接口

约定好规范，然后按照规范来做。接口就是定义规范。

`java` 中的接口作用和生活中类似，它提供一种约定，使实现接口的类在形式上保持一致。

抽象类中可以有普通方法而接口中的方法默认都是抽象的，也可以说接口是一个特殊的 `抽象类`，接口不能被实例化，而且没有构造方法。

定义接口

```
[修饰符] interface 接口名{
    //接口成员
}
```

```
public interface USBInterface {

    public static final String NAME = "";//静态常量


    public void service();
}
```

接口中的方法默认是抽象方法，所以可以省略 `abstract` 修饰符

接口中的方法默认都是 `public` 的，所以可以省略 `public`

接口中的变量只能是静态常量(`static final`),所以可以省略 `static final`，静态常量在定义时就要赋值，且不可变。

使用接口

接口使用和抽象类一样，都是通过子类。子类通过 `implements` 关键字实现接口，实现接口就必须实现接口中的抽象方法

```java
public class USBDisk implements USBInterface{

    @Override
    public void service() {
        System.out.println("service()");
    }

}
```

- 一个类可以实现多个接口，接口之间使用`,`隔开
- 接口的实现类中，可以有普通方法
- 实现的方法必须是`public`的

实现多个接口

`java`中继承是单继承,使用`extends`关键字；但是一个类可以实现多个接口，使用`implements`，多个接口之间用`,`隔开。

```java
public class Computer implements USBInterface, ChargeInterface{


    public void play() {
        System.out.println("play game");
    }

    @Override
    public void charge() {
        System.out.println("充电");
    }

    @Override
    public void service() {
        System.out.println("USB接口");
    }

}
```

一个类可以同时继承和实现接口，`extends`要在`implements`之前

```java
public class LenovoComputer extends Computer implements USBInterface,
ChargeInterface{

}
```

接口与接口之间是继承关系，使用`extends`关键字。多个接口使用`,`隔开

```
public interface USBC extends USBInterface,ChargeInterface{

}
```

JDK8 接口新特性

在 `jdk8.0` 中 `default` 关键字可用于在接口中修饰方法（默认方法），`default` 修饰的方法可以有具体实现，也只能在接口中出现。 `default` 修饰的方法可以被重写。

```
public interface USBInterface {

    public static final String NAME = "";



    void service();


    public default void test() {
        service();
        System.out.println("service test");
    }
}
public class UDBDisk implements USBInterface{

    @Override
    public void service() {
        System.out.println("service()");
    }

    @Override
    public void test() {

    }

    public static void main(String[] args) {
        USBInterface u = new UDBDisk();
        u.test();
    }
}
```

> 默认方法可以在不破坏已经在使用该接口的所有代码。默认方法有时也称为防御方法（defender method）或 虚拟扩展方法（virtual extension method）

接口中还可以有`static`修饰的方法，称为静态方法(类方法)。`static`方法必须直接使用接口名.方法名调用。

```
public interface USBInterface {
    public static void test1() {
        System.out.println("test1");
    }
}
```

JDK9 接口新特性

`JDK9`接口中可以使用`private`修饰方法。供接口中其他方法调用。

```
public interface USBInterface {
    private void privateMethod() {
        System.out.println("privateMethod");
    }

    default void print(){
        privateMethod();
    }
}
```

抽象类和接口的区别

特性	接口	抽象类
组合	可以在新类中组合多个接口	只能继承一个抽象类
状态	不能包含字段（静态字段除外，但不支持对象状态（实例字段））	可以包含字段
默认方法和抽象方法	默认方法不需要在子类里实现，它只能引用接口中的方法	抽象方法必须在子类里实现
构造器	不能有构造器	有构造器
访问权限	隐式 public	可以为 protected 或包访问权限

抽象类任然是一个类，因此如果被创建的新类所继承，则该抽象类就应该是唯一被继承的类。在创建新类的过程中可以实现多个接口。

一个经验法则是“在合理的范围内尽可能抽象”。因此，和抽象类相比，我们更偏向于使用接口。但是，大多数情况下常规类可以解决问题如果不能再使用接口和抽象类。