A. **Abstraction and Interfaces:**

1. What is an interface in Java? How does it differ from an abstract class?
2. Explain the concept of abstraction in Java and how it's achieved.
3. Can an interface extend another interface? If yes, explain with an example.

B. **Collections Framework:** 4. What is the Java Collections Framework? List some core interfaces of the Collections Framework.

1. What is the difference between ArrayList and LinkedList in Java?
2. Explain the difference between HashSet and TreeSet.
3. What is the Map interface? How is it implemented in Java?

C. **Exception Handling:** 8. What is an exception in Java? Explain the hierarchy of Java exception classes.

1. What is the difference between checked and unchecked exceptions in Java?
2. How do you handle exceptions in Java? Explain try, catch, and finally blocks.

D. **Multithreading:** 11. What is multithreading in Java? How do you create and start a thread?

1. Explain the difference between the **Runnable** and **Thread** classes.
2. How do you achieve synchronization in Java? What are synchronized blocks and methods?

E. **Inheritance and Polymorphism:** 14. What is inheritance in Java? How does it facilitate code reuse?

Explain the concept of method overriding and method overloading in Java.

1. How is polymorphism achieved in Java? Provide an example.

F. **Java Memory Management:** 17. Describe the difference between stack and heap memory in Java.

1. What is garbage collection in Java? How does it work?
2. How can you explicitly request garbage collection in Java?

G. **String Handling:** 20. Explain the difference between **String**, **StringBuilder**, and **StringBuffer**.

1. How do you reverse a string in Java?
2. What is the importance of the **String** class being immutable in Java?

H. **File Handling:** 23. How do you read from and write to a file in Java?

1. Explain the difference between **FileInputStream** and **FileOutputStream**.

I. **Design Patterns:** 25. Explain the Singleton design pattern and its implementation in Java.

1. What is the Factory design pattern? How is it useful in Java?

J. **Serialization and Deserialization:** 27. What is serialization and deserialization in Java?

1. Explain the **Serializable** interface and its purpose.
2. How do you handle versioning during serialization and deserialization?

K. **Generics:** 30. What are generics in Java? Why are they useful?

1. Explain the difference between generic classes and generic methods.

L. **Java I/O:** 32. Describe the difference between input and output streams in Java.

1. How do you read user input in Java?

M. **Annotations:** 34. What are annotations in Java? Provide examples of built-in annotations.

N. **Lambda Expressions:** 35. What are lambda expressions in Java? How are they used?

O. **Enums:** 36. What are enums in Java? How do you use them?

P. **JDBC (Java Database Connectivity):** 37. Explain JDBC and its components.

1. How do you connect to a database using JDBC in Java?

Q. **Networking:** 39. What is networking in Java? Explain sockets and their types.

R. **Unit Testing:** 40. What is JUnit in Java? How do you write and run JUnit tests?

S. **Annotations:** 41. What are custom annotations in Java, and how are they used?

T. **Java 8 Features:** 42. Discuss some of the new features introduced in Java 8 (e.g., streams, functional interfaces, default methods).

U. **Reflection:** 43. What is reflection in Java? How is it used?

V. **Class Loaders:** 44. Explain the role of class loaders in Java.

W. **JVM (Java Virtual Machine):** 45. Describe the architecture of the JVM.

X. **Exception Handling:** 46. What is the difference between throw and throws in Java?

Y. **Regular Expressions:** 47. How do you use regular expressions in Java?

Z. **Core Concepts:** 48. Explain the main principles of OOP (Object-Oriented Programming) and how they are applied in Java.