New features introduced in Interfaces Java 8

Agenda

1. New features introduced in Interfaces
2. Example for default methods
3. Example for static methods
4. Functional Interface
5. Interface reference as argument

New features introduced in Interfaces

New features introduced in Interfaces

1) Default methods

- Before Java 8, interfaces can have only abstract methods and final variables
- The implementation for these methods has to be provided by all the implementing classes
- So, if a new method is added in an interface, then its implementation code must be provided by all the classes that had implemented this interface
- To overcome this issue, Java 8 has introduced the concept of default methods which allows the **interfaces to have methods with implementation**

2) Static methods

- Static methods contains the complete definition of the function
- Since the definition is complete and the methods are **static**, they **cannot be overridden**

Example for default methods

Example for default methods

Demo for using default methods

```
interface TestInterface
{
        //abstract method
        public void square(int a);
        //default method
        default void show()       {
        System.out.println("Default Method Executed");
        }
}
//continued..
```

Example for default methods

Demo for using default methods continued..

```
public class TestClass implements TestInterface
{
        //implementation of abstract method square
        public void square(int a) {
                System.out.println(a*a);
        }

        public static void main(String args[ ]) {
                TestClass d = new TestClass();
                d.square(4);
                //default method execution
                d.show();
        }
}
```

OUTPUT

16
Default Method Executed

Example for static methods

Example for static methods

Demo for using static methods

```
interface StaticInterface {
        //abstract method
        public void square (int a);
        //static method
        static void show() {
                System.out.println("Static Method Executed");
        }
}

//continued..
```

Example for static methods

Demo for using static methods continued..

```
public class TestImpl implements StaticInterface{
        //Implementation of abstract method square
        public void square (int a) {
                System.out.println(a*a);
        }
        public static void main(String args[ ]) {
                 TestImpl d = new TestImpl ();
                d.square(4);
                //Static method execution
                 StaticInterface.show();
        }
}
```

OUTPUT

16
Static Method Executed

Functional Interface

Functional Interface

- A functional interface is an interface that contains **only one abstract method**
- They can have only one functionality to exhibit
- It can be marked using @**FunctionalInterface** annotation
- Before Java 8, we need to create anonymous inner class objects or implement these interfaces
- From Java 8 onwards, **lambda expressions** can be used to represent the instance of a functional interface
- ***Runnable***, ***ActionListener***, ***Comparable*** are some of the examples of in built functional interfaces.

Functional Interface

Demo for functional interface

```
interface DefaultInterface
{
        //An abstract method
        void abstFunction(int x);
        //A non-abstract (or default) method
        default void defaultFunction() {
        System.out.println(“default function in interface");
        }
}
```

//continued..

Functional Interface

Demo for functional interface continued..

```
public class InterfaceTester {
        public static void main(String args[]) {
                //Lambda expression to implement the abstract method from java8Interface
                 DefaultInterface fobj =null;
                fobj = (int x)->System.out.println(3*x);

                //Calling the above Lambda expression
                fobj.abstFunction(5);
                //Calling the default function
                fobj.defaultlFunction();
        }
}
```

OUTPUT

15
default function in interface

Interface reference as argument

Interface reference as argument

Working with Interfaces and Lambda expressions

```
interface Interface1 {
        int operation(int a, int b);
        }
interface Interface2 {
        void getMessage(String msg);
        }
public class LambdaAndInterfaceTest {
        private int operate(int a, int b, Interface1 ref1) {
                return ref1.operation(a, b);
                }  //continued..
```

Interface reference as argument

//Working with Interfaces and Lambda expressions continued..

```
public static void main(String args[ ])  {
        LambdaAndInterfaceTest tobj = new LambdaAndInterfaceTest();
        Interface1 addoperation = (int x, int y) -> x + y;
        System.out.println("tobj.operate(6, 3, addoperation) = " + tobj.operate(6, 3, addoperation));
        //O/P: tobj.operate(6, 3, addoperation) = 9

        Interface1 multiply= (int x, int y) -> x * y;
        System.out.println("tobj.operate(6, 3, multiply) = " +tobj.operate(6, 3, multiply)); //18
        //O/P: tobj.operate(6, 3, multiply) = 18

        Interface2 printref = message->System.out.println("Hello"+ message);
        System.out.print(" printref.getMessage(\"World\") = ");
        printref.getMessage("World");
       //O/P: printref.getMessage("World") = HelloWorld

    }
}
```

Thank you