Java 9 Interface Private Methods

pramodbablad July 28, 2021 0

Before Java 8, interfaces in Java can have only constant variables and abstract methods. From Java 8 and onwards, interfaces are allowed to have two types of concrete methods – default and static methods. From Java 9, two more method types are introduced to interfaces. They are – private methods and private static methods. Let's see the journey of Java interfaces from earlier versions of Java to Java 8 and Java 9.

Interfaces : Before Java 8

Till Java 7, interfaces are allowed to have only constant variables and abstract methods. Concrete methods are not allowed in interfaces. The classes which implement interfaces need to provide implementations for the abstract methods of the interfaces.

```
1  interface InterfaceBefo
2  {
3      //Constant Variable
4
5      int constantVariab1
6      int constantVariab1
7
8      //Abstarct Methods
9
10     void abstractMethod
11     void abstractMethod
12 }
```

By default, constant variables of interfaces are public, static and final and abstract methods are public and abstract.

Interfaces : From Java 8 And Onwards

From Java 8, two types of concrete methods are allowed inside the interfaces. They are default and static methods.

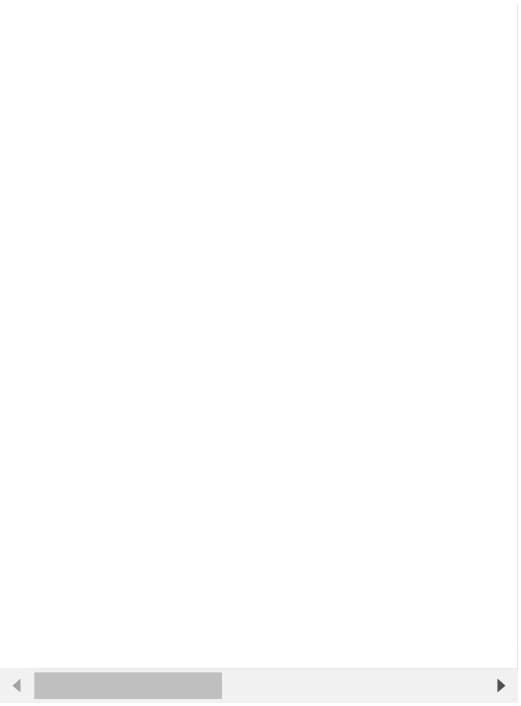
```
interface InterfaceFror
{
    //Constant Variable

    int constantVariab
    int constantVariab

    //Abstarct Methods

    void abstractMetho
    void abstractMetho

    //Default Method

    default void defau
    {
        System.out.prir
    }

    //Static Method

    static void staticM
    {
        System.out.prir
    }
}
```

Why default methods?

Default methods are introduced to add extra features to existing interfaces without disrupting their existing implementations. If there were no default methods, even if you are adding a single abstract method to an existing interface, all its existing implementations have to be updated with implementation of that method. This will be a headache if there are hundreds or thousands of implementing classes.

To overcome such overhead, default methods are introduced in interfaces from Java 8. Default methods provide default implementation for a particular method.

Why static methods?



Java API developers have followed a pattern of supplying an extra utility class along with an interface. This utility class contains only static methods to perform some basic operations on given interface type objects.

For example, `Collection` and `Collections`. `Collection` is an interface and `Collections` is an utility class which contains only static methods which perform some basic operations on `Collection` types.

From Java 8, they have break this pattern by introducing static methods in interface itself. From Java 8, interface itself will have static methods to perform some basic operations on its types.

Interfaces : From Java 9 And Onwards

From Java 9, two more type of concrete methods are allowed inside the interfaces. They are – private methods and private static methods.

```
1  interface InterfaceFror
2  {
3      //Constant Variable
4
5      int constantVariab
6      int constantVariab
7
8      //Abstarct Methods
9
```

```
10      void abstractMethod
11      void abstractMethod
12
13      //Default Method
14
15      default void defau
16      {
17          System.out.prin
18      }
19
20      //Static Method
21
22      static void staticM
23      {
24          System.out.prin
25      }
26
27      //Private Method
28
29      private void privat
30      {
31          System.out.prin
32      }
33
34      //Private Static Me
35
36      private static void
37      {
38          System.out.prin
39      }
40  }
```

Hence, from Java 9 onwards, interfaces can have 6 type of members. They are,

1. Constant Variables
2. Abstract Methods
3. Default Methods
4. Static Methods
5. Private Methods
6. Private Static Methods

Why Private Methods?

1. Private methods enhance the code re-usability inside the interfaces. For example, if two or more methods have the common code to execute then put that code block in a private method and call it whenever you require.
2. Using private methods, you can have control over what to hide and what to expose to outside the interface. If you have a sensitive data and want to use inside the interface only, then private methods will be of great use.

The following table shows the differences between abstract class and interface after Java 9.

Interface Vs Abstract Class After Java 9		
	Interface	Abstract Class
Constructors	✗	✓
Static Fields	✓	✓
Non-static Fields	✗	✓
Final Fields	✓	✓
Non-final Fields	✗	✓
Private Fields	✗	✓
Protected Fields & Methods	✗	✓
Public Fields & Methods	✓	✓
Abstract methods	✓	✓
Static Methods	✓	✓
Non-static Methods	✓	✓
Final Methods	✗	✓
Non-final Methods	✓	✓
Default Methods	✓	✗
Private Methods	✓	✓
Private Static Methods	✓	✓

Also Read :