1. **What is an interface in Java?**

   An interface in Java is a reference type, similar to a class, that can contain only constants, method signatures, default methods, static methods, and nested types. Interfaces cannot contain instance fields or constructors. They are used to specify a set of methods that a class must implement.

2. **Which modifiers are allowed for methods in an interface? Explain with an example.**

   In an interface, methods can have the following modifiers:

- **public**: All methods in an interface are implicitly public.
- **abstract**: Methods in an interface are implicitly abstract, meaning they do not have a body.

3. **What is the use of interface in Java? Or, why do we use an interface in Java?**

   Interfaces are used in Java for several reasons:

   Abstraction: They provide a way to define methods that must be implemented by a class, without specifying how these methods should be implemented.

   Multiple Inheritance: Java does not support multiple inheritance with classes, but a class can implement multiple interfaces, allowing for a form of multiple inheritance.

   Loose Coupling: Interfaces allow for the decoupling of code, making it easier to change and maintain.

4. **What is the difference between abstract class and interface in Java?**

   Abstract Class:

- Can have instance fields, constructors, and methods with implementations.
- Can have both abstract and non-abstract methods.
- A class can extend only one abstract class.

  Interface:

- Cannot have instance fields or constructors.
- Can have only abstract methods (until Java 8, which introduced default and static methods).
- A class can implement multiple interfaces**.**