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。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。

接口与类的区别：

- 接口不能用于实例化对象。但可声明引用变量类型。
- 接口中所有的方法必须是（隐式，不需要**abstract**关键子）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。
- 接口是隐式抽象的，不必使用**abstract**关键字。

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
}
```

1. 访问修饰符：只能是public（默认）。
2. 接口名：和类名采用相同命名机制。
3. extends：接口可以多继承。
4. 常量：接口中的属性只能是变量（public static final 可省）。
5. 方法：接口中的方法只能是抽象方法：public abstract（可省，JDK 1.8 省略为default）。JDK1.8后，接口中包含普通的静态方

法。JDK 1.9时，接口中的方法可以是private的。

接口的实现

类实现接口要**实现接口中所有的方法**，并且这些方法只能是public的。否则，类必须声明为抽象的类。

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

- 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。

接口的对象可以利用子类对象的向上转型进行实例化：

```
interface A{//定义一个接口A
    public static final String MSG = "hello";//全局常量
    public abstract void print();//抽象方法
}

interface B{//定义一个接口B
    public abstract void get();
}

class X implements A,B{//X类实现了A和B两个接口
    @Override
    public void print() {
        System.out.println("接口A的抽象方法print()");
    }
    @Override
    public void get() {
        System.out.println("接口B的抽象方法get()");
    }
```

```
}

public class TestDemo {
    public static void main(String[] args){
        X x = new X();//实例化子类对象
        A a = x;//向上转型
        B b = x;//向上转型
        a.print();
        b.get();
    }
}
```

结果：

```
接口A的抽象方法print()
接口B的抽象方法get()
```

```
    public static void main(String[] args){
        A a = new X();
        B b = (B) a;
        b.get();
    }
```

运行结果：

```
接口B的抽象方法get()
```

接口的继承

接口的继承使用extends关键字，子接口继承父接口的方法。

```
// 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);
}
```

接口的多重继承：

```
public interface Hockey extends Sports, Event
```

Sports及 Event 可能定义或是继承**相同的方法**

标记接口

没有任何方法和属性的接口

- **建立一个公共的父接口：**

如EventListener接口（由几十个其他接口扩展的Java API），可以使用一个标记接口来建立一组接口的父接口。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：**

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。