Basics & Syntax

1. Write a Java program to print "Hello, World!".
2. Explain the difference between `==` and `.equals()` in Java. Show with code examples and outputs.
3. What is the use of the `main` method in Java?
4. Write a Java program to add two numbers entered by the user.
5. What is the difference between `int`, `Integer`, and `String`?

Control Structures

6. Write a program to check if a number is even or odd.
7. Write a program to find the largest among three numbers.
8. Explain the difference between `while`, `for`, and `do-while` loops in Java.
9. Write a Java program to print the multiplication table of any number.

Intermediate-Level Questions

OOP Concepts

10. Explain the four pillars of OOP in Java.
11. Create a class `Student` with properties `name`, `matricNo`, and `score`, and add methods to display the student's info.
12. What is method overloading? Give a code example.
13. What is inheritance? Create a base class `Person` and a subclass `Teacher`.

General Practices

14. What does it mean to write “clean code”? Give 3 practices that make code clean and maintainable.
15. Why should you avoid writing very long methods in Java programs?

16. What naming conventions should be followed in Java for: Classes, Variables, Methods. Give examples with screenshot of code and output.
17. What is the importance of breaking your Java program into methods?
18. Explain the concept of DRY (Don't Repeat Yourself) with a Java code example.
19. What are the benefits of using classes and objects instead of writing all logic in the main method?

Testing & Debugging

20. Why is testing important during program development?
21. What is the difference between syntax error, runtime error, and logic error?
22. How would you test a method that calculates the average of five numbers?

Documentation & Comments

23. Why should Java developers write comments in their code?
24. What are JavaDoc comments and how are they different from regular comments?
25. Write a sample Java method with JavaDoc comments.

Versioning & Collaboration

26. What is version control and why is it important in team projects?
27. How would you explain the concept of "code refactoring" to a junior developer?
28. What tools can Java developers use to collaborate on large projects? Attach screenshots of 3 examples.

Good Practices Summary

29. Mention 5 best practices you follow when developing a Java program.
30. What is code readability, and why is it more important than "smart" code?

Advanced-Level Questions

Mini Projects / Logic Building

31. Build a command-line application that keeps track of student grades and allows adding, updating, and viewing records.

32. Write a program that simulates a basic ATM system (check balance, deposit, withdraw).

Note: For all code/ program examples, create a github repository and attach the whole code and outputs.

Attach proof of answer to all questions by attaching a screenshot of code and their output.