Java Fundamentals

1. What is Java and what are its key features?

Answer: Java is a high-level, object-oriented programming language that is platform-independent. Key features include:

- Platform independence (Write Once, Run Anywhere)
- Object-oriented programming
- Automatic memory management (Garbage Collection)
- Security
- Multithreading support
- Robust and reliable

2. Explain the difference between JVM, JRE, and JDK?

Answer:

- **JVM (Java Virtual Machine):** Runtime environment that executes Java bytecode
- **JRE (Java Runtime Environment):** JVM + libraries + runtime tools
- **JDK (Java Development Kit):** JRE + development tools (compiler, debugger, etc.)

3. How does Java achieve platform independence?

Answer: Java source code is compiled into bytecode, which runs on the JVM. Since each platform has its own JVM implementation, the same bytecode can run on different platforms.

4. What are the differences between primitive and reference data types?

Answer:

- **Primitive:** Basic data types (int, boolean, char) stored in stack memory
- **Reference:** Object references stored in heap memory, can be null

5. Explain method overloading vs method overriding?

Answer:

- **Overloading:** Same method name with different parameters (compile-time polymorphism)
- **Overriding:** Subclass redefines parent class method (runtime polymorphism)

6. What is the static keyword and its uses?

Answer: Static members belong to the class rather than instances. Can be accessed without creating objects. Only one copy exists in memory.

7. Explain different uses of the final keyword?

Answer:

- **final variable:** Cannot be reassigned
- **final method:** Cannot be overridden
- **final class:** Cannot be inherited (e.g., String, Integer)

8. What's the difference between abstract class and interface?

Answer:

- **Abstract class:** Can have concrete methods, constructors, single inheritance
- **Interface:** Java 8+ allows default methods, multiple inheritance, only public methods

9. What are packages and why are they used?

Answer: Packages group related classes, provide namespace, and control access. They organize code and prevent naming conflicts.

10. What are access modifiers in Java?

Answer:

- **public:** Accessible everywhere
- **protected:** Same package + subclasses
- **default:** Same package only
- **private:** Same class only

11. What is a constructor and its characteristics?

Answer: Special method called when object is created. Same name as class, no return type, can be overloaded.

12. Explain this and super keywords?

Answer:

- **this:** Reference to current object
- **super:** Reference to parent class

13. What is Garbage Collection and how does it work?

Answer: Automatic memory management that removes unreferenced objects. Uses generational collection with different heap areas.

14. Difference between == and equals() method?

Answer:

- **==:** Compares references (values for primitives)
- **equals():** Compares content/values

15. What is the hashCode() and equals() contract?

Answer: If two objects are equal (equals() returns true), they must have the same hashCode(). Used in hash-based collections.

16. What is autoboxing and unboxing?

Answer:

- **Autoboxing:** Automatic conversion of primitives to wrapper objects
- **Unboxing:** Automatic conversion of wrapper objects to primitives

17. Explain the concept of immutability in Java?

Answer: Immutable objects cannot be modified after creation. String is immutable. Benefits include thread safety and caching.

18. What are wrapper classes and their purpose?

Answer: Object representations of primitive types (Integer, Boolean, etc.). Used in collections, generics, and when objects are required.

19. Explain pass-by-value in Java?

Answer: Java passes copies of values. For objects, the reference value is copied, not the object itself.

20. What is the difference between StringBuffer and StringBuilder?

Answer:

- **StringBuffer:** Thread-safe, synchronized methods
- **StringBuilder:** Not thread-safe, better performance in single-threaded environments

21. What are enums and their advantages?

Answer: Type-safe constants. Advantages include type safety, namespace, and additional methods/constructors.

22. Explain instanceof operator?

Answer: Tests whether an object is an instance of a specific class or interface. Returns boolean.

23. What is method signature?

Answer: Method name + parameter types (not including return type or parameter names).

24. Explain variable arguments (varargs)?

Answer: Allows methods to accept variable number of arguments using ellipsis (…). Treated as arrays internally.

25. What is the difference between Comparable and Comparator?

Answer:

- **Comparable:** Natural ordering, compareTo() method, single sorting logic
- **Comparator:** Custom ordering, compare() method, multiple sorting strategies

Object-Oriented Programming

26. What are the four pillars of OOP?

Answer:

1. **Encapsulation:** Bundling data and methods, hiding internal details
2. **Inheritance:** Creating new classes based on existing ones
3. **Polymorphism:** One interface, multiple implementations
4. **Abstraction:** Hiding complex implementation details

27. What is encapsulation and how to achieve it?

Answer: Hiding internal state and requiring interaction through methods. Achieved using private fields and public getter/setter methods.

28. Explain inheritance and its types?

Answer: Mechanism where one class inherits properties from another. Types: single, multilevel, hierarchical (multiple inheritance through interfaces).

29. What is polymorphism and its types?

Answer:

- **Compile-time:** Method overloading, operator overloading
- **Runtime:** Method overriding, achieved through inheritance and interfaces

30. What is abstraction and how is it implemented?

Answer: Hiding complex implementation details. Implemented using abstract classes and interfaces.

31. Can we override static methods?

Answer: No, static methods belong to the class and are resolved at compile time. They can be hidden, not overridden.

32. Can we override private methods?

Answer: No, private methods are not visible to subclasses, so they cannot be overridden.

33. What is method hiding?

Answer: When a subclass defines a static method with the same signature as a static method in the parent class.

34. Explain covariant return types?

Answer: Subclass method can return a subtype of the return type declared in the parent class method.

35. What is composition vs inheritance?

Answer:

- **Composition:** "Has-a" relationship, more flexible, runtime behavior change
- **Inheritance:** "Is-a" relationship, compile-time binding, can lead to tight coupling

36. What is aggregation vs composition?

Answer:

- **Aggregation:** Weak relationship, parts can exist without whole
- **Composition:** Strong relationship, parts cannot exist without whole

37. Can a class be both abstract and final?

Answer: No, abstract classes are meant to be extended, while final classes cannot be extended.

38. What is multiple inheritance and why doesn't Java support it?

Answer: Class inheriting from multiple classes. Java doesn't support it to avoid diamond problem, but supports it through interfaces.

39. What are marker interfaces?

Answer: Interfaces with no methods, used to mark classes (e.g., Serializable, Cloneable).

40. What is the diamond problem?

Answer: Ambiguity that arises when a class inherits from two classes that have a common parent, leading to confusion about which parent's method to use.

String and Memory Management

41. Why is String immutable in Java?

Answer: For security, thread safety, caching (string pool), and performance optimizations.

42. What is String Pool?

Answer: Special memory area in heap where string literals are stored to optimize memory usage through string interning.

43. How to create strings in Java?

Answer:

- String literal: `String s = "Hello"` (uses string pool)
- Using new: `String s = new String("Hello")` (creates new object)

44. What is string interning?

Answer: Process of storing only one copy of each distinct string value in the string pool.

45. Difference between String, StringBuffer, and StringBuilder?

Answer:

- **String:** Immutable, thread-safe
- **StringBuffer:** Mutable, thread-safe, slower
- **StringBuilder:** Mutable, not thread-safe, faster

46. What are different memory areas in JVM?

Answer:

- **Heap:** Object storage, garbage collected
- **Stack:** Method calls, local variables
- **Method Area:** Class metadata, static variables
- **PC Register:** Current instruction pointer

47. What is heap memory structure?

Answer:

- **Young Generation:** Eden, Survivor spaces
- **Old Generation:** Long-lived objects
- **Permanent Generation:** (Java 7) or Metaspace (Java 8+)

48. What are different types of garbage collectors?

Answer:

- **Serial GC:** Single-threaded
- **Parallel GC:** Multi-threaded
- **CMS GC:** Concurrent Mark Sweep
- **G1 GC:** Low-latency collector

49. What is memory leak in Java?

Answer: When objects are no longer used but still referenced, preventing garbage collection.

50. How to avoid memory leaks?

Answer:

- Close resources properly
- Remove listeners
- Use weak references
- Avoid static collections
- Profile memory usage

Collections Framework

51. What is Java Collections Framework?

Answer: Unified architecture for representing and manipulating collections. Includes interfaces, implementations, and algorithms.

52. What's the difference between Collection and Collections?

Answer:

- **Collection:** Root interface for collections
- **Collections:** Utility class with static methods for collection operations

53. What are the main interfaces in Collections Framework?

Answer:

- **Collection:** Root interface
- **List:** Ordered collection, allows duplicates
- **Set:** No duplicates allowed
- **Queue:** FIFO operations
- **Map:** Key-value pairs

54. Difference between List, Set, and Map?

Answer:

- **List:** Ordered, allows duplicates, indexed access
- **Set:** No duplicates, unique elements
- **Map:** Key-value pairs, unique keys

55. When to use ArrayList vs LinkedList?

Answer:

- **ArrayList:** Random access, less memory overhead, better for read-heavy operations
- **LinkedList:** Better for frequent insertions/deletions, implements Deque

56. What is the difference between Vector and ArrayList?

Answer:

- **Vector:** Synchronized, thread-safe, legacy class
- **ArrayList:** Not synchronized, better performance, preferred choice

57. What is the difference between HashMap and Hashtable?

Answer:

- **HashMap:** Not synchronized, allows null keys/values, better performance
- **Hashtable:** Synchronized, thread-safe, no null keys/values

58. How does HashMap work internally?

Answer: Uses array of buckets, hash function determines bucket index, handles collisions using chaining (linked lists/trees).

59. What is the load factor in HashMap?

Answer: Ratio of number of entries to number of buckets. Default is 0.75, triggers resizing when exceeded.

60. What happens when two objects have the same hashcode?

Answer: Hash collision occurs, objects are stored in the same bucket using chaining (linked list or tree).

61. Difference between HashMap and ConcurrentHashMap?

Answer:

- **HashMap:** Not thread-safe
- **ConcurrentHashMap:** Thread-safe, uses segment locking, better performance than Hashtable

62. What is the difference between HashSet and TreeSet?

Answer:

- **HashSet:** Hash table, O(1) operations, no ordering
- **TreeSet:** Red-black tree, O(log n) operations, sorted

63. How is HashSet implemented internally?

Answer: Uses HashMap internally, stored elements as keys with dummy value.

64. What is LinkedHashMap?

Answer: HashMap that maintains insertion order using doubly-linked list.

65. What is WeakHashMap?

Answer: Map with weak references to keys, allows garbage collection of keys when no strong references exist.

66. Explain PriorityQueue?

Answer: Heap-based priority queue, elements ordered by natural ordering or custom comparator.

67. What is the difference between Iterator and ListIterator?

Answer:

- **Iterator:** Forward-only, works with all collections
- **ListIterator:** Bidirectional, only for List, supports modification

68. What is fail-fast and fail-safe iterators?

Answer:

- **Fail-fast:** Throws ConcurrentModificationException if collection is modified during iteration
- **Fail-safe:** Works on copy, doesn't throw exception

69. How to synchronize collections?

Answer: Using Collections.synchronizedXXX() methods or concurrent collections like ConcurrentHashMap.

70. What are concurrent collections?

Answer: Thread-safe collections designed for concurrent access (ConcurrentHashMap, ConcurrentLinkedQueue, etc.).

71. Difference between Comparable and Comparator interfaces?

Answer:

- **Comparable:** Natural ordering, single sorting logic, compareTo() method
- **Comparator:** Custom ordering, multiple sorting strategies, compare() method

72. What is the difference between Queue and Deque?

Answer:

- **Queue:** FIFO operations, add/remove from ends
- **Deque:** Double-ended queue, add/remove from both ends

73. How to reverse a List?

Answer: Using Collections.reverse() method or implementing custom logic.

74. How to sort a List?

Answer: Using Collections.sort() method with natural ordering or custom comparator.

75. What is the difference between Arrays.sort() and Collections.sort()?

Answer:

- **Arrays.sort():** For arrays, uses quicksort for primitives, mergesort for objects
- **Collections.sort():** For collections, converts to array, sorts, and copies back

Exception Handling

76. What is Exception Handling?

Answer: Mechanism to handle runtime errors gracefully, maintaining normal program flow.

77. What is the hierarchy of exceptions in Java?

Answer:

```
CopyThrowable
├── Error (OutOfMemoryError, StackOverflowError)
└── Exception
    ├── RuntimeException (unchecked)
    └── Other exceptions (checked)
```

78. What's the difference between checked and unchecked exceptions?

Answer:

- **Checked:** Compile-time checked, must be handled or declared (IOException, SQLException)
- **Unchecked:** Runtime exceptions, not required to handle (NullPointerException, ArrayIndexOutOfBoundsException)

79. What is the difference between Error and Exception?

Answer:

- **Error:** System-level problems, not recoverable (OutOfMemoryError)
- **Exception:** Application-level problems, can be handled

80. Explain try-catch-finally block?

Answer:

- **try:** Code that might throw exception
- **catch:** Exception handling code

- **finally:** Code that always executes, used for cleanup

81. Can finally block be skipped?

Answer: Finally block executes except when System.exit() is called or JVM crashes.

82. What is try-with-resources?

Answer: Automatic resource management introduced in Java 7, automatically closes resources that implement AutoCloseable.

83. Can we have multiple catch blocks?

Answer: Yes, but more specific exceptions should be caught before general ones.

84. What is the difference between throw and throws?

Answer:

- **throw:** Used to explicitly throw an exception
- **throws:** Used in method declaration to declare exceptions

85. Can we rethrow exceptions?

Answer: Yes, caught exceptions can be rethrown using throw keyword.

86. What is exception propagation?

Answer: When an exception is not handled in a method, it propagates up the call stack.

87. What are custom exceptions?

Answer: User-defined exceptions that extend Exception or RuntimeException classes.

88. What is the difference between ClassNotFoundException and NoClassDefFoundError?

Answer:

- **ClassNotFoundException:** Checked exception, class not found at runtime
- **NoClassDefFoundError:** Error, class was available at compile time but not at runtime

89. What is suppressed exception?

Answer: Exception that is suppressed when another exception is thrown in try-with-resources or finally block.

90. Best practices for exception handling?

Answer:

- Use specific exception types
- Don't ignore exceptions
- Clean up resources
- Log exceptions appropriately
- Don't use exceptions for control flow

Multithreading and Concurrency

91. What is multithreading?

Answer: Concurrent execution of multiple threads within a program, allowing better resource utilization.

92. What's the difference between process and thread?

Answer:

- **Process:** Independent execution environment with separate memory space
- **Thread:** Lightweight subprocess sharing memory space within a process

93. How to create threads in Java?

Answer:

1. Extending Thread class
2. Implementing Runnable interface
3. Using Callable and Future
4. Using ThreadPoolExecutor

94. What is the difference between Runnable and Callable?

Answer:

- **Runnable:** run() method, no return value, cannot throw checked exceptions
- **Callable:** call() method, returns value, can throw checked exceptions

95. What are thread states?

Answer:

- **NEW:** Thread created but not started
- **RUNNABLE:** Thread executing or ready to execute
- **BLOCKED:** Thread blocked waiting for monitor lock
- **WAITING:** Thread waiting indefinitely
- **TIMED_WAITING:** Thread waiting for specific time
- **TERMINATED:** Thread completed execution

96. What is synchronization?

Answer: Mechanism to control access to shared resources by multiple threads, preventing race conditions.

97. What are different ways to achieve synchronization?

Answer:

- synchronized methods
- synchronized blocks
- volatile keyword
- Locks (ReentrantLock, ReadWriteLock)
- Atomic classes

98. What is the synchronized keyword?

Answer: Provides mutual exclusion, ensuring only one thread can access synchronized code at a time.

99. What is the difference between synchronized method and synchronized block?

Answer:

- **Synchronized method:** Entire method is synchronized, uses object lock
- **Synchronized block:** Only specific code is synchronized, can use different locks

100. What is volatile keyword?

Answer: Ensures variable changes are visible to all threads, prevents compiler optimizations.

101. What is the difference between wait() and sleep()?

Answer:

- **wait():** Releases lock, object must be synchronized, woken by notify()
- **sleep():** Doesn't release lock, static method, woken by time or interrupt

102. What are notify() and notifyAll() methods?

Answer:

- **notify():** Wakes up one waiting thread
- **notifyAll():** Wakes up all waiting threads

103. What is deadlock?

Answer: Situation where two or more threads are permanently blocked, waiting for each other.

104. How to prevent deadlock?

Answer:

- Avoid nested locks
- Use timeout for lock acquisition
- Order locks consistently
- Use deadlock detection tools

105. What is thread pool?

Answer: Reusable collection of threads that can execute tasks, improving performance by avoiding thread creation overhead.

106. What are different types of thread pools?

Answer:

- **Fixed Thread Pool:** Fixed number of threads
- **Cached Thread Pool:** Creates threads as needed
- **Single Thread Pool:** Single thread for sequential execution
- **Scheduled Thread Pool:** For scheduled tasks

107. What is ExecutorService?

Answer: Higher-level API for managing threads, provides methods for submitting tasks and managing their lifecycle.

108. What is the difference between submit() and execute()?

Answer:

- **execute():** Takes Runnable, no return value
- **submit():** Takes Runnable/Callable, returns Future

109. What is Future and CompletableFuture?

Answer:

- **Future:** Represents result of asynchronous computation
- **CompletableFuture:** Enhanced Future with more features and better composability

110. What are atomic classes?

Answer: Thread-safe classes that support atomic operations without synchronization (AtomicInteger, AtomicBoolean, etc.).

111. What is the difference between ConcurrentHashMap and HashMap?

Answer:

- **HashMap:** Not thread-safe, better performance in single-threaded environment
- **ConcurrentHashMap:** Thread-safe, uses segment locking, better than Hashtable

112. What is ThreadLocal?

Answer: Provides thread-local variables, each thread has its own copy of the variable.

113. What is the producer-consumer problem?

Answer: Classic synchronization problem where producers generate data and consumers process it, requiring coordination.

114. What is CountDownLatch?

Answer: Synchronization aid that allows threads to wait until a set of operations completes.

115. What is CyclicBarrier?

Answer: Synchronization aid where threads wait for each other to reach a common barrier point.

Java 8+ Features

116. What are lambda expressions?

Answer: Anonymous functions that can be used to implement functional interfaces concisely.

117. What are functional interfaces?

Answer: Interfaces with exactly one abstract method, can be implemented using lambda expressions.

118. What is the Stream API?

Answer: API for processing collections in a functional style, supporting operations like filter, map, reduce.

119. What's the difference between intermediate and terminal operations?

Answer:

- **Intermediate:** Return streams, lazy evaluation (filter, map, sorted)
- **Terminal:** Return non-stream results, trigger evaluation (collect, forEach, reduce)

120. What are method references?

Answer: Shorthand for lambda expressions that call existing methods (::operator).

121. What are default methods in interfaces?

Answer: Methods with implementation in interfaces, allowing interface evolution without breaking existing implementations.

122. What are static methods in interfaces?

Answer: Utility methods that belong to the interface, called using interface name.

123. What is Optional class?

Answer: Container that may or may not contain a value, helps avoid NullPointerException.

124. What are the new Date and Time APIs?

Answer: java.time package with LocalDate, LocalTime, LocalDateTime, ZonedDateTime for better date/time handling.

125. What is the forEach() method?

Answer: Method to iterate over collections using lambda expressions or method references.

126. What are collectors?

Answer: Utility class providing common reduction operations for streams (toList, toSet, groupingBy).

127. What is parallel stream?

Answer: Stream that can be processed in parallel using multiple threads for better performance.

128. What is the difference between map() and flatMap()?

Answer:

- **map():** Transforms each element to another object
- **flatMap():** Transforms each element to a stream and flattens the result

129. What are predicates?

Answer: Functional interface representing boolean-valued functions, commonly used in filtering.

130. What is the difference between Collection.stream() and Collection.parallelStream()?

Answer:

- **stream():** Sequential processing
- **parallelStream():** Parallel processing using multiple threads

Design Patterns

131. What are design patterns?

Answer: Reusable solutions to common problems in software design, providing templates for solving recurring design problems.

132. What is Singleton pattern?

Answer: Ensures only one instance of a class exists and provides global access to it.

133. How to implement thread-safe Singleton?

Answer: Using synchronized methods, double-checked locking, or enum-based implementation.

134. What is Factory pattern?

Answer: Creates objects without specifying exact classes, using a factory method to decide which class to instantiate.

135. What is Abstract Factory pattern?

Answer: Provides interface for creating families of related objects without specifying their concrete classes.

136. What is Builder pattern?

Answer: Constructs complex objects step by step, separating construction from representation.

137. What is Observer pattern?

Answer: Defines one-to-many dependency between objects, notifying dependents when subject changes.

138. What is Strategy pattern?

Answer: Defines family of algorithms, encapsulates each one, and makes them interchangeable.

139. What is Command pattern?

Answer: Encapsulates requests as objects, allowing parameterization and queuing of requests.

140. What is Template Method pattern?

Answer: Defines skeleton of algorithm in base class, letting subclasses override specific steps.

141. What is Adapter pattern?

Answer: Allows incompatible interfaces to work together by wrapping existing class with new interface.

142. What is Decorator pattern?

Answer: Adds new functionality to objects dynamically without altering their structure.

143. What is Facade pattern?

Answer: Provides simplified interface to complex subsystem, hiding its complexity.

144. What is Proxy pattern?

Answer: Provides placeholder or surrogate for another object to control access to it.

145. What is MVC pattern?

Answer: Separates application into Model (data), View (presentation), and Controller (logic) components.

Spring Framework

146. What is Spring Framework?

Answer: Comprehensive framework for enterprise Java development, providing infrastructure support for developing Java applications.

147. What are the main features of Spring?

Answer:

- Dependency Injection
- Aspect-Oriented Programming
- Transaction Management

- MVC Framework
- Data Access Integration

148. What is Dependency Injection?

Answer: Design pattern where dependencies are injected into objects rather than created by them, promoting loose coupling.

149. What are different types of dependency injection?

Answer:

- Constructor Injection
- Setter Injection
- Field Injection

150. What is Inversion of Control (IoC)?

Answer: Principle where control of object creation and lifecycle is transferred from application code to framework.

151. What is Spring IoC Container?

Answer: Core of Spring framework that manages object creation, configuration, and lifecycle.

152. What's the difference between BeanFactory and ApplicationContext?

Answer:

- **BeanFactory:** Basic container, lazy initialization
- **ApplicationContext:** Advanced container, eager initialization, additional features

153. What are Spring Bean scopes?

Answer:

- **Singleton:** Single instance per container
- **Prototype:** New instance per request
- **Request:** Per HTTP request
- **Session:** Per HTTP session

- **Global Session:** Per global HTTP session

154. What is Spring Boot?

Answer: Opinionated framework that simplifies Spring application development with auto-configuration and embedded servers.

155. What are Spring Boot starters?

Answer: Pre-configured dependency descriptors that simplify Maven/Gradle configuration.

156. What is auto-configuration in Spring Boot?

Answer: Feature that automatically configures Spring application based on dependencies in classpath.

157. What is Spring MVC?

Answer: Web framework based on Model-View-Controller pattern for building web applications.

158. What is DispatcherServlet?

Answer: Front controller that handles all HTTP requests and dispatches them to appropriate handlers.

159. What are Spring annotations?

Answer: Metadata that provides configuration information:

- @Component, @Service, @Repository, @Controller
- @Autowired, @Qualifier, @Value
- @RequestMapping, @GetMapping, @PostMapping

160. What is Aspect-Oriented Programming (AOP)?

Answer: Programming paradigm that allows separation of cross-cutting concerns like logging, security, transactions.

161. What are Spring profiles?

Answer: Way to segregate application configuration for different environments (dev, test, prod).

162. What is Spring Data JPA?

Answer: Framework that simplifies data access layer implementation using JPA.

163. What is Spring Security?

Answer: Comprehensive security framework providing authentication and authorization capabilities.

164. What is Spring Transaction Management?

Answer: Framework for managing database transactions declaratively or programmatically.

165. What is @Transactional annotation?

Answer: Declarative transaction management annotation that handles transaction boundaries automatically.

Database and JPA

166. What is JPA?

Answer: Java Persistence API — specification for managing relational data using object-relational mapping.

167. What is Hibernate?

Answer: Popular JPA implementation providing ORM capabilities for Java applications.

168. What is the difference between JPA and Hibernate?

Answer:

- **JPA:** Specification/interface
- **Hibernate:** Implementation of JPA specification

169. What is Entity in JPA?

Answer: Plain Java object that represents a table in database, annotated with @Entity.

170. What are JPA annotations?

Answer:

- @Entity, @Table, @Id, @GeneratedValue
- @Column, @JoinColumn, @OneToMany, @ManyToOne
- @ManyToMany, @OneToOne

171. What is EntityManager?

Answer: Interface for interacting with persistence context, managing entity lifecycle.

172. What is persistence context?

Answer: Set of managed entity instances associated with EntityManager.

173. What are entity states in JPA?

Answer:

- **New/Transient:** Not associated with persistence context
- **Managed/Persistent:** Associated with persistence context
- **Detached:** Previously managed but no longer associated
- **Removed:** Marked for deletion

174. What is the difference between save() and persist()?

Answer:

- **save():** Hibernate method, returns generated ID
- **persist():** JPA method, void return type

175. What is lazy loading vs eager loading?

Answer:

- **Lazy:** Data loaded on demand
- **Eager:** Data loaded immediately with parent entity

176. What is N+1 problem?

Answer: Performance issue where one query for main entities triggers N additional queries for related entities.

177. How to solve N+1 problem?

Answer:

- Use JOIN FETCH queries
- Use @BatchSize annotation
- Use entity graphs
- Use proper fetch strategies

178. What is JPQL?

Answer: Java Persistence Query Language — object-oriented query language for JPA entities.

179. What is Criteria API?

Answer: Programmatic way to create type-safe queries using Java code instead of strings.

180. What is the difference between JPQL and SQL?

Answer:

- **JPQL:** Object-oriented, works with entities
- **SQL:** Relational, works with tables

181. What are named queries?

Answer: Pre-defined queries with names, defined using @NamedQuery annotation.

182. What is connection pooling?

Answer: Technique to reuse database connections, improving performance by avoiding connection creation overhead.

183. What is transaction isolation?

Answer: Property that defines how transaction changes are visible to other transactions.

184. What are different isolation levels?

Answer:

- **READ_UNCOMMITTED:** Lowest isolation, allows dirty reads
- **READ_COMMITTED:** Prevents dirty reads
- **REPEATABLE_READ:** Prevents dirty and non-repeatable reads

- **SERIALIZABLE:** Highest isolation, prevents all phenomena

185. What is optimistic vs pessimistic locking?

Answer:

- **Optimistic:** Assumes conflicts are rare, checks at commit time
- **Pessimistic:** Assumes conflicts are common, locks immediately

Testing

186. What is unit testing?

Answer: Testing individual components or modules in isolation to ensure they work correctly.

187. What is JUnit?

Answer: Popular Java testing framework for writing and running unit tests with annotations like @Test, @Before, @After.

188. What are JUnit annotations?

Answer:

- **@Test:** Marks test method
- **@Before/@BeforeEach:** Runs before each test
- **@After/@AfterEach:** Runs after each test
- **@BeforeClass/@BeforeAll:** Runs once before all tests
- **@AfterClass/@AfterAll:** Runs once after all tests

189. What is Mockito?

Answer: Mocking framework for creating mock objects in unit tests, allowing isolation of code under test.

190. What's the difference between @Mock and @Spy?

Answer:

- **@Mock:** Creates mock object, all methods return default values
- **@Spy:** Creates spy object, real methods are called unless stubbed

191. What is integration testing?

Answer: Testing interaction between integrated components or systems to detect interface defects.

192. What is TestNG?

Answer: Testing framework inspired by JUnit with additional features like parallel execution, data providers, and flexible test configuration.

193. What is the difference between JUnit and TestNG?

Answer:

- **JUnit:** Simpler, annotation-based, good for unit testing
- **TestNG:** More features, parallel execution, better for integration testing

194. What is test-driven development (TDD)?

Answer: Development approach where tests are written before code, following Red-Green-Refactor cycle.

195. What is behavior-driven development (BDD)?

Answer: Extension of TDD focusing on behavior specification using natural language constructs.

196. What are test doubles?

Answer: Objects that replace real dependencies in tests: Mock, Stub, Spy, Fake, Dummy.

197. What is code coverage?

Answer: Metric measuring percentage of code executed during testing, includes line, branch, and method coverage.

198. What is parameterized testing?

Answer: Running same test with different input parameters, supported by @ParameterizedTest in JUnit 5.

199. What is @SpringBootTest?

Answer: Annotation for integration testing in Spring Boot applications, loads full application context.

200. What is @WebMvcTest?

Answer: Annotation for testing Spring MVC controllers in isolation, loads only web layer components.

Performance and Optimization

201. How to optimize Java application performance?

Answer:

- Use appropriate data structures
- Optimize algorithms
- Profile and identify bottlenecks
- Minimize object creation
- Use connection pooling
- Implement caching

202. What is JVM tuning?

Answer: Adjusting JVM parameters to optimize performance, including heap size, garbage collection settings, and compiler options.

203. What are important JVM parameters?

Answer:

- **-Xms:** Initial heap size
- **-Xmx:** Maximum heap size
- **-XX:NewRatio:** Ratio of young to old generation
- **-XX:+UseG1GC:** Use G1 garbage collector

204. What is profiling?

Answer: Process of analyzing application performance to identify bottlenecks and optimization opportunities.

205. What are popular Java profilers?

Answer:

- **JProfiler:** Commercial profiler with GUI
- **YourKit:** Commercial profiler
- **VisualVM:** Free profiler included with JDK

- **JConsole:** Basic monitoring tool

206. What is caching and its types?

Answer:

- **Application-level:** In-memory caching (HashMap, Ehcache)
- **Database-level:** Query result caching
- **Distributed:** Redis, Hazelcast
- **Browser:** HTTP caching

207. What is the difference between Ehcache and Redis?

Answer:

- **Ehcache:** In-process cache, faster access, limited to single JVM
- **Redis:** Distributed cache, network overhead, shared across applications

208. What is database connection pooling?

Answer: Technique to reuse database connections, reducing connection creation overhead and improving performance.

209. What are popular connection pool libraries?

Answer:

- **HikariCP:** High-performance connection pool
- **Apache DBCP:** Commons Database Connection Pool
- **C3P0:** Mature connection pooling library

210. What is lazy loading and when to use it?

Answer: Loading data on demand rather than upfront, useful for large datasets or expensive operations.

211. What is batch processing?

Answer: Processing multiple records together instead of one by one, improving performance for large datasets.

212. What are microservices performance considerations?

Answer:

- Service communication overhead
- Circuit breaker patterns
- Load balancing
- Caching strategies
- Database per service

213. What is the difference between vertical and horizontal scaling?

Answer:

- **Vertical:** Adding more power to existing machine
- **Horizontal:** Adding more machines to pool of resources

214. What is load balancing?

Answer: Distributing incoming requests across multiple servers to prevent overload and ensure high availability.

215. What are different load balancing algorithms?

Answer:

- **Round Robin:** Sequential distribution
- **Least Connections:** Route to server with fewest connections
- **Weighted:** Based on server capacity
- **IP Hash:** Based on client IP

Advanced Java Concepts

216. What is reflection in Java?

Answer: Ability to inspect and modify runtime behavior of applications, accessing classes, methods, and fields dynamically.

217. What are the uses of reflection?

Answer:

- Frameworks (Spring, Hibernate)
- Testing frameworks
- IDE features
- Serialization
- Dynamic proxy creation

218. What are annotations in Java?

Answer: Metadata that provides information about code, processed at compile time or runtime.

219. How to create custom annotations?

Answer: Using @interface keyword with retention policy and target specification.

220. What is generics in Java?

Answer: Feature allowing type parameterization, providing type safety and eliminating casting.

221. What are wildcards in generics?

Answer:

- **? extends T:** Upper bounded wildcard
- **? super T:** Lower bounded wildcard
- **?:** Unbounded wildcard

222. What is type erasure?

Answer: Process where generic type information is removed at runtime, ensuring backward compatibility.

223. What is serialization?

Answer: Process of converting object into byte stream for storage or transmission.

224. What is the difference between Serializable and Externalizable?

Answer:

- **Serializable:** Automatic serialization, uses default mechanism
- **Externalizable:** Custom serialization, more control over process

225. What is serialVersionUID?

Answer: Unique identifier for serializable classes, used for version control during deserialization.

226. What are transient and volatile keywords?

Answer:

- **transient:** Excludes field from serialization
- **volatile:** Ensures variable changes are visible to all threads

227. What is ClassLoader?

Answer: Component that loads classes into JVM, follows delegation model with hierarchy.

228. What are different types of ClassLoaders?

Answer:

- **Bootstrap:** Loads core Java classes
- **Extension:** Loads extension classes
- **System/Application:** Loads application classes

229. What is dynamic proxy?

Answer: Runtime creation of proxy classes implementing interfaces, used in AOP and frameworks.

230. What is the difference between deep copy and shallow copy?

Answer:

- **Shallow:** Copies object references, shared nested objects

- **Deep:** Copies entire object graph, independent nested objects

231. What is immutable class and how to create one?

Answer: Class whose state cannot be changed after creation. Requirements:

- Make class final
- Make fields private and final
- No setter methods
- Return defensive copies

232. What is the Builder pattern implementation?

Answer: Pattern for constructing complex objects step by step, often used with immutable classes.

233. What is method chaining?

Answer: Technique where methods return the same object, allowing multiple method calls in sequence.

234. What is the difference between composition and inheritance?

Answer:

- **Composition:** "Has-a" relationship, more flexible
- **Inheritance:** "Is-a" relationship, tighter coupling

235. What is covariance and contravariance?

Answer:

- **Covariance:** Subtype can be used where supertype is expected
- **Contravariance:** Supertype can be used where subtype is expected

Spring Boot Advanced

236. What is Spring Boot Actuator?

Answer: Production-ready features for monitoring and managing Spring Boot applications.

237. What are Actuator endpoints?

Answer:

- **/health:** Application health information
- **/metrics:** Application metrics
- **/info:** Application information
- **/env:** Environment properties

238. What is Spring Boot configuration?

Answer: Externalized configuration using properties files, YAML, environment variables, or command-line arguments.

239. What is @ConfigurationProperties?

Answer: Annotation for type-safe configuration properties binding to POJOs.

240. What is Spring Boot DevTools?

Answer: Development-time tools providing automatic restart, live reload, and enhanced development experience.

241. What is embedded server in Spring Boot?

Answer: Web server (Tomcat, Jetty, Undertow) embedded in application JAR, eliminating need for external server.

242. How to create custom starter in Spring Boot?

Answer: Create library with auto-configuration, dependencies, and properties, following naming convention.

243. What is @EnableAutoConfiguration?

Answer: Annotation that enables Spring Boot's auto-configuration mechanism based on classpath dependencies.

244. What is Spring Boot testing?

Answer: Testing support with annotations like @SpringBootTest, @WebMvcTest, @DataJpaTest for different testing scenarios.

245. What is @MockBean?

Answer: Annotation for creating mock beans in Spring application context for testing.

246. What is Spring Boot security configuration?

Answer: Security configuration using Spring Security with auto-configuration and customization options.

247. What is CORS in Spring Boot?

Answer: Cross-Origin Resource Sharing configuration for allowing requests from different domains.

248. What is Spring Boot logging?

Answer: Logging configuration with support for Logback, Log4j2, and Java Util Logging.

249. What is externalized configuration?

Answer: Configuration stored outside application code, supporting different environments and deployment scenarios.

250. What is Spring Boot packaging?

Answer: Creating executable JAR files with embedded server and dependencies included.

Microservices and Architecture

251. What are microservices?

Answer: Architectural style structuring application as collection of small, independent services communicating over network.

252. What are advantages of microservices?

Answer:

- Independent deployment
- Technology diversity
- Scalability
- Fault isolation
- Team autonomy

253. What are challenges of microservices?

Answer:

- Service communication
- Data consistency
- Network latency
- Distributed debugging
- Operational complexity

254. What is service discovery?

Answer: Mechanism for services to find and communicate with each other dynamically.

255. What is API Gateway?

Answer: Single entry point for all client requests, handling routing, authentication, and cross-cutting concerns.

256. What is Circuit Breaker pattern?

Answer: Pattern preventing cascading failures by stopping calls to failing services and providing fallback responses.

257. What is distributed tracing?

Answer: Method for tracking requests across multiple services to understand system behavior and performance.

258. What is event sourcing?

Answer: Pattern storing state changes as sequence of events rather than current state.

259. What is CQRS?

Answer: Command Query Responsibility Segregation — separating read and write operations using different models.

260. What is saga pattern?

Answer: Pattern for managing distributed transactions across multiple services using compensating actions.

261. What is container orchestration?

Answer: Automated deployment, scaling, and management of containerized applications.

262. What is Docker?

Answer: Platform for containerizing applications, ensuring consistent deployment across environments.

263. What is Kubernetes?

Answer: Container orchestration platform for automating deployment, scaling, and management of containerized applications.

264. What is service mesh?

Answer: Infrastructure layer handling service-to-service communication, providing security, observability, and traffic management.

265. What is blue-green deployment?

Answer: Deployment strategy using two identical environments to minimize downtime and risk.

Security

266. What is authentication vs authorization?

Answer:

- **Authentication:** Verifying user identity
- **Authorization:** Determining user permissions

267. What is JWT (JSON Web Token)?

Answer: Compact, URL-safe token format for securely transmitting information between parties.

268. What is OAuth 2.0?

Answer: Authorization framework enabling third-party applications to access user resources without sharing credentials.

269. What is Spring Security?

Answer: Comprehensive security framework providing authentication, authorization, and protection against common attacks.

270. What is HTTPS and SSL/TLS?

Answer:

- **HTTPS:** HTTP over encrypted connection
- **SSL/TLS:** Cryptographic protocols securing communication

271. What is CSRF (Cross-Site Request Forgery)?

Answer: Attack where malicious site tricks user into performing unwanted actions on authenticated site.

272. What is XSS (Cross-Site Scripting)?

Answer: Attack injecting malicious scripts into web pages viewed by other users.

273. What is SQL injection?

Answer: Attack inserting malicious SQL code into application queries to manipulate database.

274. How to prevent SQL injection?

Answer:

- Use prepared statements
- Input validation
- Parameterized queries
- Stored procedures
- Principle of least privilege

275. What is encryption vs hashing?

Answer:

- **Encryption:** Reversible transformation using key
- **Hashing:** One-way transformation, irreversible

276. What is salting in password hashing?

Answer: Adding random data to password before hashing to prevent rainbow table attacks.

277. What is OWASP Top 10?

Answer: List of most critical web application security risks published by OWASP.

278. What is input validation?

Answer: Process of checking user input for correctness and security before processing.

279. What is session management?

Answer: Process of managing user sessions throughout application lifecycle, including creation, maintenance, and termination.

280. What is rate limiting?

Answer: Controlling rate of requests to prevent abuse and ensure fair usage.

Best Practices and Code Quality

281. What are SOLID principles?

Answer:

- **S:** Single Responsibility Principle
- **O:** Open/Closed Principle
- **L:** Liskov Substitution Principle
- **I:** Interface Segregation Principle
- **D:** Dependency Inversion Principle

282. What is clean code?

Answer: Code that is easy to read, understand, and maintain, following consistent style and good practices.

283. What are code smells?

Answer: Indicators of poor code quality: long methods, large classes, duplicate code, inappropriate naming.

284. What is refactoring?

Answer: Process of improving code structure without changing its external behavior.

285. What is technical debt?

Answer: Cost of additional rework caused by choosing quick solution instead of better approach.

286. What is pair programming?

Answer: Development technique where two programmers work together on same code.

287. What is code review?

Answer: Systematic examination of code to find bugs, improve quality, and share knowledge.

288. What is continuous integration?

Answer: Practice of frequently integrating code changes into shared repository with automated testing.

289. What is continuous deployment?

Answer: Practice of automatically deploying code changes to production after passing automated tests.

290. What is DevOps?

Answer: Culture and practices bringing together development and operations teams for faster delivery.

291. What is version control?

Answer: System for tracking changes to files over time, enabling collaboration and history management.

292. What is Git?

Answer: Distributed version control system for tracking changes in source code during development.

293. What is branching strategy?

Answer: Approach for managing code branches in version control (Git Flow, GitHub Flow, etc.).

294. What is logging best practices?

Answer:

- Use appropriate log levels
- Include contextual information
- Avoid logging sensitive data
- Use structured logging

- Configure log rotation

295. What is monitoring and observability?

Answer:

- **Monitoring:** Collecting and analyzing metrics
- **Observability:** Understanding system behavior from external outputs

296. What is documentation?

Answer: Written information describing code functionality, APIs, and system architecture.

297. What is API documentation?

Answer: Documentation describing how to use and integrate with APIs, including endpoints, parameters, and examples.

298. What is disaster recovery?

Answer: Process of restoring system functionality after catastrophic failure.

299. What is backup and restore?

Answer: Process of creating copies of data and restoring them when needed.

300. What are future trends in Java development?

Answer:

- Project Loom (Virtual Threads)
- Project Panama (Foreign Function Interface)
- Project Valhalla (Value Types)
- GraalVM and native compilation
- Reactive programming
- Cloud-native development