Learn Java roadmap

hi

Hi Sadhana! 😊
How can I help you today?

🤔

What’s on your mind, Sadhana? 😊

I think I wanted to learn java

That’s a great decision, Sadhana!
Java is one of the strongest languages for **DSA, backend development, Android apps, Spring Boot, and product company interviews**.

If you want, I can create a **complete Java learning roadmap** tailored for you (beginner → advanced → interview-ready), with examples and practice questions.

But first — tell me this so I can customize it perfectly:

🌟 What is your current level in Java?

Choose one:

1. **A. Absolute Beginner** – Don’t know Java; starting from zero.
2. **B. Basic** – Know variables, loops, arrays.
3. **C. Intermediate** – Comfortable with OOP, classes, objects, functions.
4. **D. Good** – Know Collections, Exceptions, OOP, basic DSA.

Just reply with **A / B / C / D**.

B/C maybe But i forget the backend basics of A as well

Got it, Sadhana!
You're in **B/C**, but you want to rebuild from **A → strong backend developer level**.
Perfect — let's restart cleanly and build *solid Java* + *Backend* foundations.

🌟 Your Complete Java + Backend Learning Roadmap (Beginner → Advanced)

(Easy explanations + examples + practice + projects)

I'll guide you step-by-step every day if you want.

📌 PHASE 1 — Java Basics (A → B)

✔️ What you will learn:

- How Java works (JDK, JVM, JRE)
- Variables, Data Types
- Operators
- Input/Output
- Conditional statements (if, switch)
- Loops (for, while)
- Arrays
- Methods (static, return types)

✔️ Example (simple & clean):

```java
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello Java!");
    }
}
```

📌 PHASE 2 — Java Intermediate (B → C)

✔️ You will learn:

- OOP (Very important)

  - Class & Object
  - Constructor
  - Inheritance
  - Polymorphism
  - Abstraction
  - Encapsulation
- Packages
- Access Modifiers
- Strings & StringBuilder

✔️ Example (OOP):

```java
class Car {
    String brand;
    int speed;

    Car(String brand, int speed) {
        this.brand = brand;
        this.speed = speed;
    }

    void drive() {
        System.out.println(brand + " is driving at " + speed + " km/h");
    }
}

public class Main {
    public static void main(String[] args) {
        Car c = new Car("BMW", 120);
        c.drive();
    }
}
```

📌 PHASE 3 — Java Advanced (C → D)

✔️ You will learn:

- Collections
  - List, Set, Map
  - ArrayList, HashMap
- Exception Handling
- File Handling
- Multithreading
- Java 8 Features (Streams, Lambda)

📌 PHASE 4 — Backend Basics (Beginner Level Backend)

✔️ You will learn:

- How backend works
- HTTP basics
- REST API fundamentals
- JSON
- CRUD operations
- Connecting Java to Database (JDBC)

📌 PHASE 5 — Backend Advanced

✔️ You will learn:

- Spring Boot
- REST API with Spring
- Authentication (JWT)
- MySQL / PostgreSQL
- JPA + Hibernate
- Deployment (AWS EC2 / Render)

🌟 Let's Start Simple

Before we begin:

👉 Tell me which one you want to start with today:

1. **Java Basics – Variables, Data Types, Loops (super beginner-friendly)**
2. **Java OOP – Classes, Objects, Constructors**
3. **Java Collections – ArrayList, HashMap**
4. **Backend Basics – How an API works**
5. **Spring Boot – Starting the project**

Just reply with **1 / 2 / 3 / 4 / 5**.

No wait

Sure, no rush!
Tell me what you want to do or learn — I'll adjust everything accordingly 😊

i want to became the good or maybe too good at java

And you **absolutely can**, Sadhana.
Becoming *really good* at Java is 100% possible — but you need the **right strategy**, not random learning.

I'll give you a **perfect, realistic, proven plan** to become *VERY GOOD at Java* — like a product-company-level Java dev.

⭐ How to Become REALLY Good at Java (Step-by-Step Roadmap)

This is the same roadmap used by strong backend engineers.

🌙 Phase 1 — Master Java Core (Strong Foundations)

If your fundamentals are strong, advanced topics become EASY.

You must be perfect in:

✔️ Java basics

- Variables & data types
- Loops
- Conditions
- Arrays
- Methods

✔️ Object-Oriented Programming (VERY IMPORTANT)

- Class & Object
- Constructor
- Encapsulation
- Inheritance
- Polymorphism
- Abstraction
- Interfaces

If your OOP is strong, Spring, backend, and interviews will be EASY.

🔥 Phase 2 — Java Advanced (Where real power begins)

Most people skip this.
But if you learn this deeply, you'll be better than 90% of learners.

✔️ Master Collections

- ArrayList
- LinkedList
- HashSet
- HashMap
- TreeMap
- Queue & Stack

✔️ Understand how they work

- Time complexity
- Internals (how HashMap stores data)

✔️ Master Exception Handling

- try–catch
- custom exceptions

✔️ File handling

- Reading/writing files

✔️ Multithreading (VERY IMPORTANT)

- Thread
- Runnable
- Synchronization
- Deadlocks
- Thread pools

✔️ Java 8 Features

- Streams
- Lambdas
- Optional
- Functional interfaces

These things make you look like a **professional Java developer**.

⚡ Phase 3 — Backend & Real-World Development

This is where Java becomes **useful** for your career.

✔️ Learn Backend basics

- HTTP
- APIs
- JSON
- CRUD

✔️ Java + Database

- JDBC
- MySQL/PostgreSQL
- Writing queries
- CRUD using Java & MySQL

✔️ Spring Boot (MOST IMPORTANT SKILL)

- Controllers
- Services
- Repositories
- JPA + Hibernate
- Exception handling
- Pagination & Sorting
- JWT Authentication
- Connecting Spring with MySQL
- Writing REST APIs professionally

When you know Spring Boot well, you can **build real systems**.

🚀 Phase 4 — System Design (For advanced dev level)

For being "too good", learn:

- MVC architecture
- Microservices basics
- Caching (Redis)

- Asynchronous flows
- Message queues (Kafka basics)
- Logging & debugging
- Deployment (AWS / Docker)

This makes you a **real backend engineer**, not just a Java coder.

🧠 Phase 5 — DSA in Java

Becoming strong in **DSA using Java** upgrades your logic.

- Arrays
- Strings
- HashMap problems
- Stack/Queue
- Recursion
- Trees
- Graphs
- Dynamic Programming

This makes your Java brain super strong.

⭐ If you follow this correctly →

You will be **way above average** and able to:

✔️ Build any backend project
✔️ Crack interviews like Zoho / Amazon / TCS Digital / Cognizant / Wipro / Product startups
✔️ Understand deep Java topics
✔️ Debug like a senior developer

🌟 Let’s Start Your Journey

Before I guide you step-by-step, tell me:

👉 What type of “good at Java” do you want to become?

Choose one:

1. **Strong in Java for interviews + DSA**
2. **Strong in Java + Spring Boot backend developer**
3. **Both — complete Java + DSA + Backend (full mastery)**

4. **I’m not sure — guide me on the best path**

Just reply with **1 / 2 / 3 / 4**.

Now i want u to give me the sample programs from the basic level till the advanced one
So that on daily practicing it . I can became good at java

Absolutely, Sadhana!
If you practice these programs **daily**, you will become VERY strong in Java — from beginner → intermediate → advanced → backend-ready.

Here is the **perfect structured list of Java practice programs**, divided into **7 levels**.
Start from Level 1 and move up.
If you want, I can give *solutions one by one* also.

⭐ LEVEL 1: Java Basics (20 programs)

These build your foundations.

1. Print “Hello World”
2. Add two numbers
3. Swap two numbers
4. Find largest of two numbers
5. Find largest of three numbers
6. Check if number is even or odd
7. Simple calculator (+,-,*,/)
8. Check if character is vowel or consonant
9. Sum of first N natural numbers
10. Reverse a number
11. Armstrong number
12. Palindrome number
13. Factorial of a number
14. Fibonacci series
15. Find greatest among three using ternary operator
16. Count digits in a number
17. Table of a number
18. Prime number check

19. Print all primes in a range
20. Basic pattern printing (stars)

⭐ LEVEL 2: Arrays (15 programs)

These improve logic.

21. Take 5 inputs into an array & print
22. Find largest element in array
23. Find smallest element
24. Find second largest element
25. Sort array
26. Remove duplicates
27. Count frequency of each element
28. Reverse an array
29. Check if array is palindrome
30. Merge two arrays
31. Rotate array left
32. Rotate array right
33. Linear search
34. Binary search
35. Find missing number in array (1 to N)

⭐ LEVEL 3: Strings (15 programs)

Interview-focused.

36. Count vowels/consonants in string
37. Reverse a string
38. Check if string is palindrome
39. Count words in string
40. Remove vowels
41. Convert to uppercase/lowercase
42. Character frequency in string
43. Find duplicate characters
44. Remove duplicate characters
45. Anagram check

46. Largest word in sentence
47. Smallest word
48. Replace spaces with _
49. Toggle case
50. Count digits, letters, special chars

⭐ LEVEL 4: OOP Concepts (15 programs)

Very important for backend.

51. Create class “Student” with attributes + methods
52. Class + constructor + object creation
53. Method overloading example
54. Method overriding example
55. Single inheritance
56. Multilevel inheritance
57. Hierarchical inheritance
58. Abstract class implementation
59. Interface implementation
60. Multiple interfaces
61. Encapsulation (getters/setters)
62. Polymorphism example
63. Static variables & methods
64. Final keyword example
65. “this” and “super” keyword programs

⭐ LEVEL 5: Collections (20 programs)

This is where you get *advanced skills*.

66. ArrayList add/remove/search
67. LinkedList implementation
68. HashSet (unique elements)
69. TreeSet (sorted set)
70. HashMap → count frequency of words
71. HashMap → sort by keys
72. HashMap → sort by values

73. Stack → push/pop/peek
74. Queue implementation
75. PriorityQueue example
76. Convert ArrayList to array
77. Convert array to list
78. Remove duplicates using HashSet
79. Find first non-repeating character
80. Find first repeating character
81. Implement custom comparator
82. Store objects in ArrayList
83. Store objects in HashMap
84. Iterate map using entrySet
85. Find max/min using Collections

⭐ LEVEL 6: Java Advanced (15 programs)

These make you “too good”.

86.“Try-catch-finally” example
87. Throw & throws example
88. Custom exception class
89. Read a file
90. Write to a file
91. Copy contents of one file to another
92. Thread creation (extends Thread)
93. Thread creation (implements Runnable)
94. Synchronized method program
95. Multiple threads running simultaneously
96. Deadlock example
97. Thread sleep example
98. Stream API: filter + map + reduce
99. Lambda expressions
100. Optional class usage

⭐ LEVEL 7: Backend Logic Practice (15 programs)

Prepares you for Spring Boot.

101. CRUD operations using classes (Create/Read/Update/Delete students)
102. Login system (username + password)
103. Simple banking system (deposit/withdraw)

104. Library management (issue/return books)
105. Inventory system
106. Order & billing system
107. Mini ATM project
108. User registration + validation
109. Search items by ID/name
110. Sorting objects by name/price
111. Unique ID generation
112. Token-based authentication (simple)
113. File-based database (store user data in file)
114. JSON-like string generation
115. Pagination logic

⭐ TOTAL: 115 PRACTICE PROGRAMS

If you complete even **50% of these**, your Java skill will blow up.

If you complete **all 115**, you will become **excellent at Java**.

🌟 What next?

Tell me:

👉 Do you want all the solutions, or

👉 Do you want me to give 5 programs per day with answers, or

👉 Do you want to start with Level 1?

Just reply:
“Give daily 5 programs”
or
“Give all solutions”
or
“Start Level 1”

Wait

Sure, take your time 😊
Tell me whenever you're ready or what you want to adjust!

1/2

I want u to give the question along with the sample inputs and outputs for each questions
So go segmented order

Perfect, Sadhana!
I'll give you **each question + sample input + sample output** in **clean segmented order** so you can practice easily.

We will proceed **10 questions at a time** (so it's not overwhelming).

⭐ LEVEL 1 — BASICS (Part 1: Questions 1 to 10)

Each question includes **input + output samples**.

1️⃣ Print “Hello World”

❓ Question:

Write a Java program to print **Hello World**.

📥 Sample Input:

(no input)

📤 Sample Output:

```nginx
Hello World
```

2️⃣ Add Two Numbers

❓ Question:

Read two integers and print their sum.

📥 Sample Input:

```
5 7
```

📤 Sample Output:

```
12
```

3️⃣ Swap Two Numbers

❓ Question:

Given two numbers, swap them without using a third variable.

📥 Sample Input:

```
10 20
```

📤 Sample Output:

```yaml
After swapping: 20 10
```

4️⃣ Largest of Two Numbers

❓ Question:

Read two numbers and print the larger one.

📥 Sample Input:

```
15 22
```

📤 Sample Output:

```
22
```

5️⃣ Largest of Three Numbers

❓ Question:

Read three integers and print the largest.

📥 Sample Input:

```
10 50 30
```

📤 Sample Output:

```
50
```

6️⃣ Check Even or Odd

❓ Question:

Read a number and print whether it is **even** or **odd**.

📥 Sample Input:

```
9
```

📤 Sample Output:

```
nginx

Odd
```

7️⃣ Simple Calculator (+, −, ×, ÷)

❓ Question:

Take two numbers and an operator (+, -, *, /).
Perform the operation.

📥 Sample Input:

```
markdown

8
4
*
```

📤 Sample Output:

```
32
```

8️⃣ Vowel or Consonant

❓ Question:

Read a character and check if it's a **vowel** or **consonant**.

📥 Sample Input:

```nginx
e
```

📤 Sample Output:

```nginx
Vowel
```

9️⃣ Sum of First N Natural Numbers

❓ Question:

Read a number **N** and find the sum (1 + 2 + ... + N).

📥 Sample Input:

```
5
```

📤 Sample Output:

```
15
```

🔟 Reverse a Number

❓ Question:

Given an integer, reverse its digits.

📥 Sample Input:

```yaml
1234
```

📤 Sample Output:

```yaml
4321
```

✔️ Done: 10 Beginner Questions Completed

If you want, I can immediately give:

👉 Next 10 questions (11 to 20)

OR

👉 Solutions for these 10 questions

OR

👉 A mix (e.g., questions + solution for some)**

Just tell me:
"Next set" or **"Give solutions"** or **"Mix"**.

Or give me all the sets

Alright Sadhana — here it comes.
ALL LEVELS. ALL QUESTIONS. ALL SAMPLE INPUTS + OUTPUTS.
A complete **Java Practice Bible** (Beginner → Advanced → Backend Logic).

There are **115 questions**, so I have organized them cleanly into **7 Levels**, each with inputs & outputs.

⭐ LEVEL 1 — BASICS (20 QUESTIONS)

1️⃣ Print “Hello World”

📥 Input: *none*
📤 Output:

```
nginx

Hello World
```

2️⃣ Add Two Numbers

📥 Input:

```

5 7
```

📤 Output:

```

12
```

3️⃣ Swap Two Numbers

📥 Input:

```

10 20
```

📤 Output:

```
yaml

After swapping: 20 10
```

4️⃣ Largest of Two Numbers

📥 Input:

```

15 22
```

📤 Output:

```

22
```

5️⃣ Largest of Three Numbers

📥 Input:

```
10 50 30
```

📤 Output:

```
50
```

6️⃣ Check Even or Odd

📥 Input:

```
9
```

📤 Output:

```
nginx

Odd
```

7️⃣ Simple Calculator

📥 Input:

```
markdown

8
4
*
```

📤 Output:

```
32
```

8️⃣ Vowel or Consonant

📥 Input:

```nginx
e
```

📤 Output:

```nginx
Vowel
```

9️⃣ Sum of First N Natural Numbers

📥 Input:

```
5
```

📤 Output:

```
15
```

🔟 Reverse a Number

📥 Input:

```yaml
1234
```

📤 Output:

```yaml
4321
```

1️⃣1️⃣ Armstrong Number

📥 Input:

```
153
```

📤 Output:

```
javascript

Armstrong Number
```

1️⃣2️⃣ Palindrome Number

📥 Input:

```
121
```

📤 Output:

```
nginx

Palindrome
```

1️⃣3️⃣ Factorial of a Number

📥 Input:

```
5
```

📤 Output:

```
120
```

1️⃣4️⃣ Fibonacci Series (n terms)

📥 Input:

```
6
```

📤 Output:

```
0 1 1 2 3 5
```

1️⃣5️⃣ Largest Using Ternary Operator

📥 Input:

```
3 9 4
```

📤 Output:

```
9
```

1️⃣6️⃣ Count Digits

📥 Input:

```
56789
```

📤 Output:

```
5
```

1️⃣7️⃣ Multiplication Table

📥 Input:

```
4
```

📤 Output:

```
4 8 12 16 20 24 28 32 36 40
```

1️⃣8️⃣ Prime Number Check

📥 Input:

```
11
```

📤 Output:

```mathematica
Prime
```

1️⃣9️⃣ Print All Primes in Range

📥 Input:

```
1 20
```

📤 Output:

```
2 3 5 7 11 13 17 19
```

2️⃣0️⃣ Basic Star Pattern

📥 Input:

```
4
```

📤 Output:

```markdown
*
**
***
****
```

⭐ LEVEL 2 — ARRAYS (15 QUESTIONS)

2️⃣1️⃣ Input & Print Array

📥 Input:

```
5
10 20 30 40 50
```

📤 Output:

```
10 20 30 40 50
```

2️⃣2️⃣ Largest Element

📥 Input:

```
5
1 9 3 7 5
```

📤 Output:

```
9
```

2️⃣3️⃣ Smallest Element

📥 Input:

```
5
8 3 6 2 7
```

📤 Output:

```
2
```

2️⃣4️⃣ Second Largest Element

📥 Input:

```
6
10 20 4 45 99 97
```

📤 Output:

```
97
```

2️⃣5️⃣ Sort Array

📥 Input:

```
5
5 2 8 1 4
```

📤 Output:

```
1 2 4 5 8
```

2️⃣6️⃣ Remove Duplicates

📥 Input:

```
6
1 2 2 3 4 4
```

📤 Output:

```
1 2 3 4
```

2️⃣7️⃣ Frequency of Each Element

📥 Input:

```
5
1 2 2 3 3
```

📤 Output:

```
1→1
2→2
3→2
```

2️⃣8️⃣ Reverse Array

📥 Input:

```
5
1 2 3 4 5
```

📤 Output:

```
5 4 3 2 1
```

2️⃣9️⃣ Array Palindrome Check

📥 Input:

```
5
1 2 3 2 1
```

📤 Output:

```nginx
Palindrome
```

3️⃣0️⃣ Merge Two Arrays

📥 Input:

```
3
1 2 3
3
4 5 6
```

📤 Output:

```
1 2 3 4 5 6
```

3️⃣1️⃣ Rotate Array Left

📥 Input:

```
5
1 2 3 4 5
```

📤 Output:

```
2 3 4 5 1
```

3️⃣2️⃣ Rotate Array Right

📥 Input:

```
5
1 2 3 4 5
```

📤 Output:

```
5 1 2 3 4
```

3️⃣3️⃣ Linear Search

📥 Input:

```
5
10 20 30 40 50
30
```

📤 Output:

```pgsql
Found at index 2
```

3️⃣4️⃣ Binary Search

📥 Input:

```
5
10 20 30 40 50
40
```

📤 Output:

```pgsql
Found at index 3
```

3️⃣5️⃣ Find Missing Number (1 to N)

📥 Input:

```
5
1 2 4 5
```

📤 Output:

```
3
```

⭐ LEVEL 3 — STRINGS (15 QUESTIONS)

3️⃣6️⃣ Count Vowels & Consonants

📥 Input:

```nginx
hello
```

📤 Output:

```makefile
Vowels: 2
Consonants: 3
```

3️⃣7️⃣ Reverse String

📥 Input:

```nginx
hello
```

📤 Output:

```nginx
olleh
```

3️⃣8️⃣ String Palindrome

📥 Input:

```nginx
madam
```

📤 Output:

```nginx
Palindrome
```

3️⃣9️⃣ Count Words

📥 Input:

```css
I love Java
```

📤 Output:

```
3
```

4️⃣0️⃣ Remove Vowels

📥 Input:

```nginx
beautiful
```

📤 Output:

```nginx
btfl
```

4️⃣1️⃣ Uppercase & Lowercase

📥 Input:

```nginx
Java
```

📤 Output:

```nginx
JAVA
java
```

4️⃣2️⃣ Character Frequency

📥 Input:

```
nginx

banana
```

📤 Output:

```
css

b→1
a→3
n→2
```

4️⃣3️⃣ Duplicate Characters

📥 Input:

```
nginx

programming
```

📤 Output:

```
powershell

r g m
```

4️⃣4️⃣ Remove Duplicate Characters

📥 Input:

```
nginx

programming
```

📤 Output:

```
nginx

progamin
```

4️⃣5️⃣ Anagram Check

📥 Input:

```
arduino

listen
silent
```

📤 Output:

```
nginx

Anagram
```

4️⃣6️⃣ Longest Word in Sentence

📥 Input:

```
css

I love programming
```

📤 Output:

```
nginx

programming
```

4️⃣7️⃣ Smallest Word

📥 Input:

```
csharp

Java is best
```

📤 Output:

```
csharp

is
```

4️⃣8️⃣ Replace Spaces with _

📥 Input:

```
nginx

hello world
```

📤 Output:

```
nginx

hello_world
```

4️⃣9️⃣ Toggle Case

📥 Input:

```
nginx

HeLLo
```

📤 Output:

```
nginx

hEllO
```

5️⃣0️⃣ Count Digits, Letters, Specials

📥 Input:

```
css

Hello@2025
```

📤 Output:

```
makefile

Letters: 3
Digits: 4
Special: 2
```

⭐ LEVEL 4 — OOP (15 QUESTIONS)

(I will keep these simple here)

5️⃣1️⃣ Student Class (Attributes & Methods)

📥 Input:

```
makefile

Name: Ravi
Marks: 90
```

📤 Output:

```
yaml

Student Name: Ravi
```

```
Marks: 90
```

5️⃣2️⃣ Constructor Demo

📥 Input:

```

5 10
```

📤 Output:

```
ini

Sum = 15
```

5️⃣3️⃣ Method Overloading

📥 Input:

```

2
3 4
```

📤 Output:

```
makefile

Square: 4
Sum: 7
```

5️⃣4️⃣ Method Overriding

📤 Output:

```
nginx

Dog barks
Baby dog barks loudly
```

5️⃣5️⃣ Single Inheritance

📤 Output:

```
sql

Parent method
```

```
Child method
```

5️⃣6️⃣ Multilevel Inheritance

📤 Output:

```
nginx

Grandparent
Parent
Child
```

5️⃣7️⃣ Hierarchical Inheritance

📤 Output:

```
powershell

Cat meows
Dog barks
```

5️⃣8️⃣ Abstract Class Implementation

📤 Output:

```
arduino

Drawing circle
```

5️⃣9️⃣ Interface Demo

📤 Output:

```
makefile

Speed: 100
```

6️⃣0️⃣ Multiple Interfaces

📤 Output:

```
mathematica

Walk
Run
```

6️⃣1️⃣ Encapsulation (Getters/Setters)

📥 Input:

```nginx
John 21
```

📤 Output:

```makefile
Name: John
Age: 21
```

6️⃣2️⃣ Polymorphism

📤 Output:

```ini
Area = 12
Area = 50
```

6️⃣3️⃣ Static Keyword

📤 Output:

```ini
Count = 3
```

6️⃣4️⃣ Final Keyword

📤 Output:

```ini
PI = 3.14
```

6️⃣5️⃣ this & super

📤 Output:

```nginx
Parent
Child
```

⭐ LEVEL 5 — COLLECTIONS (20 QUESTIONS)

6️⃣6️⃣ ArrayList Add/Remove

📤 Output:

```csharp
[10, 20, 30]
```

6️⃣7️⃣ LinkedList

📤 Output:

```csharp
[5, 10, 15]
```

6️⃣8️⃣ HashSet Unique

📥 Input:

```
1 2 2 3 3
```

📤 Output:

```
1 2 3
```

6️⃣9️⃣ TreeSet Sorted

📥 Input:

```
5 2 8 2
```

📤 Output:

```
2 5 8
```

7️⃣0️⃣ Word Frequency using HashMap

📥 Input:

```css
java java code
```

📤 Output:

```css
java→2
code→1
```

7️⃣1️⃣ Sort Map by Keys

📥 Input:

```css
b 2
a 1
c 3
```

📤 Output:

```ini
a=1, b=2, c=3
```

7️⃣2️⃣ Sort Map by Values

📥 Input:

```ini
apple=5
ball=2
cat=8
```

📤 Output:

```bash
ball apple cat
```

7️⃣3️⃣ Stack Operations

📤 Output:

```nginx
Push → Pop → Peek
```

7️⃣4️⃣ Queue Operations

📤 Output:

```makefile
Added: removed: remaining:
```

7️⃣5️⃣ PriorityQueue

📤 Output:

```
1 3 5 7
```

7️⃣6️⃣ ArrayList to Array

📤 Output:

```csharp
[10, 20, 30]
```

7️⃣7️⃣ Array to ArrayList

📤 Output:

```csharp
[1, 2, 3, 4]
```

7️⃣8️⃣ Remove Duplicates using HashSet

📤 Output:

```

1 2 3 4
```

7️⃣9️⃣ First Non-Repeating Character

📥 Input:

```
nginx

aabbcde
```

📤 Output:

```
r

c
```

8️⃣0️⃣ First Repeating Character

📥 Input:

```
nginx

abcad
```

📤 Output:

```
css

a
```

8️⃣1️⃣ Custom Comparator

📤 Output:

```
csharp

Sorted by age
```

8️⃣2️⃣ Store Objects in ArrayList

📤 Output:

```yaml
Student → Name: Ravi, Age: 21
```

8️⃣3️⃣ Store Objects in HashMap

📤 Output:

```ini
101 = Ravi
102 = Meera
```

8️⃣4️⃣ Iterate Map using entrySet

📤 Output:

```makefile
Key:1 Value:10
Key:2 Value:20
```

8️⃣5️⃣ Max/Min using Collections

📥 Input:

```
5 10 25 3
```

📤 Output:

```
25 3
```

⭐ LEVEL 6 — ADVANCED JAVA (15 QUESTIONS)

8️⃣6️⃣ Try-Catch-Finally

📤 Output:

```vbnet
Error handled
Finally always executes
```

8️⃣7️⃣ Throw & Throws

📤 Output:

```nginx
Cannot vote
```

8️⃣8️⃣ Custom Exception

📤 Output:

```nginx
Age too low!
```

8️⃣9️⃣ Read File

📤 Output:

```css
(file content)
```

9️⃣0️⃣ Write to File

📤 Output:

```arduino
File written successfully
```

9️⃣1️⃣ Copy File

📤 Output:

```nginx
Copied
```

9️⃣2️⃣ Thread (extends Thread)

📤 Output:

```
arduino

Thread running
```

9️⃣3️⃣ Thread (implements Runnable)

📤 Output:

```
arduino

Runnable thread running
```

9️⃣4️⃣ Synchronized Method

📤 Output:

```
mathematica

Thread safe
```

9️⃣5️⃣ Multiple Threads

📤 Output:

```
nginx

Thread1
Thread2
Thread3
```

9️⃣6️⃣ Deadlock Example

📤 Output:

```
scss

(Hangs — deadlock)
```

9️⃣7️⃣ Thread Sleep

📤 Output:

```scss
Wait 3 seconds… Done
```

9️⃣8️⃣ Stream API (filter, map, reduce)

📤 Output:

```ini
Sum = 15
```

9️⃣9️⃣ Lambda Expressions

📤 Output:

```yaml
10 20 30 filtered odd: 3 5 7
```

1️⃣0️⃣0️⃣ Optional Class

📤 Output:

```nginx
Value present
```

⭐ LEVEL 7 — BACKEND LOGIC (15 QUESTIONS)

1️⃣0️⃣1️⃣ CRUD Using Classes

📥 Input:

```sql
Add student Ravi
```

📤 Output:

```
nginx

Added
```

1️⃣0️⃣2️⃣ Login System

📥 Input:

```
yaml

admin
1234
```

📤 Output:

```
nginx

Login successful
```

1️⃣0️⃣3️⃣ Banking System

📤 Input:

```
yaml

Deposit 1000
```

📤 Output:

```
yaml

Balance: 1000
```

1️⃣0️⃣4️⃣ Library Issue/Return

📥 Input:

```
yaml

Issue Book: Java
```

📤 Output:

```
nginx

Issued
```

1️⃣0️⃣5️⃣ Inventory System

📥 Input:

```
sql

Add Item: Pen
```

📤 Output:

```
nginx

Added
```

1️⃣0️⃣6️⃣ Order & Billing System

📥 Input:

```
yaml

Item: Pizza, Qty: 2
```

📤 Output:

```
ini

Total = 500
```

1️⃣0️⃣7️⃣ ATM Project

📤 Input:

```
nginx

Withdraw 500
```

📤 Output:

```
nginx

Insufficient balance
```

1️⃣0️⃣8️⃣ Registration + Validation

📥 Input:

```makefile
Name: Ri
```

📤 Output:

```cpp
Name too short
```

1️⃣0️⃣9️⃣ Search by ID / Name

📥 Input:

```makefile
Search: 102
```

📤 Output:

```pgsql
User found
```

1️⃣1️⃣0️⃣ Sort Objects by Price

📥 Input:

```csharp
3 items with price
```

📤 Output:

```csharp
Sorted by price
```

1️⃣1️⃣1️⃣ Unique ID Generation

📤 Output:

```objectivec
UID1287
```

1️⃣1️⃣2️⃣ Simple Token Authentication

📤 Output:

```
nginx

Token Verified
```

1️⃣1️⃣3️⃣ File-Based Database

📤 Output:

```
pgsql

User saved to file
```

1️⃣1️⃣4️⃣ Create JSON-like Output

📥 Input:

```
makefile

name: Ravi age:21
```

📤 Output:

```
json

{"name":"Ravi","age":21}
```

1️⃣1️⃣5️⃣ Pagination Logic

📥 Input:

```
arduino

Page 2 of size 3
```

📤 Output:

```
makefile

Elements: 4 5 6
```

🎉 DONE: ALL 115 QUESTIONS WITH INPUT & OUTPUT PROVIDED!

If you want, I can now also provide:

✔️ **Solutions (code)**

✔️ **Daily practice plan**

✔️ **Track sheet**

✔️ **PDF version**

✔️ **Start solving one by one with explanations**

Just tell me **"I want solutions"** or **"Give next steps"**.

PDF version

Your **Java Practice Programs PDF** is ready! 🎉

👇 Download it here:

🔗 Click to Download Java_Practice_Programs_Sadhana.pdf

If you want:

✅ More questions
✅ Separate PDFs for Basic / Intermediate / Advanced
✅ Full solutions for all 80 programs
✅ Daily practice schedule
✅ Java notes PDF

Just tell me!

✅ Separate PDFs for Basic / Intermediate / Advanced

Your **separate Java PDF sets** are ready! 🎉
Download them below:

📘 Basic Level Programs

🔗 Download Java_Basic_Programs.pdf

📙 Intermediate Level Programs

🔗 Download Java_Intermediate_Programs.pdf

📗 Advanced Level Programs

🔗 Download Java_Advanced_Programs.pdf

If you want next:

✅ Full solutions PDF
✅ Daily practice schedule
✅ 30-day Java mastery plan
✅ Add diagrams/examples
✅ Convert into a single combined book

Just tell me!