JAVA - 9
Private Methods in Interfaces

Private Methods in Interfaces

Need Of Default Methods inside interfaces:

Prior to Java 8,Every method present inside interface is always public and abstract whether we are declaring or not.

```
1)  interface Prior2Java8Interf
2) {
3)     public void m1();
4)     public void m2();
5)  }
```

Assume this interface is implemented by 1000s of classes and each class provided implementation for both methods.

```
1)  interface Prior2Java8Interf
2) {
3)     public void m1();
4)     public void m2();
5) }
6)  class Test1 implements  Prior2Java8Interf
7)  {
8)     public void m1(){}
9)     public void m2(){}
10) }
11) class Test2 implements  Prior2Java8Interf
12) {
13)    public void m1(){}
14)    public void m2(){}
15) }
16) class Test3 implements  Prior2Java8Interf
17) {
18)    public void m1(){}
19)    public void m2(){}
20) }
21) class Test1000 implements  Prior2Java8Interf
22) {
23)    public void m1(){}
24)    public void m2(){}
25) }
```

It is valid b'z all implementation classes provided implementation for both m1() and m2().

Assume our programming requirement is we have to extend the functionality of this interface by adding a new method m3().

```
1)  interface Prior2Java8Interf
2)  {
3)     public void m1();
4)     public void m2();
5)     public void m3();
6)  }
```

If we add new method m3() to the interface then all the implementation classes will be effected and won't be compiled,b'z every implementation class should implement all methods of interface.

```
1)  class Test1 implements  Prior2Java8Interf
2)  {
3)     public void m1(){}
4)     public void m2(){}
5)  }
```

CE: Test1 is not abstract and does not override abstract method
m3() in PriorJava8Interf

Hence prior to java 8,it is impossible to extend the functionality of an existing interface without effecting implementation classes. JDK 8 Engineers addresses this issue and provides solution in the form of Default methods,which are also known as Defender methods or Virtual Extension Methods.

How to Declare Default Methods inside interfaces:

In Java 8,inside interface we can define default methods with implementation as follows.

```
1)  interface Java8Interf
2)  {
3)     public void m1();
4)     public void m2();
5)     default void m3()
6)     {
7)        //"Default Implementation
8)     }
9)  }
```

Interface default methods are by-default available to all implementation classes.Based on requirement implementation class can ignore these methods or use these default methods directly or can override.

Hence the main advantage of Default Methods inside interfaces is, without effecting implementation classes we can extend functionality of interface by adding new methods.(backward compatibility).

Need of private Methods inside interface:

If several default methods having same common functionality then there may be a chance of duplicate code(Redundant Code).

Eg:

```
1)  public interface Java8DBLogging
2)  {
3)     //Abstract Methods List
4)     default void logInfo(String message)
5)     {
6)        Step1: Connect to DataBase
7)        Setp2: Log Info Message
8)        Setp3: Close the DataBase connection
9)       }
10)    default void logWarn(String message)
11)    {
12)       Step1: Connect to DataBase
13)       Setp2: Log Warn Message
14)       Setp3: Close the DataBase connection
15)      }
16)    default void logError(String message)
17)    {
18)       Step1: Connect to DataBase
19)       Setp2: Log Error Message
20)       Setp3: Close the DataBase connection
21)      }
22)    default void logFatal(String message)
23)    {
24)       Step1: Connect to DataBase
25)       Setp2: Log Fatal Message
26)       Setp3: Close the DataBase connection
27)      }
28) }
```

In the above code all log methods having some common code,which increases length of the code and reduces readability.It creates maintenance problems also. In Java8 there is no solution for this.

How to declare private Methods inside interface:

JDK 9 Engineers addresses this issue and provided private methods inside interfaces. We can seperate that common code into a private method and we can call that private method from every default method which required that functionality.

```
1)  public interface Java9DBLogging
2)  {
3)     //Abstract Methods List
4)     default void logInfo(String message)
5)     {
6)        log(message,"INFO");
7)        }
8)     default void logWarn(String message)
9)     {
10)       log(message,"WARN");
11)       }
12)    default void logError(String message)
13)    {
14)       log(message,"ERROR");
15)       }
16)    default void logFatal(String message)
17)    {
18)       log(message,"FATAL");
19)       }
20)    private void log(String msg,String logLevel)
21)    {
22)       Step1: Connect to DataBase
23)       Step2: Log Message with the Provided logLevel
24)       Step3: Close the DataBase Connection
25)    }
26) }
```

Demo Program for private instance methods inside interface:

private instance methods will provide code reusability for default methods.

```
1)  interface Java9Interf
2)  {
3)     default void m1()
4)     {
5)        m3();
6)     }
7)     default void m2()
8)     {
9)        m3();
10)    }
11)    private void m3()
```

```
12)    {
13)       System.out.println("common functionality of methods m1 & m2");
14)    }
15) }
16) class Test implements Java9Interf
17) {
18)    public static void main(String[] args)
19)    {
20)       Test t = new Test();
21)       t.m1();
22)       t.m2();
23)       //t.m3(); ==>CE
24)    }
25) }
```

o/p:
D:\java9durga>java Test
common functionality of methods m1 & m2
common functionality of methods m1 & m2

Inside Java 8 interaces,we can take public static methods also.If several static methods having some common functionality,we can seperate that common functionality into a private static method and we can call that private static method from public static methods where ever it is required.

Demo Program for private static methods:

private static methods will provide code reusability for public static methods.

```
1)  interface Java9Interf
2) {
3)    public static void m1()
4)    {
5)       m3();
6)    }
7)    public static void m2()
8)    {
9)       m3();
10)   }
11)   private static void m3()
12)   {
13)      System.out.println("common functionality of methods m1 & m2");
14)   }
15) }
16) class Test implements Java9Interf
17) {
```

```
18)    public static void main(String[] args)
19)    {
20)       Java9Interf.m1();
21)       Java9Interf.m2();
22)    }
23) }
```

o/p:
D:\durga_classes>java Test
common functionality of methods m1 & m2
common functionality of methods m1 & m2

Note: Interface static methods should be called by using interface name only even in implementation classes also.

Advantages of private Methods inside interfaces:

The main advantages of private methods inside interfaces are :

1. Code Reusability
2. We can expose only intended methods to the API clients(Implementation classes),b'z interface private methods are not visible to the implementation classses.

Note:

1. private methods cannot be abstract and hence compulsory private methods should have the body.
2. private method inside interface can be either static or non-static.

JDK 7 vs JDK 8 vs JDK9:

1. Prior to java 8,we can declare only public-abstract methods and public-static-final variables inside interfaces.

```
1)  interface Prior2Java8Interface
2) {
3)     public-static-final variables
4)     public-abstract methods
5)  }
```

2. In Java 8 ,we can declare default and public-static methods also inside interface.

```
1)  interface Java8Interface
2)  {
3)     public-static-final variables
4)     public-abstract methods
```

```
5)     default methods with implementation
6)     public static methods with implementation
7)  }
```

3. In Java 9,We can declare private instance and private static methods also inside interface.

```
1)  interface Java9Interface
2) {
3)     public-static-final variables
4)     public-abstract methods
5)     default methods with implementation
6)     public static methods with implementation
7)     private instance methods with implementation
8)     private static methods with implementation
9) }
```

Note: The main advantage of private methods inside interface is Code Reusablility without effecting implemenation classes.