Priority comparison between default method and Concrete Method :

While working with class and interface, default method is having low priority than concrete method, In the same way class is more powerfull than interface.

```
interface A
{
}
class B
{
}

class C extends B implements A  {}  //Valid


class C implements A extends B {} //Invalid
```

```
//Program
package com.ravi.default_methdo;

interface A
{
  default void m1()
  {
     System.out.println("Default Method of interface A");
  }
}
class B
{
   public void m1()
   {
      System.out.println("Concrete Method of class B");
   }
}
class C extends B implements A
{
}
public class PriorityDemo
{
   public static void main(String[] args)
   {
      C c1 = new C();
        c1.m1();
   }
}
```

Can we achieve multiple inheritance using interface through default method :

```
interface Alpha                    interface Beta
{                                  {
  default void m1()                  default void m1()
  {                                  {

  }                                  }
}                                  }
         class Gamma implements Alpha, Beta
         {
            //We can achieve MI using default method also
         }
```

* We can achieve multiple Inheritance by using default method also because interface never contain constructor.

* If a class implements from two or more than two interface and If all the interfaces contains same deafult method then the implementer class must override the default method from the super interface otherwise we will get CE.

* If we want to call super interface default method from the overridden method then we should use interface name and super keyword [Alpha.super.m1()]

```
package com.ravi.default_methdo;

interface Alpha
{
    default void m1()
    {
        System.out.println("Default Method of Alpha interface...");
    }
}
interface Beta
{
    default void m1()
    {
        System.out.println("Default Method of Beta interface...");
    }
}

class Implementer implements Alpha, Beta
{
    @Override
    public void m1()
    {
        Alpha.super.m1();
        Beta.super.m1();
        System.out.println("MI is possible");
    }
}

public class MultipleInheritance
{
    public static void main(String[] args)
    {
        Implementer i = new Implementer();
        i.m1();

    }

}
```

What is static method inside an interafce ?

We can write static method (method with body) inside an interface from java 8 version.

Example :

```
public interface Calculator
{
  static double getCube(int num) //java 8 [AM is public]
  {
  }
}
```

A static method must be marked with the static keyword and contains method body.

A static method without any access modifier is assumed to be public.

A static method cannot be marked as abstract OR final.

A static method is not inherited and cannot be accessed in a clas directly, It can be accessible through interface name only.

//Programs :

Program that describes static method of an interface is by default public so we can access from another package.

```
package com.nit;

public interface Calculator
{
   static double doSum(double x, double y) //JDK 1.8
   {
      return (x+y);
   }

   static double doSquare(double x)
   {
      return (x*x);
   }
}


package com.ravi;

import com.nit.Calculator;

public class StaticDemo1
{
    public static void main(String[] args)
    {
        double result = Calculator.doSum(12, 24);
        System.out.println("Sum is :"+result);

        result = Calculator.doSquare(5);
        System.out.println("Square is :"+result);

    }

}
```

---

* As we know static method of a class is by default available to all the sub classes, so sub class can access the static method on the other hand **static method of an interface is only available to interface but not in the implementer classes so WE CAN ACCESS THE INTERFACE STATIC METHDO THROUGH INTERFACE NAME ONLY.**

```
interface Callable
{
    static void call()
    {
        System.out.println("Static Method of Callable interface");
    }
}
public class StaticDemo2 implements Callable
{
    public static void main(String[] args)
    {
        //StaticDemo2.call();    [Invalid]
        //StaticDemo2 s = new StaticDemo2();
        //s.call();  [Invalid]
        Callable.call();

    }
}
```

We can write main method inside an interface from JDK 1.8v

```
interface Drawable
{
    public static void main(String [] args)
    {
        System.out.println("Main Method");
    }
}
```

```
package com.nit;

interface A
{
   int x = 100;
}
abstract class B
{
   int x = 200;
}
class C extends B implements A
{
   int x = 300; //Variable Hiding
}
public class VariableHiding
{
   public static void main(String[] args)
   {
      C c1 = new C();
      System.out.println(c1.x);

   }

}
```

What is Functional Programming in Java ?
---

* As we know, Java always concentrate on Objects.

* From Java 1.8v, Java started writing concise code for Method OR Function.

* Here Java has provided more priority to Function so It is known as Functional Programming.

What is a Functional Interface ?
---

* If an interface contains **exactly one abstract method then It is called Functional interface.**

* It may contain 'n' number of default OR static methods but It must contain exactly one abstract method. [Excluding Object class Methds]

* In order to restrict the develeoper to accept only one abstract method we can use an annotation i.e @FunctionalInterface annotation.

Example :

```
@FunctionalInterface
public interface Printable
{
   void print();  //SAM [Single Abstract Method]

   default void m1()
   {}
}
```