**Java 8 Multiple Inheritance Conflict Resolution Rules and Diamond Problem**

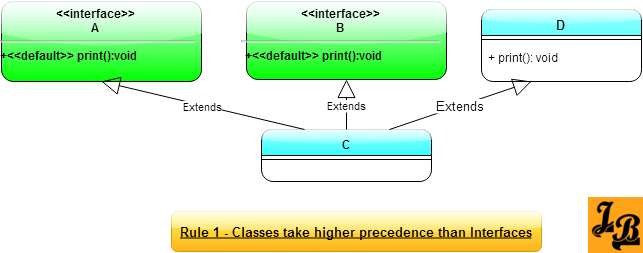
This article explains how to resolve conflicts when inheriting [default methods](https://www.javabrahman.com/java-8/default-methods-in-java-8-with-examples/) with same signatures from multiple interfaces using java 8’s conflict resolution rules. It then looks in to the classic Diamond Problem and its resolution in Java 8.

**What is default method resolution conflict**  
In my previous article I explained how Java 8 supports [multiple inheritance](https://www.javabrahman.com/java-8/java-8-multiple-inheritance-of-behavior-from-interfaces-using-default-methods/) of behaviour using default methods. However, *what-if* the multiple default interfaces implemented have default methods with the same signatures. Then which of the default implementations from the many parent interfaces will be invoked in the implementing class.

Java 8 designers have thought of this conflict and have defined resolution rules for such scenarios. Let us now take a look at the possible conflict scenarios and the resolution rules in-built in Java 8 for avoiding them.

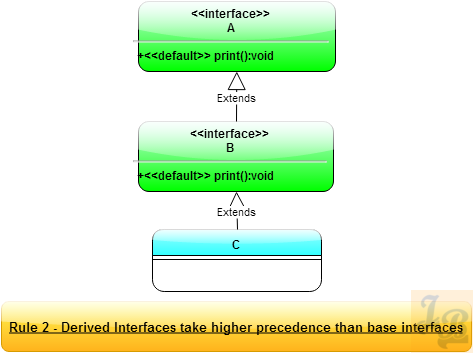
**Conflict resolution rules for inherited default methods**  
Conflict Resolution Rules for inherited default methods **in order of precedence are –**

* **Rule 1 – Classes take higher precedence than interfaces** – Any method inherited from a class or a super class is invoked over any default method inherited from an interface.  
  *Example to explain Rule 1*



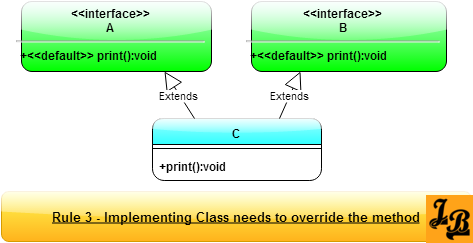
In the above class diagram, Class C inherits default method print() from interface A, interface B and super class C. If print() method is invoked in Class C then the implementation in super class C is executed.

* **Rule 2 – Derived interfaces or sub-interfaces take higher precedence than the interfaces higher-up in the inheritance hierarchy** – If default methods with the same method signature exist in an interface and its child interfaces, then the default method from the child interface is invoked.  
  *Example to explain Rule 2*



In the above class diagram, interface B inherits from interface A. Both have a default method print() with the same signature. Class C implements both interfaces A & B. When print() method is invoked on an instance of class C then the implementation in interface B is invoked as it is the lowest child/most derived interface in the inheritance hierarchy.

* **Rule 3 – In case Rule 1 and Rule 2 are not able to resolve the conflict then the implementing class has to specifically override and provide a method with the same method definition** – The implementing class can, of course, invoke the specific default method from the specific parent interface to get desired behavior. But still the class needs to override and invoke the default method to resolve the conflict.  
  *Example to explain Rule 3*



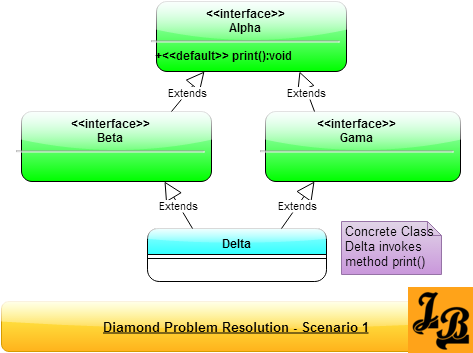
In the above class diagram, class C inherits from interfaces A & B, both of which have the default implementations of print(). Since, both interfaces A & B are parents of C, they are at the same hierarchy level, and hence, C has to provide its own implementation of method print().  
*Important Note* – Inside Class C’s implementation of print() method it should invoke the specific implementation of interface A or B. For this Java 8 has a special syntax –

**<super-interface-name>.super<method-name>**

In this case print() method in class C will invoke print() method of B,its parent, like this – **B.super.print()**

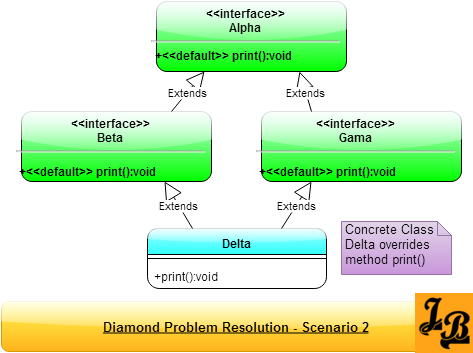
**Diamond Problem Revisited with Java 8’s Default Methods in Interfaces**  
Diamond Problem, wherein the grand-child class has both of its parents inheriting from a single grand-parent class, is a common problem faced in languages with multiple inheritance. Java 8 is affected by the diamond problem in 2 ways/2 scenarios which I will cover below –

* **Scenario 1 of diamond problem in Java 8** –



In the above class diagram, interfaces Beta & Gama implement interface Alpha. Class Delta is a concrete class and it implements both the interfaces Beta & Gama. The question in this scenario is that if Delta invokes the print() method, then which one will be invoked – the one inherited from Beta or that which was inherited from Gama?  
(*On a side note – If one sees the 4 inheritance arrows marked “extends” then they form a diamond shape and hence the name Diamond Problem*).  
**Resolution for Scenario 1 of Diamond Problem** –  
Java 8 resolves this situation by considering that there is only one implementation of print() method, which is in class Alpha. Hence, when Delta invokes print() then the implementation of print() in Alpha is executed.

* **Scenario 2 of diamond problem in Java 8** –



In the above class diagram, interfaces Beta & Gama implement interface Alpha like in scenario 1 above. However, here both of them override the print() method of Alpha. Now, when Delta wants to invoke print() which instance of print() will be invoked, the one inherited from Beta or the one inherited from Gama?  
**Resolution for Scenario 2 of Diamond Problem** –  
The answer lies in Rule 3 of the conflict resolution scenarios described earlier in this tutorial. Delta will have to override print() method and explicitly invoke the print() method on one of its parents – Beta or Gama. This will resolve the issue.