1. 接口的定义

接口类似于抽象类，也是定义一些抽象方法，通过实现类来实现这些接口。接口就是一种规范标准，只要符合标准，大家都可以用。

1. 接口的特点
2. 接口用关键字interface修饰

Public interface 接口名{}

1. 类实现接口用implements表示

public class 类名 implements 接口名 {}

1. 接口不能直接实例化

和抽象类一样，接口也不能直接实例化，因为定义了抽象方法。因此，只有通过实现类实现接口，重写接口中的所有抽象方法，才能将实现类对象实例化。这叫接口多态。

1. 接口的实现类
2. 重写接口中的所有抽象方法，能够对象实例化
3. 没有重写接口中的所有抽象方法，依然是抽象类。
4. 接口的成员特点
5. 接口的成员变量

接口中的成员变量只能是常量，且默认的修饰符是 public static final。即如果你仅仅写 int num ，编译器默认添加的修饰符就是public static final int num。

1. 接口的构造方法

接口是没有构造方法的，因为接口只是对于行为的抽象，是没有具体的实现的。那么我们在接口的实现类的构造函数中调用super()为什么不会报错呢？

因为，所有类的父类都是Object类，默认继承Object类，因此，调用super()并不会报错。

1. 接口的成员方法

接口中的成员方法只能是抽象方法，就算定义了普通方法，也会添加修饰符定义为抽象方法，默认修饰符为：public abstract。

1. 类和接口的关系

（一）类和类的关系

继承关系，只能单继承，但是可以多层继承

（二）类和接口的关系

实现关系，可以单实现，也可以多实现，还可以在继承一个类的同时实现多个接口

（三）接口与接口的关系

继承关系，可以单继承，也可以多继承

1. 抽象类和接口的区别
2. 成员的区别
3. 抽象类

可以有变量和常量；可以有构造方法；可以有抽象方法，也可以有非抽象方法。

1. 接口

只有常量；没有构造方法；只有抽象方法。

1. 设计理念区别
2. 抽象类

对类进行抽象，一般是行为和属性。

1. 接口

对行为进行抽象。

1. 抽象类和接口的案例思考

实现门和警报，如果我们将门和警报都定义在一个抽象类中，是不合适的。因为门的基本共性是打开和关闭，但是警报功能并不是所有门通用的，因此需要用接口实现警报功能。

Public interface Alarm{

Void alarm();

}

Public abstract class Door{

Public abstract void open();

Public abstract void close();

}

Public class AlarmDoor extends Door implements Alarm{

Public void open(){…}

Public void close(){…}

Public void alarm(){…}

}