**Section 1: Java Basics (Q1–Q10)**

1. **What is Java?**
Java is an object-oriented, platform-independent, high-level programming language that follows the "Write Once, Run Anywhere" principle.
2. **What is JVM?**
JVM (Java Virtual Machine) is the runtime environment that executes Java bytecode.
3. **What is the difference between JDK and JRE?**
	* **JDK**: JRE + development tools (compilers, debuggers).
	* **JRE**: JVM + libraries for running Java programs.
4. **What is bytecode?**
Bytecode is platform-independent code generated by the Java compiler, executed by the JVM.
5. **Why is Java platform-independent?**
Because compiled bytecode runs on any system with a JVM.
6. **What is the signature of the main method in Java?**
public static void main(String[] args)
7. **Difference between a variable and a constant?**
A variable’s value can change; a constant’s value (declared with final) cannot.
8. **What is a wrapper class?**
A class that wraps primitive types into objects (e.g., Integer, Double).
9. **Difference between == and .equals()?**
== compares references; .equals() compares values.
10. **Difference between final, finally, and finalize()?**
	* **final**: Used for constants, prevents method overriding or inheritance.
	* **finally**: Block that executes after try-catch.
	* **finalize()**: Called before garbage collection.

**Section 2: OOP Concepts (Q11–Q20)**

1. **What are the four pillars of OOP?**
Encapsulation, Inheritance, Polymorphism, Abstraction.
2. **What is encapsulation?**
Wrapping data and methods in a single unit and restricting direct access.
3. **What is abstraction?**
Hiding implementation details and showing only essential features.
4. **Types of inheritance in Java?**
Single, Multilevel, Hierarchical (multiple inheritance not directly supported).
5. **What is method overloading?**
Same method name with different parameter lists.
6. **What is method overriding?**
Redefining a parent class method in a child class.
7. **Difference between interface and abstract class?**
Interface: All methods abstract by default.
Abstract class: Can have both abstract and non-abstract methods.
8. **What is super keyword?**
Refers to the parent class and can be used to call parent methods/constructors.
9. **What is constructor overloading?**
Multiple constructors in the same class with different parameter lists.
10. **What is this keyword?**
Refers to the current object.

**Section 3: Strings & Collections (Q21–Q30)**

1. **Why are strings immutable in Java?**
For security, caching, and thread safety.
2. **Difference between StringBuilder and StringBuffer?**
StringBuilder is faster but not synchronized; StringBuffer is synchronized.
3. **What is the String pool?**
A special memory area where string literals are stored.
4. **Difference between ArrayList and LinkedList?**
ArrayList is faster for random access; LinkedList is faster for insert/delete.
5. **Difference between HashMap and Hashtable?**
HashMap is not synchronized; Hashtable is synchronized.
6. **Why doesn’t HashSet allow duplicates?**
It uses hashing which ensures unique keys.
7. **Difference between Comparable and Comparator?**
Comparable defines natural order; Comparator defines custom order.
8. **Difference between fail-fast and fail-safe iterators?**
Fail-fast throws an exception if modified during iteration; fail-safe iterates over a copy.
9. **What is a PriorityQueue?**
A queue that orders elements based on natural or custom order.
10. **Difference between TreeMap and HashMap?**
TreeMap is sorted; HashMap is unordered.

**Section 4: Exception Handling (Q31–Q40)**

1. **Difference between checked and unchecked exceptions?**
Checked: Checked at compile-time.
Unchecked: Checked at runtime.
2. **Flow of try-catch-finally?**
try → catch (if exception) → finally (always executes).
3. **Difference between throw and throws?**
throw: Used to throw a single exception.
throws: Declares possible exceptions.
4. **How to create a custom exception?**
Extend Exception or RuntimeException.
5. **How do multiple catch blocks work?**
The first matching exception type is executed.
6. **Does finally block always execute?**
Yes, except when JVM exits using System.exit().
7. **What is try-with-resources?**
Automatically closes resources declared inside try.
8. **Difference between Error and Exception?**
Error: Serious problem not handled by programs.
Exception: Can be handled.
9. **Example of RuntimeException?**
NullPointerException, ArrayIndexOutOfBoundsException.
10. **What is SQLException?**
Exception related to database operations in JDBC.

**Section 5: Multithreading (Q41–Q50)**

1. **What is a thread?**
The smallest unit of execution in a program.
2. **Advantages of multithreading?**
Faster execution, better CPU utilization, and responsiveness.
3. **Two ways to create a thread in Java?**
Extend the Thread class or implement the Runnable interface.
4. **Purpose of synchronized keyword?**
Prevents multiple threads from accessing shared resources simultaneously.
5. **What is a deadlock?**
When two or more threads wait indefinitely for each other’s resources.
6. **Example of a thread-safe class?**
StringBuffer, Vector.
7. **What is volatile keyword?**
Ensures visibility of variable changes across threads immediately.
8. **Difference between sleep() and wait()?**
sleep() pauses for a time without releasing the lock; wait() releases the lock and waits.
9. **What is thread priority?**
The scheduling hint for threads (range 1–10).
10. **What is the Executor framework?**
A set of utilities for managing thread pools.

**Section 6: Java 8 Features (Q51–Q60)**

1. **What is a lambda expression?**
A short block of code that takes parameters and returns a value without a name.
2. **What is a functional interface?**
An interface with exactly one abstract method.
3. **What is the Streams API?**
Allows functional-style operations on collections.
4. **What is the Optional class?**
A container to handle null values safely.
5. **What is a method reference?**
A shorthand for calling a method via ::.
6. **What is a default method in an interface?**
A method with a default implementation inside an interface.
7. **What are Predicate, Function, and Consumer?**
Predefined functional interfaces in Java 8.
8. **Difference between filter() and map() in Streams?**
filter() removes elements; map() transforms elements.
9. **What is a parallel stream?**
Processes stream elements concurrently.
10. **What are Instant and LocalDateTime?**
Classes from Java 8 Date-Time API.

**Section 7: Memory Management (Q61–Q70)**

1. **What is garbage collection?**
Automatic removal of unused objects from memory.
2. **Purpose of finalize() method?**
Runs before an object is garbage collected (deprecated in newer Java).
3. **Difference between heap and stack memory?**
Heap stores objects; stack stores method calls and local variables.
4. **What is a memory leak in Java?**
Unused objects remain referenced, preventing GC.
5. **Difference between SoftReference and WeakReference?**
Weak references are collected more aggressively than soft references.
6. **Difference between PermGen and Metaspace?**
PermGen was fixed-size; Metaspace grows dynamically (Java 8+).
7. **What causes OutOfMemoryError?**
Heap is full and cannot allocate more memory.
8. **What is StackOverflowError?**
Too deep recursion or excessive stack usage.
9. **How to tune JVM memory?**
Using options like -Xmx and -Xms.
10. **What is ReferenceQueue?**
Holds references after GC collects them.

**Section 8: File I/O (Q71–Q80)**

1. **Difference between FileReader and FileWriter?**
FileReader reads characters; FileWriter writes characters.
2. **Why use BufferedReader?**
Improves efficiency when reading large files.
3. **What is serialization?**
Converting an object into a byte stream.
4. **What is deserialization?**
Converting a byte stream back into an object.
5. **What is transient keyword?**
Skips a variable during serialization.
6. **What are Path and Files classes?**
Utility classes from Java NIO for file operations.
7. **What is RandomAccessFile?**
Allows read/write at specific file positions.
8. **Difference between InputStream and Reader?**
InputStream handles bytes; Reader handles characters.
9. **Difference between OutputStream and Writer?**
OutputStream writes bytes; Writer writes characters.
10. **What is ObjectOutputStream?**
Writes objects in serialized form to a stream.

**Section 9: Advanced Java (Q81–Q90)**

1. **What is JDBC?**
Java Database Connectivity API for database operations.
2. **Difference between Statement and PreparedStatement?**
PreparedStatement is precompiled, faster, and safer.
3. **What is a ResultSet?**
A table of data returned by a database query.
4. **What is connection pooling?**
Reusing database connections for efficiency.
5. **What is a servlet?**
A Java program that handles HTTP requests on the server.
6. **What is JSP?**
Java Server Pages for embedding Java into HTML.
7. **What is JPA?**
Java Persistence API for ORM.
8. **What is Hibernate?**
A popular ORM framework for Java.
9. **What is RMI?**
Remote Method Invocation for calling remote objects.
10. **What is JMS?**
Java Message Service for asynchronous communication.

**Section 10: Miscellaneous (Q91–Q100)**

1. **What is an enum?**
A type that defines a set of constants.
2. **What is an annotation?**
Metadata for code elements.
3. **What is the Reflection API?**
Allows inspection and modification of code at runtime.
4. **What is a marker interface?**
An interface with no methods (e.g., Serializable).
5. **What is the Singleton pattern?**
Ensures only one instance of a class exists.
6. **What is the Factory pattern?**
Encapsulates object creation.
7. **What is dependency injection?**
Providing required objects from outside a class.
8. **How to make a class immutable?**
Declare it final, fields final, no setters.
9. **Why override equals() and hashCode()?**
To ensure correct behavior in hash-based collections.
10. **Java best practices?**
Follow naming conventions, handle exceptions, manage resources, write thread-safe code.

**Section 11: String Handling & Logic (Q101–Q120)**

1. **How to count uppercase and lowercase letters in a String?**
Iterate characters and use Character.isUpperCase() / isLowerCase().
2. **How to toggle case of characters in a String?**
Iterate and convert uppercase to lowercase and vice versa.
3. **How to reverse a String without built-in methods?**
Use a loop from the end to the start to build a new string.
4. **How to remove special characters from a String?**
Use replaceAll("[^a-zA-Z0-9]", "").
5. **How to check if a String contains only letters and spaces?**
Use matches("[a-zA-Z ]+").
6. **How to check if one String is a rotation of another?**
Concatenate the first string with itself and check contains().
7. **How to check palindrome using recursion?**
Compare first and last characters, then recurse on the substring.
8. **How to capitalize the first letter of each word?**
Split into words, capitalize first letter, and join.
9. **How to replace only the first occurrence in a String?**
Use replaceFirst().
10. **How to find duplicate words in a String?**
Split into words and count using a Map.
11. **How to convert a String to title case?**
Capitalize the first letter of each word.
12. **How to reverse words but keep punctuation positions?**
Skip non-letter characters during reversal.
13. **How to convert String to char array without spaces?**
replace(" ", "").toCharArray().
14. **How to split comma-separated values into an array?**
Use split(",").
15. **How to swap the first and last character of a String?**
Use a StringBuilder and set characters.
16. **How to print ASCII values of characters?**
Cast each char to (int).
17. **How to swap two characters in a String?**
Convert to char array and swap.
18. **How to count palindrome subsequences in a String?**
Use recursion or dynamic programming.
19. **How to check balanced brackets in a String?**
Use a stack to track opening/closing brackets.
20. **How to convert String to Morse code?**
Map each letter to its Morse equivalent and join.

**Section 12: Number & Math Problems (Q121–Q140)**

1. **How to check prime using bitwise method?**
Test divisibility up to √n.
2. **How to find the next prime number?**
Increment from N+1 until prime found.
3. **How to check palindrome number using recursion?**
Reverse digits recursively and compare.
4. **How to find factorial recursively?**
Base case 1; return n \* factorial(n-1).
5. **How to find factorial iteratively?**
Multiply in a loop.
6. **How to generate Fibonacci recursively?**
Base case 0, 1; else f(n-1)+f(n-2).
7. **How to generate Fibonacci iteratively?**
Update two variables in a loop.
8. **How to find prime factors recursively?**
Divide by smallest divisor and recurse.
9. **How to check if number is a power of two?**
(n & (n-1)) == 0 && n > 0.
10. **How to check Armstrong number recursively?**
Sum of digits to power of count equals number.
11. **How to reverse a number?**
Loop extracting digits and build reversed.
12. **How to check Strong number recursively?**
Sum of factorial of digits equals number.
13. **How to check perfect square?**
Square of sqrt(n) equals n.
14. **What is a Spy number?**
Digit sum equals digit product.
15. **What is a Happy number?**
Sum of squares of digits leads to 1.
16. **What is a Duck number?**
Contains zero but doesn’t start with zero.
17. **What is a Buzz number?**
Divisible by 7 or ends with 7.
18. **What is an Evil number?**
Binary representation has even 1s.
19. **What is a Magic number?**
Recursive digit sum equals 1.
20. **What is a Disarium number?**
Sum of digits raised to their position equals number.

**Section 13: Arrays & Collections (Q141–Q160)**

1. **How to sum elements at even indexes?**
Loop with step 2 from index 0.
2. **How to sum elements at odd indexes?**
Loop with step 2 from index 1.
3. **How to find pair sum equal to target?**
Use a HashSet to check complements.
4. **How to find triplet sum equal to target?**
Sort and use two pointers.
5. **How to find median of array?**
Sort and pick middle.
6. **How to find mode of array?**
Count frequency, return max frequency element.
7. **How to find average of array?**
Sum / length.
8. **How to print palindrome elements from array?**
Apply palindrome check on each.
9. **How to print prime elements from array?**
Apply prime check on each.
10. **How to merge sorted arrays without extra space?**
Use gap method.
11. **How to find kth largest element?**
Use PriorityQueue or Quickselect.
12. **How to remove duplicates from array?**
Use LinkedHashSet.
13. **How to find maximum subarray sum?**
Use Kadane’s algorithm.
14. **How to find minimum subarray sum?**
Use modified Kadane’s.
15. **How to rotate array right cyclically?**
Reverse method.
16. **How to rotate array left cyclically?**
Reverse method.
17. **How to move zeros to end keeping order?**
Stable shifting.
18. **How to sort elements by frequency?**
Use Map + custom comparator.
19. **How to find first non-repeating element?**
Map count then first count==1.
20. **How to find leaders in array?**
Traverse from right keeping max so far.

**Section 14: File, Date & Miscellaneous (Q161–Q180)**

1. **How to append to file?**
FileWriter(file, true).
2. **How to read CSV file?**
BufferedReader + split(",").
3. **How to write CSV file?**
FileWriter.
4. **How to search word in text file?**
Read line by line, use contains().
5. **How to replace word in file?**
Read, replaceAll(), write back.
6. **How to list all files in folder?**
File.listFiles().
7. **How to filter only .txt files?**
FilenameFilter.
8. **How to get current time in milliseconds?**
System.currentTimeMillis().
9. **How to measure execution time?**
Start time – end time.
10. **How to get current working directory?**
System.getProperty("user.dir").
11. **How to create ZIP file?**
ZipOutputStream.
12. **How to extract ZIP file?**
ZipInputStream.
13. **How to read PDF in Java?**
Apache PDFBox or iText.
14. **How to read Excel file?**
Apache POI.
15. **How to write Excel file?**
Apache POI.
16. **How to get image file size?**
File.length().
17. **How to resize image?**
BufferedImage.
18. **How to change image format?**
ImageIO.write().
19. **How to check file permissions?**
canRead(), canWrite(), canExecute().
20. **How to check hidden file?**
isHidden().

**Section 15: Advanced Problem Solving (Q181–Q200)**

1. **How to generate all permutations of a string?**
Use recursion and swapping.
2. **How to generate all combinations?**
Use backtracking.
3. **How to build a Sudoku solver?**
Use backtracking with constraints checking.
4. **How to solve N-Queens problem?**
Backtracking with column/diagonal checks.
5. **How to solve a maze problem?**
Use DFS or BFS.
6. **How to detect cycles in a graph?**
Use DFS with visited and recursion stack tracking.
7. **What is Bellman-Ford algorithm?**
Finds shortest paths by relaxing edges repeatedly.
8. **What is Prim’s algorithm?**
Builds minimum spanning tree using a priority queue.
9. **What is Kruskal’s algorithm?**
Builds MST using sorted edges and disjoint sets.
10. **What is topological sort?**
Linear ordering of vertices using DFS or Kahn’s algorithm.
11. **How to insert into a Binary Search Tree (BST)?**
Recursively place nodes based on value.
12. **How to delete from a BST?**
Handle leaf, one-child, and two-child cases separately.
13. **What is Heap Sort?**
Sorting using heapify and extracting the root repeatedly.
14. **What is Counting Sort?**
Sort using frequency count of elements.
15. **What is Radix Sort?**
Sort digit by digit using counting sort.
16. **What is Shell Sort?**
Sort using decreasing gap-based insertion sort.
17. **What are bitwise AND, OR, XOR operations?**
&, |, and ^ perform bit-level logical operations.
18. **How to swap numbers without a temp variable using XOR?**
a = a ^ b; b = a ^ b; a = a ^ b;
19. **How to count set bits in an integer?**
Use n & (n-1) repeatedly.
20. **How to do fast exponentiation?**
Use divide-and-conquer method.

**Section 16: Extra Simple Core Java Q&A (Q201–Q220)**

1. **What is the file extension of Java source code?**
.java for source, .class for compiled bytecode.
2. **Who invented Java?**
James Gosling.
3. **When was Java first released?**
1995.
4. **What is the latest Java LTS version (as of 2025)?**
Java 21.
5. **What is the default package in Java?**
java.lang.
6. **What is a package?**
A namespace for organizing classes and interfaces.
7. **Syntax to import a class?**
import packageName.ClassName;
8. **What are access modifiers in Java?**
public, protected, default, private.
9. **Examples of non-access modifiers?**
static, final, abstract, synchronized, transient, volatile.
10. **Default value of int in Java?**
0.
11. **Default value of boolean in Java?**
false.
12. **How to define a static variable?**
static type varName;
13. **How to call a static method?**
ClassName.methodName();
14. **What is the parameter of main method?**
String[] args.
15. **What is System.out.println?**
Prints output to console with a newline.
16. **Types of comments in Java?**
Single-line, multi-line, and documentation comments.
17. **Variable naming rules in Java?**
Start with letter, $, or \_, case-sensitive, no spaces.
18. **Is Java case-sensitive?**
Yes.
19. **Do local variables have default values?**
No, they must be initialized before use.
20. **What is an instance variable?**
Variable defined in a class but outside any method.

**Section 17: Keywords, Operators & Basics (Q221–Q250)**

1. **How many reserved keywords in Java?**
Around 50 (varies by version).
2. **What does break do?**
Exits loop or switch.
3. **What does continue do?**
Skips current iteration.
4. **What does return do?**
Exits from a method, optionally returning a value.
5. **Arithmetic operators in Java?**
+, -, \*, /, %.
6. **Relational operators?**
==, !=, >, <, >=, <=.
7. **Logical operators?**
&&, ||, !.
8. **Assignment operators?**
=, +=, -=, \*=, /=, %=.
9. **Unary operators?**
++, --, +, -.
10. **Syntax of ternary operator?**
condition ? valueIfTrue : valueIfFalse;
11. **Types of type casting in Java?**
Widening (implicit), narrowing (explicit).
12. **What does instanceof do?**
Checks object type.
13. **Bitwise operators?**
&, |, ^, ~, <<, >>, >>>.
14. **Shift operators?**
<<, >>, >>>.
15. **Difference between pre-increment and post-increment?**
Pre (++i) increments before use; post (i++) increments after.
16. **Why use wrapper classes?**
To treat primitives as objects.
17. **What is autoboxing?**
Automatic primitive → wrapper conversion.
18. **What is unboxing?**
Automatic wrapper → primitive conversion.
19. **What is a default constructor?**
No-argument constructor, provided by compiler if not defined.
20. **What is a parameterized constructor?**
Constructor with parameters.
21. **What is overloaded constructor?**
Multiple constructors with different parameter lists.
22. **What is a static block?**
Runs once when class is loaded.
23. **What is an initialization block?**
Runs when an object is created.
24. **How to request garbage collection?**
System.gc();
25. **What is null in Java?**
Reference with no object.
26. **Use of this()?**
Calls another constructor in the same class.
27. **Use of super()?**
Calls parent class constructor.
28. **What is method signature?**
Method name + parameter list.
29. **What are varargs?**
Variable-length arguments (type... args).
30. **What is a marker annotation?**
Annotation with no methods.

**Section 18: Java Basics & Syntax (Q251–Q270)**

1. **Three types of comments in Java?**
Single-line, multi-line, documentation.
2. **Class naming convention?**
PascalCase.
3. **Method naming convention?**
camelCase.
4. **How to define a constant?**
final type NAME = value;
5. **Can Java run without main method?**
No (except with special tools or older tricks).
6. **Method for single-line output without newline?**
System.out.print().
7. **Method for output with newline?**
System.out.println().
8. **How to format output?**
System.out.printf() or String.format().
9. **Do local variables have default values?**
No.
10. **What does \n mean?**
Newline.
11. **What does \t mean?**
Tab space.
12. **How to define char literal?**
Single quotes, e.g., 'A'.
13. **Boolean literal values?**
true, false.
14. **How to define string literal?**
Double quotes.
15. **Run code without method?**
Place in static block.
16. **How to write infinite loop?**
while(true) or for(;;).
17. **What is block scope?**
Scope inside {}.
18. **Array index start?**
0.
19. **Get array length?**
array.length.
20. **Last array index?**
length - 1.

**Section 19: Java Keywords & Small Concepts (Q281–Q300)**

1. **Is goto used in Java?**
No, it’s reserved but not implemented.
2. **Is const used in Java?**
No, use final instead.
3. **What is native keyword?**
Calls non-Java (e.g., C/C++) code.
4. **What is strictfp keyword?**
Enforces IEEE floating-point standard.
5. **What is assert keyword?**
For debugging/testing conditions.
6. **What is synchronized method?**
Method accessed by only one thread at a time.
7. **What is volatile variable?**
Always read from main memory.
8. **What is transient keyword?**
Skips variable in serialization.
9. **When is default keyword used?**
In switch or interface default methods.
10. **What is enum short for?**
Enumeration.
11. **Enum constant naming convention?**
Uppercase.
12. **Can switch use String?**
Yes, since Java 7.
13. **Can switch have duplicate case values?**
No.
14. **What happens without break in switch?**
Fall-through.
15. **Can continue be used with label?**
Yes, in nested loops.
16. **Effect of final method parameter?**
Cannot reassign inside method.
17. **When must super() be called?**
First line of constructor.
18. **Multiple public classes in a file?**
No, only one allowed.
19. **Private main method?**
Compiles, but JVM won’t call it.
20. **What is static import?**
Imports static members for direct access.

**Section 20: Java Memory & Miscellaneous (Q311–Q350)**

1. **Heap memory stores?**
Objects and instance variables.
2. **Stack memory stores?**
Local variables, method calls.
3. **Main JVM components?**
Class loader, memory, execution engine, GC.
4. **Class loader types?**
Bootstrap, Extension, System.
5. **Class loading lazy?**
Yes.
6. **Is overloading runtime?**
No, compile-time.
7. **Is overriding compile-time?**
No, runtime.
8. **Does System.gc() guarantee GC?**
No.
9. **Does finalize() always run?**
Not guaranteed.
10. **Change JVM stack size?**
-Xss option.
11. **Set heap size?**
-Xms and -Xmx.
12. **When OutOfMemoryError occurs?**
Heap is full.
13. **When StackOverflowError occurs?**
Stack limit exceeded.
14. **Where is String pool?**
Heap, in a special area.
15. **What is interned string?**
String stored in pool.
16. **Infinite recursion causes?**
StackOverflowError.
17. **Immutable object?**
State cannot change.
18. **Mutable object?**
State can change.
19. **Marker interface?**
Interface with no methods.
20. **Serializable interface?**
Enables object serialization.
21. **Externalizable interface?**
Custom serialization control.
22. **Where is readObject() defined?**
ObjectInputStream.
23. **Where is writeObject() defined?**
ObjectOutputStream.
24. **Parent of Object class?**
None, it’s the root.
25. **Methods in Object class?**
toString(), equals(), hashCode(), clone(), wait(), notify(), notifyAll().
26. **equals() vs == ?**
equals() checks values; == checks references.
27. **hashCode() purpose?**
Hash-based collection support.
28. **Default toString() output?**
ClassName@HexHashCode.
29. **wait() outside synchronized?**
Throws IllegalMonitorStateException.
30. **notify() vs notifyAll()?**
One thread vs all waiting threads.
31. **sleep() vs wait()?**
sleep() keeps lock; wait() releases.
32. **Methods in Runnable?**
run().
33. **Methods in Callable?**
call() returns value.
34. **Thread vs Runnable?**
Extend vs implement.
35. **Daemon thread?**
Background thread.
36. **Set daemon thread?**
setDaemon(true) before start().
37. **join() method?**
Wait for another thread to finish.
38. **yield() method?**
Suggests thread scheduler to pause.
39. **Default thread priority?**
5.
40. **What is TimerTask?**
Task scheduled by java.util.Timer.