COURSEWARE

Interfaces

Contents

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!

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.