Here are some fundamental Java interview questions that often come up for junior or entry-level roles:

1. What are the main features of Java?

Java is object-oriented, platform-independent, secure, robust, portable, multi-threaded, high-performance, and has automatic garbage collection.

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

JDK (Java Development Kit): Contains tools for developing, compiling, and running Java programs (includes JRE).

JRE (Java Runtime Environment): Provides the libraries, Java Virtual Machine (JVM), and other components to run Java applications.

JVM (Java Virtual Machine): An abstract machine that provides the runtime environment for executing Java bytecode.

3. What is the difference between == and equals() in Java?

== checks for reference equality, i.e., if two references point to the same memory location.

equals() checks for content equality and is used to compare the actual content of objects.

4. What is an object in Java?

An object is an instance of a class. It is created in memory and has its own state and behavior defined by the class.

5. What is a constructor in Java?

A constructor is a special method that initializes new objects. It has the same name as the class and doesn’t have a return type.

6. What are the different types of constructors in Java?

Default constructor: A no-argument constructor provided by Java if no other constructors are defined.

Parameterized constructor: A constructor that takes parameters to initialize an object with specific values.

7. What is inheritance in Java?

Inheritance is a concept where one class (subclass) inherits properties and behaviors (methods) from another class (superclass), promoting code reuse.

8. What is the difference between method overloading and method overriding?

Method Overloading: Defining multiple methods with the same name but different parameter lists within the same class.

Method Overriding: Providing a specific implementation for a method defined in a superclass, done in the subclass.

9. What is polymorphism in Java?

Polymorphism allows a single action to behave differently based on the context. In Java, polymorphism is mainly achieved through method overloading (compile-time) and method overriding (runtime).

10. What is an interface in Java?

An interface is a reference type in Java that can contain abstract methods (methods without a body) and constants. Classes can implement interfaces, providing implementations for the abstract methods.

11. What is an abstract class, and how is it different from an interface?

An abstract class can have abstract methods as well as concrete methods (methods with a body). It cannot be instantiated and is used to provide a base class for other classes.

Difference: Abstract classes can have method implementations, while interfaces (before Java 8) only had method declarations. Interfaces support multiple inheritance, whereas classes do not.

12. What is the final keyword in Java?

final can be used with variables (making them constants), methods (preventing overriding), and classes (preventing inheritance).

13. What is the purpose of static keyword?

static indicates that a member (variable or method) belongs to the class rather than any specific instance, meaning it can be accessed without creating an object of the class.

14. What is garbage collection in Java?

Garbage collection is the process of automatically freeing memory by removing objects that are no longer reachable in the program. Java has an automatic garbage collector to manage memory.

15. Explain try, catch, and finally in exception handling.

try: Block where code that may throw an exception is written.

catch: Block that catches and handles the exception.

finally: Block that executes regardless of whether an exception occurred, usually for cleanup code.

16. What is the difference between Array and ArrayList?

Array: Fixed-size, can store both primitive data types and objects.

ArrayList: Resizable, part of the Java Collection framework, and can only store objects.

17. What are the main differences between String, StringBuilder, and StringBuffer?

String: Immutable, meaning any modification creates a new String object.

StringBuilder: Mutable and not thread-safe, used for single-threaded applications.

StringBuffer: Mutable and thread-safe (synchronized), used for multi-threaded applications.

18. What is multithreading in Java?

Multithreading allows concurrent execution of two or more threads, improving CPU utilization. Threads can be created by implementing the Runnable interface or extending the Thread class.

19. What are access modifiers in Java?

Java has four access modifiers:

public: Accessible from any other class.

protected: Accessible within the same package and by subclasses.

default (no modifier): Accessible within the same package.

private: Accessible only within the same class.

20. What is a Java package, and why is it used?

A package is a namespace used to organize a group of related classes and interfaces, helping to avoid naming conflicts and making code modular and easier to maintain.

21. What are the main concepts of OOP in Java?

The four main OOP concepts are:

Encapsulation: Bundling data and methods within a class.

Inheritance: Mechanism where one class inherits properties of another.

Polymorphism: Allows methods to do different things based on the object type.

Abstraction: Hiding implementation details, showing only essential features.

22. What is the this keyword in Java?

this is a reference variable that refers to the current object. It’s used to distinguish instance variables from local variables when they have the same name.

23. What are public, private, protected, and package-private (default) access?

These are access levels that determine visibility:

public: Visible to all classes.

private: Visible only within the same class.

protected: Visible to subclasses and within the same package.

default (no modifier): Visible within the same package only.

24. What is the super keyword in Java?

super refers to the superclass (parent class) and is used to access superclass methods, variables, and constructors from a subclass.

25. What is serialization in Java?

Serialization is the process of converting an object into a byte stream, making it possible to save the object’s state to a file or transmit it over a network. Serializable interface is used to mark a class as serializable.

These are foundational concepts, and being comfortable with them will help you answer most introductory Java questions effectively.

Here are some commonly asked questions about inheritance in Java that may come up in interviews, along with brief explanations to help you prepare.

1. What is inheritance in Java?

Inheritance is a mechanism in Java where one class (subclass) inherits fields and methods from another class (superclass). This allows code reusability and establishes a relationship between classes. It’s achieved using the extends keyword.

2. What are the types of inheritance in Java?

Java supports:

Single inheritance (a class extends one superclass).

Multilevel inheritance (a class is derived from another derived class).

Hierarchical inheritance (multiple classes extend the same superclass).

Note: Java does not support multiple inheritance (where a class has more than one superclass) to avoid ambiguity. However, it can be achieved through interfaces.

3. What is the difference between super and this keywords?

super: Refers to the superclass object and is used to access superclass methods and constructors from a subclass.

this: Refers to the current class instance and is used to access class-level variables and methods within the same class.

4. Can a constructor be inherited in Java?

No, constructors are not inherited in Java. However, a subclass can call a superclass constructor using super().

5. What is method overriding? How does it relate to inheritance?

Method overriding occurs when a subclass provides a specific implementation of a method already defined in its superclass. It enables polymorphism and allows a subclass to modify the behavior of the inherited method.

6. What is the final keyword in inheritance?

The final keyword can be used to restrict inheritance:

A final class cannot be subclassed.

A final method cannot be overridden by subclasses.

final fields are constants and cannot be changed once initialized.

7. Explain the instanceof keyword and its relation to inheritance.

The instanceof keyword checks if an object is an instance of a specific class or subclass. It's often used with inheritance to verify the type of an object before performing type-specific operations.

8. What are the advantages of inheritance?

Code reuse (reusability of common functionality).

Improved maintainability (centralizes common code in one superclass).

Enables polymorphism, allowing dynamic method dispatch at runtime.

9. Why doesn't Java support multiple inheritance?

Java avoids multiple inheritance to prevent the diamond problem, where a class inherits from two classes that share a common superclass, leading to ambiguity. Java resolves this by allowing multiple inheritance through interfaces.

10. Can you override a private method in Java?

No, private methods are not accessible outside their class, so they cannot be overridden. However, if a subclass defines a method with the same signature, it is treated as a separate method, not an overridden one.

11. What is an abstract class, and how is it related to inheritance?

An abstract class cannot be instantiated and is meant to be subclassed. It can have both abstract (unimplemented) methods and concrete (implemented) methods, allowing subclasses to provide specific implementations for abstract methods.

12. What is the difference between an interface and an abstract class?

Abstract Class: Can have both implemented and unimplemented methods, but only single inheritance is allowed.

Interface: All methods are abstract by default (until Java 8), and multiple inheritance is possible since a class can implement multiple interfaces.

13. What is polymorphism, and how does inheritance support it?

Polymorphism allows objects to be treated as instances of their superclass, enabling dynamic method invocation. Inheritance supports polymorphism by letting subclasses override methods and enabling method calls based on the object's runtime type.

14. How does Java handle multiple inheritance with interfaces?

A class can implement multiple interfaces, allowing it to inherit multiple sets of behaviors. Since interfaces don’t have state, they avoid the diamond problem. However, from Java 8 onwards, interfaces can have default methods, so careful handling is required to avoid ambiguity.

15. What is covariant return type, and how does it relate to inheritance?

Covariant return type allows a subclass to override a method and return a more specific type than the superclass method. This is helpful in inheritance as it maintains type consistency and improves type safety.

16. What happens if a subclass does not implement all abstract methods of its superclass?

If a subclass does not implement all abstract methods, it must also be declared as abstract. Otherwise, the code will not compile.

17. Explain the difference between method hiding and method overriding.

Method hiding occurs when a subclass defines a static method with the same signature as a static method in the superclass. It is not the same as overriding, as it depends on the reference type, not the object type.

18. What is the purpose of protected access modifier in inheritance?

The protected access modifier allows fields and methods to be accessible within the same package and by subclasses. It’s often used in inheritance to grant access to subclass methods.

19. What is a superclass reference and a subclass object?

A superclass reference can hold a subclass object, allowing access to superclass methods while enabling polymorphism. However, to access subclass-specific methods, explicit casting is needed.

20. What is constructor chaining, and how does it work with inheritance?

Constructor chaining occurs when one constructor calls another constructor within the same class (using this()) or superclass (using super()). In inheritance, super() is used to invoke the superclass constructor to ensure proper initialization.

These questions cover essential aspects of inheritance in Java and will help demonstrate a solid understanding during interviews.