Java doesn’t support multiple inheritance for classes but does support for interfaces.

**Do you know the reason ?**

Assume we have a class A which extends two classes B and C. What happen if B and C contains a method with same name.

**class B {**

**void show() {}**

**}**

**class C {**

**void show() {}**

**}**

**class A extends B, C {**

**void print() {**

**show();**

**}**

**}**

Inside print() method of A class we have a call to show() method. And as you can see, show() method belong to B class as well as C class.

In such scenario, java can’t decide which class show() method should be call. So to avoid, such kind of ambiguity in the code. Java decide, not to support multiple inheritance.

But multiple inheritance is supported for interface. Reason is simple, because interface only contains abstract method so even if a class implements two interfaces having a method with same name. But still ambiguity is not there because we can’t call abstract methods.

But from JSE 8, java introduced the concept of **default methods**.

Default methods are nothing but the concrete methods for interface.

So now, interfaces can have concrete method in their body.

**Here one confusion should come in your mind !**

Java support multiple inheritance for interfaces and now from JSE 8 java introduced **default methods**.

Can you imagine ! We’ll face the same issue of ambiguity which were we facing in case of Multiple inheritance with classes.

Now if a class is extending two interfaces both are containing a default method with same name.

What if in the class we have a call for that default method. In this case, which default method will be called ?

**Same Ambiguity problem will be there !**

**Then why java introduced default method? What is the reason ?**

Let’s understand with a example !

It is obvious that a interface can be implemented with any number of classes.

Let’s assume we have a interface called Cloud

**interface Cloud {**

**void instaces();**

**}**

Now if any cloud comes in the market so that particular cloud has to implement this Cloud interface.

**class AWS implements Cloud {**

**public void instaces() { }**

**}**

**class Azure implements Cloud {**

**public void instaces() { }**

**}**

**class GCP implements Cloud {**

**public void instaces() { }**

**}**

So there can be lot many cloud server exist in the market. For the time being all they are providing instance service only as per the Cloud guidelines.

Now suppose, there is little changes in Cloud guidelines

**interface Cloud {**

**void instaces();**

**void storage();**

**}**

Here, you can see Cloud guidelines saying that now onwards every cloud has to provide two services i.e. instances and storage.

What if, AWS cloud don’t want to provide the storage service. But because the storage() method is added as abstract method so every cloud has to forcefully override the storage() method.

This situation is very common. Nobody, can imagine what is going to be happen in future. So java just introduced default method so that if in future Cloud interface wants to add some more guidelines without forcing it’s implemented classes to override those methods. What Cloud can do is ? Just add the storage() methods as a default method.

Now, whoever wants to provide storage service they just simply override the default storage() method.

**interface Cloud {**

**void instaces();**

**default void storage();**

**}**