Java Technical Interview Questions:

Core Java Concepts

1. **What distinguishes JDK from JRE?**
    - **JDK (Java Development Kit)** includes tools for developing Java applications, such as a compiler and debugger. **JRE (Java Runtime Environment)** is a subset of JDK that provides libraries and other components necessary for running Java applications but does not include development tools.
2. **Why is Java considered platform-independent?**
    - Java is platform-independent because it compiles code into bytecode, which can be executed on any system that has a Java Virtual Machine (JVM), making it platform-agnostic.
3. **What is the difference between an abstract class and an interface?**
    - An **abstract class** can have both abstract (without implementation) and concrete (with implementation) methods, whereas an **interface** can only have abstract methods (until Java 8, which introduced default methods).
4. **What are the differences between final, finally, and finalize in Java?**
    - **final** is used to declare constants, prevent method overriding, and prevent inheritance of a class.
    - **finally** is used in exception handling to define a block of code that will always execute after a try-catch, regardless of whether an exception is thrown.
    - **finalize** is a method in the Object class that is invoked by the garbage collector before an object is removed from memory.
5. **How do stack and heap memory differ in Java?**
    - **Stack memory** is used for storing method calls, local variables, and references. It is automatically managed. **Heap memory** is used for storing objects and is managed by the garbage collector.
6. **How does method overloading differ from method overriding?**
    - **Method overloading** occurs when multiple methods have the same name but differ in parameters (number, type, or order). **Method overriding** occurs when a subclass provides its own implementation for a method already defined in the superclass.
7. **How do private and protected modifiers differ?**

   - **private** restricts access to the member within the same class only. **protected** allows access within the same package and subclasses.

8. **What is constructor overloading in Java?**
   - Constructor overloading occurs when a class has more than one constructor with different parameter lists, allowing the creation of objects in different ways.
9. **What is the role of the super keyword in Java?**
   - The super keyword refers to the superclass of the current object and is used to call superclass methods and constructors.
10. **What is the difference between static methods, static variables, and static classes in Java?**
    - **Static methods** belong to the class rather than instances. **Static variables** are shared among all instances of the class. **Static classes** are nested classes that can be instantiated without a reference to an outer class.
11. **What does System.out.println do in Java?**
    - System.out.println is a method that prints text to the console and moves the cursor to a new line after the output.
12. **Which part of memory is cleaned during garbage collection in Java?**
    - The **heap memory** is cleaned during garbage collection, as it stores dynamically created objects.

Object-Oriented Programming (OOP)

1. **What are the Object-Oriented features in Java?**
   - Java supports features such as **encapsulation**, **inheritance**, **polymorphism**, and **abstraction**.
2. **What access specifiers are used in Java?**
   - The access specifiers in Java are **public**, **private**, **protected**, and **default** (no modifier).
3. **What is the difference between composition and inheritance?**
   - **Composition** involves building classes using objects of other classes, whereas **inheritance** involves a subclass inheriting properties and behaviors from a superclass.
4. **Why use an abstract class in Java?**
   - An **abstract class** allows defining methods that must be implemented by subclasses, providing a common structure while leaving specific details to the subclasses.
5. **How does a constructor differ from a method in Java?**
   - A **constructor** initializes objects when they are created, while a **method** defines behavior that can be called after the object is created.

6. **What is the diamond problem in Java, and how is it solved?**
    - The **diamond problem** occurs when a class inherits from two classes that have a common ancestor. It is solved in Java using **interfaces**, which allow multiple inheritance without ambiguity.
7. **What is the difference between local and instance variables?**
    - **Local variables** are defined inside methods and are not accessible outside. **Instance variables** are associated with an object and are accessible throughout the class.
8. **What is a Marker interface in Java?**
    - A **Marker interface** is an interface with no methods. It is used to indicate that a class possesses some property or behavior, such as **Serializable** or **Cloneable**.

Data Structures and Algorithms

1. **Why are strings immutable in Java?**
    - Strings are **immutable** in Java to provide security, ensure thread-safety, and optimize memory usage by allowing string pooling.
2. **How does creating a string using new() differ from using a literal?**
    - Using **new()** creates a new object in the heap, while using a **string literal** checks the string pool for an existing object before creating a new one.
3. **What is the Collections framework in Java?**
    - The **Collections framework** provides a set of interfaces and classes for working with data structures like lists, sets, maps, and queues.
4. **How do ArrayList and LinkedList differ?**
    - **ArrayList** is backed by an array, providing fast access but slow insertions/removals. **LinkedList** uses nodes with pointers, making insertions/removals faster but slower for access.
5. **How do HashMap and TreeMap differ?**
    - **HashMap** stores key-value pairs using a hash table, providing constant-time performance for lookups. **TreeMap** stores keys in a sorted order, which makes it slower for lookups but useful for ordered data.
6. **How do HashSet and TreeSet differ?**
    - **HashSet** stores unique elements without maintaining any order. **TreeSet** stores unique elements in a sorted order.
7. **What is the difference between Iterator and ListIterator?**
    - **Iterator** allows traversal in one direction (forward) and can remove elements. **ListIterator** allows bidirectional traversal and can modify elements during iteration.
8. **What is the purpose of the Comparable interface?**

- The **Comparable** interface defines a method for objects to be compared and sorted, enabling sorting in collections like lists.

9. **What is the difference between HashSet and TreeSet?**
   - **HashSet** is unordered and faster for operations, while **TreeSet** maintains elements in a sorted order.
10. **What is the java.util.concurrent package used for?**
    - The **java.util.concurrent** package provides classes for concurrent programming, like thread pools, locks, and atomic variables.

Exception Handling

1. **What is an exception in Java?**
   - An **exception** is an event that disrupts the normal flow of execution, often caused by errors like invalid input or resource issues.
2. **How does exception propagation work in Java?**
   - When an exception occurs, it is propagated up the call stack until it is caught by a corresponding catch block or results in program termination.
3. **What's the difference between checked and unchecked exceptions?**
   - **Checked exceptions** must be either caught or declared using throws. **Unchecked exceptions** (runtime exceptions) do not require explicit handling.
4. **What is the use of a try-catch block?**
   - A **try-catch block** is used to handle exceptions by enclosing code that may throw an exception and defining how to handle it.
5. **What is the difference between throw and throws?**
   - **throw** is used to explicitly throw an exception, while **throws** is used to declare that a method may throw exceptions.
6. **What is the use of the finally block?**
   - The **finally** block ensures that certain code is executed regardless of whether an exception occurs or not, such as closing resources.
7. **What's the base class of all exception classes in Java?**
   - The base class of all exception classes in Java is **Throwable**, with **Exception** and **Error** as its main subclasses.
8. **What is Java EE?**
   - **Java EE (Enterprise Edition)** is a set of APIs for building large-scale, distributed, and multi-tiered enterprise applications.
9. **What's the difference between a Servlet and a JSP?**

    - A **Servlet** is a Java class that handles HTTP requests and responses, while a **JSP (JavaServer Pages)** allows embedding Java code directly in HTML to generate dynamic content.
10. **What is the purpose of the Java Persistence API (JPA)?**
    - **JPA** is used to manage relational data in Java applications by providing an object-relational mapping (ORM) framework.
11. **What is the difference between stateful and stateless session beans?**
    - **Stateful session beans** maintain client-specific data across method calls, while **stateless session beans** do not maintain any client-specific state.

Multithreading

1. **What is a thread and what are its lifecycle stages?**
    - A **thread** is a lightweight process that executes tasks concurrently. Its lifecycle includes the **New**, **Runnable**, **Blocked**, **Waiting**, **Timed Waiting**, and **Terminated** states.
2. **How do processes and threads differ?**
    - A **process** is an independent program in execution, while a **thread** is a lightweight sub-process within a program.
3. **What are the different thread priorities in Java?**
    - Java provides **MIN_PRIORITY**, **NORM_PRIORITY**, and **MAX_PRIORITY** for setting thread priorities.
4. **What is context switching?**
    - **Context switching** is the process of saving and loading the state of threads to allow multitasking.
5. **What's the difference between user threads and daemon threads?**
    - **User threads** are regular threads that keep running until completion. **Daemon threads** run in the background and terminate when the program ends.
6. **What is synchronization?**
    - **Synchronization** ensures that only one thread can access a resource at a time, preventing conflicts in multi-threaded programs.
7. **What is deadlock?**
    - **Deadlock** occurs when two or more threads are blocked forever, each waiting for the other to release a resource.
8. **What are wait() and notify() used for?**
    - **wait()** is used to pause the execution of a thread until it is notified, while **notify()** wakes up a waiting thread.

9. **How do threads differ from processes?**
    - **Threads** are lightweight and share the same memory space, while **processes** are independent and have their own memory space.
10. **What's the difference between synchronized and volatile?**
    - **synchronized** ensures that only one thread can access a block of code at a time. **volatile** ensures that a variable's value is always up-to-date across all threads.
11. **What's the purpose of the sleep() method?**
    - The **sleep()** method pauses the execution of a thread for a specified amount of time.
12. **How does wait() differ from sleep()?**
    - **wait()** is used for thread coordination and pauses a thread until it is notified, while **sleep()** pauses the thread for a specific time without needing notification.
13. **What's the difference between notify() and notifyAll()?**
    - **notify()** wakes up one waiting thread, while **notifyAll()** wakes up all waiting threads.