**Java 接口**

接口（英文：Interface），在JAVA编程语言中是一个抽象类型，是抽象方法的集合，接口通常以interface来声明。一个类通过继承接口的方式，从而来继承接口的抽象方法。

接口并不是类，编写接口的方式和类很相似，但是它们属于不同的概念。类描述对象的属性和方法。接口则包含类要实现的方法。

除非实现接口的类是抽象类，否则该类要定义接口中的所有方法。

接口无法被实例化，但是可以被实现。一个实现接口的类，必须实现接口内所描述的所有方法，否则就必须声明为抽象类。另外，在Java中，接口类型可用来声明一个变量，他们可以成为一个空指针，或是被绑定在一个以此接口实现的对象。

接口与类相似点：

* 一个接口可以有多个方法。
* 接口文件保存在.java结尾的文件中，文件名使用接口名。
* 接口的字节码文件保存在.class结尾的文件中。
* 接口相应的字节码文件必须在与包名称相匹配的目录结构中。

接口与类的区别：

* 接口不能用于实例化对象。
* 接口没有构造方法。
* 接口中所有的方法必须是抽象方法。
* 接口不能包含成员变量，除了static和final变量。
* 接口不是被类继承了，而是要被类实现。
* 接口支持多重继承。

**接口的声明**

接口的声明语法格式如下：

[可见度] interface 接口名称 [extends 其他的类名] {

// 声明变量

// 抽象方法

}

Interface关键字用来声明一个接口。下面是接口声明的一个简单例子。

/\* 文件名 : NameOfInterface.java \*/

import java.lang.\*;

//引入包

public interface NameOfInterface

{

//任何类型 final, static 字段

//抽象方法

}

接口有以下特性：

* 接口是隐式抽象的，当声明一个接口的时候，不必使用**abstract**关键字。
* 接口中每一个方法也是隐式抽象的，声明时同样不需要**abstract**关键子。
* 接口中的方法都是公有的。

**实例**

/\* 文件名 : Animal.java \*/

interface Animal {

public void eat();

public void travel();

}

**接口的实现**

当类实现接口的时候，类要实现接口中所有的方法。否则，类必须声明为抽象的类。

类使用implements关键字实现接口。在类声明中，Implements关键字放在class声明后面。

实现一个接口的语法，可以使用这个公式：

... implements 接口名称[, 其他接口, 其他接口..., ...] ...

**实例**

/\* 文件名 : MammalInt.java \*/

public class MammalInt implements Animal{

public void eat(){

System.out.println("Mammal eats");

}

public void travel(){

System.out.println("Mammal travels");

}

public int noOfLegs(){

return 0;

}

public static void main(String args[]){

MammalInt m = new MammalInt();

m.eat();

m.travel();

}

}

以上实例编译运行结果如下:

Mammal eats

Mammal travels

重写接口中声明的方法时，需要注意以下规则：

* 类在实现接口的方法时，不能抛出强制性异常，只能在接口中，或者继承接口的抽象类中抛出该强制性异常。
* 类在重写方法时要保持一致的方法名，并且应该保持相同或者相兼容的返回值类型。
* 如果实现接口的类是抽象类，那么就没必要实现该接口的方法。

在实现接口的时候，也要注意一些规则：

* 一个类可以同时实现多个接口。
* 一个类只能继承一个类，但是能实现多个接口。
* 一个接口能继承另一个接口，这和类之间的继承比较相似。

**接口的继承**

一个接口能继承另一个接口，和类之间的继承方式比较相似。接口的继承使用extends关键字，子接口继承父接口的方法。

下面的Sports接口被Hockey和Football接口继承：

// 文件名: Sports.java

public interface Sports

{

public void setHomeTeam(String name);

public void setVisitingTeam(String name);

}

// 文件名: Football.java

public interface Football extends Sports

{

public void homeTeamScored(int points);

public void visitingTeamScored(int points);

public void endOfQuarter(int quarter);

}

// 文件名: Hockey.java

public interface Hockey extends Sports

{

public void homeGoalScored();

public void visitingGoalScored();

public void endOfPeriod(int period);

public void overtimePeriod(int ot);

}

Hockey接口自己声明了四个方法，从Sports接口继承了两个方法，这样，实现Hockey接口的类需要实现六个方法。

相似的，实现Football接口的类需要实现五个方法，其中两个来自于Sports接口。

**接口的多重继承**

在Java中，类的多重继承是不合法，但接口允许多重继承，。

在接口的多重继承中extends关键字只需要使用一次，在其后跟着继承接口。 如下所示：

public interface Hockey extends Sports, Event

以上的程序片段是合法定义的子接口，与类不同的是，接口允许多重继承，而 Sports及 Event 可能定义或是继承相同的方法

**标记接口**

最常用的继承接口是没有包含任何方法的接口。

标识接口是没有任何方法和属性的接口.它仅仅表明它的类属于一个特定的类型,供其他代码来测试允许做一些事情。

标识接口作用：简单形象的说就是给某个对象打个标（盖个戳），使对象拥有某个或某些特权。

例如：java.awt.event包中的MouseListener接口继承的java.util.EventListener接口定义如下：

package java.util;

public interface EventListener

{}

没有任何方法的接口被称为标记接口。标记接口主要用于以下两种目的：

* **建立一个公共的父接口：**

正如EventListener接口，这是由几十个其他接口扩展的Java API，你可以使用一个标记接口来建立一组接口的父接口。例如：当一个接口继承了EventListener接口，Java虚拟机(JVM)就知道该接口将要被用于一个事件的代理方案。

* **向一个类添加数据类型：**

这种情况是标记接口最初的目的，实现标记接口的类不需要定义任何接口方法(因为标记接口根本就没有方法)，但是该类通过多态性变成一个接口类型。