**抽象类**

**1.抽象类概述 abstract**

**抽象方法:没有方法体的方法 public abstract void eat();**

**抽象类:有抽象方法的类 public abstract class Animal{}**

**一个没有方法体的方法定义为抽象方法,一个有抽象方法的类必须是抽象类**

**2.抽象类的特点**

**抽象类和抽象方法必须用 abstract 关键字修饰**

**抽象方法: public abstract void eat();**

**抽象类: public abstract class Animal{}**

**抽象类中不一定有抽象方法,由抽象方法的类一定是抽象类**

**抽象类不能实例化**

**需要通过子类对象实例化,这叫抽象类多态**

**抽象类的子类**

**要么本身也是子类**

**要么重写父类的所有抽象方法**

**3.抽象类的成员特点**

**成员变量**

**可以是变量,也可以是常量**

**构造方法**

**可以有无参和带参构造方法,但是本身不能实例化,是用于子类访问父类数据的初始化,也就是给子类用的.**

**成员方法**

**有抽象方法:限定子类必须要重写**

**有非抽象方法:提高代码的复用性**

**4.猫狗案例抽象类版**

**接口**

**1.接口概述**

**接口就是一种公共的规范标准,只要符合标准,大家都可以用.**

**Java中的接口主要用在对行为的抽象**

**2.接口的特点 interface implements**

**接口用关键字interface修饰**

**Public interface 接口名 {}**

**类实现接口用implements表示**

**Public class 类名 implements 接口名{}**

**接口不能实例化**

**接口通过实现类对象进行实例化,也叫接口多态.**

**多态的形式:具体类多态,抽象类多态,接口多态**

**多态的前提:有继承或者实现关系;有方法重写;有父(类/接口)引用指向子(类/实现)对象.**

**接口的实现类**

**要么重写接口中的所有抽象方法**

**要么本身也是抽象类**

**3.接口的成员特点**

**成员变量**

**只能是常量,并且可以通过类名调用**

**默认修饰符: public static final**

**构造方法**

**没有构造方法,因为接口主要是对行为进行抽象的,没有具体存在**

**如果一个类没有具体的父类,那就默认继承自Object类**

**成员方法**

**只能是抽象方法**

**默认修饰符: public abstract**

**JDK8版中接口成员特点:**

**JDK8版本后:**

**<1>.允许在接口中定义非抽象方法,但是余姚使用关键字default修饰,这些方法就是默认方法**

**作用:解决接口升级的问题,在接口中增加新功能,不影响实现类,因为默认方法不是抽象的,不需要重写.而实现类可以重写使用,也可以不重写,直接使用**

**接口中默认方法的定义格式:**

**格式 : public default 返回值类型 方法名(参数列表){}**

**接口中默认方法的注意事项:**

**1.默认方法不是抽象方法,所以不强制被重写,但是可以被重写,重写的时候去掉default关键字**

**2.public可以省略,default不能省略**

**3.如果实现了多个接口,多个接口中存在相同的方法声明,子类就必须对这个默认方法进行重写**

**<2>.接口中允许定义static静态方法**

**接口中静态方法的定义格式:**

**public static 返回值类型 方法名(参数列表){}**

**接口中静态方法的注意事项:**

**1.静态方法只能通过接口名调用,不能通过实现类名或者实现类对象名调用**

**2.public可以省略,static不能省略**

**接口的使用思路:**

**1.如果发现一个类中所有的方法都是抽象方法,那么就可以将该类改进为一个接口;**

**2.涉及到接口大面积更新方法,而不想去修改每一个实现类,就可以将更新的方法定义为带有方法体的默认方法**

**3.希望默认方法在调用的时候更加简单,就可以用静态修饰,使用的时候要去掉default关键字;**

**4.默认方法中出现了重复的代码,可以抽取出一个私有方法,用private修饰, 使用的时候要去掉default关键字;**

**4.猫狗案例接口版**

**练习**

**5.类和接口的关系**

**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情况下实现多个接口**

**public class Demo extends Object implements Inter1, Inter2, Inter3{}**

**类和抽象类的关系**

**继承关系,只能单继承,但是可以多层继承**

**public class Demo extends Fu{} public class Fu extends Object{}**

**接口和接口的关系**

**继承关系,可以多继承,互相继承**

**public interface Inter extends Inter1, Inter2, Inter3{}**

**6.抽象类和接口的区别**

**语法区别**

**成员区别**

**抽象类:有变量,常量;有构造方法;有抽象方法,也有非抽象方法**

**接口:只有常量;没有构造方法;只有抽象方法**

**关系区别**

**类与抽象类:继承,单继承,但是可以多层继承**

**类与接口:实现,多实现,可以在继承一个类的情况下多实现**

**接口与接口:继承,多继承,网格继承相互继承**

**设计理念区别**

**抽象类: 对类抽象,包括属性行为**

**接口: 对行为抽象,主要是行为**

**7.运动员和教练案例**

**练习**

**补充点**

**1.类名作为形参和返回值**

**2.抽象类名作为形参和返回值**

**方法的形参是抽象类名,其实需要的是该抽象类的子类作为对象**

**方法的返回值是抽象类名,其实返回的是该抽象类的子类对象**

**抽象类不能实例化,多态为子类参与实际操作**

**3.接口名作为形参和返回值**

**方法的形参是接口名,其实需要的是该接口的实现类作为对象**

**方法的返回值是接口名,其实返回的是该接口的实现类对象**

**实现类也不能实例化,多态为实现类参与实际操作,所以需要创建一个实现类**