Table of contents

Core Java Questions

1. **Palindrome Check**
   - Write a program to check if a given string is a palindrome.
   - Optimize it for case insensitivity and ignore non-alphanumeric characters.
2. **Anagram Check**
   - Write a program to check if two strings are anagrams of each other.
3. **Find the First Non-Repeated Character in a String**
   - Given a string, find the first non-repeated character.
   - Example: Input: "stress" → Output: 't'.
4. **Reverse Words in a Sentence**
   - Input: `"Hello World"`
   - Output: `"World Hello"`
5. **Find Duplicates in an Array**
   - Given an integer array, find and print all duplicates.
6. **Remove Duplicates from a Sorted Array**
   - Implement a method to remove duplicates from a sorted array without using extra space.

Data Structures and Algorithms

1. **Two Sum Problem**
   - Given an array of integers, find two numbers such that they add up to a specific target. Return the indices of the two numbers.
2. **Longest Substring Without Repeating Characters**
   - Given a string, find the length of the longest substring without repeating characters.
3. **Merge Two Sorted Arrays**
   - Merge two sorted integer arrays into a single sorted array without using additional space.
4. **Find the Missing Number in an Array**
   - Given an array containing n distinct numbers taken from 0, 1, 2, ..., n, find the missing number.
5. **Find the Intersection of Two Arrays**
   - Write a method to find common elements between two arrays.

Collections Framework

1. **How Does HashMap Work Internally?**
   - Explain the internal working of `HashMap`, focusing on hashing, bucket creation, and handling collisions.
2. **Implement LRU Cache Using LinkedHashMap**
   - Design an LRU (Least Recently Used) cache system.
3. **Difference Between HashSet and TreeSet**

  - Explain the difference in terms of implementation, time complexity, and use cases.

Multi-threading and Concurrency

1. **Producer-Consumer Problem**
   - Implement a producer-consumer scenario using threads and synchronization.
2. **Deadlock Example**
   - Write a simple code example of a deadlock scenario and explain how to avoid it.
3. **Difference Between Synchronized and Lock**
   - Explain `synchronized` keyword vs. `Lock` interface in Java. Provide a code example demonstrating both.

Java 8+ Features

1. **Stream API Examples**
   - Find the maximum value in a list using `Stream API`.
   - Filter out even numbers and collect them in a list.
2. **Lambda Expressions and Functional Interfaces**
   - Provide examples of using lambda expressions with common functional interfaces like `Predicate`, `Function`, and `Consumer`.
3. **Optional Class**
   - Write a method using `Optional` to avoid `NullPointerException`.

Spring Boot and Hibernate

1. **Write a Simple REST API Using Spring Boot**
   - Create a REST API to perform CRUD operations on a `Product` entity.
2. **Explain the Difference Between `@Controller` and `@RestController`**
   - What is the main difference, and in what scenarios would you use each?
3. **Spring Boot Application Properties Configuration**
   - How do you externalize configuration in Spring Boot using `application.properties` or `application.yml`?

Design Patterns

1. **Singleton Pattern**
   - Write a thread-safe singleton class in Java.
2. **Factory Design Pattern**
   - Explain the Factory design pattern with a real-world example and provide a code implementation.

Java Collections

Questions:

- **Difference Between List, Set, and Map:**
  - What are the main differences between `List`, `Set`, and `Map` in Java?
- **HashMap vs. ConcurrentHashMap:**
  - Explain the differences and how `ConcurrentHashMap` prevents thread interference.
- **Sorting with Comparator and Comparable:**
  - How would you sort a list of custom objects using `Comparable` and `Comparator`?
- **Internal Working of ArrayList and LinkedList:**
  - Explain the internal working, including capacity handling in `ArrayList` and node management in `LinkedList`.
- **Fail-Fast vs. Fail-Safe Iterators:**
  - Explain the difference between fail-fast and fail-safe iterators with examples.

Coding Questions:

- **Implement a Custom ArrayList:**
  - Implement a simple version of `ArrayList` using an array, supporting basic operations like `add()`, `get()`, and `remove()`.
- **Find the Frequency of Words in a String:**
  - Given a string, use a `HashMap` to count the frequency of each word.
- **Reverse a LinkedList:**
  - Implement a method to reverse a singly linked list.

Exception Handling

Questions:

- **Checked vs. Unchecked Exceptions:**
  - What is the difference between checked and unchecked exceptions? Give examples.
- **Custom Exception:**
  - How do you create a custom exception in Java? When would you use it?
- **Try-With-Resources:**
  - What is the `try-with-resources` statement in Java 7+? Explain with an example.
- **Difference Between `throw` and `throws`:**
  - Explain the difference between `throw` and `throws` with examples.

Coding Questions:

- **Divide by Zero Handling:**
    - Write a method that takes two integers and divides them. Handle the `ArithmeticException` if the denominator is zero.
- **Custom Exception Implementation:**
    - Create a custom exception called `InvalidInputException` that is thrown when a negative integer is passed to a method.

Java 8 Features

Questions:

- **Stream API:**
    - Explain the use of `filter()`, `map()`, and `reduce()` methods in `Stream API`.
- **Lambda Expressions:**
    - What are lambda expressions, and how do they simplify code? Provide examples.
- **Functional Interfaces:**
    - What is a functional interface? Give examples of commonly used functional interfaces.
- **Difference Between `map()` and `flatMap()`:**
    - What is the difference between `map()` and `flatMap()` in streams?
- **Optional Class:**
    - Explain the `Optional` class and how it helps in avoiding `NullPointerException`.

Coding Questions:

- **Filter and Collect Using Streams:**
    - Given a list of integers, filter out even numbers and collect them into a list.
- **Find the Longest Word in a Sentence:**
    - Use `Stream API` to find the longest word in a sentence.
- **Sorting Using Lambda:**
    - Sort a list of custom objects using lambda expressions based on a specific field.

Multi-Threading and Concurrency

Questions:

- **Thread Lifecycle:**
    - Describe the lifecycle of a thread in Java.
- **Synchronized Keyword:**
    - What is the `synchronized` keyword in Java? Provide examples.

- **Volatile Keyword:**
    - What is the `volatile` keyword, and when should it be used?
- **Callable vs. Runnable:**
    - What is the difference between `Callable` and `Runnable` interfaces?
- **Thread Pool Executor:**
    - Explain the usage of `ThreadPoolExecutor` and its advantages.

Coding Questions:

- **Print Even and Odd Numbers Using Two Threads:**
    - Write a program where two threads print even and odd numbers alternately.
- **Implement a Thread-Safe Singleton:**
    - Implement a singleton class using the double-checked locking principle.
- **Producer-Consumer Problem Using BlockingQueue:**
    - Implement the producer-consumer problem using `BlockingQueue`.

Object-Oriented Programming (OOP) Concepts

Questions:

- **SOLID Principles:**
    - Explain the SOLID principles with examples.
- **Inheritance vs. Composition:**
    - What is the difference between inheritance and composition in Java?
- **Method Overloading vs. Overriding:**
    - Explain method overloading and method overriding with examples.
- **Abstract Class vs. Interface:**
    - What are the differences between an abstract class and an interface?
- **Polymorphism:**
    - Explain polymorphism in Java with examples.

Coding Questions:

- **Override `equals()` and `hashCode()` Methods:**
    - Implement `equals()` and `hashCode()` for a custom `Employee` class.
- **Design a Class Hierarchy:**
    - Design a class hierarchy for a simple vehicle system with `Car`, `Bike`, and `Truck` as subclasses.
- **Factory Design Pattern Implementation:**
    - Implement a simple factory design pattern for creating objects of different types (e.g., `Circle`, `Square`, `Rectangle`).

String Manipulation

Questions:

- **Immutable String:**
  - Why is the `String` class immutable in Java? Explain the benefits.
- **String vs. StringBuilder vs. StringBuffer:**
  - What are the differences between `String`, `StringBuilder`, and `StringBuffer`?
- **Interning of Strings:**
  - What is string interning in Java?

Coding Questions:

- **Check if a String Contains Only Digits:**
  - Write a method to check if a given string contains only numeric characters.
- **Permutations of a String:**
  - Write a recursive method to print all permutations of a given string.
- **Count Vowels and Consonants in a String:**
  - Implement a method to count the number of vowels and consonants in a given string.

Miscellaneous

Questions:

- **Serialization in Java:**
  - What is serialization in Java? Explain with an example.
- **Marker Interface:**
  - What is a marker interface in Java? Provide examples.
- **ClassLoaders in Java:**
  - Explain the concept of class loaders and the types of class loaders in Java.
- **JVM Architecture:**
  - Explain the architecture of the Java Virtual Machine (JVM).

Coding Questions:

- **Implement a Simple Cache Using HashMap:**
  - Create a basic in-memory cache using `HashMap` with a maximum size limit.
- **Read a File Line by Line:**
  - Write a program to read a text file line by line and print its content.
- **Fibonacci Sequence Using Recursion and Iteration:**
  - Implement a method to print the Fibonacci sequence using both recursion and iteration.