Java Functional Interfaces

An Interface that contains exactly one abstract method is known as functional interface. It can have any number of default, static methods but can contain only one abstract method. It can also declare methods of object class.

Functional Interface is also known as Single Abstract Method Interfaces or SAM Interfaces. It is a new feature in Java, which helps to achieve functional programming approach.

Example 1

```
@FunctionalInterface
interface sayable{
    void say(String msg);
}
public class FunctionalInterfaceExample implements sayable{
    public void say(String msg){
        System.out.println(msg);
    }
    public static void main(String[] args) {
        FunctionalInterfaceExample fie = new FunctionalInterfaceExample();
        fie.say("Hello there");
    }
}
```

Test it Now

Output:

```
Hello there
```

A functional interface can have methods of object class. See in the following example.

Example 2

```
@FunctionalInterface
interface sayable{
    void say(String msg);   // abstract method
    // It can contain any number of Object class methods.
    int hashCode();
    String toString();
    boolean equals(Object obj);
}
public class FunctionalInterfaceExample2 implements sayable{
    public void say(String msg){
        System.out.println(msg);
    }
    public static void main(String[] args) {
        FunctionalInterfaceExample2 fie = new FunctionalInterfaceExample2();
        fie.say("Hello there");
    }
}
```

Test it Now

Output:

```
Hello there
```

Invalid Functional Interface

A functional interface can extends another interface only when it does not have any abstract method.

```
interface sayable{
    void say(String msg);   // abstract method
}
@FunctionalInterface
interface Doable extends sayable{
    // Invalid '@FunctionalInterface' annotation; Doable is not a functional interface
    void doIt();
}
```

Output:

```
compile-time error
```

Example 3

In the following example, a functional interface is extending to a non-functional interface.

```
interface Doable{
    default void doIt(){
        System.out.println("Do it now");
    }
}
@FunctionalInterface
interface Sayable extends Doable{
    void say(String msg);   // abstract method
}
public class FunctionalInterfaceExample3 implements Sayable{
    public void say(String msg){
        System.out.println(msg);
    }
    public static void main(String[] args) {
        FunctionalInterfaceExample3 fie = new FunctionalInterfaceExample3();
        fie.say("Hello there");
        fie.doIt();
    }
}
```

Test it Now

Output:

```
Hello there
Do it now
```