

Professional Skills
Agile Fundamentals
Jira
Git
Databases Introduction
Java Beginner
<input checked="" type="radio"/> What is Java?
<input checked="" type="radio"/> Installation
<input checked="" type="radio"/> Hello World Example
<input checked="" type="radio"/> Data Types
<input checked="" type="radio"/> Packages
<input checked="" type="radio"/> Naming Conventions Cheat Sheet
<input checked="" type="radio"/> Flow of Control
<input checked="" type="radio"/> Class Members
<input checked="" type="radio"/> Operators
<input checked="" type="radio"/> Conditionals
<input checked="" type="radio"/> Iteration
<input checked="" type="radio"/> Arrays
<input checked="" type="radio"/> ArrayList
<input checked="" type="radio"/> Enhanced For Loops
<input checked="" type="radio"/> String Manipulation
<input checked="" type="radio"/> Class Constructors
<input checked="" type="radio"/> Access Modifiers
<input checked="" type="radio"/> Installing Java & Maven To PATH
<input checked="" type="radio"/> Object-Oriented Programming Principles
<input checked="" type="radio"/> Encapsulation
<input checked="" type="radio"/> Inheritance
<input checked="" type="radio"/> Polymorphism
<input checked="" type="radio"/> Abstraction
<input checked="" type="radio"/> Interfaces
<input type="radio"/> Type Casting
<input type="radio"/> Static
<input type="radio"/> Final
<input type="radio"/> Garbage Collection
<input type="radio"/> Input With Scanner
<input type="radio"/> Pass by Value/Reference
<input type="radio"/> JUnit

Interfaces

Contents

- [Overview](#)
- [Tutorial](#)
 - [Implementation](#)
 - [Multiple Interfaces](#)
- [Exercises](#)

Overview

An interface is a completely abstract class that is used to group related methods with empty bodies.

[This should not be confused with *abstract classes*, which are covered here.](#)

Tutorial

```
public interface Animal {
    void sound();
    void sleep();
}
```

Implementation

To access the interface methods, the interface must be 'implemented' by another class with the `implements` keyword. The body of the empty methods is provided by the class.

```
public class Pig implements Animal {
    public void sound() {
        System.out.println("The pig says: Oink");
    }

    public void sleep() {
        System.out.println("Zzz");
    }
}

public class MainClass {
    public static void main(String[] args) {
        Pig lilPig = new Pig();
        lilPig.sound();
        lilPig.sleep();
    }
}
```

On implementation of an interface, you must `@Override` all of its methods. Like abstract classes, interfaces cannot be used to create objects. For example; we could not instantiate an `Animal` object. For the same reason interfaces cannot contain a constructor method.

Java, unlike its close counterpart C#, does not support "multiple inheritance" - a subclass can only inherit from one superclass.

However, multiple interface implementations are allowed, which circumvents the problem somewhat.

In fact, a subclass that inherits from a superclass which has multiple interface implementations will also implement those interfaces!

<div><div></div><div>Test Driven Development</div></div> <div><div></div><div>UML Basics</div></div> <div><div></div><div>JavaDoc</div></div> <div><div></div><div>Peer Programming</div></div> <div><div></div><div>Code Reviews</div></div>
Maven
Testing (Foundation)
Java Intermediate
HTML
CSS
Javascript
Spring Boot
Selenium
Sonarqube
Advanced Testing (Theory)
Cucumber
MongoDB
Express
NodeJS
React
Express-Testing
Networking
Security
Cloud Fundamentals
AWS Foundations
AWS Intermediate
Linux
DevOps
Jenkins Introduction
Jenkins Pipeline
Markdown
IDE Cheatsheet

Multiple Interfaces

To implement multiple interfaces, simply separate them with a comma:

```
interface InterfaceOne {
    public void foo();
}

interface InterfaceTwo {
    public void bar();
}

class FooBar implements InterfaceOne, InterfaceTwo {
    public void foo() {
        System.out.println("Some text..");
    }
    public void bar() {
        System.out.println("Some other text...");
    }
}

class MainClass {
    public static void main(String[] args) {
        FooBar newObj = new FooBar();
        newObj.foo();
        newObj.bar();
    }
}
```

Exercises

There are no exercise for this module.

