1. Why Java is not 100% Object-Oriented?

Java is not 100% object-oriented because it supports primitive data types like int, double, char, etc., which are not objects.

Pure OOP languages require everything to be an object.

Java introduced primitives for performance, memory efficiency, and simplicity.

Static methods and variables also break pure object-oriented behavior.

2. Why Pointers Are Not Present in Java?

Java does not support pointers to ensure security, memory safety, and platform independence.

Pointers can cause memory corruption, dangling references, and security issues.

Java uses references instead of pointers, which are managed by the JVM.

3. What is JIT (Just-In-Time Compiler)?

JIT is a JVM component that converts frequently executed bytecode into native machine code at runtime.

It improves performance by avoiding repeated interpretation of hot code.

Java uses a combination of interpreter and JIT compilation.

4. Why Strings Are Immutable in Java?

Strings are immutable for security, thread safety, memory efficiency, and hashcode caching.

Immutability allows safe usage in String Constant Pool and HashMap keys.

Any modification creates a new String object instead of changing the existing one.

5. What is a Marker Interface?

A marker interface is an empty interface used to mark a class.

JVM or frameworks check marker interfaces to apply special behavior.

Examples include Serializable, Cloneable, and RandomAccess.

6. Can We Override Private or Static Methods?

Private methods cannot be overridden because they are not inherited.

Static methods cannot be overridden; they are method-hidden.

Overriding works only with instance methods using runtime polymorphism.

7. Does Finally Always Execute?

Finally executes in almost all cases except when JVM terminates.

It does not execute if System.exit() is called or JVM crashes.

Finally always runs before return statements.

8. What Methods Does Object Class Have?

Object class provides methods like toString, equals, hashCode, getClass.

It also provides thread-related methods wait, notify, notifyAll.

clone and finalize support object copying and cleanup.

9. How Can We Make a Class Immutable?

Make the class final and fields private and final.

Do not provide setter methods.

Use defensive copying for mutable objects.

10. What is Singleton Class and How to Create It?

A Singleton class allows only one instance in the JVM.

Implemented using private constructor and static instance.

Thread-safe approaches include Double-Checked Locking, Bill Pugh, and Enum Singleton.