Interface in Java

Definition:

An interface in Java is a reference type, similar to a class, that can contain only abstract methods, default methods, static methods, and constants. It is used to achieve abstraction and multiple inheritance in Java.

Syntax:

```
interface InterfaceName {
   // constant fields
   // abstract methods
   // default methods (Java 8+)
   // static methods (Java 8+)
}
```

Example:

```
interface Animal {
   void sound();  // abstract method
}

class Dog implements Animal {
   public void sound() {
      System.out.println("Dog barks");
   }
}
```

Key Points:

- - Interfaces cannot be instantiated.
- - All methods in interfaces are public and abstract by default (except static and default).
- - All variables in interfaces are public, static, and final by default.
- - A class uses the implements keyword to inherit an interface.
- - A class can implement multiple interfaces.

Why Use Interface?

- - To achieve abstraction.
- - To support multiple inheritance.
- - To define a contract that other classes must follow.

Types of Methods in Interface:

Type	Description	Since
Abstract Method	Must be implemented in implementing class	Java 1.0
Default Method	Has body, can be overridden	Java 8
Static Method	Has body, called using interface name	Java 8
Private Method	Used only within the interface (helper)	Java 9

Example with Default and Static Methods:

```
interface Vehicle {
   void start();

   default void fuel() {
      System.out.println("Petrol");
   }

   static void stop() {
      System.out.println("Vehicle stopped");
   }
}
```

Multiple Interface Example:

```
interface A {
   void display();
}

interface B {
   void show();
}
```

```
class C implements A, B {
   public void display() {
      System.out.println("Display from A");
   }

   public void show() {
      System.out.println("Show from B");
   }
}
```