BCS-403: Object Oriented Programming With Java

60-Hour / 36-Session (100 Minutes Each) Full Course Plan

🔹 CORE JAVA FOUNDATION

SESSION 1 – Introduction to Java & Environment

Objective:

Help students understand *what Java is*, *why it is used*, and *how Java programs execute internally*.

Detailed Content

- **Evolution of Java and why it dominates backend development**
- **Platform independence: *Write Once, Run Anywhere***
- **Difference between compiler vs interpreter**
- **JDK (tools), JRE (runtime), JVM (execution engine)**
- **Java execution flow:**
  `.java → compiler → .class → JVM → OS`
- **Structure of a Java program**
- **Role of `main()` method**
- **Command-line compilation vs IDE execution**

Example

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Welcome to Java Programming");
    }
}
```

LeetCode

1. **Two Sum – https://leetcode.com/problems/two-sum/**
2. **Add Digits – https://leetcode.com/problems/add-digits/**

SESSION 2 – Data Types, Variables & Input

Objective:

Understand how data is stored, typed, and taken from users.

Detailed Content

- **Primitive data types & memory size**
- **Reference variables (objects, arrays)**
- **Variable declaration rules & naming conventions**
- **Type casting (implicit & explicit)**
- **Data loss during narrowing conversion**
- **Using `Scanner` for real-world inputs**

Example

```
Scanner sc = new Scanner(System.in);
```

```
int age = sc.nextInt();
double salary = sc.nextDouble();
```

LeetCode

1. **Reverse Integer – https://leetcode.com/problems/reverse-integer/**
2. **Palindrome Number – https://leetcode.com/problems/palindrome-number/**

SESSION 3 – Operators & Expressions

Objective:
Enable students to write logical and mathematical expressions.

Detailed Content

- **Arithmetic & relational operators**
- **Logical operators and short-circuiting**
- **Unary operators (`++`, `--`)**
- **Ternary operator for compact conditions**
- **Operator precedence & evaluation order**

Example

```
int a = 10, b = 20;
int max = (a > b) ? a : b;
```

LeetCode

1. **Single Number – https://leetcode.com/problems/single-number/**
2. **Plus One – https://leetcode.com/problems/plus-one/**

SESSION 4 – Conditional Statements

Objective:
Teach decision-making logic for real programs.

Detailed Content

- **if, if–else, nested if**
- **else-if ladder**
- **switch-case usage**
- **Menu-driven logic examples**
- **Avoiding fall-through errors**

Example

```
int marks = 85;
if (marks >= 90) System.out.println("A");
else if (marks >= 75) System.out.println("B");
else System.out.println("C");
```

LeetCode

1. **Power of Two – https://leetcode.com/problems/power-of-two/**
2. **Number of Steps – https://leetcode.com/problems/number-of-steps-to-reduce-a-number-to-zero/**

SESSION 5 – Loops

Objective:
Develop repetition logic and iteration thinking.

Detailed Content

- for vs while vs do-while
- Loop control using break & continue
- Nested loops
- Pattern generation
- Mathematical series problems

Example

```
for(int i=1;i<=5;i++){
    System.out.println(i);
}
```

LeetCode

1. Fizz Buzz – https://leetcode.com/problems/fizz-buzz/
2. Count Primes – https://leetcode.com/problems/count-primes/

SESSION 6 – Arrays (1D & 2D)

Objective:
Store and process multiple values efficiently.

Detailed Content

- Array declaration & initialization
- Indexing & traversal
- Common mistakes (out-of-bounds)
- 2D arrays as matrices

- Basic searching & aggregation

Example

```
int[] arr = {10,20,30};
for(int x : arr){
    System.out.println(x);
}
```

LeetCode

1. Maximum Subarray – https://leetcode.com/problems/maximum-subarray/
2. Best Time to Buy and Sell Stock – https://leetcode.com/problems/best-time-to-buy-and-sell-stock/

SESSION 7 – Strings & StringBuilder

Objective:
Work with text data efficiently.

Detailed Content

- String immutability
- String pool vs heap
- Common String methods
- Performance issue with concatenation
- StringBuilder for modification

Example

```
String s = "Java";
String rev = new StringBuilder(s).reverse().toString();
```

LeetCode

1. **Valid Anagram – https://leetcode.com/problems/valid-anagram/**

2. **First Unique Character – https://leetcode.com/problems/first-unique-character-in-a-string/**

SESSION 8 – Methods & Modular Programming

Objective:
Break large programs into reusable units.

Detailed Content

- **Method declaration & calling**
- **Parameters vs arguments**
- **Return values**
- **Method overloading**
- **Code reusability**

Example

```
static int add(int a, int b){
    return a + b;
}
```

LeetCode

1. **Search Insert Position – https://leetcode.com/problems/search-insert-position/**

2. **Remove Element – https://leetcode.com/problems/remove-element/**

🔹 OBJECT ORIENTED PROGRAMMING

SESSION 9 – Classes & Objects

Objective:
Model real-world entities using OOP.

Detailed Content

- **Class as blueprint**
- **Object creation using** `new`
- **Instance variables & methods**
- **Heap memory concept**

Example

```
class Student {
    String name;
    void show(){
        System.out.println(name);
    }
}
```

LeetCode

1. **Merge Two Sorted Lists –** https://leetcode.com/problems/merge-two-sorted-lists/
2. **Move Zeroes –** https://leetcode.com/problems/move-zeroes/

SESSION 10 – Constructors

Objective:
Initialize objects automatically.

Detailed Content

- **Default vs parameterized constructor**
- **Constructor overloading**
- `this` **keyword**
- **Object initialization flow**

Example

```
class Box {
    int l;
    Box(int l){
        this.l = l;
    }
}
```

LeetCode

1. **Length of Last Word – https://leetcode.com/problems/length-of-last-word/**
2. **Richest Customer Wealth – https://leetcode.com/problems/richest-customer-wealth/**

SESSION 11 – Static & Final

Objective:
Understand class-level members and immutability.

Detailed Content

- **Static variables & methods**
- **Static block execution**
- **Final variable, method, class**
- **Memory behavior**

Example

```
class Counter {
    static int count = 0;
    Counter(){ count++; }
}
```

LeetCode

1. **Missing Number – https://leetcode.com/problems/missing-number/**
2. **Contains Duplicate – https://leetcode.com/problems/contains-duplicate/**

SESSION 12 – Inheritance

Objective:
Reuse and extend existing functionality.

Detailed Content

- **Parent-child relationship**
- `extends` **keyword**
- **Method reuse**
- **super keyword**

Example

```
class A { void show(){} }
class B extends A {}
```

LeetCode

1. **Linked List Cycle – https://leetcode.com/problems/linked-list-cycle/**
2. **Middle of the Linked List – https://leetcode.com/problems/middle-of-the-linked-list/**

SESSION 13 – Polymorphism

Objective:
Enable dynamic behavior at runtime.

Detailed Content

- **Method overriding**
- **Runtime binding**
- **Upcasting**
- **Real-world use cases**

Example

```
A obj = new B();
obj.show();
```

LeetCode

1. **Remove Linked List Elements – https://leetcode.com/problems/remove-linked-list-elements/**
2. **Reverse Linked List – https://leetcode.com/problems/reverse-linked-list/**

SESSION 14 – Abstraction

Objective:
Hide implementation details.

Detailed Content

- **Abstract classes**
- **Abstract methods**
- **Design advantages**
- **Partial abstraction**

Example

```
abstract class Shape {
    abstract void area();
}
```

LeetCode

1. **Valid Parentheses – https://leetcode.com/problems/valid-parentheses/**
2. **Min Stack – https://leetcode.com/problems/min-stack/**

SESSION 15 – Interfaces

Objective:
Achieve multiple inheritance & loose coupling.

Detailed Content

- **Interface vs abstract class**

- Multiple inheritance
- Default methods
- Real-world contracts

Example

```
interface Payment {
    void pay();
}
```

LeetCode

1. Climbing Stairs – https://leetcode.com/problems/climbing-stairs/
2. Pascal's Triangle – https://leetcode.com/problems/pascals-triangle/

SESSION 16 – OOP Design Practice

Objective:
Apply all OOP principles together.

Detailed Content

- Composition vs inheritance
- Object relationships
- Mini design exercise
- Best practices

Example

```
class Car {
    Engine engine = new Engine();
}
```

LeetCode

1. **House Robber – https://leetcode.com/problems/house-robber/**
2. **Min Cost Climbing Stairs – https://leetcode.com/problems/min-cost-climbing-stairs/**

ADVANCED JAVA + SPRING

Sessions 17–36 (Elaborated)

SESSION 17 – Exception Handling Basics

Objective:
Teach students how to prevent abnormal program termination and write fault-tolerant applications.

Detailed Content

- What are runtime errors and why programs crash
- Difference between **Error vs Exception**
- Checked vs unchecked exceptions (compile-time vs runtime)
- try–catch block execution flow
- Multiple catch blocks and order of exception handling
- Importance of exception handling in production systems
- Best practices (never ignore exceptions)

Example

```
try {
    int result = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Division by zero not allowed");
}
```

LeetCode

1. Implement Queue Using Stacks
   https://leetcode.com/problems/implement-queue-using-stacks/
2. Implement Stack Using Queues
   https://leetcode.com/problems/implement-stack-using-queues/

SESSION 18 – Custom Exceptions & throw/throws

Objective:
Design domain-specific validations using custom exceptions.

Detailed Content

- Limitation of predefined exceptions
- `throw` vs `throws`
- Creating user-defined exceptions
- Propagating exceptions across methods
- Real-world validation scenarios (age, login, balance)
- Best practices for exception messages

Example

```
class AgeException extends Exception {
    AgeException(String msg) {
        super(msg);
    }
}

static void validateAge(int age) throws AgeException {
    if(age < 18)
        throw new AgeException("Not eligible");
}
```

LeetCode

1. Flood Fill
   https://leetcode.com/problems/flood-fill/

2. Number of Islands
   https://leetcode.com/problems/number-of-islands/

SESSION 19 – File Handling (Streams Combined)

Objective:
Enable persistent storage using files.

Detailed Content

- Why file handling is needed
- Byte streams vs character streams
- Reading and writing text files
- try-with-resources for safe closing
- File handling use cases (logs, reports)

Example

```
FileWriter fw = new FileWriter("data.txt");
```

```
fw.write("Java File Handling");
fw.close();
```

LeetCode

1. Running Sum of 1D Array
   https://leetcode.com/problems/running-sum-of-1d-array/
2. Find Pivot Index
   https://leetcode.com/problems/find-pivot-index/

SESSION 20 – Multithreading Basics

Objective:
Introduce parallel execution for performance improvement.

Detailed Content

- Process vs thread
- Why multithreading is required
- Thread lifecycle (conceptual)
- Creating thread using Thread class
- Difference between `start()` and `run()`

Example

```
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread Running");
    }
}
new MyThread().start();
```

LeetCode

1. Binary Tree Level Order Traversal
   https://leetcode.com/problems/binary-tree-level-order-traversal/

2. Symmetric Tree
   https://leetcode.com/problems/symmetric-tree/

SESSION 21 – Runnable & Thread Control

Objective:
Use industry-preferred Runnable approach and manage threads.

Detailed Content

- Why Runnable is better than Thread
- Thread constructor with Runnable
- Thread methods: sleep(), join()
- Thread priorities (conceptual)
- Real-world thread coordination

Example

```
Runnable r = () -> System.out.println("Runnable Thread");
Thread t = new Thread(r);
t.start();
```

LeetCode

1. Rotting Oranges
   https://leetcode.com/problems/rotting-oranges/

2. Max Area of Island
   https://leetcode.com/problems/max-area-of-island/

SESSION 22 – Synchronization

Objective:
Prevent data inconsistency in multithreaded programs.

Detailed Content

- Race condition explanation
- Critical section
- synchronized method vs synchronized block
- Object-level locking
- Real-world examples (bank balance)

Example

```
synchronized void withdraw(int amt) {
    balance -= amt;
}
```

LeetCode

1. Merge Intervals
   https://leetcode.com/problems/merge-intervals/
2. Sort Characters by Frequency
   https://leetcode.com/problems/sort-characters-by-frequency/

SESSION 23 – Collections Framework Overview

Objective:
Understand Java’s dynamic data structures.

Detailed Content

- Limitations of arrays
- Collection hierarchy
- List, Set, Map overview
- Generics for type safety
- Iteration techniques

Example

```
List<Integer> list = new ArrayList<>();
list.add(10);
```

LeetCode

1. Ransom Note
   https://leetcode.com/problems/ransom-note/
2. Intersection of Two Arrays II
   https://leetcode.com/problems/intersection-of-two-arrays-ii/

SESSION 24 – List Interface

Objective:
Store ordered and duplicate elements efficiently.

Detailed Content

- ArrayList internal working
- LinkedList use cases
- Stack (LIFO)

- Performance comparison

Example

```
List<String> names = new ArrayList<>();
names.add("Java");
```

LeetCode

1. Design Linked List
   https://leetcode.com/problems/design-linked-list/
2. Min Stack
   https://leetcode.com/problems/min-stack/

SESSION 25 – Set Interface

Objective:
Handle unique data efficiently.

Detailed Content

- HashSet hashing concept
- LinkedHashSet ordering
- TreeSet sorting
- Removing duplicates use cases

Example

```
Set<Integer> set = new HashSet<>();
set.add(10);
```

LeetCode

1. Happy Number
   https://leetcode.com/problems/happy-number/

2. Longest Consecutive Sequence
   https://leetcode.com/problems/longest-consecutive-sequence/

SESSION 26 – Map Interface

Objective:
Manage key–value based data.

Detailed Content

- HashMap working
- Collision concept (intro)
- TreeMap sorting
- Frequency problems

Example

```
Map<String, Integer> map = new HashMap<>();
map.put("Java", 1);
```

LeetCode

1. Word Pattern
   https://leetcode.com/problems/word-pattern/

2. Two Sum
   https://leetcode.com/problems/two-sum/

SESSION 27 – Comparable & Comparator

Objective:
Implement custom sorting logic.

Detailed Content

- Natural ordering
- Comparable interface
- Comparator for multiple criteria
- Lambda sorting

Example

```
Collections.sort(list, (a, b) -> a - b);
```

LeetCode

1. Kth Largest Element in an Array
   https://leetcode.com/problems/kth-largest-element-in-an-array/
2. Top K Frequent Elements
   https://leetcode.com/problems/top-k-frequent-elements/

SESSION 28 – Stream API

Objective:
Process data in a functional style.

Detailed Content

- Stream vs collection
- filter, map, reduce
- forEach

- Pipeline execution

Example

```
list.stream()
    .filter(x -> x % 2 == 0)
    .forEach(System.out::println);
```

LeetCode

1. Group Anagrams
   https://leetcode.com/problems/group-anagrams/

2. Unique Number of Occurrences
   https://leetcode.com/problems/unique-number-of-occurrences/

SESSION 29 – Lambda & Functional Interfaces

Objective:
Write concise, modern Java code.

Detailed Content

- Functional interface concept
- Lambda syntax
- Predicate, Function, Consumer
- Real use cases

Example

```
Predicate<Integer> p = x -> x > 10;
```

LeetCode

1. Sort Array by Increasing Frequency
   https://leetcode.com/problems/sort-array-by-increasing-frequency/

2. Largest Number
   https://leetcode.com/problems/largest-number/

🚀 SPRING (PRACTICAL – NO LEETCODE)

SESSION 30 – Spring Core & IoC

- What is Spring
- IoC & DI concepts
- @Component, @Autowired
- Loose coupling demo

SESSION 31 – Bean Lifecycle & Scopes

- Bean lifecycle phases
- Singleton vs prototype
- Real configuration demo

SESSION 32 – Spring AOP

- Cross-cutting concerns
- @Aspect, @Before
- Logging example

SESSION 33 – Spring Boot Introduction

- Spring Boot vs Spring
- Auto-configuration
- Project structure

SESSION 34 – REST API Development

- @RestController
- CRUD operations
- JSON request/response

SESSION 35 – REST Exception Handling

- @ControllerAdvice
- Global exception handling
- Standard API responses

SESSION 36 – Mini Project

Student Management REST API

- Controller → Service → Repository
- CRUD APIs
- Postman testing
- Runnable JAR