1. GIT

- **What is Git, and how is it different from centralized version control systems like SVN?**
- **Explain the difference between a working directory, staging area, and repository in Git.**
- **What is a commit in Git? What best practices should be followed while writing commit messages?**
- **What does** git status **and** git log **show? How are they useful during development?**
- **What is a branch in Git, and why is branching important in collaborative development?**
- **Explain the difference between git merge and git rebase. When would you prefer one over the other?**
- **What is a merge conflict? In which situations do merge conflicts usually occur?**
- **How do you resolve a merge conflict in Git? Explain the steps.**
- What happens internally during a git rebase? How does it affect commit history?
- Why is rebasing a shared/public branch considered risky?
- Pull, Push & Remote Repositories
- What is the difference between git pull and git fetch?
- What happens when you run git push? What are common reasons for a push failure?
- How is Git used in code review processes (e.g., Pull Requests)? What are reviewers expected to check?
- How do you undo a commit that has already been pushed to a remote repository?

2. SDLC

- What is SDLC? Why is it important in software development?
- Explain the phases of SDLC: Requirements, Design, Implementation, Testing, Deployment, and Maintenance.
- What is requirements engineering? How do you differentiate between functional and non-functional requirements?
- Design & Implementation
- What activities are performed during the design phase of SDLC? How does design impact maintainability?
- What is the difference between verification and validation? Can you give examples for each?
- What types of maintenance exist in SDLC? Explain corrective, adaptive, perfective, and preventive maintenance.
- **Compare Waterfall, Agile, Spiral, and DevOps models. When would you choose each?**
- **How does SDLC change when developing a cloud-native or microservices-based application?**
- **What are entry and exit criteria for each SDLC phase? Why are they important?**
- **How do you handle ambiguous or frequently changing requirements?**
- **What techniques do you use for requirement elicitation (e.g., interviews, workshops, prototyping)?**

- **What is a Software Requirements Specification (SRS)? What key sections does it contain?**
- **How do you ensure requirements are testable and traceable throughout the SDLC?**
- **What is the difference between High-Level Design (HLD) and Low-Level Design (LLD)?**
- **How do design patterns improve software quality and maintainability? Give an example.**
- **How do you address non-functional requirements (performance, security, scalability) during design?**
- **How do coding standards and code reviews fit into the SDLC?**
- **What role does version control play in implementation and maintenance phases?**
- **How does Continuous Integration (CI) support the SDLC?**
- **What testing levels exist in SDLC (unit, integration, system, UAT)?**
- **How do you decide what to automate vs test manually?**
- **What is regression testing and when is it required?**
- **What is release management, and how does it differ in Agile vs traditional SDLC?**
- **Explain blue-green deployment and canary releases. Where do they fit in SDLC?**
- **How do you handle production defects? Explain the lifecycle of a production bug.**
- **What KPIs or metrics do you track during maintenance (MTTR, defect density, SLA)?**
- **What are common risks in SDLC, and how do you mitigate them?**
- **How does documentation evolve across SDLC phases?**
- **Explain traceability matrix (RTM). Why is it important for audits and compliance?**
- **How does SDLC align with Agile ceremonies like sprint planning, review, and retrospectives?**
- **Explain "Shift Left" and "Shift Right" testing concepts in SDLC.**
- **A critical requirement was missed and discovered during UAT. How would you handle it?**
- **How do you manage SDLC in a project involving multiple vendors and distributed teams?**

- **How does IntelliJ simplify Git operations?**
- **Difference between commit from CLI vs IDE?**
- **How do you resolve conflicts using IntelliJ?**

3. Core Java

- **What is the difference between Primitive vs Reference types**
- **What is the efault value of variables int and double**
- **Explain difference between Local vs instance vs static variables**

- Difference between == and `equals()`
- What are Short-circuit operators
- What is the Operator precedence in Java
- When to use `switch` vs `if-else`
- What are Infinite loops and why does it happens
- Difference between `while` and `do-while`
- Why String is immutable?
- What is String vs StringBuilder

- Explain Array vs ArrayList with example
- What are Wrapper classes and why wrapper classes exist?
- What is Boxing and Unboxing
- Autoboxing performance impact
- What do we mean by `NullPointerException` risk in unboxing
- **What is the Java Collections Framework? What are its main components?**
- **Difference between `Collection` and `Collections` in Java?**
- **Explain the difference between `List`, `Set`, and `Map`.**
- **Why is `Map` not a subtype of `Collection`?**
- **How does `Vector` differ from `ArrayList`? Is `Vector` still recommended?**
- **What is the difference between `Iterator` and `ListIterator`?**
- **Difference between `HashSet`, `LinkedHashSet`, and `TreeSet`.**
- **How does `HashSet` ensure uniqueness of elements?**
- **What happens if `equals()` and `hashCode()` are not overridden properly?**
- **Difference between `HashMap`, `LinkedHashMap`, and `TreeMap`.**
- **Difference between `HashMap` and `Hashtable`.**
- **Can a `HashMap` store `null` keys and values?**
- **What are fail-fast and fail-safe iterators?**
- **How does `ConcurrentHashMap` work internally?**
- **What is the difference between `Comparable` and `Comparator`?**
- **Which collection would you use for fast search and why?**
- **How would you remove duplicate elements from a list?**
- **How do you sort a list of custom objects?**
- **Difference between checked and unchecked exceptions?**
- **Explain the exception hierarchy in Java.**
- **Can we have multiple `catch` blocks? What is the order?**
- **Is `finally` always executed? When is it not?**
- **Can we write `try` without `catch`?**
- **Difference between `throw` and `throws` keyword?**
- **Can we throw multiple exceptions from a method? How?**
- **How do you create a custom exception? When should you use one?**
- **Should custom exceptions be checked or unchecked? Why?**
- **What is exception chaining?**
- **Difference between `printStackTrace()` and logging an exception?**
- **Why should exceptions not be used for normal program flow?**
- **How would you handle exceptions in a multi-layer application?**
- **What happens if an exception is thrown in the `catch` block itself?**
- **How do you handle resources to avoid memory leaks?**
- **What exception is thrown when modifying a collection while iterating?**
- **What is `NullPointerException` in collections and how can it be avoided?**
- **How does `Optional` help reduce exceptions?**

4. OOPS (OOPS, Collections, Exceptions, Framework features, Streaming and Threading)

- **What is a class and what is an object in Java?**

- **How does object creation work internally using the new keyword?**
- Can a class exist without objects? Give real-world examples.
- **Explain `public`, `protected`, `default`, and `private` access modifiers.**
- **Difference between `protected` and default access?**
- **Can a class be declared `private` or `protected`? Why or why not?**
- **What is the purpose of static variables and methods?**
- **Why can't a static method access non-static members directly?**
- **Explain static blocks and their use cases.**
- **What is inheritance and how is it implemented in Java?**
- **Why does Java not support multiple inheritance using classes?**
- **Explain method overriding rules in Java.**
- **What is the role of the `super` keyword?**
- **What is polymorphism? Difference between compile-time and runtime polymorphism.**
- **How does dynamic method dispatch work in Java?**
- **Can static methods be overridden? Why?**
- **What is abstraction? How is it achieved using abstract classes and interfaces?**
- **When would you choose an abstract class over an interface?**
- **Can an abstract class have constructors? Why?**
- **What is encapsulation and how is it implemented in Java?**
- **Why should instance variables be private?**
- **How does encapsulation improve maintainability and security?**
- **What is a constructor? How is it different from a method?**
- **Explain constructor overloading and chaining.**
- **Does Java have destructors? How does garbage collection work?**
- **Difference between composition and inheritance.**
- **Why is composition often preferred over inheritance?**
- **Give a real-world example where composition is better than inheritance.**
- **What is an interface? How is it different from an abstract class?**
- **What are default and static methods in interfaces?**
- **What are packages and why are they important?**
- **How does Java resolve naming conflicts using packages?**
- **What methods are defined in the `Object` class?**
- **Why must `hashCode()` and `equals()` be overridden together?**
- **How is `toString()` useful during debugging and logging?**
- **What happens if `equals()` is overridden but `hashCode()` is not?**
- **Design a system using interfaces for loose coupling.**
- **Explain a real scenario where polymorphism improved extensibility.**
- **How would you design an immutable class?**
- **Where would you use final keyword with classes, methods, and variables?**
- **What is a lambda expression? Why were lambdas introduced in Java 8?**
- **Explain the syntax of a lambda expression. What are its limitations?**
- **How do lambda expressions improve code readability and performance?**
- **Can lambda expressions access local variables? What is "effectively final"?**
- **What is a functional interface? What is the role of @FunctionalInterface?**

- **Explain the commonly used functional interfaces:**
  - Predicate<T>
  - Function<T, R>
  - Consumer<T>
  - Supplier<T>
- **Difference between Function, BiFunction, and UnaryOperator.**
- **How does Predicate differ from Function? When would you use each?**
- **What is a method reference? How is it different from a lambda expression?**
- **Explain different types of method references with examples.**
  - Static method reference
  - Instance method reference
  - Constructor reference
- **When would you prefer a method reference over a lambda?**
- **What is function composition? Explain `andThen()` and `compose()`.**
- **Difference between `andThen()` and `compose()` with execution order.**
- **Can `Predicate` be composed? How (`and`, `or`, `negate`)?**
- **What is a Stream in Java? How is it different from a Collection?**
- **Are streams reusable? Why or why not?**
- **Explain intermediate vs terminal operations in streams.**
- **What is lazy evaluation in Stream API?**
- **Explain `map()` vs `flatMap()` with real examples.**
- **How does `filter()` work internally?**
- **What is the purpose of `sorted()`? How do you sort custom objects?**
- **Difference between `limit()` and `distinct()` operations.**
- **What is the `reduce()` operation? Explain its use cases.**
- **Difference between `reduce()` and `collect()`?**
- **How does `reduce()` behave in parallel streams?**
- **Difference between `forEach()` and `forEachOrdered()`?**
- **Explain `count()`, `anyMatch()`, `allMatch()`, and `noneMatch()`.**
- **What happens if a terminal operation is not called on a stream?**
- **Explain `Collectors.toList()` vs `Collectors.toSet()`.**
- **How does `Collectors.joining()` work? Provide a use case.**
- **Explain `groupingBy()` with an example.**
- **Difference between `groupingBy()` and `partitioningBy()`.**
- **How would you perform multi-level grouping using streams?**
- **What are parallel streams? When should they be avoided?**
- **How does Stream API handle immutability and thread safety?**
- **Common performance pitfalls while using streams.**
- **Convert a nested list into a flat list using streams.**
- **Group employees by department and count them using streams.**
- **Find the second highest number using Stream API.**
- **Explain a real project use case where Stream API simplified logic.**
- **Explain the complete thread lifecycle in Java (New → Runnable → Running → Blocked/Waiting → Terminated).**
- **What causes a thread to move from Runnable to Waiting or Blocked state?**
- **Can a thread be restarted once it reaches the Terminated state? Why?**

- **Difference between extending `Thread` and implementing `Runnable`. Which is preferred and why?**
- **Why does Java support only single inheritance but still allow multiple threads using `Runnable`?**
- **Explain different ways to create threads in Java.**
- **What happens internally when `start()` is called instead of `run()`?**
- **What problem does `ExecutorService` solve compared to manual thread management?**
- **Explain different types of thread pools provided by Executors.**
- **How does `Executors.newFixedThreadPool()` work internally?**
- **Difference between `execute()` and `submit()` methods.**
- **How do you retrieve results from a thread using `Future.get()`?**
- **What happens if `Future.get()` is called before task completion?**
- **Difference between `Runnable` and `Callable`.**
- **What is `FutureTask` and when would you use it?**
- **Can a `Callable` throw checked exceptions? How are they handled?**
- **What is synchronization and why is it required in multithreaded programs?**
- **Difference between synchronized method and synchronized block.**
- **What is intrinsic lock (monitor lock) in Java?**
- **Explain `wait()`, `notify()`, and `notifyAll()` methods.**
- **Why must `wait()` and `notify()` be called inside synchronized blocks?**
- **Difference between `sleep()` and `wait()`?**
- **What are thread-safe collections in Java? Give examples.**
- **Difference between `Collections.synchronizedList()` and `CopyOnWriteArrayList`.**
- **Why is `ConcurrentHashMap` preferred over `HashMap` in multithreaded environments?**
- **What issues arise when multiple threads modify a collection concurrently?**
- **What is a fail-fast iterator? Which exception is thrown?**
- **How do concurrent collections avoid `ConcurrentModificationException`?**
- **What is deadlock? Explain with an example.**
- **What are the four necessary conditions for deadlock?**
- **How can deadlocks be prevented or detected?**
- **Design a producer-consumer problem using `wait()` and `notify()`.**
- **How would you limit the number of concurrent API calls in an application?**
- **Explain a real project scenario where thread pool improved performance.**
- **How would you debug a deadlock in production?**
-