1. Default and Static Method in Interfaces :

Student.java

**package** com.java8.model;

**public** **class** Student **implements** Comparable<Student> {

**private** String name;

**private** Integer age;

**public** Student(String name, Integer age) {

**super**();

**this**.name = name;

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Integer getAge() {

**return** age;

}

**public** **void** setAge(Integer age) {

**this**.age = age;

}

@Override

**public** **int** compareTo(Student o) {

**return** **this**.getName().compareTo(o.getName());

}

}

MyInterface.java

**package** com.java8;

**import** java.util.Collections;

**import** java.util.List;

**import** com.java8.model.Student;

**public** **interface** MyInterface {

**default** **public** List<Student> sortStudent(List<Student> stuList) // default method can be overwritten by the implementation class

{

Collections.*sort*(stuList);

**return** stuList;

}

**static** **public** **void** greet(String name) // static method can't be overwritten by the implementation class. As static method is related to class not in the instance level.

{

System.***out***.println("Welcome : "+name);

}

**public** **abstract** Integer getMaximum(List<Integer> intList); // Before Java 8, Interface can contain only abstract method which needs to be implemented in the class level.

}

MyInterfaceImpl.java

**package** com.java8.interfaceImpl;

**import** java.util.Collections;

**import** java.util.List;

**import** com.java8.MyInterface;

**public** **class** MyInterfaceImpl **implements** MyInterface {

@Override

**public** Integer getMaximum(List<Integer> intList) {

**return** Collections.*max*(intList);

}

}

ClientTest.java

**package** com.java8.client;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**import** com.java8.MyInterface;

**import** com.java8.interfaceImpl.MyInterfaceImpl;

**import** com.java8.model.Student;

**public** **class** ClientTest {

**public** **static** **void** main(String[] args) {

MyInterface myInterface = **new** MyInterfaceImpl();

List<Student> stuList = **new** ArrayList<>();

stuList.add(**new** Student("Sean",20));

stuList.add(**new** Student("Andrew",16));

stuList.add(**new** Student("Frank",26));

List<Student > sortStudent = myInterface.sortStudent(stuList);

**for**(Student student : sortStudent)

{

System.***out***.println(student.getName()+"\t"+student.getAge());

}

System.***out***.println("-----------------------------------");

MyInterface.*greet*("Andrew");

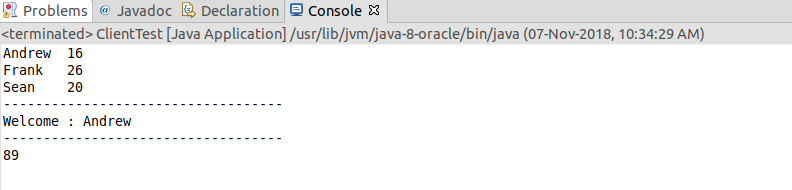
System.***out***.println("-----------------------------------");

List<Integer> myIntList = **new** ArrayList(Arrays.*asList*(10,50,0,5,34,23,89));

System.***out***.println(myInterface.getMaximum(myIntList));

}

}



From Java 8, Interfaces are enhanced with the implementation of Method. There can be two kind of method implementation – Static Method and default method.

1. default method can be overwritten by the implementation class

2. static method can't be overwritten by the implementation class. As static method is related to class not in the instance level.

3. Before Java 8, Interface can contain only abstract method which needs to be implemented in the class level.

**Override Methods on ‘Object’ class methods :**

It is not possible for an interface to provide the default implementation of the methods which is present in Object class. Trying to do so it will give compile error.

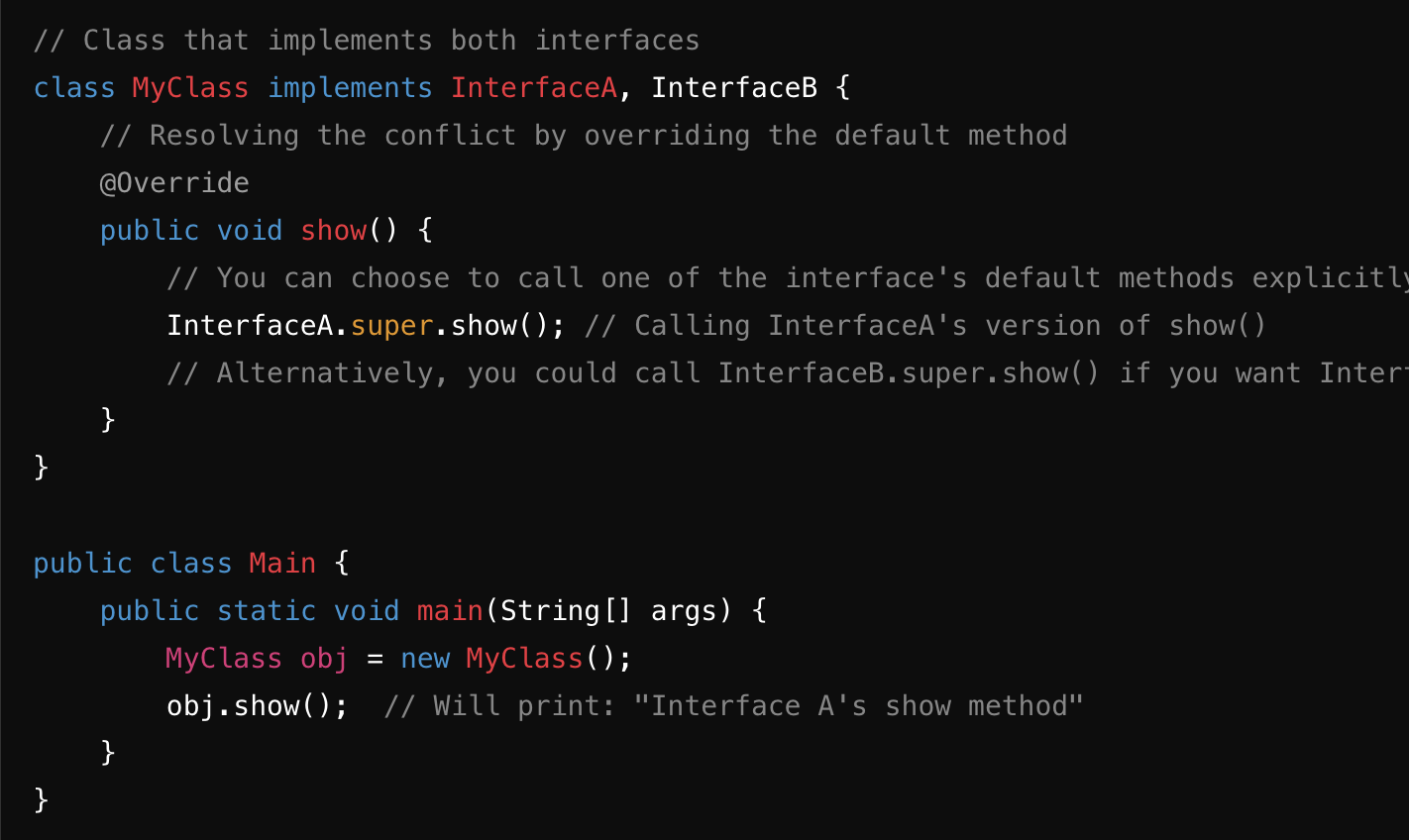
If any class in the hierarchy has a method with a same signature with the same method of Object class, then default methods become irrelevant. Since any class implementing the interface has been extended from the super class as Object, if we have deafult method as equals(), hashCode() in the interface then it is completely irreleavnt. That’s why interfaces are not allowed to implement the default method of Object class.

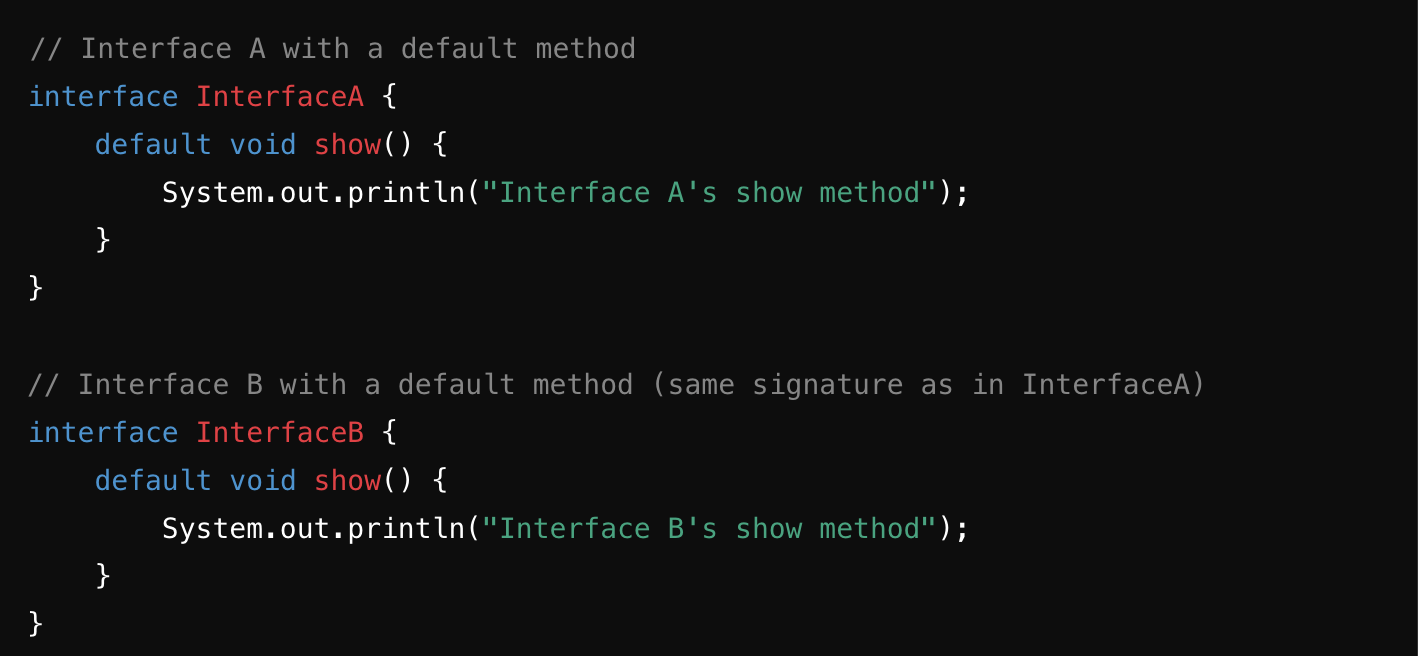
**public** **class** MyInterfaceImpl **extends** Object **implements** MyInterface

if MyInterface allows to implement the non-final methods of Object class, then compiler will be confused in this case, which methods needs to be accepted as every class extend Object class.

**Key points about the Static and Default Method Implementation in Interface :**

1. Default methods enable you to add new functionality to the interfaces of your libraries and ensure binary compatibility with code written for older versions of those interfaces.
2. Interface default methods help in extending interfaces without having the fear of breaking implementation classes.
3. Normally Static Methods in Interface is used as Helper Methods while default methods are used as a default implementation for classes that implements the interface.
4. Default method can be overridden in implementation classes, while static cannot.
5. Both implementation class and Interface can have static method with the same name and neither overrides others.
6. Interface can’t provide the default implementation of the methods which are declared in Object class.
7. In Java, when a class implements multiple interfaces that define the same default method signature, a **conflict** arises. The implementing class must **override** the default method to resolve this conflict; otherwise, the compiler will throw an error.
8. if the method in I1 is abstract, and default in I2, you cannot implement both of them. So this is also a compile-time error





Interface I1

{

default public void display()

{

System.out.println(“I1:Display”);

}

}

Interface I2

{

default public void display()

{

System.out.println(“I2:Display”);

}

}

class Welcome implements I1, I2

{

public void display()

{

I1.super.display();

I2.super.display();

}

public static void main(String [] args)

{

Welcome welcome = new Welcome();

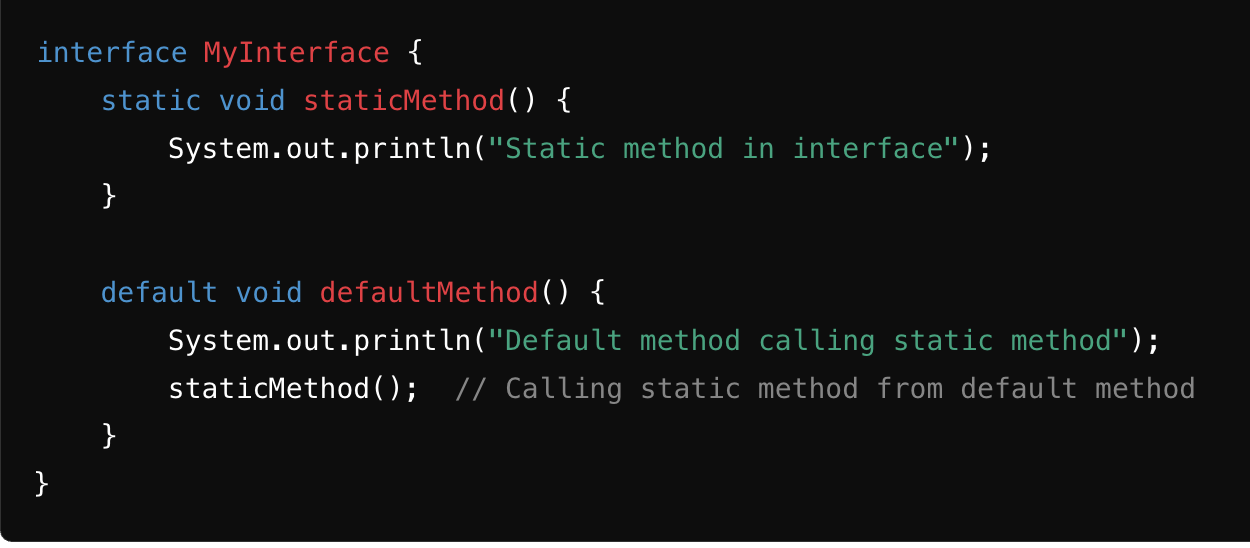
welcome.display();

}

}

**Can a default method call a static method in the same interface?**

Yes, a default method can call a static method in the same interface.



**Can a class inherit both a static and default method from an interface?**

Yes, a class can inherit both a static method and a default method from an interface, but the behaviour for each is different:

* The default method can be called on an instance of the class.
* The static method can only be called on the interface itself.

A computer screen shot of a program code

Description automatically generated

**What are the restrictions of default methods?**

**Answer:**

* A default method cannot override a method from the java.lang.Object class (e.g., toString(), equals(), hashCode()).
* Default methods are not allowed in classes or abstract classes; they are specific to interfaces.
* If a class implements an interface with a default method and also extends a class that has a method with the same signature, the class’s method will be used.