Java & Spring Boot Basics Interview Q&A; (Full 51 + 52)

Java Basics Q&A; (51)

1. What is Java? - High-level, object-oriented, platform-independent language.
2. Features of Java? - OOP, platform independent, robust, secure, multithreaded.
3. JVM vs JRE vs JDK? - JVM runs bytecode, JRE = JVM + libraries, JDK = JRE + tools.
4. == vs equals()? - == compares references, equals() compares content.
5. OOP Concepts? - Encapsulation, Inheritance, Polymorphism, Abstraction.
6. Encapsulation example? - Wrapping data & methods in class.
7. Inheritance example? - Dog extends Animal.
8. Polymorphism example? - Method overloading & overriding.
9. Abstraction? - Hiding implementation details, using abstract classes/interfaces.
10. Abstract class vs Interface? - Abstract class can have impl, interface pure contract (Java 8+ default methods).
11. Constructor? - Initializes object state.
12. Overloading vs Overriding? - Overloading diff params, Overriding modifies parent method.
13. final vs finally vs finalize? - final keyword, finally block, finalize() before GC.
14. Access Modifiers? - public, protected, default, private.
15. static keyword? - Shared across objects for vars/methods.
16. Array vs ArrayList? - Array fixed size, ArrayList dynamic.
17. StringBuilder vs StringBuffer? - Builder fast, Buffer thread-safe.
18. throw vs throws? - throw raises exception, throws in method declaration.
19. Multithreading? - Concurrent execution of multiple threads.
20. Runnable vs Thread? - Runnable preferred for flexibility.
21. Garbage Collection? - Automatic memory cleanup.
22. synchronized? - Lock access in multithreading.
23. Checked vs Unchecked Exception? - Checked compile-time, unchecked runtime.
24. Package? - Group of classes/interfaces.
25. super keyword? - Access parent class.
26. this keyword? - Refers to current object.
27. Shallow vs Deep Copy? - Shallow copies ref, Deep copies object.
28. Singleton class? - One instance only.
29. transient keyword? - Skip serialization.
30. volatile keyword? - Ensure visibility across threads.
31. HashMap vs Hashtable? - HashMap non-synced, Hashtable synced.
32. LinkedHashMap? - Maintains insertion order.
33. HashSet vs TreeSet? - HashSet unordered, TreeSet sorted.

34. Enum? - Constants in class-like structure.
35. Optional? - Avoid null pointer exceptions.
36. Functional Interface? - One abstract method.
37. Stream API vs Collections? - Collections store, Streams process.
38. Lambda? - Short impl of functional interface.
39. Default methods in Interface? - Impl inside interface (Java 8).
40. Predicate? - Functional interface returning boolean.
41. Method reference? - System.out::println style.
42. ArrayDeque vs LinkedList? - ArrayDeque faster.
43. var keyword? - Type inference (Java 10).
44. Comparable vs Comparator? - Natural vs custom order.
45. try-with-resources? - Auto-close resources.
46. Reflection? - Inspect classes at runtime.
47. JIT vs JVM? - JIT optimizes, JVM runs bytecode.
48. ClassLoader? - Loads classes at runtime.
49. Immutability? - Object state unchangeable (String).
50. String vs StringBuilder vs StringBuffer? - Immutable vs mutable.
51. Stack vs Heap Memory? - Stack local vars, Heap objects.

Spring Boot Basics Q&A; (52)

1. What is Spring Boot? - Build production-ready Spring apps easily.
2. Spring vs Spring Boot? - Spring XML-heavy, Boot auto-config.
3. @SpringBootApplication? - Combines @Configuration, @EnableAutoConfiguration, @ComponentScan.
4. application.properties? - Config file.
5. Change server port? - server.port=9090.
6. Starters? - Pre-configured dependencies.
7. @RestController? - @Controller + @ResponseBody.
8. @Controller vs @RestController? - Views vs JSON.
9. @Autowired? - Inject dependencies.
10. Dependency Injection? - Spring manages object creation.
11. Default embedded server? - Tomcat.
12. @Component? - Generic bean.
13. @Component vs @Service vs @Repository? - General, business logic, DAO.
14. @Configuration? - Source of bean defs.
15. @Bean? - Declares bean.
16. @Value? - Injects property value.
17. @PathVariable? - Extracts from URI.
18. @RequestParam? - Extracts query param.

19. @RequestBody? - Bind JSON to object.
20. @RequestBody vs @ResponseBody? - Request vs Response.
21. Exception Handling? - @ControllerAdvice + @ExceptionHandler.
22. application.yml? - Alternative config format.
23. Spring Data JPA? - Simplifies DB operations.
24. @Entity? - Marks DB entity.
25. @Id and @GeneratedValue? - Primary key and auto-gen.
26. @EnableAutoConfiguration? - Enables auto config.
27. Actuator? - Monitoring endpoints.
28. Swagger? - API docs.
29. CommandLineRunner? - Runs after startup.
30. @ComponentScan vs @EnableAutoConfiguration? - Bean scan vs auto config.
31. Profiles? - Dev/test/prod configs.
32. DevTools? - Auto-reload.
33. Mono vs Flux? - Mono = 0/1, Flux = many.
34. RestTemplate? - Sync API client.
35. WebClient? - Reactive client.
36. Security? - Add Spring Security.
37. JWT? - Stateless auth.
38. Session vs JWT? - Session server-side, JWT client-side.
39. @Scheduled? - Run tasks on schedule.
40. @EnableScheduling? - Enable scheduling.
41. Caching? - Use @Cacheable etc.
42. spring-boot-starter-test? - JUnit, Mockito, etc.
43. @SpringBootTest? - Integration test.
44. @MockBean? - Mock object for tests.
45. @DataJpaTest? - JPA repository tests.
46. Unit vs Integration test? - Single vs flow.
47. Flyway/Liquibase? - DB migration.
48. PUT vs PATCH? - PUT replaces, PATCH partial.
49. Run app? - mvn spring-boot:run or java -jar.
50. SpringApplication.run vs SpringBootServletInitializer? - Standalone vs WAR deploy.
51. Logging? - Uses SLF4J + Logback.
52. @EnableJpaRepositories? - Enable JPA repo scan.