接口：相当于一个（抽象类），比抽象类更抽象

抽象类：

注意：

1.在一个类中，存在方法没有方法体，则该方法需要用abstract修饰，该类也必须用abstracr修饰

2.创建抽象类的目的是为了被继承，根据子类的实际需求来进行不同的实现，所以它无需创建实例对象，也不能创建对象。

因为创建对象的目的是为了调用非抽象的成员方法，但是抽象类不能创建对象来，~~所以抽象类也就没有必要出现非抽象的方法~~（可以出现非抽象的方法）。

即：~~实际开发中，抽象类里面只会出现抽象方法~~（可能出现成员变量和构造方法，因为可以被子类继承）。

3.子类必须实现抽象类的所有抽象方法，如果子类没有实现父类的抽象方法，必须用abstract修饰子类。

子类需要被继承，即为了只出现父类中需要实现的抽象方法！！！

应用场景：

1. 子类对父类里面的方法实际需求不同，则考虑父类用抽象类
2. 强制子类必须重写父类里面的方法，即强调规范

1.接口的定义：

interface 接口名{

成员变量；（成员变量都是常量，默认修饰符为public abstract final,一般不写常量）

——>没有构造方法（构造方法是用来初始化变量的）

成员方法；（成员方法都是抽象方法,默认修饰符为public abstract）

——>非抽象类实现一个接口，必须要把接口的全部方法实现

~~抽象类实现某个接口，可以不实现接口的一些方法（一般不会用抽象类去实现某个接口，因为抽象类自己就可以声名一个非抽象的方法）~~

}

接口和抽象类和非抽象类的关系

接口{

}

抽象类1 implements 接口{

}

抽象类2 extends 抽象类1{

}

非抽象类1extends 抽象类2{

}

2.接口的实现：

class 类名 implements 接口名1,接口名2（可以实现多个接口）{

}

3.接口与接口的继承：一个接口可以继承多个接口(因为接口的方法是抽象的，)

interface 接口1{

}

interface 接口2 {

}

interface 接口3 extends 接口2 extends 接口3{

}