ALACRITI

HIRING CHALLENGE 2026

5 MUST-PRACTICE CODING QUESTIONS

with SOLUTIONS

Java • Payments • AWS • Microservices

Prepared for Alacriti Aspirants

March 2026

INTRODUCTION

About This Guide

This PDF contains 5 essential coding questions that are most likely to appear in the Alacriti Hiring Challenge 2026. Each question is carefully selected based on Alacriti's payment domain focus and Java technology stack.

Each Question Includes

- Problem Statement
- Sample Input/Output
- Why Alacriti Asks
- Java Solution
- Time Complexity
- Screenshot Space

Challenge Details

- Dates: March 13–29, 2026
- Duration: 90 minutes
- Location: Hyderabad (Work from Office)
- Roles: Associate/Software/Senior Engineer
- Salary: ₹3.6 – 24 LPA

How to Use This Guide

1. Try solving without looking at solution first
2. Type the code yourself (don't copy-paste)
3. Run and test with different inputs
4. Take your own screenshots
5. Review time complexity analysis

QUESTION 1: LUHN ALGORITHM

> Problem Statement:
> Write a Java function to validate a credit card number using the Luhn algorithm.

Java Solution

```
public static boolean validateCreditCard(String cardNumber) {
 int sum = 0;
 boolean alternate = false;

 for (int i = cardNumber.length() - 1; i >= 0; i--) {
 int n = Integer.parseInt(cardNumber.substring(i, i + 1));

 if (alternate) {
 n *= 2;
 if (n > 9) {
 n = (n % 10) + 1;
}
}
 sum += n;
 alternate = !alternate;
}
 return (sum % 10 == 0);
}
```

Sample Output

```
Input: 4532015112830366
Output: true

Input: 4532015112830367
Output: false
```

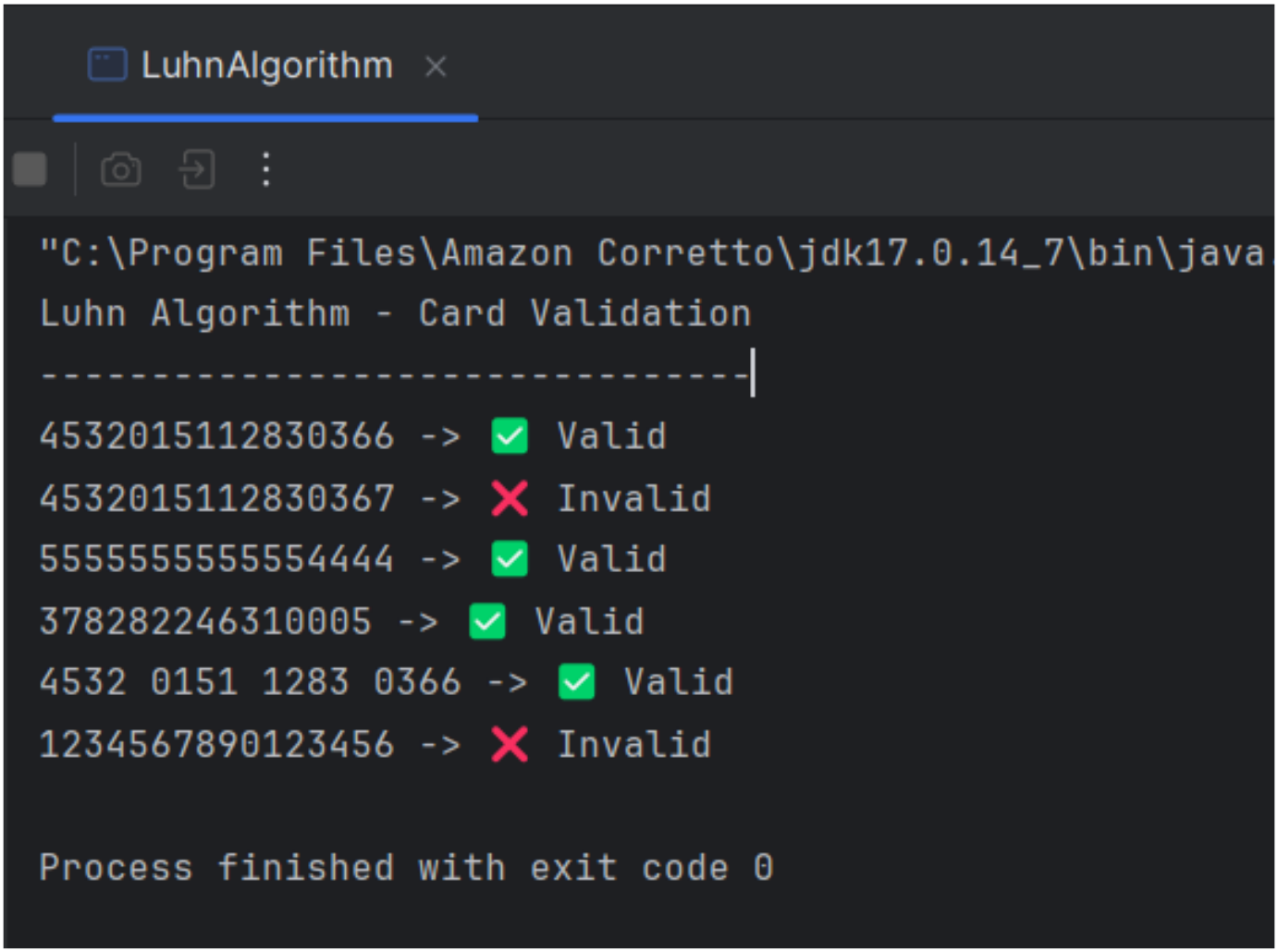
Metric	Value

Time Complexity	O(n)
Space Complexity	O(1)

Why Alacriti Asks This

Every payment system needs card validation. This question tests attention to detail and basic algorithm implementation.

Console Output



QUESTION 2: FIND DUPLICATE TRANSACTIONS

> Problem Statement:
> Given a list of transaction IDs, find all duplicate transaction IDs. Return them as a Set.

Java Solution

```
public static Set<String> findDuplicateTransactions(List<String> transactions) {
 Set<String> seen = new HashSet<>();
 Set<String> duplicates = new HashSet<>();

 for (String txn : transactions) {
 if (!seen.add(txn)) {
 duplicates.add(txn);
}
}
 return duplicates;
}
```

Sample Output

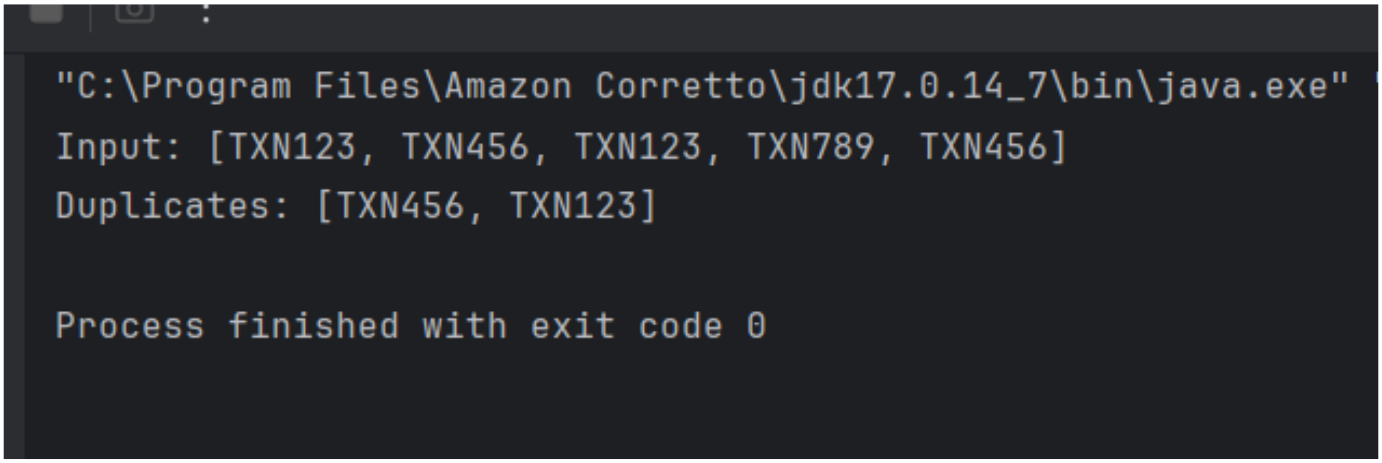
```
Input: [TXN123, TXN456, TXN123, TXN789, TXN456]
Output: [TXN123, TXN456]
```

Metric	Value
Time Complexity	O(n)
Space Complexity	O(n)

Why Alacriti Asks This

Payment systems must handle duplicate transactions. This tests knowledge of Collections and idempotency concepts.

Console Output



QUESTION 3: PRODUCER-CONSUMER

> Problem Statement:
> Implement Producer-Consumer problem using BlockingQueue in Java.

Java Solution

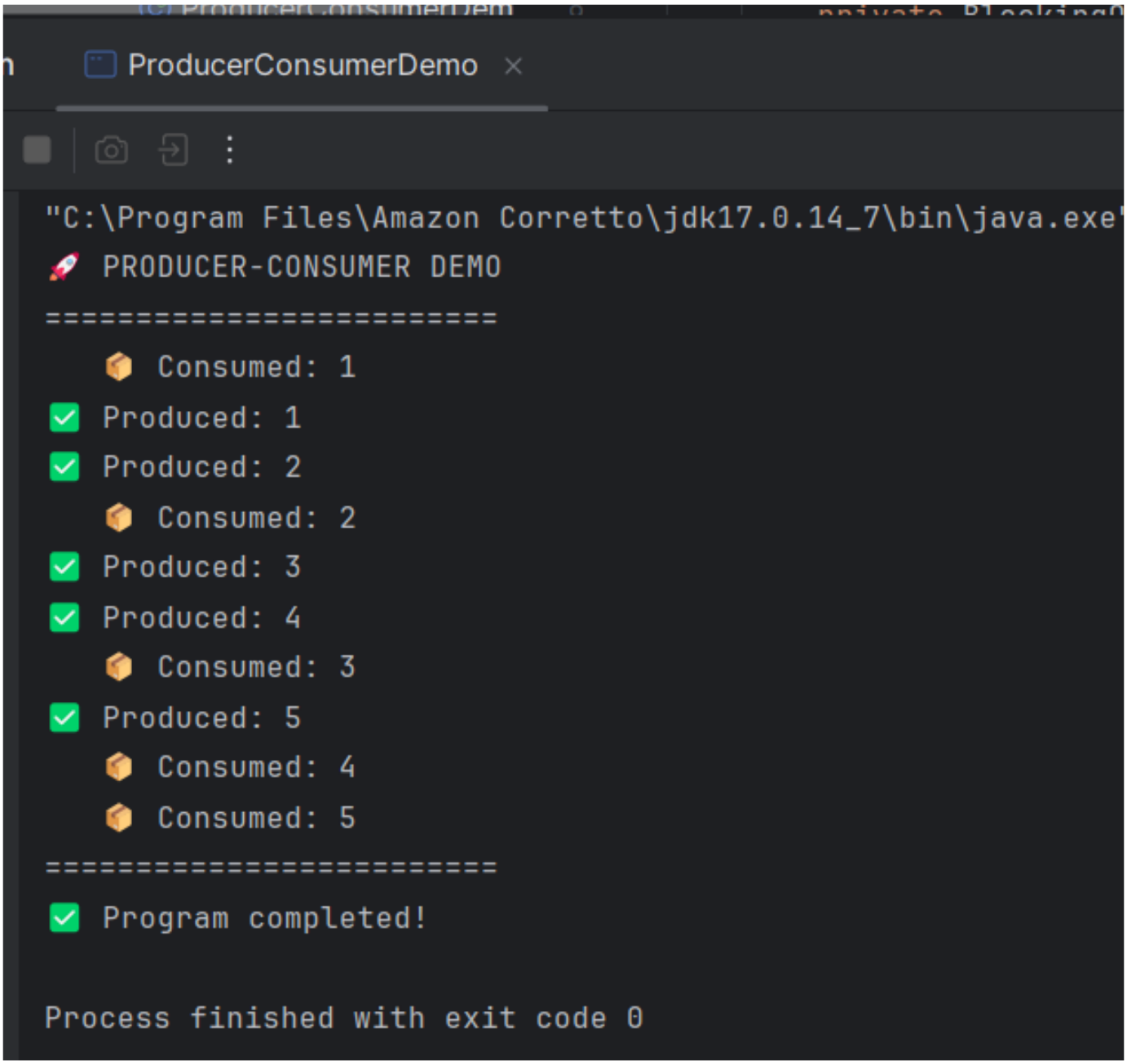
```
class Producer implements Runnable {
 BlockingQueue<Integer> queue;
 Producer(BlockingQueue<Integer> q) { queue = q; }

 public void run() {
 for (int i = 1; i <= 5; i++) {
 queue.put(i);
 System.out.println("Produced: " + i);
}
}
}

class Consumer implements Runnable {
 BlockingQueue<Integer> queue;
 Consumer(BlockingQueue<Integer> q) { queue = q; }

 public void run() {
 while (true) {
 int val = queue.take();
 System.out.println("Consumed: " + val);
 if (val == 5) break;
}
}
}
```

Console Output


ProducerConsumerDemo
"C:\Program Files\Amazon Corretto\jdk17.0.14_7\bin\java.exe"
🚀 PRODUCER-CONSUMER DEMO
==========
📦 Consumed: 1
✅ Produced: 1
✅ Produced: 2
📦 Consumed: 2
✅ Produced: 3
✅ Produced: 4
📦 Consumed: 3
✅ Produced: 5
📦 Consumed: 4
📦 Consumed: 5
==========
✅ Program completed!
Process finished with exit code 0

QUESTION 4: SORT TRANSACTIONS BY DATE

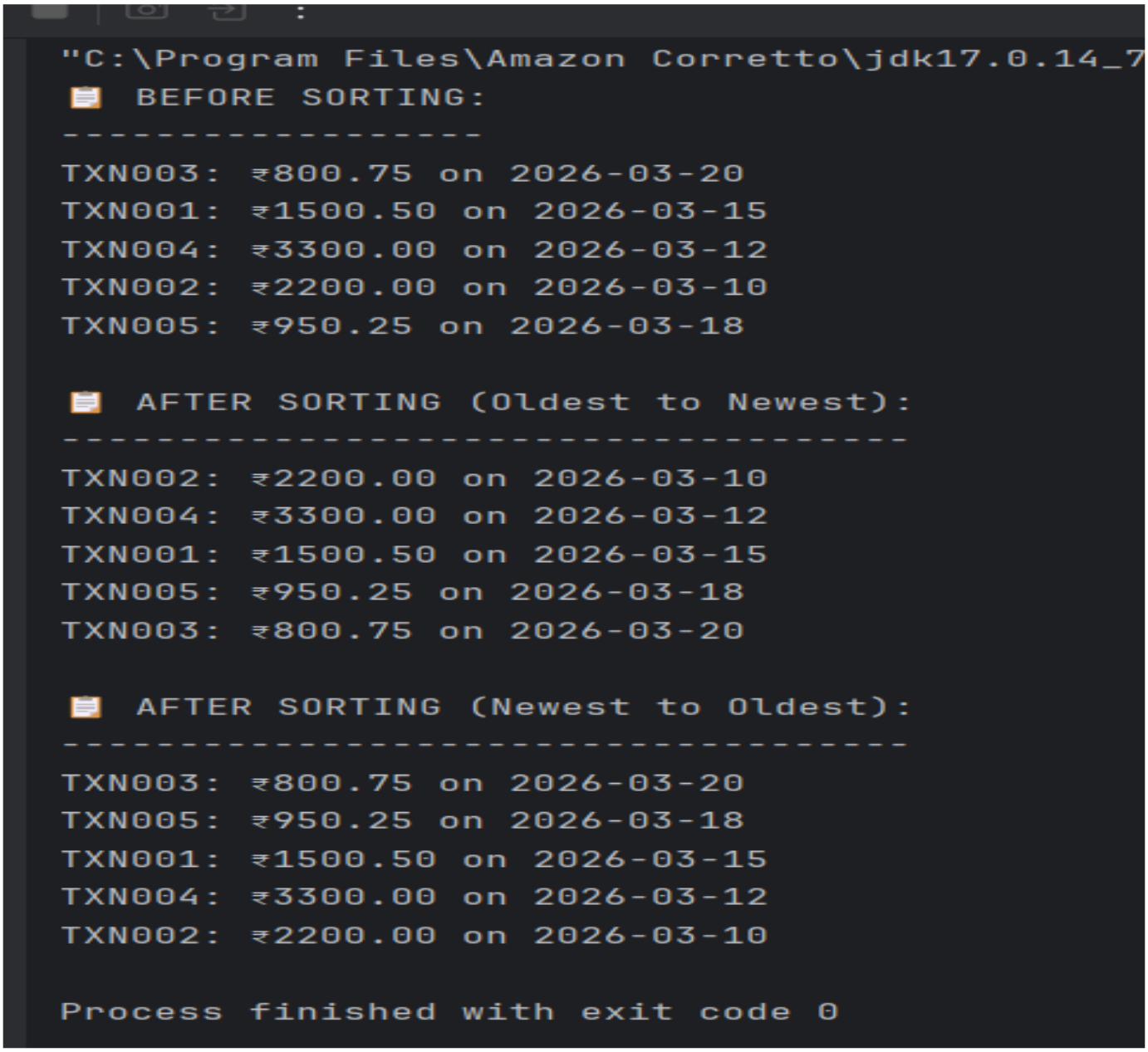
> Problem Statement:
> Sort a list of transactions by date using Comparator.

Java Solution

```
transactions.sort(Comparator.comparing(Transaction::getDate));

// Or reverse order
transactions.sort(Comparator.comparing(Transaction::getDate).reversed());
```

Console Output



QUESTION 5: SINGLETON PAYMENT GATEWAY

> Problem Statement:
> Implement thread-safe Singleton pattern for PaymentGateway.

Java Solution

```
public class PaymentGateway {
 private static volatile PaymentGateway instance;

 private PaymentGateway() { }

 public static PaymentGateway getInstance() {
 if (instance == null) {
 synchronized (PaymentGateway.class) {
 if (instance == null) {
 instance = new PaymentGateway();
}
}
}
 return instance;
}
}
```

Console Output

```
📌 TEST 1: Multiple calls in single thread
------------------------------------------
🏦 PaymentGateway initialized (once only)
   💰 Processing payment: TXN001 for ₹1500.0
   🔑 Instance hashcode: 1452126962
   💰 Processing payment: TXN002 for ₹2200.5
   🔑 Instance hashcode: 1452126962

   Same instance? true

📌 TEST 2: Multiple threads accessing simultaneously
------------------------------------------------
   💰 Processing payment: TXN001 for ₹1000.0
   💰 Processing payment: TXN004 for ₹4000.0
   🔑 Instance hashcode: 1452126962
   🔑 Instance hashcode: 1452126962
   💰 Processing payment: TXN003 for ₹3000.0
   🔑 Instance hashcode: 1452126962
   💰 Processing payment: TXN005 for ₹5000.0
   🔑 Instance hashcode: 1452126962
   💰 Processing payment: TXN002 for ₹2000.0
   🔑 Instance hashcode: 1452126962

==================================
✅ All tests completed

Process finished with exit code 0
```

BONUS TIPS

Before the Challenge

- Type all programs yourself - muscle memory helps
- Practice explaining your code out loud
- Review PCI-DSS compliance basics
- Understand transaction lifecycle
- Get comfortable with Java 21 features

During the Challenge

- Read questions twice before starting
- Start with what you know
- Add comments to show thought process
- Handle edge cases
- Manage time - 90 minutes goes fast

Alacriti Hiring Challenge 2026

March 13–29, 2026
90 Minutes | 10 Openings
Hyderabad (Work from Office)