Backend Development

Introduction

Welcome to our comprehensive programming course designed to take you from the basics of computing to advanced object-oriented programming concepts. This syllabus is structured into three parts:

1. **First Part - Python and Computer Basics**
2. **Second Part - Programming Basics with Java**
3. **Third Part - Advanced Object-Oriented Programming & Java 8 Features**
4. **Fourth Part-Databases, SQL, JDBC, Virtualization, Docker**
5. **Fifth Part-Java EE & Spring Boot**

A. First Part - Python and Computer Basics (2 Months)

Introduction to Computing and Programming

- **Introduction to Computers and Linux**
  - Basics of computers and components (CPU, RAM, Storage)
  - Understanding how RAM works
  - Introduction to Linux operating system
  - Linux vs other OS (Windows, Mac)
  - Basic Linux commands: `ls`, `cd`, `pwd`, `mkdir`, `rm`
  - Working with the terminal/shell
  - **Quiz & Practice**
- **Introduction to Python and IDEs**
  - What is Python? Why choose Python?
  - Setting up Python on Linux/Windows
  - Writing your first Python program in the terminal (`Hello, World!`)
  - Python IDEs and editors: PyCharm, VSCode, Spyder
- **Version Control with Git**
  - Introduction to Git and GitHub
  - What is version control?
  - Installing Git on Linux
  - Basic Git commands:
    - `git init`, `git clone`
    - `git status`, `git add`, `git commit`
    - `git push`, `git pull`
  - Working with GitHub: creating repositories, pushing code
  - **Quiz & Practice**

Python Basics

- **Python Syntax and Data Types**
  - Variables, basic data types (`int`, `float`, `bool`, `str`)
  - Input/output in Python (`input()`, `print()`)
  - Type conversions
  - **Quiz & Practice**
- **Basic Operations**
  - Arithmetic, comparison, logical operations
  - String operations
  - **Quiz & Practice**
- **Control Flow in Python**
  - **Conditional Statements**
    - `if`, `elif`, `else`
    - Nested conditionals
    - **Quiz & Practice**
  - **Loops**
    - `for`, `while` loops

        - Loop control statements: `break`, `continue`, `pass`
        - **Quiz & Practice**
- **Functions and Modules**
    - **Functions**
        - Defining and calling functions
        - Function arguments and return values
        - Variable scope (global vs local)
        - **Quiz & Practice**
    - **Modules and Packages**
        - Importing modules (`math`, `os`)
        - Writing and using your own modules
        - **Quiz & Practice**
- **Collections (Lists, Sets, Tuples, Dictionaries)**
    - **Introduction to Python Collections**
        - **Lists**: creation, manipulation, methods (`append`, `remove`, etc.)
        - **Tuples**: immutable sequences, use cases
        - **Sets**: uniqueness and set operations (union, intersection)
        - **Dictionaries**: key-value pairs, accessing and modifying elements
        - **Quiz & Practice**
- **Sorting and Searching Algorithms**
    - Basic sorting algorithms: Bubble Sort, Merge Sort, Quick Sort
    - Searching algorithms: Linear Search, Binary Search
    - Time complexity analysis (Big O notation)
    - **Quiz & Practice**
- **Project and Final Exam**
    - **Mini Python Project**
        - Choose a small project (e.g., to-do list, file manager, or calculator)
        - Using Git to version control the project
        - Presenting the project
    - **Final Exam**

B. Second Part - Programming Basics with Java

Introduction/Software Setup

- **Introduction to ICT (Career plan)**
  - Syllabus introduction & course procedures
  - Requirements
- **Basics of ICT**
  - Information in digital form, number systems, binary system
  - What is programming?
- **Introduction to Programming**
  - Algorithmic thinking, reasoning
  - What are Programming Languages (PLs)?
  - How to choose a PL to learn?
  - **JDK, JRE, JVM, IDE?**
    - Platform independency, C++ vs Java
    - `javac` vs `java`
    - How to compile & run Java code from terminal/cmd
    - `.java` and `.class` files - source code & bytecode & machine code
    - JDK & JRE & JVM?
    - IDEs - IntelliJ IDEA, NetBeans, Eclipse
- **What is VCS (Git / GitHub)?**
  - Git download and installing
  - Overview about version control systems
  - Initializing or cloning a repository
  - Basic git commands: `clone`, `init`, `status`, `add`, `commit`, `push`, `pull`
  - **Quiz & Practice**

Java Basics

- **Java syntax, writing first “Hello, World!” app in Java**
  - Manifest: `public static void main (String[] args) { ... }`
- **Print to console**
  - `System.out.print("Hello, World");`
  - `System.out.println("Hello, World");`
  - `System.out.printf("Hello, %s", "World");`
  - `System.out.printf("Hello, World: %.2f\n", 50.0);`
- **Storing data - Variables - declaration & initialization**
- **Data types**
  - **Primitive types**
    - `byte`, `short`, `int`, `long`, `float`, `double`
    - `char`, `boolean`
  - **Reference types**
- **Comments**
  - Single line comment
  - Multiple lines (block) comment

    - Documentation comment
- **Operations**
    - Arithmetic operations
    - Relational operations
    - Logical operations
    - Assignment operations
    - Miscellaneous operations
    - **Quiz & Practice**

Control Flow

- **Input from console - Scanner class**
    - Code structure: input -> process -> output
- **Conditional statements**
    - `if`
    - `if - else`
    - `if - else if - else`
    - `switch - case`
    - Ternary operator
- **Loops**
    - `for`
    - `while`
    - `do - while`
    - `break`, `continue`
    - Nested conditions and loops
    - **Quiz & Practice**

Arrays

- **Declaration, initialization of arrays**
- **Operations on an array** (fill, print, find max, min, copy etc.)
- **Enhanced for loop ("for-each")**
- **How memory works for arrays** (stack vs heap memory)
- **Two and more dimensional arrays**
- **Quiz & Practice**

Methods

- **Declaration of methods, method signature**
- **Parametric & non-parametric methods**
- **Void & value methods**
- **Overloading, rules for overloading**
- **Quiz & Practice**

String Class

- **Character array and understanding String**

- **String under the hood**
- **Methods of String class** (some)
    - `toLowerCase()` & `toUpperCase()`
    - `substring()` & `trim()`
    - `indexOf(String s)` & `indexOf(int i)`
    - `split()`, `replace()`, `length()`, `concat()`
- **Memory (RAM) intro (stack vs heap)**
    - Memory for String management, String pool
    - Reference and how this works?
- **Passing values**
    - Passing-by-value
    - Passing-by-reference
- **String concatenation:**
    - `+` operator for strings
    - `concat()`
    - `StringBuilder`
    - `StringBuffer`
    - Comparison of above solutions
    - **Quiz & Practice**

Problem Solving with Predefined Tasks/Games

- **Number guess game**
- **Ship battle game**
- **Week planner game**
- **Module 02 Final Exam**

C. Third Part - Advanced Object-Oriented Programming & Java 8

"Java Standard Edition & Java 8 Features"

Object-Oriented Programming (OOP)

- **Object and class**
- **Constructors, object initialization**
- **Types of variables**
  - Instance variables
  - Local variables
  - Static (global) variables
- **Static vs non-static methods and variables**
- **References/Garbage Collection**
- **Getters and setters**
- **Encapsulation**
  - Access modifiers
  - Getters-setters
  - **Quiz & Practice**
- **Inheritance**
  - IS-A relationship
  - HAS-A relationship
  - Object class
    - `toString()`, `equals()`, `hashCode()`
  - **Quiz & Practice**
- **Polymorphism**
  - **Quiz & Practice**
- **Abstraction**
  - **Quiz & Practice**
- **Keywords: `this` & `super` & `instanceof`**
- `@Override`
- Compile-time (overloading) vs runtime (overriding) polymorphism

Object-Oriented Programming (OOP)

- **Abstract classes**
  - **Quiz & Practice**
- **Interfaces**
  - **Quiz & Practice**
- **Abstract classes vs interfaces in depth**
  - Functional & Marker Interfaces
  - **Quiz & Practice**

Object-Oriented Programming (OOP)

- **Enumeration**

- **Immutability**
    - Final class
    - Final method
    - Final fields, parameters
    - Recursive immutability
- **Var keyword**

Object-Oriented Programming (OOP)

- **Packaging, built-in packages**
    - Importing: single vs whole imports, static imports
    - UML diagrams for class designing
- **Wrapper types**
- **Casting (upcasting, downcasting)**
- **Boxing and unboxing. Autoboxing**
- **Quiz & Practice**

Date and Time API

- **LocalDate**
- **LocalTime**
- **LocalDateTime**
- **Date vs LocalDate**
- `java.util.Date` vs `java.sql.Date`

Exceptions

- **Exception hierarchy**
- **Error vs Exception**
- **Checked and unchecked exceptions**
- **Try-catch**
- **Multiple catch and union catch**
- **Swallowing exceptions**
- **Custom Exceptions**
- **Throw vs Throws**
- **Quiz & Practice**
- **Module 03 Midterm/Part01 Exam**

Generics & Optional

- **Need for Generics**
    - Diamond operator
    - Type wildcards (lower and upper bounds)
    - Generic class definitions
    - Generic method definitions
- **Optional class and its usage**
- **Introduction to Functional Programming**
    - Method chaining strategy

    - **Practice**

Sorting and Comparing

- **Comparable vs Comparator**
    - **Hackerrank Task Link!!!**

Functional Interfaces, Common Functional Interfaces in Java 8

- **Anonymous classes & methods, lambda expressions**
- **Method references**
- `Arrays.sort()`
- **Quiz & Practice**

Introduction to Algorithms

- **Introduction to complexity analysis**
    - Worst case scenario (Big O)
    - Best case scenario (Big Omega)
- **Searching**
    - Linear search
    - Binary search
- **Sorting**
    - Bubble sort
    - Selection sort
    - Merge sort

Introduction to Data Structures

- **Introduction to Java Collection Framework (API)**
    - **ArrayList**
    - **LinkedList**
    - **Map**
        - Hashing vs Encoding vs Encryption
        - Contract between `equals()` and `hashCode()`
    - **Set**
    - **Queue vs Deque (Stack)**
    - **HashSet vs LinkedHashSet vs TreeSet**
    - **HashMap vs LinkedHashMap vs TreeMap**
    - **Quiz & Practice**

Introduction to Java Stream API

- **Introduction to Java Stream API**
    - Input --> Process --> Output
    - Controller --> Service --> DAO
    - Source --> Intermediate --> Terminal Operations

    - **Quiz & Practice**

File Input/Output

- **File reading and writing with `io`**
    - Input, output, error with System class (`in`, `out`, `err`)
    - Character streams vs byte streams
    - `FileReader` and `FileWriter`
    - Buffered file operations
- **File reading and writing with `nio`**
- **Try-with-finally**
- **Try-with-resources**
- **Quiz & Practice**

Serialization & Reflection

- **Serialization, object streams**
    - Writing object into file (text, binary & object)
    - Binary vs XML vs JSON serialization
    - `Transient` keyword and its mechanism
- **Introduction to Reflection API**
    - Java class object, fields, methods, constructors
    - Dynamic invocation, annotations

Multithreading

- **Introduction to multithreading, process vs thread vs task**
- **Thread class**
- **Runnable interface**
- **Callable interface**
- **Execution service**
- **Concurrency API**
- **Atomic Scalars**

Creating Proper Project Structure

- **Build tools & packaging with Maven & Gradle**
- **Step Project Intro**
- **Module02 final exam preparation**
- **Quiz & Practice**

Student Management App Coding via DAO (in-memory & file) GitHub

- **Module 03 Final Exam**

D. Fourth Part - Databases, SQL, JDBC, Virtualization, Docker

Introduction to Databases

1. **Introduction / Software Setup / Database Fundamentals**
   - Installing PostgreSQL
   - Database Tools
     - DataGrip
     - DBeaver
   - Database Creation and Connection
   - Understanding Relational vs Non-relational Databases
   - SQL Syntax Overview
   - SQL Command Categories
     - DML (Data Manipulation Language)
     - DDL (Data Definition Language)
     - TCL (Transaction Control Language)
     - DCL (Data Control Language)
   - SQL Data Types
   - Basic DDL Commands
     - `CREATE DATABASE` / `SCHEMA` / `TABLE` / `TRUNCATE`
     - `DROP DATABASE` / `SCHEMA` / `TABLE`
     - `ALTER DATABASE` / `SCHEMA` / `TABLE`
     - `SELECT` / `SELECT DISTINCT` / `INSERT INTO`
   - **Additional Learning Resources**
     - W3Schools SQL Lessons
     - PostgreSQL Documentation and Tutorials
2. **SQL Fundamentals: DML for Basic CRUD Operations**
   - **Filtering and Sorting Data**
     - `SELECT`, `SELECT DISTINCT`, `LIMIT`, `ORDER BY`, `CASE`
     - `WHERE` Clause with `AND`, `OR`, `NOT`, `NULL` Operators
     - `UPDATE`, `SET`, `DELETE` Commands
     - **Aliases and Wildcards:** `AS`, `LIKE`, `IN`, `BETWEEN`
   - **Aggregate Functions**
     - `MIN()`, `MAX()`, `SUM()`, `COUNT()`, `AVG()`
     - **Quiz & Practice**
3. **SQL Constraints**
   - **Types of Constraints**
     - `NOT NULL`
     - `UNIQUE`
     - `AUTO INCREMENT`
     - `PRIMARY KEY`
     - `CHECK`
     - `DEFAULT`

        - `INDEX`
        - `FOREIGN KEY`
    - **Relational Database Modeling**
        - Entity-Relationship Diagrams (ERDs)
        - **Quiz & Practice**
4. **Relationships in Databases**
    - **One-to-One Relationships**
    - **One-to-Many and Many-to-One Relationships**
    - **Many-to-Many Relationships**
    - **Joins in SQL**
        - `INNER JOIN`
        - `LEFT JOIN`
        - `RIGHT JOIN`
        - `FULL OUTER JOIN`
        - Cartesian Products
    - **Advanced SQL Clauses**
        - `UNION`
        - `GROUP BY`
        - `HAVING`
        - `EXISTS`
        - `ANY`, `ALL`
    - **Developing Database Structure for a Booking App**
5. **JDBC (Java Database Connectivity)**
    - **Introduction to JDBC API**
    - **Database Drivers**
    - **Establishing Database Connections**
    - **Executing SQL Statements**
        - `Statement`
        - `PreparedStatement`
        - `CallableStatement`
    - **Processing ResultSets**
    - **Managing Transactions**
        - JDBC Transactions
        - Auto-commit Mode
        - **Quiz & Practice**
6. **Virtualization and Containerization with Docker**
    - **Understanding Virtualization and VMs**
    - **Introduction to Containerization**
    - **Docker Fundamentals**
    - **Creating PostgreSQL Containers**
    - **Connecting Java Applications to Docker Containers**
    - **Basic Docker Commands**
    - **Docker Architecture Overview**
    - **Using Docker Hub**
    - **Networking in Docker**
        - Network Modes for Applications

                - **Quiz & Practice**

7. **Practical Project**
    - **Developing a Booking Application with PostgreSQL Database**
    - **Applying Database Concepts**
    - **Implementing JDBC for Data Access**
    - **Utilizing Docker for Deployment**
8. **Module 4 Final Exam**

E. Fifth Part: Java EE & Spring Boot

Introduction into Web Development

1. **Internet Fundamentals**
   - How Does the Internet Work?
   - Hosting, OSI, TCP/IP Models
   - Browsers, HTTP(s), DNS
   - HTTP Request and Response
   - Server, Handler, Mapping, Servlet
   - Basic Application with Java EE
2. **Spring Framework and Spring Boot**
   - Inversion of Control (IoC), ApplicationContext
   - Dependency Injection (DI) Strategies
   - Spring vs Spring Boot
   - Migrating from Java EE to Spring Boot (Demo)
   - Conventional Project Structure
   - Spring Beans and Bean Configurations
3. **Spring Boot Annotations**
   - `@SpringBootApplication`
   - `@EnableAutoConfiguration`
   - `@SpringBootConfiguration`
   - `@ComponentScan`, `@AliasFor`
   - `@Configuration`, `@Bean`
   - `@Controller`, `@RequestMapping`
   - `@GetMapping`, `@PostMapping`, `@PutMapping`, `@DeleteMapping`
   - `@ResponseStatus`, `@ResponseBody`, `@RestController`
   - `@PathVariable`, `@RequestParam`, `@RequestBody`
   - `@Service`, `@Repository`, `@Autowired*`
   - `@ConfigurationProperties(prefix="custom")`
   - **Application Configuration**
     - `application.yaml` vs `application.properties`
     - `@Value`
4. **Project Lombok and Spring Validation**
   - `@Setter`, `@Getter`, `@ToString`, `@EqualsAndHashCode`, `@Data`
   - `@NoArgsConstructor`, `@AllArgsConstructor`
   - **Validation Annotations**
     - `@Valid`, `@Validated`
     - `@NotNull`, `@NotBlank`, `@NotEmpty`
     - `@Min`, `@Max`, `@Size`, `@Email`, `@Pattern`
5. **Global Exception Handling, Logging, File Operations**
   - `@RestControllerAdvice`
   - `@ExceptionHandler`
   - **Logging with Log4j2**

        - `@Log4j2`
        - Logging Levels: `ERROR`, `INFO`, `DEBUG`, `TRACE`
    - **Uploading and Downloading Files**
        - Handling `byte[]` & `MultipartFile`

6. **Spring Web**
    - **Thymeleaf Template Engine**
    - **Internationalization**
    - **Static Resources Management**
7. **Data Access Layer #1: CRUD Operations with JDBC Template**
8. **Data Access Layer #2: Spring Data JPA and MapStruct**
    - **Understanding ORM**
    - Hibernate vs Spring Data JPA
    - **Repositories**
        - `@Repository`, `JpaRepository`
    - **Database Connection Configuration**
        - Via `application.yaml`
    - **Entity Mapping**
        - `@Table`, `@Entity`, `@Id`, `@GeneratedValue`, `@Column`
    - **Transaction Management**
        - `@Transactional`, `@Transactional(readOnly = true)`
    - **Object Mapping with MapStruct**
        - `@Mapper`, `@Mapping`
9. **Data Access Layer #3: Relationships**
    - **One-to-One (`@OneToOne`)**
    - **One-to-Many (`@OneToMany`)**
    - **Many-to-Many (`@ManyToMany`)**
10. **Liquibase Database Migration & Versioning**
11. **Making HTTP Requests and Scheduled Jobs**
    - **REST Clients**
        - `RestTemplate`
        - `FeignClient`
    - **Feign Configuration**
        - `FeignConfig`, `ErrorDecoder`
    - **Scheduled Tasks**
        - `@Scheduled`, `@EnableScheduling`
        - Parameters: `fixedRate`, `fixedDelay`, `initialDelay`, Cron Expressions
12. **Unit Testing and Swagger Documentation**
    - **Types of Testing**
    - **Testing Frameworks**
        - Mockito
        - JUnit
    - **API Documentation**
        - OpenAPI
        - SpringFox

13. **Spring Security**
14. **Module 4 Final Project and Exam**