family of algorithms
  - Interchangeable
  - Strategy interface, Context
  - Eliminate conditionals
- **Observer Pattern**
  - One-to-many dependency
  - Subject and Observers

  - attach(), detach(), notify()
  - Pub-Sub model

▧ MODULE 6: SOLID & CLEAN CODE (2 BUỔI)

BUỔI 21: SOLID Principles

Mục tiêu: Design principles mastery

Nội dung:

- **S - Single Responsibility Principle**
  - One class, one responsibility
  - Violations & fixes
- **O - Open/Closed Principle**
  - Open for extension, closed for modification
  - Use abstraction
- **L - Liskov Substitution Principle**
  - Subtypes must be substitutable
  - Rectangle-Square problem
- **I - Interface Segregation Principle**
  - Many specific interfaces > fat interface
  - Client-specific interfaces
- **D - Dependency Inversion Principle**
  - Depend on abstractions
  - Dependency Injection

BUỔI 22: Clean Code & Code Smells

Mục tiêu: Professional code quality

Nội dung:

- **Clean Code Principles**
  - Meaningful names
  - Small functions (< 20 lines)
  - Function arguments (0-3)
  - Comments vs self-documenting
  - DRY, KISS
- **Code Smells**
  - Long Method
  - Large Class
  - Long Parameter List
  - Duplicate Code
  - Dead Code
  - Primitive Obsession
  - Switch Statements

- **Refactoring Techniques**
  - Extract Method
  - Extract Class
  - Replace Conditional with Polymorphism
  - Introduce Parameter Object
- **Error Handling**
  - Use exceptions
  - Provide context
  - Don't return null (Optional)

▧ MODULE 7: EXCEPTION HANDLING (2 BUỔI)

BUỔI 23: Exception Fundamentals

Mục tiêu: Error handling basics

Nội dung:

- **Exception Hierarchy**
  - Throwable → Error, Exception
  - Checked vs Unchecked
  - RuntimeException
- **Try-Catch-Finally**
  - Basic try-catch
  - Multiple catch blocks
  - Multi-catch (Java 7+)
  - Finally block
- **Try-with-Resources (Java 7+)**
  - AutoCloseable interface
  - Resource management
  - Multiple resources

BUỔI 24: Custom Exceptions & Best Practices

Mục tiêu: Professional error handling

Nội dung:

- **Throwing Exceptions**
  - throw keyword
  - throws clause
- **Custom Exceptions**
  - Creating custom exceptions
  - Exception hierarchy
  - Exception chaining
- **Best Practices**
  - When to catch vs throw

  - Specific exceptions
  - Logging exceptions
  - Fail-fast
  - Don't catch Throwable/Exception
  - Clean up resources

▧ MODULE 8: LAMBDA & FUNCTIONAL PROGRAMMING (3 BUỔI)

BUỔI 25: Lambda Expressions

Mục tiêu: Functional programming introduction

Nội dung:

- **Lambda Basics**
  - What are Lambdas?
  - Syntax: (parameters) -> expression
  - Syntax: (parameters) -> { statements; }
  - Type inference
- **Functional Interfaces**
  - @FunctionalInterface
  - Single Abstract Method (SAM)
  - Built-in functional interfaces:
    - Predicate: test()
    - Function<T, R>: apply()
    - Consumer: accept()
    - Supplier: get()
- **Method References**
  - Static method: Class::staticMethod
  - Instance method: object::instanceMethod
  - Instance method of arbitrary object: Class::instanceMethod
  - Constructor: Class::new

BUỔI 26: Advanced Functional Interfaces

Mục tiêu: More functional programming

Nội dung:

- **More Built-in Interfaces**
  - BiPredicate<T, U>
  - BiFunction<T, U, R>
  - BiConsumer<T, U>
  - UnaryOperator extends Function<T, T>
  - BinaryOperator extends BiFunction<T, T, T>
- **Specialized Interfaces**
  - IntPredicate, LongPredicate, DoublePredicate

    - IntFunction, LongFunction, DoubleFunction
    - IntConsumer, LongConsumer, DoubleConsumer
    - IntSupplier, LongSupplier, DoubleSupplier
    - ToIntFunction, ToLongFunction, ToDoubleFunction
- **Combining Functions**
    - and(), or(), negate() for Predicate
    - andThen(), compose() for Function
    - andThen() for Consumer

BUỔI 27: Optional & Functional Design

Mục tiêu: Null safety và functional patterns

Nội dung:

- **Optional**
    - Creating Optional:
        - Optional.of()
        - Optional.ofNullable()
        - Optional.empty()
    - Checking value:
        - isPresent(), isEmpty()
        - ifPresent()
    - Retrieving value:
        - get() (avoid!)
        - orElse()
        - orElseGet()
        - orElseThrow()
    - Transforming:
        - map()
        - flatMap()
        - filter()
- **Optional Best Practices**
    - When to use Optional
    - Anti-patterns to avoid
    - Optional in method returns
- **Functional Design Patterns**
    - Strategy with Lambdas
    - Command pattern
    - Template method
    - Chain of responsibility

▧ MODULE 9: STREAM API (2 BUỔI)

BUỔI 28: Stream API Basics

Mục tiêu: Stream fundamentals

Nội dung:

- **Stream Introduction**
  - What is Stream?
  - Stream vs Collection
  - Stream pipeline: source → intermediate → terminal
  - Lazy evaluation
  - Stream không thay đổi source
- **Creating Streams**
  - collection.stream()
  - Arrays.stream()
  - Stream.of()
  - Stream.generate()
  - Stream.iterate()
  - IntStream, LongStream, DoubleStream
- **Intermediate Operations**
  - filter(Predicate)
  - map(Function)
  - flatMap(Function)
  - distinct()
  - sorted(), sorted(Comparator)
  - peek()
  - limit(), skip()
- **Terminal Operations**
  - forEach()
  - count()
  - collect()
  - toArray()
  - reduce()
  - min(), max()
  - anyMatch(), allMatch(), noneMatch()
  - findFirst(), findAny()

BUỔI 29: Advanced Stream API

Mục tiêu: Stream mastery

Nội dung:

- **Collectors**
  - Collectors.toList()
  - Collectors.toSet()
  - Collectors.toMap()
  - Collectors.joining()
  - Collectors.counting()
  - Collectors.summingInt/Long/Double()
  - Collectors.averagingInt/Long/Double()

    - Collectors.summarizingInt/Long/Double()
    - Collectors.maxBy(), minBy()
    - Collectors.groupingBy()
    - Collectors.partitioningBy()
- **Reduce Operations**
    - reduce(BinaryOperator)
    - reduce(identity, BinaryOperator)
    - reduce(identity, BiFunction, BinaryOperator)
    - Sum, product, concatenation
- **Parallel Streams**
    - parallelStream()
    - parallel()
    - When to use parallel
    - Performance considerations
    - Thread safety
- **Stream Best Practices**
    - Avoid side effects
    - Use appropriate operations
    - Don't reuse streams
    - Performance tips

▧ MODULE 10: COLLECTIONS FRAMEWORK (4 BUỔI)

BUỔI 30: Collections Overview & List

Mục tiêu: Collection hierarchy & List

Nội dung:

- **Collections Framework**
    - Iterable → Collection
    - List, Set, Queue interfaces
    - Map (not in Collection hierarchy)
- **List Interface**
    - Ordered, allows duplicates
    - Index-based access
- **ArrayList**
    - Dynamic array
    - Internal working
    - Capacity vs size
    - Time complexity: O(1) get, O(n) add/remove
    - When to use
- **LinkedList**
    - Doubly-linked list
    - Time complexity: O(1) add/remove at ends
    - Implements List, Deque
    - When to use

- **ArrayList vs LinkedList**
  - Performance comparison
  - Use cases

BUỔI 31: Set & Map

Mục tiêu: Unique collections & key-value pairs

Nội dung:

- **Set Interface**
  - No duplicates
- **HashSet**
  - Hash table
  - O(1) operations
  - Unordered
  - hashCode() & equals() importance
- **LinkedHashSet**
  - Insertion order
- **TreeSet**
  - Sorted (natural or Comparator)
  - Red-black tree
  - O(log n) operations
  - NavigableSet features
- **Map Interface**
  - Key-value pairs
  - Unique keys
- **HashMap**
  - Hash table
  - O(1) operations
  - null key/values allowed
  - Load factor, capacity
- **LinkedHashMap**
  - Insertion/access order
  - LRU cache
- **TreeMap**
  - Sorted by keys
  - NavigableMap

BUỔI 32: Queue & Utilities

Mục tiêu: FIFO & utility classes

Nội dung:

- **Queue Interface**
  - FIFO

    - offer(), poll(), peek()
- **PriorityQueue**
    - Heap-based
    - Natural/custom ordering
- **Deque Interface**
    - Double-ended queue
    - Stack, Queue operations
- **ArrayDeque**
    - Resizable array
    - Faster than LinkedList
- **Collections Utilities**
    - sort(), reverse(), shuffle()
    - binarySearch(), min(), max()
    - frequency(), disjoint()
    - synchronizedXxx()
    - unmodifiableXxx()
- **Arrays Utilities**
    - sort(), binarySearch()
    - asList(), copyOf()
    - equals(), toString()

BUỔI 33: Comparable & Comparator

Mục tiêu: Custom sorting

Nội dung:

- **Comparable**
    - Natural ordering
    - compareTo() method
    - Implementing Comparable
- **Comparator**
    - Custom ordering
    - compare() method
    - Multiple Comparators
- **Comparator Methods (Java 8+)**
    - comparing()
    - thenComparing()
    - reversed()
    - nullsFirst(), nullsLast()
- **Sorting**
    - Collections.sort()
    - List.sort()
    - Stream.sorted()

▧ MODULE 11: INPUT/OUTPUT (2 BUỔI)

BUỔI 34: File I/O & Streams

Mục tiêu: File operations

Nội dung:

- **File Class**
  - Creating files/directories
  - File properties
  - Delete, rename
- **Byte Streams**
  - InputStream, OutputStream
  - FileInputStream, FileOutputStream
  - BufferedInputStream, BufferedOutputStream
  - DataInputStream, DataOutputStream
- **Character Streams**
  - Reader, Writer
  - FileReader, FileWriter
  - BufferedReader, BufferedWriter
  - PrintWriter
- **Try-with-Resources**
  - AutoCloseable
  - Resource management

BUỔI 35: NIO & Serialization

Mục tiêu: Modern I/O & object persistence

Nội dung:

- **NIO (java.nio)**
  - Path, Paths, Files
  - Reading/writing efficiently
  - Walking file tree
- **Serialization**
  - Serializable
  - ObjectOutputStream, ObjectInputStream
  - transient
  - serialVersionUID
  - Custom serialization

▧ MODULE 12: SQL & DATABASE (4 BUỔI)

BUỔI 36: SQL Fundamentals

Mục tiêu: Database & SQL basics

Nội dung:

- **Database Concepts**
  - RDBMS
  - Tables, rows, columns
  - Primary key, Foreign key
  - Relationships
- **SQL Basics**
  - DDL: CREATE, ALTER, DROP
  - DML: INSERT, UPDATE, DELETE
  - DQL: SELECT
  - WHERE, ORDER BY, LIMIT
- **Operators**
  - =, !=, >, <, LIKE, IN, BETWEEN
  - AND, OR, NOT

BUỔI 37: Advanced SQL

Mục tiêu: Complex queries

Nội dung:

- **Aggregate Functions**
  - COUNT, SUM, AVG, MIN, MAX
  - GROUP BY
  - HAVING
- **Joins**
  - INNER JOIN
  - LEFT JOIN
  - RIGHT JOIN
- **Subqueries**
  - WHERE, FROM, SELECT subqueries

BUỔI 38: JDBC Basics

Mục tiêu: Connect Java to Database

Nội dung:

- **JDBC Architecture**
  - Driver types
  - JDBC URL
- **JDBC Steps**
  1. Load driver (optional Java 6+)
  2. Get connection
  3. Create statement
  4. Execute query
  5. Process results
  6. Close resources

- **Statement Types**
  - Statement
  - PreparedStatement (recommended)
  - CallableStatement

BUỔI 39: JDBC Advanced & DAO

Mục tiêu: Professional database layer

Nội dung:

- **Transaction Management**
  - autoCommit = false
  - commit(), rollback()
  - Savepoints
- **Batch Processing**
  - addBatch(), executeBatch()
- **Connection Pooling**
  - HikariCP setup
- **DAO Pattern**
  - Data Access Object
  - CRUD interface
  - Separation of concerns

▧ MODULE 13: MULTITHREADING (2 BUỔI)

BUỔI 40: Thread Fundamentals

Mục tiêu: Concurrency basics

Nội dung:

- **Thread Basics**
  - Thread vs Process
  - Thread lifecycle
- **Creating Threads**
  - Extend Thread
  - Implement Runnable (preferred)
- **Thread Methods**
  - start(), run()
  - sleep(), join()
  - isAlive()
  - getName(), setName()
  - Priority

BUỔI 41: Synchronization & Thread Safety

Mục tiêu: Thread-safe programming

Nội dung:

- **Synchronization**
  - Race condition
  - synchronized keyword
  - wait(), notify(), notifyAll()
  - Producer-Consumer
- **Thread Safety**
  - Atomic classes
  - volatile
  - Thread-safe collections

📚 MODULE 14: GENERICS (1 BUỔI)

BUỔI 42: Generics Complete

Mục tiêu: Type-safe programming

Nội dung:

- **Generics Basics**
  - Generic classes
  - Generic methods
  - Generic interfaces
  - Type parameters
- **Bounded Types**
  - Upper bound:
  - Multiple bounds
- **Wildcards**
  - , ,
  - PECS principle
- **Type Erasure**

🎯 CAPSTONE PROJECT

Project: Task Management System

Mô tả: Xây dựng hệ thống quản lý công việc hoàn chỉnh

Features:

1. User Management (Register, Login)
2. Project Management (CRUD)
3. Task Management (CRUD, assign, priority, status)
4. Data Persistence (Database)
5. Reporting (statistics, export CSV)

6. Logging system

Technical Stack:

- ✅ OOP Design
- ✅ Design Patterns (min 3)
- ✅ SOLID Principles
- ✅ Exception Handling
- ✅ Lambda & Stream API
- ✅ Collections Framework
- ✅ File I/O
- ✅ JDBC & DAO Pattern
- ✅ Multithreading (optional)
- ✅ Generics

Deliverables:

- Source code (GitHub)
- Database schema
- README
- Demo presentation

📚 TÀI LIỆU HỌC TẬP

Sách Recommended:

1. **"Head First Java"** - Kathy Sierra
2. **"Effective Java"** - Joshua Bloch ⭐
3. **"Clean Code"** - Robert C. Martin
4. **"Modern Java in Action"** - Stream API & Lambdas

Online Resources:

- Oracle Java Tutorials (Official)
- Baeldung.com
- Java Brains (YouTube)
- GeeksforGeeks

Practice Platforms:

- LeetCode
- HackerRank
- Codewars
- Exercism

🚀 SAU KHÓA HỌC

Bạn sẽ có khả năng:

✅ Viết Java code chuyên nghiệp
✅ Thiết kế OOP applications
✅ Áp dụng Design Patterns
✅ Làm việc với Databases
✅ Handle concurrency
✅ Modern Java (Lambda, Stream API)
✅ Sẵn sàng cho Spring Framework

Next Steps:

1. **Spring Framework / Spring Boot**
2. **Hibernate / JPA**
3. **REST API Development**
4. **Microservices Architecture**
5. **Cloud Deployment (AWS, Azure, GCP)**

Good luck! 💪

Version 2.0 - Complete Edition