# Programming Problems in Java language

Core Java Concepts:

1. Explain the difference between the == operator and .equals() method for comparing objects.
2. Discuss the purpose of the transient keyword in Java.
3. Implement a custom class loader in Java.
4. Explain the principles of Java Reflection and provide an example.
5. Discuss the concept of the this reference in Java.

Advanced Java Features:

1. Implement a simple annotation processor in Java.
2. Explain the use of the enum keyword in Java and provide an example.
3. Write a program to demonstrate the use of lambda expressions in Java.
4. Discuss the principles of functional interfaces and provide an example.
5. Implement a custom functional interface and use it in a program.

Collections and Generics:

1. Discuss the differences between ArrayList and LinkedList.
2. Implement a generic class for a binary search tree.
3. Explain the wildcard in generics and when to use ? extends and ? super.
4. Write a program to demonstrate the use of the Java Stream API.
5. Discuss the principles of the Observer design pattern and implement it.

Concurrency:

1. Write a program demonstrating the use of Java threads.

Discuss the principles of the Java Memory Model.

1. Implement a simple producer-consumer problem using java.util.concurrent.
2. Explain the concept of thread pools and implement one in Java.
3. Write a program using the synchronized keyword to ensure thread safety.

Exception Handling:

1. Discuss the differences between checked and unchecked exceptions.
2. Write a program to demonstrate the use of Java's try-with-resources statement.
3. Implement a custom exception and handle it in a program.
4. Discuss the principles of exception chaining in Java.

Networking:

1. Write a simple client-server program using Java sockets.
2. Implement a program to download a file using java.net.URL.
3. Discuss the principles of Java's NIO (New I/O) package.
4. Write a program to implement a basic HTTP server in Java.

Design Patterns:

1. Explain the principles of the Singleton design pattern and implement it.
2. Implement a factory pattern in Java.
3. Discuss the principles of the Observer design pattern and implement it.
4. Write a program to demonstrate the use of the Decorator pattern.
5. Implement a simple builder pattern in Java.

Database Connectivity:

1. Write a program to connect to a MySQL database in Java.
2. Discuss the use of JDBC (Java Database Connectivity) in Java.
3. Implement a program to perform CRUD (Create, Read, Update, Delete) operations on a database.

Advanced Data Structures:

1. Implement a trie data structure and perform insert and search operations.
2. Discuss the implementation and advantages of a skip list.
3. Write a program to detect a cycle in a directed graph.
4. Implement a priority queue using a binary heap.
5. Explain the concept of a self-adjusting list and its applications.

File Handling:

1. Write a program to copy the contents of one file to another.
2. Implement a file compression and decompression algorithm.
3. Discuss the use of java.nio.file for file operations in Java.
4. Write a program to serialize and deserialize a binary tree.

Algorithms:

1. Implement Dijkstra's algorithm for finding the shortest path in a graph.
2. Discuss the advantages and disadvantages of various graph traversal algorithms.
3. Write a program to find the longest common subsequence of two strings.
4. Implement the A\* algorithm for pathfinding in a grid.
5. Discuss the concept of topological sorting and its applications.

JavaFX:

1. Implement a simple JavaFX application with a graphical user interface.
2. Write a program to create a basic animation using JavaFX.
3. Discuss the principles of MVC (Model-View-Controller) architecture in JavaFX.

Web Development in Java:

1. Discuss the principles of servlets in Java.
2. Write a program to create a basic Java web application using servlets.
3. Implement a Java program to connect to a RESTful API.

Spring Framework:

1. Discuss the principles of Dependency Injection in the Spring framework.
2. Write a simple Spring Boot application.
3. Implement a basic Spring MVC application with annotations.

Hibernate:

1. Explain the principles of Hibernate ORM (Object-Relational Mapping).
2. Write a program to perform CRUD operations using Hibernate.
3. Discuss the differences between Hibernate and JDBC.

Java Security:

1. Discuss the principles of Java Security.
2. Write a program to demonstrate the use of Java Cryptography Architecture (JCA).

Java Native Interface (JNI):

1. Explain the purpose of JNI in Java.
2. Write a program to call a C function from Java using JNI.

Memory Management:

1. Discuss the garbage collection process in Java.
2. Write a program to create a memory leak in Java and suggest a solution.

Java Microservices:

1. Discuss the principles of microservices architecture.
2. Implement a simple microservices-based Java application.

Java Performance Optimization:

1. Discuss the principles of Java performance optimization.
2. Write a program to profile and optimize Java code.

Machine Learning in Java:

1. Implement a basic machine learning algorithm in Java.
2. Discuss the challenges of implementing machine learning in Java.

Natural Language Processing (NLP):

1. Implement a simple program using Java for natural language processing.
2. Discuss the principles of NLP and its applications in Java.

Android Development:

1. Write a simple Android application using Java.
2. Discuss the principles of Android development with Java.

Internet of Things (IoT):

1. Implement a simple IoT application using Java.
2. Discuss the challenges and security considerations in IoT programming.

Robotics:

1. Write a program to control a robot using Java.
2. Discuss the principles of robot programming and control in Java.

Advanced Java I/O:

1. Discuss the principles of Java NIO (New I/O).
2. Write a program to implement a non-blocking server using Java NIO.

Advanced String Manipulation:

1. Implement a program to check if two strings are anagrams.
2. Write a function to reverse words in a given sentence without using library functions.
3. Discuss the concept of string matching algorithms and implement one.

Advanced Reflection:

1. Write a program to dynamically instantiate a class using Java Reflection.
2. Discuss the principles of reflection and its applications in Java.

Advanced Annotations:

1. Implement a custom annotation and use it in a Java program.
2. Discuss the use of annotations in Java frameworks.
3. JavaFX Charts and Graphs:
4. Implement a JavaFX application with charts and graphs.
5. Discuss the principles of data visualization in JavaFX.

Advanced Garbage Collection:

1. Discuss the principles of advanced garbage collection algorithms in Java.
2. Write a program to analyze and optimize garbage collection in Java.

Code Analysis and Metrics:

1. Discuss the principles of code analysis and metrics in Java.
2. Write a program to analyze code complexity and suggest improvements.

Advanced Testing in Java:

1. Discuss the principles of advanced testing frameworks in Java.
2. Write a program to demonstrate unit testing using JUnit.
3. Discuss the principles of test-driven development (TDD) in Java.