参考：

《Java核心技术卷一》

前言

接口是个老生长谈的东西，也是很重要的东西，再初入门的Java新手耳边都会经常听到什么什么面向接口编程。那么这里详细给接口做个整理，顺便对比下抽象类（这个暂时就不主要整理了）。

demo

```
public interface Demo {

    void getSth();
}
```

这就是个标准的接口，并且其中写了一个空方法体方法。

接口简单介绍

结合以上demo，我们先简单介绍下接口。

1.接口区别类用interface修饰；

2.接口的普通方法没有方法体（为什么强调普通，下文再讲）；

3.接口的方法你可以不加修饰符，其实就是默认的public，你要手写也行，但也只能是public；

4.接口是没有构造方法的，别以为不写就是有个默认的，接口没有。

接口的继承问题以及实现问题

我们对之前的接口做了以下改造。

```
public interface Demo extends DemoParent, DemoBrother {

    public void getSth();
```

```
}
```

这个接口就很高级了，继承了两个父接口，有人纳闷了，Java不是单继承嘛，你这边肯定是错的，嗯，这么问的肯定是优秀的同学。但忽略了一点，单继承是针对类的，这边是接口，接口是允许多继承的。

那么如何去用这个接口呢。

```
public class DemoClass implements Demo {

	@Override
	public void getSth() {

	}

}
```

这就是一个标准的接口实现类，我们来介绍下，首先呢，“继承”接口不是通过extends而是通过implements，所以我们一般不说继承，而是实现了某某接口，顺便说下，实现接口也是多实现的。这里就可以提到类单继承的概念，C++是多继承的，Java是基于C++的，继承了C++很多特性，但在多继承这个概念上，开发人员最终还是放弃了，因为多继承引发了复杂性和低效性的问题，但是单继承肯定会让代码受限，最终开发了接口这个伪多继承的东西，但是效果意外的好。

我们可以看到getSth这个方法，这不是我们想写的，是必须写的，接口的中方法必须在实现类中得到实现。

接口的特殊方法

Java8之前你可以就像上面那样去理解接口了，然而事物的存在就是要去打破的，对于接口Java8也推出了新特性，接口不能有方法体，不存在，这样写你就可以有。

```
public interface Demo extends DemoParent, DemoBrother {

	public void getSth();

	static void testStatic() {
		System.out.println("it's static");
	}
```

```
    default void testDefault() {
        System.out.println("it's default");
    }
}
```

可以看到我们多了俩方法testStatic和testDefault，不过这俩方法有点特殊，一个被static修饰，一个被default修饰，这就是Java8的新特性了。

被这样修饰后，接口中的方法也就能有自己的方法体了，具体用法就像这样。

```
public class DemoClass implements Demo {

    @Override
    public void getSth() {
        testDefault();
        Demo.testStatic();
    }

    public static void main(String[] args) {
        new DemoClass().getSth();
    }

}
```

default修饰的方法就像你继承父类时调父类方法一样用就是了，static更简单了，平常的static方法咋用，这边就咋用。

默认方法的二义性问题

比如两个接口中都有testDefault方法，这时候有个类同时实现这两个接口，那么到底算哪个接口的testDefault呢，这时候会编译报错，我们必须在实现类中重写该方法，并指明是哪个接口的方法。

```
public class DemoClass implements Demo, Demo2 {

    @Override
    public void getSth() {
        testDefault();
        Demo.testStatic();
    }

    @Override
    public void testDefault() {
        Demo.super.testDefault();
```

```
    }

    public static void main(String[] args) {
        new DemoClass().getSth();
    }

}
```

那么还有种呢，比如我的类继承了一个类，父类和接口的testDefault重复了会咋样呢，其实这时候你不要动啥，这里有个超类优先原则，就像这样。

```
public class DemoClass2 {

    public void testDefault() {
        System.out.println("it's a class");
    }
}
```

```
public class DemoClass extends DemoClass2 implements Demo {

    @Override
    public void getSth() {
        testDefault();
        Demo.testStatic();
    }

    public static void main(String[] args) {
        new DemoClass().getSth();
    }

}
```

结果。

```
it's a class
it's static
```

这时候我们调用testDefault，虽然接口中也有，但因为超类中存在，直接会无视接口。

与抽象类的对比

1. 抽象类可以有构造方法，但不能被实例化，构造方法只能在构造匿名内部类时使用，也就是说抽象类比接口多个构造方法的用途就在匿名内部类可以赋参数；

2.抽象类可以有普通属性，方法，静态属性和静态方法，而接口怎么说呢，至少没有普通属性和方法，Java8提供了静态方法，同时静态属性也不存粹只能是static final，相当于常量；

3.抽象类可以有抽象方法（abstract修饰），接口可以这么写却不正确，JVM会把它当作抽象类对待，会有隐患问题；

4.抽象类只能被继承一个，接口可以被实现好多个。