**Default Methods in Java Interfaces**

In Java, a **default method** is a method that is defined in an interface with a default implementation. This feature was introduced in Java 8 to allow interfaces to evolve without breaking the implementing classes.

**Key Points about Default Methods:**

* **Backward Compatibility:** Before Java 8, if a new method was added to an interface, all classes that implemented that interface would need to provide an implementation for the new method. Default methods allow new methods to be added to interfaces without affecting existing classes.
* **Default Implementation:** The method has a body (implementation) in the interface, which means classes that implement the interface don't have to override it unless they want to provide a specific implementation.

**Syntax**

The syntax for a default method in an interface is similar to a method in a class, except it uses the default keyword.

public interface MyInterface {

void regularMethod(); // Abstract method

default void defaultMethod() {

System.out.println("This is a default method.");

}

}

**Example**

Here’s an example to demonstrate how default methods work:

interface MyInterface {

void regularMethod();

default void defaultMethod() {

System.out.println("This is the default implementation.");

}

}

class MyClass implements MyInterface {

@Override

public void regularMethod() {

System.out.println("Implementation of the regular method.");

}

// You can override the default method if needed

@Override

public void defaultMethod() {

System.out.println("Overridden default method.");

}

}

public class DefaultMethodExample {

public static void main(String[] args) {

MyClass obj = new MyClass();

obj.regularMethod(); // Outputs: Implementation of the regular method.

obj.defaultMethod(); // Outputs: Overridden default method.

}

}

In this example:

* MyInterface defines both a regular abstract method (regularMethod) and a default method (defaultMethod).
* MyClass implements MyInterface. It overrides both methods, including the default method, providing its own implementation.

If MyClass didn't override defaultMethod, it would inherit the default implementation provided by MyInterface.

**Multiple Inheritance with Default Methods**

If a class implements multiple interfaces that contain default methods with the same signature, the compiler will throw an error, and the class must override the conflicting default method.

**Example:**

interface InterfaceA {

default void myMethod() {

System.out.println("InterfaceA default method");

}

}

interface InterfaceB {

default void myMethod() {

System.out.println("InterfaceB default method");

}

}

class MyClass implements InterfaceA, InterfaceB {

@Override

public void myMethod() {

System.out.println("Overridden method to resolve conflict");

}

}

public class DefaultMethodConflictExample {

public static void main(String[] args) {

MyClass obj = new MyClass();

obj.myMethod(); // Outputs: Overridden method to resolve conflict

}

}

In this case, the MyClass must override myMethod to resolve the conflict between InterfaceA and InterfaceB.

**Conclusion**

Default methods in interfaces are a powerful feature in Java, providing flexibility and maintaining backward compatibility as interfaces evolve. They allow developers to add new functionality to interfaces without breaking existing implementations and provide a mechanism to resolve conflicts in multiple inheritance scenarios.