Topic – Default Methods in JAVA 8

Java 8 was released on 18th March 2014 of Java. Java 8 is a product version. It's development version is JDK1.8 (JAva Development Kit). Java 8 has some new features that are most awaited in the development environment.

Let us see one of the features of Java 8 that is

Default Methods -

We know that an interface does not have method body. Classes which implement the interface have to define the method. Java 8 comes with the feature where interface methods could have a method body. Methods implemented in interfaces use **default** keyword.

For Example -

```
interface A
{double calculate(int a);
default double cube(int a){
return (a*a*a);
}
}


class Myclass
{

public static void main(String str[])
{


A a = new A(){

public double calculate (int a)
{
return cube(a);
}
};
```

```
double di = a.calculate(6);
System.out.println(di);
}
```

In this example, method **calculate** is the abstract method of the interface and **cube** method is default method. Abstract method has to implement by the class but the default method can be used directly.

The default keyword suggests that the class which implements the interface does not need to implement the method. It can use the default definition of the method provided by the interface. If we want to provide different implementation for the default method, we have to implement the method in class.

For Example -

```
interface printable{
default void printString(String s){
System.out.println("Interface default method");
}
}

class Test1{
public static void main(String args[]){
printable p =new printable(){
public void printString(String str)
{
System.out.println(str);}
};
p.printString("Method implemented to provide other definiition ");
}
}
```

In this example, interface **printable** have one default method **printString**. The class implements the method to provide different definition.

In java , interface is introduced to overcome the problem of multiple inheritance. When there are common methods in extended classes then it creates an ambiguity. To avoid this, java does not support multiple inheritance. But with the introduction of default

method, the same problem can arise when class implements more than one interface having common methods.

For example -

```
interface interface1{
void method1();
default void printStr(String str)
{
System.out.println("Interface1 print string method" + str);
}
}

interface interface2{
void method2();
default void printStr(String str)
{
System.out.println("Interface2 print string method" + str);
}
}

class myClass1 implements interface1,interface2
{
@Override
public void method1(){
}

@Override
public void method2(){
}

@Override
public void printStr(String str){
System.out.println("Myclass print string method " + str);
}

public static void main(String str[]){
myClass1 m = new myClass1();
m.printStr("HELLO");
}

}
```

In above example, **interface1** and **interface2** both have common method **printStr**. Class **myClass1** is implementing both interfaces. In this situation, compiler will throw an error. It is difficult for compiler to decide which method to call.

To solve this problem, java 8 makes it mandatory to implement the common method. In our example, we have implemented the common method in class **myClass1**. So by implementing the common method we can solve the problem occurred by implementing more than one interfaces.

If you want to use the default implementation provided by one of the interfaces, you can use the **super** keyword. In this case you don't have to provide the different implementation you can just use the default method provided by one of the interface.

For example -

```
class myClass1 implements interface1,interface2
{
@Override
public void method1(){
}

@Override
public void method2(){
}

@Override
public void printStr(String str){
interface1.super.printStr(str);
}

public static void main(String str[]){
myClass1 m = new myClass1();
m.printStr("HELLO");
}

}
```

In this example we just change the **printStr** method. The method calls the **printStr** method of **interface1** using the **super** keyword.

The main advantage of default method is that it provides **backward compatibility**. In earlier versions, interfaces are abstract and concrete. If we have to add any method in the interface, we have to change all classes that have implemented the interface. This is a very difficult task. But default method solves this problem. Now we can add methods in interfaces and use them directly without affecting the existing classes. Classes which already had implemented the interface do not have to implement the new method if it is a default method.