# Java 100 Q&A Glossary (Q1–Q350)

## Q1. What is Java?

Java is a programming language. It is object-oriented and works on any computer with JVM.

## Q2. What is JVM?

JVM is Java Virtual Machine. It runs Java bytecode on any device.

## Q3. What is the difference between JDK and JRE?

JDK = JRE + developer tools. JRE = JVM + libraries to run Java.

## Q4. What is bytecode?

Bytecode is compiled Java code that JVM can run.

## Q5. Why is Java platform-independent?

Because bytecode can run on any system with JVM.

## Q6. What is the signature of the main method in Java?

public static void main(String[] args)

## Q7. Difference between variable and constant?

Variable can change. Constant (final) cannot change.

## Q8. What is a wrapper class?

Class that turns primitive values into objects, like Integer or Double.

## Q9. Difference between == and .equals()?

== checks memory reference. .equals() checks actual value.

## Q10. Difference between final, finally, finalize()?

final = constant/stop changes. finally = cleanup block. finalize() = before GC.

## Q11. What are the four pillars of OOP?

Encapsulation, Inheritance, Polymorphism, Abstraction.

## Q12. What is encapsulation?

Wrapping data and methods into one class, hiding details.

## Q13. What is abstraction?

Showing only important features, hiding details.

## Q14. Types of inheritance in Java?

Single, Multilevel, Hierarchical. (No multiple inheritance directly).

## Q15. What is method overloading?

Same method name, different parameter lists.

## Q16. What is method overriding?

Child class redefines parent class method.

## Q17. Difference between interface and abstract class?

Interface = only abstract methods. Abstract class = abstract + normal methods.

## Q18. What is super keyword?

Refers to parent class, can call parent methods/constructors.

## Q19. What is constructor overloading?

Having multiple constructors with different parameters.

## Q20. What is this keyword?

Refers to the current object.

## Q21. Why are strings immutable in Java?

For security, memory saving, and thread safety.

## Q22. Difference between StringBuilder and StringBuffer?

StringBuilder = faster, not safe for threads. StringBuffer = slower, safe for threads.

## Q23. What is the String pool?

Special memory space for string literals.

## Q24. Difference between ArrayList and LinkedList?

ArrayList = fast random access. LinkedList = fast insert/delete.

## Q25. Difference between HashMap and Hashtable?

HashMap = not synchronized. Hashtable = synchronized.

## Q26. Why doesn’t HashSet allow duplicates?

It uses hashing, which ensures only unique values.

## Q27. Difference between Comparable and Comparator?

Comparable = natural order. Comparator = custom order.

## Q28. Difference between fail-fast and fail-safe iterators?

Fail-fast = errors if collection changes. Fail-safe = works on a copy.

## Q29. What is a PriorityQueue?

Queue that orders elements by priority.

## Q30. Difference between TreeMap and HashMap?

TreeMap = sorted order. HashMap = no order.

## Q31. Difference between checked and unchecked exceptions?

Checked = at compile time. Unchecked = at runtime.

## Q32. Flow of try-catch-finally?

try → catch (if error) → finally (always runs).

## Q33. Difference between throw and throws?

throw = to throw an exception. throws = to declare possible exceptions.

## Q34. How to create a custom exception?

Make a class that extends Exception or RuntimeException.

## Q35. How do multiple catch blocks work?

First matching block runs.

## Q36. Does finally block always execute?

Yes, except if System.exit() is called.

## Q37. What is try-with-resources?

try block that closes resources automatically.

## Q38. Difference between Error and Exception?

Error = serious, not fixable. Exception = can be handled.

## Q39. Example of RuntimeException?

NullPointerException, ArrayIndexOutOfBoundsException.

## Q40. What is SQLException?

Error related to database in JDBC.

## Q41. What is a thread?

Smallest unit of execution in Java.

## Q42. Advantages of multithreading?

Faster work, better CPU use.

## Q43. Two ways to create a thread?

Extend Thread class or implement Runnable.

## Q44. Purpose of synchronized keyword?

Stops multiple threads from accessing shared resource at same time.

## Q45. What is a deadlock?

When threads wait forever for each other’s resources.

## Q46. Example of a thread-safe class?

StringBuffer, Vector.

## Q47. What is volatile keyword?

Makes sure variable changes are visible to all threads immediately.

## Q48. Difference between sleep() and wait()?

sleep() pauses thread but keeps lock. wait() pauses and releases lock.

## Q49. What is thread priority?

Number (1–10) that gives hint for scheduling thread.

## Q50. What is Executor framework?

Set of tools to manage thread pools.

## Q51. What is a lambda expression?

Short block of code without a name, added in Java 8.

## Q52. What is a functional interface?

Interface with one abstract method.

## Q53. What is Streams API?

Helps process data collections in functional style.

## Q54. What is Optional class?

Container that may or may not hold a value, helps avoid null.

## Q55. What is method reference?

Shorthand to call a method using ::.

## Q56. What is default method in interface?

Method in interface with default code.

## Q57. What are Predicate, Function, Consumer?

Ready-made functional interfaces in Java 8.

## Q58. Difference between filter() and map()?

filter() removes elements. map() changes elements.

## Q59. What is a parallel stream?

Processes stream data with multiple threads.

## Q60. What are Instant and LocalDateTime?

Java 8 classes for date and time.

## Q61. What is garbage collection?

Automatic removal of unused objects.

## Q62. Purpose of finalize()?

Runs before object is garbage collected (not recommended now).

## Q63. Difference between heap and stack?

Heap = objects. Stack = method calls and local variables.

## Q64. What is a memory leak?

When unused objects still have references and cannot be collected.

## Q65. SoftReference vs WeakReference?

WeakReference = collected more easily than SoftReference.

## Q66. PermGen vs Metaspace?

PermGen = old fixed area. Metaspace = dynamic (Java 8+).

## Q67. What causes OutOfMemoryError?

Heap memory is full.

## Q68. What is StackOverflowError?

Too much recursion or deep stack.

## Q69. How to tune JVM memory?

Use -Xmx and -Xms options.

## Q70. What is ReferenceQueue?

Holds references after objects are collected.

## Q71. Difference between FileReader and FileWriter?

FileReader reads characters. FileWriter writes characters.

## Q72. Why use BufferedReader?

For faster reading of large files.

## Q73. What is serialization?

Convert object into byte stream.

## Q74. What is deserialization?

Convert byte stream back into object.

## Q75. What is transient keyword?

Skip a field during serialization.

## Q76. Path and Files classes?

Utility classes from Java NIO for file handling.

## Q77. What is RandomAccessFile?

Allows reading/writing at specific file positions.

## Q78. Difference between InputStream and Reader?

InputStream = bytes. Reader = characters.

## Q79. Difference between OutputStream and Writer?

OutputStream = bytes. Writer = characters.

## Q80. What is ObjectOutputStream?

Writes objects into byte stream.

## Q81. What is JDBC?

Java API for database work.

## Q82. Difference between Statement and PreparedStatement?

PreparedStatement = precompiled, faster, safer.

## Q83. What is ResultSet?

Table of data returned by a database query.

## Q84. What is connection pooling?

Reuse database connections to save resources.

## Q85. What is servlet?

Java program that handles web requests.

## Q86. What is JSP?

Java Server Pages = Java code inside HTML.

## Q87. What is JPA?

Java Persistence API, used for ORM.

## Q88. What is Hibernate?

A framework for ORM in Java.

## Q89. What is RMI?

Remote Method Invocation, for calling remote objects.

## Q90. What is JMS?

Java Messaging Service for async communication.

## Q91. What is an enum?

Special type for constants.

## Q92. What is an annotation?

Metadata added to code elements.

## Q93. What is Reflection API?

Lets you inspect or modify code at runtime.

## Q94. What is a marker interface?

Interface with no methods, like Serializable.

## Q95. What is Singleton pattern?

Design pattern where only one object of class exists.

## Q96. What is Factory pattern?

Design pattern that handles object creation.

## Q97. What is dependency injection?

Giving objects from outside instead of creating inside.

## Q98. How to make a class immutable?

Use final class, final fields, and no setters.

## Q99. Why override equals() and hashCode()?

To make sure collections work correctly with objects.

## Q100. Java best practices?

Follow naming rules, handle errors, close resources, write safe code.

## Q101. How to count uppercase and lowercase letters in a String?

Go through each character and use Character.isUpperCase() or isLowerCase().

## Q102. How to toggle case of characters in a String?

Loop through string and change upper to lower, lower to upper.

## Q103. How to reverse a String without built-in methods?

Start from end and build a new string by adding characters one by one.

## Q104. How to remove special characters from a String?

Use replaceAll("[^a-zA-Z0-9]", "").

## Q105. How to check if a String contains only letters and spaces?

Use matches("[a-zA-Z ]+").

## Q106. How to check if one String is a rotation of another?

Double the first string and check if the second exists inside it.

## Q107. How to check palindrome using recursion?

Compare first and last characters, then check inside substring.

## Q108. How to capitalize the first letter of each word?

Split words, uppercase first letter, join back.

## Q109. How to replace only the first occurrence in a String?

Use replaceFirst().

## Q110. How to find duplicate words in a String?

Split into words and count using a Map.

## Q111. How to convert a String to title case?

Uppercase first letter of each word, lowercase rest.

## Q112. How to reverse words but keep punctuation positions?

Skip non-letter characters while reversing.

## Q113. How to convert String to char array without spaces?

Use replace(" ", "").toCharArray().

## Q114. How to split comma-separated values into an array?

Use split(",").

## Q115. How to swap the first and last character of a String?

Convert to StringBuilder, swap chars.

## Q116. How to print ASCII values of characters?

Loop and cast each char to int.

## Q117. How to swap two characters in a String?

Convert to char array and swap indexes.

## Q118. How to count palindrome subsequences in a String?

Use recursion or dynamic programming.

## Q119. How to check balanced brackets in a String?

Use stack to push opening and match with closing.

## Q120. How to convert String to Morse code?

Map each letter to its Morse code and join.

## Q121. How to check prime using bitwise method?

Check divisibility up to √n with math tricks.

## Q122. How to find the next prime number?

Start from n+1 and test until a prime is found.

## Q123. How to check palindrome number using recursion?

Reverse digits recursively and compare with original.

## Q124. How to find factorial recursively?

If n=1 return 1 else return n \* factorial(n-1).

## Q125. How to find factorial iteratively?

Multiply numbers in a loop from 1 to n.

## Q126. How to generate Fibonacci recursively?

If n=0 return 0, if n=1 return 1 else f(n-1)+f(n-2).

## Q127. How to generate Fibonacci iteratively?

Start with 0,1 then add last two numbers in loop.

## Q128. How to find prime factors recursively?

Divide by smallest divisor and recurse on quotient.

## Q129. How to check if number is power of two?

Check (n & (n-1)) == 0 and n > 0.

## Q130. How to check Armstrong number recursively?

Sum of digits raised to power of count should equal number.

## Q131. How to reverse a number?

Use loop: take digit with %, build reversed with \*10 + digit.

## Q132. How to check Strong number recursively?

Sum of factorial of digits should equal number.

## Q133. How to check perfect square?

Square root squared should equal the number.

## Q134. What is a Spy number?

Sum of digits equals product of digits.

## Q135. What is a Happy number?

Sum of squares of digits becomes 1.

## Q136. What is a Duck number?

Number contains 0 but does not start with 0.

## Q137. What is a Buzz number?

Number ends with 7 or divisible by 7.

## Q138. What is an Evil number?

Binary form has even number of 1s.

## Q139. What is a Magic number?

Sum of digits repeatedly gives 1.

## Q140. What is a Disarium number?

Sum of digits raised to position equals number.

## Q141. How to sum elements at even indexes?

Loop with step 2 from index 0.

## Q142. How to sum elements at odd indexes?

Loop with step 2 from index 1.

## Q143. How to find pair sum equal to target?

Use HashSet to check for complement values.

## Q144. How to find triplet sum equal to target?

Sort and use two-pointer method.

## Q145. How to find median of array?

Sort array and pick middle value.

## Q146. How to find mode of array?

Count frequency of each element, return max.

## Q147. How to find average of array?

Sum of elements divided by length.

## Q148. How to print palindrome elements from array?

Check palindrome for each element.

## Q149. How to print prime elements from array?

Check prime for each element.

## Q150. How to merge sorted arrays without extra space?

Use gap method to merge in-place.

## Q151. How to find kth largest element?

Use PriorityQueue or Quickselect.

## Q152. How to remove duplicates from array?

Use LinkedHashSet to remove duplicates.

## Q153. How to find maximum subarray sum?

Use Kadane’s algorithm.

## Q154. How to find minimum subarray sum?

Use modified Kadane’s algorithm.

## Q155. How to rotate array right cyclically?

Use reverse method.

## Q156. How to rotate array left cyclically?

Use reverse method.

## Q157. How to move zeros to end keeping order?

Shift non-zeros forward, fill rest with zeros.

## Q158. How to sort elements by frequency?

Use Map to count, then custom sort.

## Q159. How to find first non-repeating element?

Use Map count, return first count==1 element.

## Q160. How to find leaders in array?

From right, track maximum seen so far.

## Q161. How to append to file?

Use FileWriter(file, true).

## Q162. How to read CSV file?

Use BufferedReader and split lines by comma.

## Q163. How to write CSV file?

Use FileWriter with commas between values.

## Q164. How to search word in text file?

Read lines and check with contains().

## Q165. How to replace word in file?

Read, replaceAll(), write back.

## Q166. How to list all files in folder?

Use File.listFiles().

## Q167. How to filter only .txt files?

Use FilenameFilter.

## Q168. How to get current time in milliseconds?

Use System.currentTimeMillis().

## Q169. How to measure execution time?

Subtract start time from end time.

## Q170. How to get current working directory?

Use System.getProperty("user.dir").

## Q171. How to create ZIP file?

Use ZipOutputStream.

## Q172. How to extract ZIP file?

Use ZipInputStream.

## Q173. How to read PDF in Java?

Use Apache PDFBox or iText.

## Q174. How to read Excel file?

Use Apache POI library.

## Q175. How to write Excel file?

Use Apache POI library.

## Q176. How to get image file size?

Use File.length().

## Q177. How to resize image?

Use BufferedImage operations.

## Q178. How to change image format?

Use ImageIO.write().

## Q179. How to check file permissions?

Use canRead(), canWrite(), canExecute().

## Q180. How to check hidden file?

Use isHidden().

## Q181. How to generate all permutations of a string?

Use recursion and swapping characters.

## Q182. How to generate all combinations?

Use backtracking method.

## Q183. How to build a Sudoku solver?

Use backtracking with rules for rows, columns, and grids.

## Q184. How to solve N-Queens problem?

Use backtracking checking row, column, diagonal.

## Q185. How to solve a maze problem?

Use DFS or BFS search.

## Q186. How to detect cycles in a graph?

Use DFS with visited + recursion stack.

## Q187. What is Bellman-Ford algorithm?

Finds shortest paths with edge relaxation method.

## Q188. What is Prim’s algorithm?

Builds minimum spanning tree using greedy method.

## Q189. What is Kruskal’s algorithm?

Builds MST using edges sorted by weight.

## Q190. What is topological sort?

Order of vertices using DFS or Kahn’s algorithm.

## Q191. How to insert into a BST?

Place node left if smaller, right if larger.

## Q192. How to delete from a BST?

Handle cases: leaf, one child, two children.

## Q193. What is Heap Sort?

Sorting by heapify and extracting root repeatedly.

## Q194. What is Counting Sort?

Sort using frequency of elements.

## Q195. What is Radix Sort?

Sort digit by digit using counting sort.

## Q196. What is Shell Sort?

Sort using decreasing gaps with insertion sort.

## Q197. What are bitwise AND, OR, XOR?

& = AND, | = OR, ^ = XOR at bit level.

## Q198. How to swap numbers with XOR?

a=a^b; b=a^b; a=a^b;

## Q199. How to count set bits in integer?

Use n & (n-1) repeatedly.

## Q200. How to do fast exponentiation?

Use divide and conquer with squaring.

# Java Q&A Glossary (Q201–Q275)

## Q201. What is the file extension of Java source code?

.java for source code, .class for compiled bytecode.

## Q202. Who invented Java?

James Gosling.

## Q203. When was Java first released?

1995.

## Q204. What is the latest Java LTS version (2025)?

Java 21.

## Q205. What is the default package in Java?

java.lang.

## Q206. What is a package?

A way to group classes and interfaces.

## Q207. Syntax to import a class?

import packageName.ClassName;

## Q208. What are access modifiers?

public, protected, default, private.

## Q209. Examples of non-access modifiers?

static, final, abstract, synchronized, transient, volatile.

## Q210. Default value of int?

0.

## Q211. Default value of boolean?

false.

## Q212. How to define static variable?

static type varName;

## Q213. How to call static method?

ClassName.methodName();

## Q214. Parameter of main method?

String[] args.

## Q215. What is System.out.println?

Prints message with newline.

## Q216. Types of comments in Java?

Single-line, multi-line, documentation.

## Q217. Variable naming rules?

Start with letter, $, or \_. No spaces. Case sensitive.

## Q218. Is Java case-sensitive?

Yes.

## Q219. Do local variables have default values?

No, must be initialized.

## Q220. What is an instance variable?

Variable inside class but outside method.

## Q221. How many reserved keywords?

Around 50.

## Q222. What does break do?

Exits loop or switch.

## Q223. What does continue do?

Skips current loop step.

## Q224. What does return do?

Exits method and may return value.

## Q225. Arithmetic operators?

+, -, \*, /, %.

## Q226. Relational operators?

==, !=, >, <, >=, <=.

## Q227. Logical operators?

&&, ||, !.

## Q228. Assignment operators?

=, +=, -=, \*=, /=, %=.

## Q229. Unary operators?

++, --, +, -.

## Q230. Ternary operator?

condition ? valueIfTrue : valueIfFalse.

## Q231. Types of type casting?

Widening (automatic), narrowing (manual).

## Q232. instanceof keyword?

Checks if object is of a class type.

## Q233. Bitwise operators?

&, |, ^, ~, <<, >>, >>>.

## Q234. Shift operators?

<<, >>, >>>.

## Q235. Pre-increment vs post-increment?

++i = increment before use. i++ = increment after use.

## Q236. Why wrapper classes?

To treat primitives as objects.

## Q237. What is autoboxing?

Primitive to wrapper automatically.

## Q238. What is unboxing?

Wrapper to primitive automatically.

## Q239. Default constructor?

Constructor with no arguments.

## Q240. Parameterized constructor?

Constructor with arguments.

## Q241. Overloaded constructor?

Multiple constructors with different parameters.

## Q242. Static block?

Runs once when class is loaded.

## Q243. Initialization block?

Runs when object is created.

## Q244. How to request GC?

System.gc().

## Q245. What is null?

Reference that points to nothing.

## Q246. Use of this()?

Calls another constructor in same class.

## Q247. Use of super()?

Calls parent constructor.

## Q248. Method signature?

Method name + parameter list.

## Q249. varargs?

Variable-length arguments (type... args).

## Q250. Marker annotation?

Annotation with no methods.

## Q251. Three types of comments?

Single-line, multi-line, documentation.

## Q252. Class naming convention?

PascalCase.

## Q253. Method naming convention?

camelCase.

## Q254. How to define constant?

final type NAME = value.

## Q255. Can Java run without main method?

No (except old tricks).

## Q256. Single-line output without newline?

System.out.print().

## Q257. Output with newline?

System.out.println().

## Q258. How to format output?

Use printf() or String.format().

## Q259. Do local variables have default values?

No, must initialize.

## Q260. What does \n mean?

New line.

## Q261. What does \t mean?

Tab space.

## Q262. Char literal?

Single quotes, e.g., 'A'.

## Q263. Boolean literals?

true, false.

## Q264. String literal?

Double quotes, e.g., "Hello".

## Q265. Run code without method?

Use static block.

## Q266. Infinite loop?

while(true) or for(;;).

## Q267. Block scope?

Variables valid inside {} only.

## Q268. Array index start?

0.

## Q269. Get array length?

array.length.

## Q270. Last array index?

length - 1.

## Q271. Is goto used in Java?

No, reserved only.

## Q272. Is const used in Java?

No, use final instead.

## Q273. native keyword?

Calls non-Java code like C/C++.

## Q274. strictfp keyword?

Enforces standard floating-point math.

## Q275. assert keyword?

Used for debugging conditions.

# Java Q&A Glossary (Q276–Q350)

## Q276. synchronized method?

Only one thread can use the method at a time.

## Q277. volatile variable?

Always read from main memory, not cache.

## Q278. transient keyword?

Skip field during serialization.

## Q279. default keyword?

Used in switch or default interface methods.

## Q280. enum short for?

Enumeration, type for constants.

## Q281. Enum constant naming?

Uppercase letters.

## Q282. Can switch use String?

Yes, since Java 7.

## Q283. Duplicate case values in switch?

Not allowed.

## Q284. No break in switch?

Causes fall-through to next case.

## Q285. continue with label?

Yes, skips to labeled loop.

## Q286. final method parameter?

Cannot reassign inside method.

## Q287. super() call?

Must be first line in constructor.

## Q288. Multiple public classes in file?

No, only one public class allowed.

## Q289. Private main method?

Compiles but JVM won’t call it.

## Q290. static import?

Imports static members directly.

## Q291. Heap memory stores?

Objects and instance variables.

## Q292. Stack memory stores?

Local variables and method calls.

## Q293. Main JVM components?

Class loader, memory, execution engine, GC.

## Q294. Class loader types?

Bootstrap, Extension, System.

## Q295. Class loading lazy?

Yes, loads classes when needed.

## Q296. Overloading runtime?

No, done at compile time.

## Q297. Overriding compile-time?

No, checked at runtime.

## Q298. Does System.gc() guarantee GC?

No, it only suggests GC.

## Q299. finalize() always run?

Not guaranteed.

## Q300. Change JVM stack size?

Use -Xss option.

## Q301. Set heap size?

Use -Xms and -Xmx options.

## Q302. OutOfMemoryError occurs?

When heap memory is full.

## Q303. StackOverflowError occurs?

Too many method calls in stack.

## Q304. Where is String pool?

Inside heap memory.

## Q305. Interned string?

Stored in String pool.

## Q306. Infinite recursion causes?

StackOverflowError.

## Q307. Immutable object?

State cannot change after creation.

## Q308. Mutable object?

State can change.

## Q309. Marker interface?

Interface with no methods.

## Q310. Serializable interface?

Lets object be saved to stream.

## Q311. Externalizable interface?

Custom control over serialization.

## Q312. readObject() defined in?

ObjectInputStream.

## Q313. writeObject() defined in?

ObjectOutputStream.

## Q314. Parent of Object class?

None, Object is the root class.

## Q315. Methods in Object class?

toString(), equals(), hashCode(), clone(), wait(), notify(), notifyAll().

## Q316. equals() vs ==?

equals() checks value. == checks reference.

## Q317. hashCode() purpose?

Helps in hash-based collections.

## Q318. Default toString()?

ClassName@HexHashCode.

## Q319. wait() outside synchronized?

Throws IllegalMonitorStateException.

## Q320. notify() vs notifyAll()?

notify() wakes one thread. notifyAll() wakes all.

## Q321. sleep() vs wait()?

sleep() keeps lock. wait() releases lock.

## Q322. Methods in Runnable?

run().

## Q323. Methods in Callable?

call() returns value.

## Q324. Thread vs Runnable?

Thread extends class. Runnable implements interface.

## Q325. Daemon thread?

Background service thread.

## Q326. Set daemon thread?

Call setDaemon(true) before start().

## Q327. join() method?

Waits for thread to finish.

## Q328. yield() method?

Suggests scheduler to pause thread.

## Q329. Default thread priority?

5.

## Q330. TimerTask?

Task scheduled by java.util.Timer.

## Q331. What is polymorphism?

Ability of one method to do different things.

## Q332. What is abstraction?

Hide details and show essential parts.

## Q333. What is inheritance?

One class getting features of another.

## Q334. What is encapsulation?

Wrapping data and code together.

## Q335. Overloading vs Overriding?

Overloading = same method different params. Overriding = redefine in subclass.

## Q336. Constructor?

Special method to create object.

## Q337. Default constructor?

No-arg constructor given by compiler.

## Q338. Parameterized constructor?

Constructor with parameters.

## Q339. Static method?

Method called using class name, not object.

## Q340. Abstract method?

Method with no body, must be implemented.

## Q341. Interface?

Collection of abstract methods.

## Q342. Abstract class?

Class with abstract + normal methods.

## Q343. Package?

Group of classes and interfaces.

## Q344. Import?

Bring in classes from other packages.

## Q345. Final class?

Cannot be inherited.

## Q346. Final method?

Cannot be overridden.

## Q347. Static block?

Runs once when class loads.

## Q348. Instance block?

Runs when object is created.

## Q349. super keyword?

Refers to parent class.

## Q350. this keyword?

Refers to current object.