**Java-8 – Default methods inside interface vs abstract classes**

* **Default methods wrt multiple inheritance:**

interface Left{

default void m1(){

System.out.println(“Left default method”);

}

}

interface Right{

System.out.println(“Right default method”);

}

class Test implements Left, Right{

}

CE: class Test inherits unrelated defaults for m1() from type Left and Right

We can resolve the above problem in two ways:

Override the method in the implementation class.

class Test implements Left, Right{

public void m1(){

System.out.println(“My own implementation”);

}

}

Call which interface method you need explicitly as follows:

class Test implements Left, Right{

public void m1(){

***Left.super.m1()***

}

}

* **Differences between interface with default methods and abstract classes:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Interface with default method** | **Abstract class** |
| 1 | Inside interface every variable is always public, static and final we can’t not declare instance variables. | Inside abstract class we can declare instance variables, which are required to the child class. |
| 2 | Interface never talks about state of object | Abstract class can talk about state of object. |
| 3 | Inside interface we can’t declare constructors. | Inside abstract class we can declare constructors. |
| 4 | Inside interface we can’t declare instance and static blocks. | Inside abstract class, we can declare instance and static blocks. |
| 5 | Functional interface with default methods can refer lambda expression. | Abstract class can’t refer lambda expression. |
| 6 | Inside interface we can’t override Object class methods. | Inside abstract class we can override Object class methods. |

**Interface with default methods != Abstract classes**