面向对象高级(二)

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

目录

Contents

什么是多态?

- 多态是在继承/实现情况下的一种现象，表现为：对象多态、行为多态。

多态的具体代码体现

```
People p1 = new Student();
p1.run();

People p2 = new Teacher();
p2.run();
```

多态的前提

- 有继承/实现关系；存在父类引用子类对象；存在方法重写。

多态的一个注意事项

- 多态是对象、行为的多态，Java中的属性(成员变量)不谈多态。

目录

Contents

使用多态的好处

- 在多态形式下，右边对象是解耦合的，更便于扩展和维护。

```
People p1 = new Teacher();

p1.run();
```

- 定义方法时，使用父类类型的形参，可以接收一切子类对象，扩展性更强、更便利。

多态下会产生的一个问题，怎么解决?

- 多态下不能使用子类的独有功能。

类型转换

- 自动类型转换：父类 变量名 = new 子类();

例如：People p = new Teacher();

- 强制类型转换：子类 变量名 = (子类) 父类变量

例如 Teacher t = (Teacher)p;

强制类型转换的一个注意事项

- 存在继承/实现关系就可以在编译阶段进行强制类型转换，编译阶段不会报错。
- 运行时，如果发现对象的真实类型与强转后的类型不同，就会报类型转换异常（ClassCastException）的错误出来。

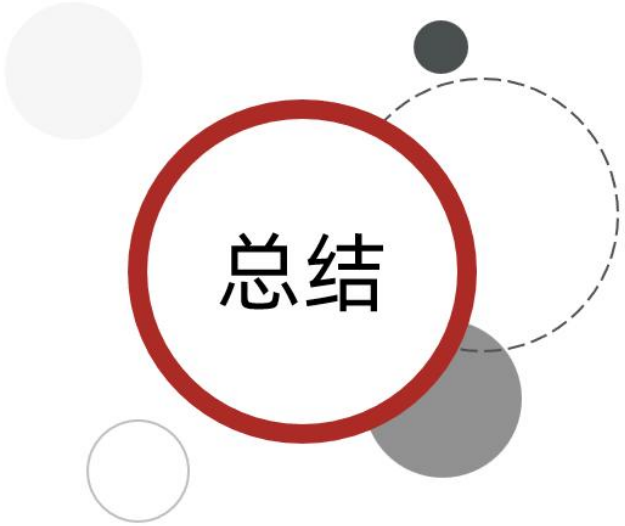
```
People p = new Teacher();

Student s = (Student) p;  // java.lang.ClassCastException
```

强转前，Java建议：

- 使用instanceof关键字，判断当前对象的真实类型，再进行强转。

```
p instanceof Student
```



1、使用多态有什么好处？存在什么问题？

- **可以解耦合，扩展性更强；使用父类类型的变量作为方法的形参时，可以接收一切子类对象。**
- **多态下不能直接调用子类的独有方法。**

2、类型转换有几种形式？能解决什么问题？

- **自动类型转换、强制类型转换。**
- **可以把对象转换成其真正的类型，从而解决了多态下不能调用子类独有方法的问题。**

3、强制类型转换需要注意什么？

- **存在继承/实现时，就可以进行强制类型转换，编译阶段不会报错。**
- **但是，运行时，如果发现对象的真实类型与强转后的类型不同会报错（ClassCastException）。**

4、强制类型转换前？Java建议我们做什么事情？

- **使用instanceof判断当前对象的真实类型：对象 instanceof 类型 。**

目录

Contents

final

- final 关键字是最终的意思，可以修饰（类、方法、变量）
- 修饰类：该类被称为最终类，特点是不能被继承了。
- 修饰方法：该方法被称为最终方法，特点是不能被重写了。
- 修饰变量：该变量只能被赋值一次。

final修饰变量的注意

- final修饰基本类型的变量，变量存储的**数据**不能被改变。
- final修饰引用类型的变量，变量存储的**地址**不能被改变，但地址所指向对象的内容是可以被改变的。

目录

Contents

常量

- 使用了 static final 修饰的成员变量就被称为常量；
- 作用：通常用于记录系统的配置信息。

```
public class Constant {
    public static final String SCHOOL_NAME = “传智教育";



}
```

注意！常量名的命名规范：建议使用大写英文单词，多个单词使用下划线连接起来。

使用常量记录系统配置信息的优势、执行原理

- 代码可读性更好，可维护性也更好。
- 程序编译后，常量会被“宏替换”：出现常量的地方全部会被替换成其记住的字面量，这样可以保证使用常量和直接用字面量的性能是一样的。

目录

Contents

什么是抽象类?

- 在Java中有一个关键字叫：abstract，它就是抽象的意思，可以用它修饰类、成员方法。
- abstract修饰类，这个类就是抽象类；修饰方法，这个方法就是抽象方法。

```
修饰符 abstract class 类名{
        修饰符 abstract 返回值类型 方法名称(形参列表);
}
```

```
public abstract class A {
    // 抽象方法：必须abstract修饰，只有方法签名，不能有方法体。
    public abstract void test();
}
```

抽象类的注意事项、特点

- 抽象类中不一定有抽象方法，有抽象方法的类一定是抽象类。
- 类该有的成员（成员变量、方法、构造器）抽象类都可以有。
- **抽象类最主要的特点：**抽象类不能创建对象，仅作为一种特殊的父类，让子类继承并实现。
- 一个类继承抽象类，必须重写完抽象类的全部抽象方法，否则这个类也必须定义成抽象类。

抽象类的作用和好处

1、抽象类、抽象方法是什么样的？

- **都是用abstract修饰的；抽象方法只有方法签名，不能写方法体。**

2、抽象类有哪些注意事项和特点？

- **抽象类中可以不写抽象方法，但有抽象方法的类一定是抽象类**
- **类有的成员（成员变量、方法、构造器）抽象类都具备。**
- **抽象类不能创建对象，仅作为一种特殊的父类，让子类继承并实现。**
- **一个类继承抽象类，必须重写完抽象类的全部抽象方法，否则这个类也必须定义成抽象类。**

认识了抽象类

抽象类的场景和好处

抽象类的场景和好处

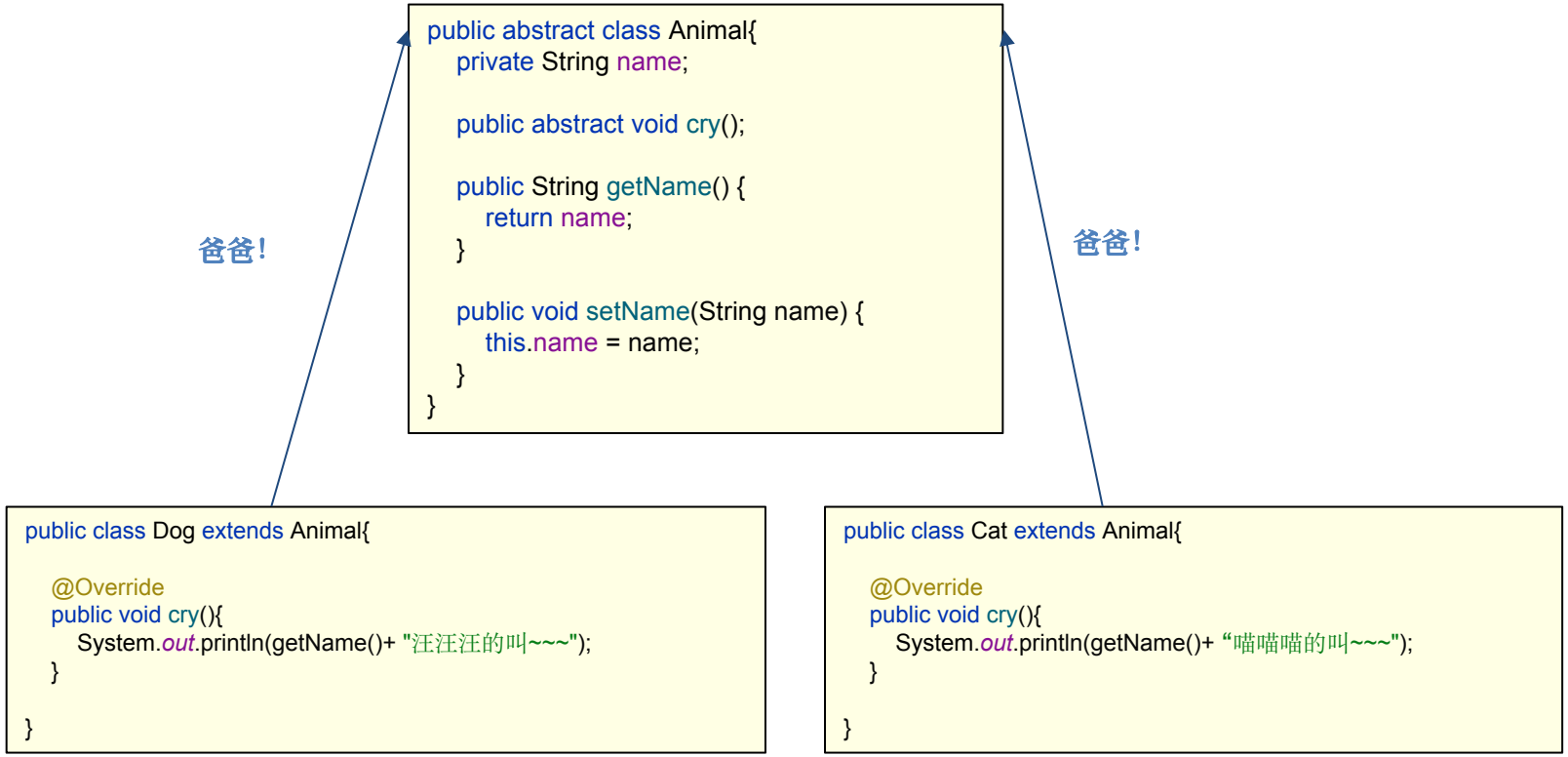
需求

某宠物游戏，需要管理猫、狗的数据。

猫的数据有：名字；行为是：喵喵喵的叫~

狗的数据有：名字；行为是：汪汪汪的叫~

请用面向对象编程设计该程序。



- 多个类中只要有重复代码（包括相同的方法签名），我们都应该抽取到父类中去，此时，父类中就有可能存在只有方法签名的方法，这时，父类必定是一个抽象类了，我们抽出这样的抽象类，就是为了更好的支持多态。

1、抽象类的应用场景和好处是什么？

- **两种主要的应用场景，一种是：用抽象类，我们可以把子类中相同的代码，包括方法签名都抽上来，这样能更好的支持多态，以提高代码的灵活性。**
- **一种是：反过来用，我们不知道系统未来具体的业务实现时，我们可以先定义抽象类，将来让子类去继承实现，以方便系统的扩展。**

目录

Contents

面试笔试

源码

1、解决了什么问题？

2、怎么写？

模板方法设计模式解决了什么问题?

- 解决方法中存在重复代码的问题。

```
public class A {
    ...

    public void sing(){
        代码一样

        代码不同

        代码一样
    }
    ...
}
```

```
public class B {
    ...

    public void sing(){
        代码一样

        代码不同

        代码一样
    }
    ...
}
```

模板方法设计模式的写法

- 1、定义一个抽象类。
- 2、在里面定义2个方法
  - 一个是模板方法：把相同代码放里面去。
  - 一个是抽象方法：具体实现交给子类完成。

多学一招：建议使用final关键字修饰模板方法，为什么？

- 模板方法是给对象直接使用的，不能被子类重写。
- 一旦子类重写了模板方法，模板方法就失效了。

案例 模板方法设计模式的应用

需求

黑马的员工管理系统中，需要管理讲师、咨询师的数据

讲师的数据有：姓名、年龄、具备的技能；

咨询的数据有：姓名、年龄、回答问题的总人数。

讲师和咨询师都需要提供打印个人介绍的方法。

模板方法模式实现步骤

- 1、定义一个抽象类。
- 2、在里面定义2个方法（一个是模板方法：把相同代码放里面去，一个是抽象方法：具体实现交给子类完成）。
- 3、定义子类继承抽象类，重写抽象方法。
- 4、创建子类对象，调用模板方法完成功能。

1、模板方法设计模式解决了什么问题？

- **解决方法中存在重复代码的问题。**

2、模板方法设计模式应该怎么写？

- **定义一个抽象类。**
- **在里面定义2个方法，一个是模板方法：放相同的代码里，一个是抽象方法：具体实现交给子类完成。**

3、模板方法建议使用什么关键字修饰？为什么

- **建议使用final关键字修饰模板方法。**

目录

Contents

认识接口

- Java提供了一个关键字interface，用这个关键字我们可以定义出一个特殊的结构：接口。

```
public interface 接口名 {
    // 成员变量（常量）
    // 成员方法（抽象方法）
}
```

- 注意：接口不能创建对象；接口是用来被类实现（implements）的，实现接口的类称为实现类。

```
修饰符 class 实现类 implements 接口1, 接口2, 接口3 , ... {

}
```

- 一个类可以实现多个接口（接口可以理解成干爹），实现类实现多个接口，必须重写完全部接口的全部抽象方法，否则实现类需要定义成抽象类。

1、接口是什么？

- **使用interface关键字定义的一种结构，JDK 8之前，接口中只能定义成员变量和成员方法。**

2、接口怎么使用？需要注意什么？

- **接口 是被类 实现的。**

```
修饰符 class 实现类 implements 接口1, 接口2, 接口3 , ... {

}
```

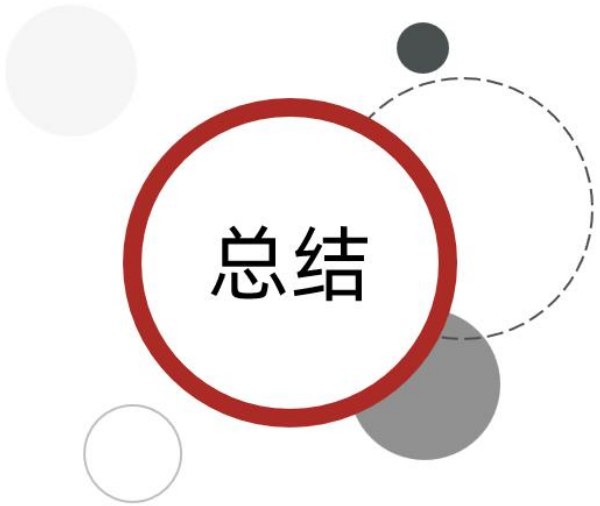
- **实现类实现多个接口，必须重写完全部接口的全部抽象方法，否则实现类需要定义成抽象类。**

3、接口的好处是啥？

- **弥补了类单继承的不足，类可以同时实现多个接口。**
- **让程序可以面向接口编程，这样既不用关心实现的细节，也可以灵活方便的切换各种实现。**

接口的好处（重点）

- 弥补了类单继承的不足，一个类同时可以实现多个接口。
- 让程序可以面向接口编程，这样程序员就可以灵活方便的切换各种业务实现。



1、使用接口有啥好处，第一个好处是什么？

- **可以解决类单继承的问题，通过接口，我们可以让一个类有一个亲爹的同时，还可以找多个干爹去扩展自己的功能。**

2、为什么我们要通过接口，也就是去找干爹，来扩展自己的功能呢？

- **因为通过接口去找干爹，别人通过你implements的接口，就可以显性的知道你是谁，从而也就可以放心的把你当作谁来用了。**

3、使用接口的第二个好处是什么？

- **一个类我们说可以实现多个接口，同样，一个接口也可以被多个类实现的。这样做的好处是我们的程序就可以面向接口编程了，这样我们程序员就可以很方便的灵活切换各种业务实现了。**

目录

Contents

案例 接口的应用案例：班级学生信息管理模块的开发

需求

请设计一个班级学生的信息管理模块：学生的数据有：姓名、性别、成绩

功能1：要求打印出全班学生的信息； 功能2：要求打印出全班学生的平均成绩。

注意！以上功能的业务实现是有多套方案的，比如：

第1套方案：能打印出班级全部学生的信息；能打印班级全部学生的平均分。

第2套方案：能打印出班级全部学生的信息（包含男女人数）；能打印班级全部学生的平均分（要求是去掉最高分、最低分）。

要求：系统可以支持灵活的切换这些实现方案。

目录

Contents

JDK 8开始，接口新增了三种形式的方法：

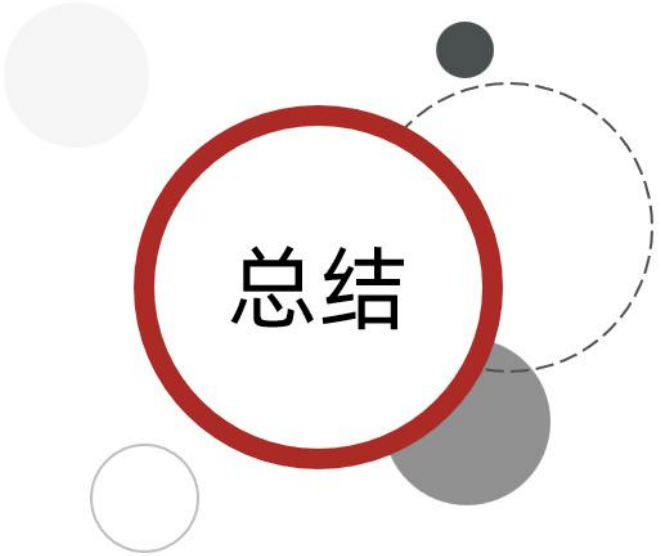
```
public interface A{
    /**
     * 1、默认方法（实例方法）：使用用default修饰，默认会被加上public修饰。
     * 注意：只能使用接口的实现类对象调用
     */
    default void test1(){
        ...
    }

    /**
     * 2、私有方法：必须用private修饰(JDK 9开始才支持)
     */
    private void test2(){
        ...
    }

    /**
     * 3、类方法（静态方法）：使用static修饰，默认会被加上public修饰。
     * 注意：只能用接口名来调用。
     */
    static void test3(){
        ...
    }
}
```

JDK8开始，接口中为啥要新增这些方法?

- 增强了接口的能力，更便于项目的扩展和维护。



1、JDK8开始，接口中新增了哪些方法?

- **默认方法：使用default修饰，使用实现类的对象调用。**
- **静态方法：static修饰，必须用当前接口名调用**
- **私有方法：private修饰，jdk9开始才有的，只能在接口内部被调用。**
- **他们都会默认被public修饰。**

2、JDK8开始，接口中为啥要新增这些方法?

- **增强了接口的能力，更便于项目的扩展和维护。**

目录

Contents

接口的多继承

一个接口可以同时继承多个接口

```
public interface C extends B , A{

}
```

接口多继承的作用

- 便于实现类去实现。

接口其他注意事项（了解）

1、一个接口继承多个接口，如果多个接口中存在方法签名冲突，则此时不支持多继承。

2、一个类实现多个接口，如果多个接口中存在方法签名冲突，则此时不支持多实现。

3、一个类继承了父类，又同时实现了接口，父类中和接口中有同名的默认方法，实现类会优先用父类的。

4、一个类实现了多个接口，多个接口中存在同名的默认方法，可以不冲突，这个类重写该方法即可。

传智教育旗下高端IT教育品牌

传智教育旗下高端IT教育品牌