**Java Interview Questions and Answers**

**✅ Core Java (Beginner to Intermediate)**

**1. What are the main features of Java?**

* Platform-independent
* Object-oriented
* Secure
* Robust
* Multithreaded
* High performance via Just-In-Time (JIT) compiler

**2. What is the difference between JDK, JRE, and JVM?**

* **JDK (Java Development Kit):** Contains tools to develop Java applications.
* **JRE (Java Runtime Environment):** Provides libraries and JVM to run applications.
* **JVM (Java Virtual Machine):** Executes Java bytecode.

**3. What are primitive data types in Java?**

* byte, short, int, long, float, double, boolean, char

**4. What is the difference between == and .equals()?**

* == compares object references.
* .equals() compares object values (can be overridden).

**5. What is a constructor?**

* A special method that initializes an object when it is created. It has no return type and the same name as the class.

**6. What is method overloading and method overriding?**

* **Overloading:** Same method name, different parameters (compile-time polymorphism).
* **Overriding:** Subclass provides specific implementation of a method in the parent class (run-time polymorphism).

**7. What is the difference between ArrayList and LinkedList?**

* ArrayList is backed by a dynamic array; better for accessing elements.
* LinkedList is a doubly-linked list; better for insertions/deletions.

**8. How is memory managed in Java (Stack vs Heap)?**

* **Stack:** Stores method calls and local variables.
* **Heap:** Stores objects and class instances.

**9. What is the use of the final, finally, and finalize keywords?**

* final: Constant variable or method/class that cannot be overridden.
* finally: Block that always executes after try-catch.
* finalize(): Called before garbage collection (deprecated).

**10. Explain access modifiers in Java.**

* public: Accessible everywhere.
* private: Accessible within the same class.
* protected: Accessible in the same package and subclasses.
* Default (no modifier): Accessible within the same package.

**🧱 Object-Oriented Programming (OOP)**

**1. What are the four pillars of OOP?**

* Encapsulation, Inheritance, Polymorphism, Abstraction

**2. Explain inheritance with an example.**

class Animal {

void sound() { System.out.println("Animal sound"); }

}

class Dog extends Animal {

void sound() { System.out.println("Bark"); }

}

**3. What is polymorphism in Java?**

* The ability to take many forms: method overloading (compile-time) and overriding (run-time).

**4. What is abstraction? How is it implemented in Java?**

* Hiding internal details and showing only functionality.
* Implemented via abstract classes and interfaces.

**5. What is encapsulation and why is it important?**

* Wrapping data and methods into a single unit (class), ensuring controlled access.

**6. What’s the difference between an interface and an abstract class?**

* Interface: All methods are abstract (Java 8+ allows default/static).
* Abstract class: Can have method implementations.

**7. Can a class extend multiple classes in Java? Why or why not?**

* No, Java does not support multiple inheritance with classes to avoid ambiguity.

**🔁 Collections & Generics**

**1. What is the Java Collections Framework?**

* A set of classes and interfaces that implement commonly reusable data structures.

**2. Difference between HashMap, TreeMap, and LinkedHashMap.**

* HashMap: Unordered.
* TreeMap: Sorted by key.
* LinkedHashMap: Maintains insertion order.

**3. How does HashSet ensure uniqueness?**

* Uses a HashMap internally, relies on hashCode() and equals().

**4. What are generics and why are they useful?**

* Allow type-safe data structures. Example: List<String>.

**5. What is the difference between List, Set, and Map?**

* List: Ordered, allows duplicates.
* Set: Unordered, no duplicates.
* Map: Key-value pairs.

**6. What’s the difference between Comparable and Comparator?**

* Comparable: Natural ordering (compareTo() in the object).
* Comparator: Custom ordering using external class.

**🧵 Concurrency & Multithreading**

**1. What is the difference between Thread and Runnable?**

* Thread: Subclass, overrides run().
* Runnable: Interface, shared by multiple threads.

**2. What are synchronized methods and blocks?**

* Ensure that only one thread can access a method/block at a time.

**3. Explain thread lifecycle in Java.**

* New → Runnable → Running → Blocked/Waiting → Terminated

**4. What is the use of volatile keyword?**

* Ensures visibility of changes to variables across threads.

**5. How does wait(), notify(), and notifyAll() work?**

* Used for thread communication inside synchronized blocks.

**6. What is a ThreadPoolExecutor?**

* Manages a pool of worker threads; efficient for handling many short tasks.

**⚙️ Exception Handling**

**1. What is the difference between checked and unchecked exceptions?**

* Checked: Must be handled (e.g., IOException).
* Unchecked: Runtime exceptions (e.g., NullPointerException).

**2. What is a try-catch-finally block?**

* Used to handle exceptions and ensure final execution.

**3. Can you catch multiple exceptions in one block?**

* Yes, using multi-catch (Java 7+): catch (IOException | SQLException e)

**4. How to create a custom exception in Java?**

class MyException extends Exception {

public MyException(String message) { super(message); }

}

**📦 Java 8+ Features (Lambdas, Streams, etc.)**

**1. What is a lambda expression in Java?**

* A concise way to represent an anonymous function.

(x, y) -> x + y

**2. What are functional interfaces?**

* Interfaces with a single abstract method. Example: Runnable, Function<T,R>

**3. What is the Stream API and how is it used?**

* Used for processing collections with a functional approach.

list.stream().filter(x -> x > 5).collect(Collectors.toList());

**4. Difference between map() and flatMap() in streams.**

* map(): Transforms each element.
* flatMap(): Flattens nested structures.

**5. What is Optional and how is it used to avoid NullPointerException?**

* A container for optional values.

Optional<String> name = Optional.ofNullable(getName());

name.ifPresent(System.out::println);

**🧪 Coding & Algorithm Style Questions (Java-based)**

**1. Reverse a string or linked list.**

String reversed = new StringBuilder(input).reverse().toString();

**2. Check for a palindrome.**

boolean isPalindrome = str.equals(new StringBuilder(str).reverse().toString());

**3. Implement a singleton class.**

public class Singleton {

private static Singleton instance = new Singleton();

private Singleton() {}

public static Singleton getInstance() {

return instance;

}

}

**4. Detect a cycle in a linked list.**

* Use Floyd’s Cycle Detection Algorithm (slow and fast pointers).

**5. Implement a simple LRU cache.**

* Use LinkedHashMap with accessOrder set to true.

class LRUCache<K, V> extends LinkedHashMap<K, V> {

private int capacity;

public LRUCache(int capacity) {

super(capacity, 0.75f, true);

this.capacity = capacity;

}

protected boolean removeEldestEntry(Map.Entry<K, V> eldest) {

return size() > capacity;

}

}