1. OOPS (Object-Oriented Programming Concepts)

- **Basic Concepts:**
    - What are the four pillars of Object-Oriented Programming (OOP)?
    - Explain the difference between class and object.
    - What is inheritance in Java? Can you provide an example?
    - What is polymorphism in Java? Explain with examples.
    - What is the difference between method overloading and method overriding?
    - Explain the concept of abstraction in Java. How is it achieved in Java?
    - What is encapsulation? Can you give an example?
    - What is the difference between final, finally, and finalize in Java?
    - How do you define a constructor in Java? What are the types of constructors?
    - What is the use of the super keyword in Java?
    - Can you explain the concept of this keyword with an example?
    - What is the difference between an abstract class and an interface?
    - Can we instantiate an interface in Java? Why or why not?
- **Advanced OOPS:**
    - What is the purpose of the instanceof operator in Java?
    - What is the difference between shallow copy and deep copy of objects in Java?
    - Explain the Object class in Java and its methods.
    - What are the access modifiers in Java and their scopes?
    - What is method chaining in Java? How is it implemented?
    - What is the difference between static and instance methods in Java?
    - What is a singleton class? How do you implement it in Java?

---

2. Strings in Java

- **Basic String Questions:**
    - What is the difference between String, StringBuilder, and StringBuffer in Java?
    - What is the difference between equals() and == for comparing strings?
    - How do you create an immutable object in Java? Is String immutable in Java? Why?
    - How can you convert a string to an integer in Java?
    - What are the performance implications of using String versus StringBuilder?
    - How do you concatenate strings in Java? What is the most efficient way?
    - How do you check if a string contains a specific character or substring?
    - How can you reverse a string in Java without using the reverse() method?
    - What does the String.intern() method do?
    - What are regular expressions in Java? How are they used with strings?
- **Advanced String Questions:**
    - How does the String pool work in Java?
    - What is the difference between String.format() and System.out.printf()?
    - How do you split a string into multiple substrings in Java?
    - How does the StringBuilder.append() method work? Can it be used with String?
    - What is the time complexity of string concatenation in Java using + and StringBuilder?

---

3. Collections in Java

- **Basic Collection Questions:**
    - What are the main differences between List, Set, and Map in Java?
    - What is the difference between ArrayList and LinkedList?
    - What is the difference between HashMap and TreeMap?

- Explain the difference between HashSet and TreeSet.
- How do you iterate through a collection in Java? What is the role of the Iterator interface?
- What is the difference between HashMap and Hashtable?
- How does ConcurrentHashMap work? What are its advantages over HashMap in a multi-threaded environment?
- What is a LinkedHashMap? How does it differ from HashMap?
- How does the containsKey() method of a Map work? What is its time complexity?
- What is the difference between ArrayList and Vector?

- **Advanced Collection Questions:**
  - Explain how a HashMap works internally in Java (hashing mechanism).
  - How can you make a collection thread-safe in Java?
  - What is the use of Comparator and Comparable interfaces in Java? How do they differ?
  - What are NavigableSet and NavigableMap in Java? How are they different from their regular counterparts?
  - How does PriorityQueue work in Java? What is its use case?
  - What is the difference between Iterator and ListIterator? Can you use a ListIterator with a Set?
  - How do you remove duplicates from a list in Java using collections?
  - What is a Queue in Java? What are the differences between Queue and Deque?
  - How does the forEach() method work in the context of a collection?

---

4. Exception Handling in Java (Java 8)

- **Basic Exception Handling Questions:**
  - What is the difference between checked and unchecked exceptions?

- Explain the try-catch-finally block in Java. What happens if you don't use finally?
- What is the use of the throws keyword in Java? How is it different from throw?
- What are the common built-in exceptions in Java?
- How does Java handle exceptions internally? What is the exception handling mechanism in Java?
- What is the role of Exception, RuntimeException, and Error in Java?
- How do you create a custom exception class in Java?

- **Advanced Exception Handling Questions:**
  - Explain the concept of exception propagation in Java.
  - What is the difference between throw and throws in Java exception handling?
  - Can you catch multiple exceptions in a single catch block in Java? How?
  - What is the use of try-with-resources in Java 8?
  - Explain the concept of "exception chaining" in Java.
  - How does the Multi-catch feature in Java 7 and later work? What's the syntax?
  - How do you handle exceptions in streams in Java 8?
  - What is CompletionException in Java and when is it thrown?
  - What are some best practices for exception handling in large Java applications?
  - What is the UncheckedIOException in Java 8, and when is it used?

Java 8 Specific Features:

- **Lambda Expressions and Functional Interface:**
  - What is a lambda expression in Java 8? Provide an example.
  - What is the significance of functional interfaces in Java 8? Can you name a few built-in functional interfaces?

- How does method reference work in Java 8? Provide examples.
- Explain the concept of the Stream API in Java 8. How does it differ from traditional collections?
- What is the difference between map() and flatMap() in streams?

- **Streams API:**
  - What is the difference between a sequential and a parallel stream in Java 8?
  - Explain the lazy evaluation feature in Java 8 Streams.
  - How do you filter elements from a stream? Provide an example.
  - What is the purpose of the Collectors class in Java 8? Provide examples of its usage.
  - What are the terminal and intermediate operations in streams? Give examples.
  - How do you use the reduce() method in a stream?
  - What is the Optional class in Java 8, and why is it used?
  - How does forEach() method work with streams?