面向对象进阶

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

目录

Contents

什么是包？

- 包是用来分门别类的管理各种不同类的，类似于文件夹、建包利于程序的管理和维护。
- 建包的语法格式：package 公司域名倒写.技术名称。报名建议全部英文小写，且具备意义

```
package com.itheima.javabean;
public class Student {

}
```

- 建包语句必须在第一行，一般IDEA工具会帮助创建

导包

- 相同包下的类可以直接访问，不同包下的类必须导包,才可以使用！导包格式：**import 包名.类名;**
- 假如一个类中需要用到不同类，而这个两个类的名称是一样的，那么默认只能导入一个类，另一个类要带包名访问。

目录

Contents

什么是权限修饰符？

- 权限修饰符：是用来控制一个成员能够被访问的范围。
- 可以修饰成员变量，方法，构造器，内部类，不同权限修饰符修饰的成员能够被访问的范围将受到限制。

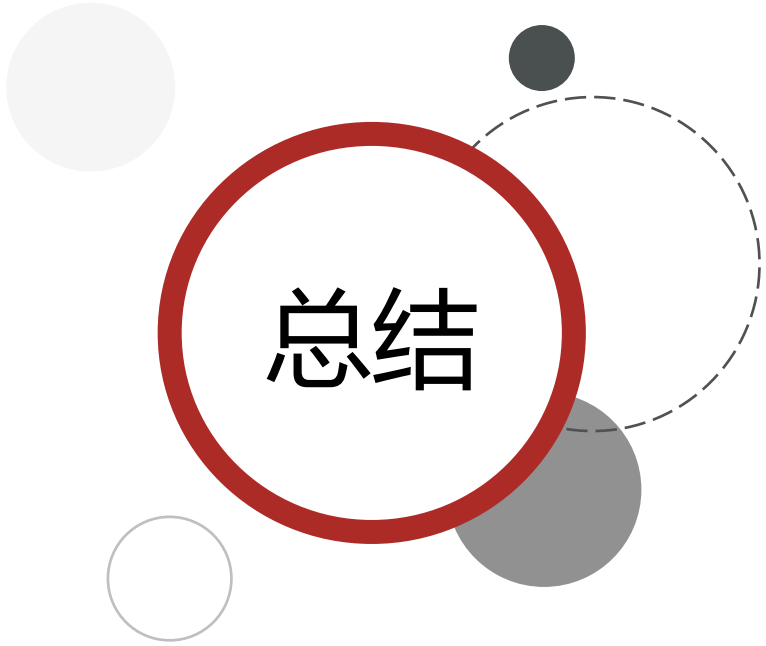
权限修饰符的分类和具体作用范围：

- 权限修饰符：有四种作用范围由小到大（ private -> 缺省 -> protected - > public ）

修饰符	同一 个类中	同一个包中 其他类	不同包下的 子类	不同包下的 无关类
private	√			
缺省	√	√		
protected	√	√	√	
public	√	√	√	√

学完权限修饰符需要具备如下能力

- 能够识别别人定义的成员的访问范围。
- 自己定义成员（方法，成员变量，构造器等）一般需要满足如下要求：
  - 成员变量一般私有。
  - 方法一般公开。
  - 如果该成员只希望本类访问，使用private修饰。
  - 如果该成员只希望本类，同一个包下的其他类和子类访问，使用protected修饰。



1、权限修饰符是什么？

有四种作用范围由小到大（private -> 缺省 -> protected - > public ）

修饰符	同一 个类中	同一个包中其他类	不同包下的子类	不同包下的无关类
private	√			
缺省	√	√		
protected	√	√	√	
public	√	√	√	√

目录

Contents

final的作用

- final 关键字是最终的意思，可以修饰（类、方法、变量）
- 修饰类：表明该类是最终类，不能被继承。
- 修饰方法：表明该方法是最终方法，不能被重写。
- 修饰变量：表示该变量第一次赋值后，不能再次被赋值(有且仅能被赋值一次)。

final修饰变量的注意

- final修饰的变量是基本类型：那么变量存储的**数据值**不能发生改变。
- final修饰的变量是引用类型：那么变量存储的**地址值**不能发生改变，但是地址指向的对象内容是可以发生变化的。

目录

Contents

常量

- 常量是使用了public static final修饰的成员变量，必须有初始化值，而且执行的过程中其值不能被改变。
- 常量名的命名规范：英文单词全部大写，多个单词下划线连接起来。
- 常量的作用：通常用来记录系统的配置数据。

```
public class Constant {
    public static final String SCHOOL_NAME = “传智教育";
    public static final String LOGIN_NAME = “admin";
    public static final String PASS_WORD = “123456";
}
```

常量做信息配置的原理、优势

- 在编译阶段会进行“宏替换”：把使用常量的地方全部替换成真实的字面量。
- 维护系统容易，可读性更好。

目录

Contents

案例说明：

- 现在开发的超级玛丽游戏需要接收用户输入的四个方向的信号（上下左右），以便控制玛丽移动的方向。

选择常量做信息标志和分类：

- 代码可读性好，实现了软编码形式。

目录

Contents

枚举的概述

- 枚举是Java中的一种特殊类型
- **枚举的作用：**"是为了做信息的标志和信息的分类"。

定义枚举类的格式：

```
修饰符 enum 枚举名称{
        第一行都是罗列枚举类实例的名称。
}
```

```
enum Season{
    SPRING , SUMMER , AUTUMN , WINTER;
}
```

反编译后观察枚举的特征：

```
enum Season{
    SPRING , SUMMER , AUTUMN , WINTER;
}
```

```
Compiled from "Season.java"
public final class Season extends java.lang.Enum<Season> {
    public static final Season SPRING = new Season();
    public static final Season SUMMER = new Season();
    public static final Season AUTUMN = new Season();
    public static final Season WINTER = new Season();
    public static Season[] values();
    public static Season valueOf(java.lang.String);
}
```

枚举的特征：

- 枚举类都是继承了枚举类型：java.lang.Enum
- 枚举都是最终类，不可以被继承。
- 构造器都是私有的，枚举对外不能创建对象。
- 枚举类的第一行默认都是罗列枚举对象的名称的。
- 枚举类相当于是多例模式。

目录

Contents

案例说明：

- 现在开发的超级玛丽游戏需要接收用户输入的四个方向的信号（上下左右），以便控制玛丽移动的方向。

选择常量做信息标志和分类：

- 虽然可以实现可读性，但是入参值不受约束，代码相对不够严谨。

枚举做信息标志和分类：

- 代码可读性好，入参约束严谨，代码优雅，是最好的信息分类技术！建议使用！

目录

Contents

抽象类

- 在Java中abstract是抽象的意思，可以修饰类、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 Animal{
    public abstract void run();
}
```

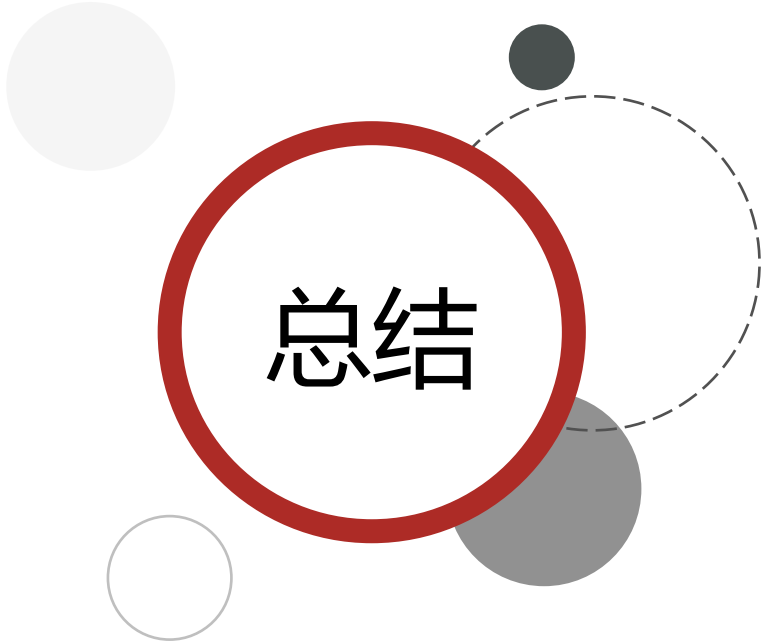
注意事项

- 抽象方法只有方法签名，不能声明方法体。
- 一个类中如果定义了抽象方法，这个类必须声明成抽象类，否则报错。

抽象的使用场景

- 抽象类可以理解成不完整的设计图，一般作为父类，让子类来继承。
- 当父类知道子类一定要完成某些行为，但是每个子类该行为的实现又不同，于是该父类就把该行为定义成抽象方法的形式，具体实现交给子类去完成。此时这个类就可以声明成抽象类。

```
public abstract class Animal{
    public abstract void run();
}
```



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。**
- **一个类中定义了抽象方法，这个类必须声明成抽象类。**

2、抽象类基本作用是啥？

- **作为父类，用来被继承的。**

3、继承抽象类有哪些要注意？

- **一个类如果继承了抽象类，那么这个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。**

目录

Contents

系统需求

- 某加油站推出了2种支付卡，一种是预存10000的**金卡**，后续加油享受**8折优惠**，另一种是预存5000的**银卡**,后续加油享受**8.5折优惠**。
- 请分别实现2种卡片进入收银系统后的逻辑，卡片需要包含主人名称，余额，支付功能。

分析实现

- 创建一张卡片父类：定义属性包括主人名称、余额、支付功能（具体实现交给子类）
- 创建一张白金卡类：重写支付功能，按照原价的8折计算输出。
- 创建一张银卡类：重写支付功能，按照原价的8.5折计算输出。

目录

Contents

特征和注意事项

- 类有的成员（成员变量、方法、构造器）抽象类都具备
- 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
- 一个类继承了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。
- 不能用abstract修饰变量、代码块、构造器。
- **最重要的特征：**得到了抽象方法，失去了创建对象的能力（**有得有失**）

final和abstract是什么关系？

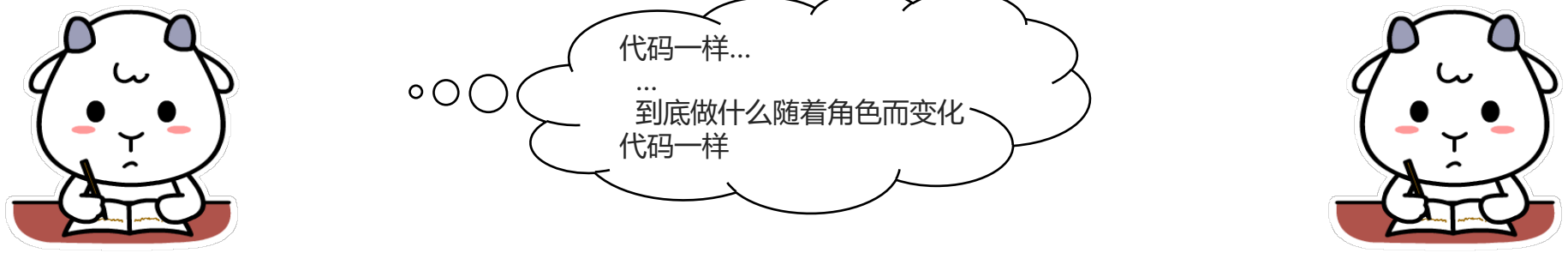
- 互斥关系
- abstract定义的抽象类作为模板让子类继承，final定义的类不能被继承。
- 抽象方法定义通用功能让子类重写，final定义的方法子类不能重写。

目录

Contents

什么时候使用模板方法模式

- **使用场景说明**：当系统中出现同一个功能多处在开发，而该功能中大部分代码是一样的，只有其中部分可能不同的时候。



模板方法模式实现步骤

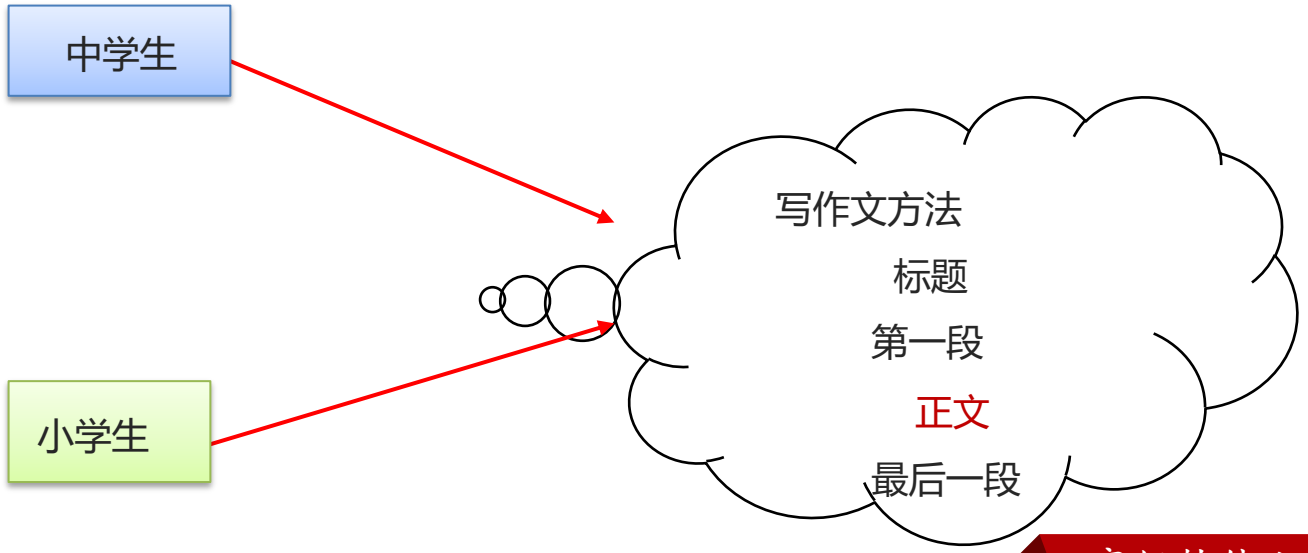
- 1、定义一个抽象类。2、定义2个方法，一个是模板方法：把相同代码放里面去，不同代码定义成抽象方法
- 3、子类继承抽象类，重写抽象方法。

案例 理解模板方法：写作文案例

需求：

- 现在有两类学生，一类是中学生，一类是小学生，他们都要写《我的爸爸》这篇作文。
- 要求每种类型的学生，标题第一段和最后一段，内容必须一样。正文部分自己发挥。
- 请选择最优的面向对象方案进行设计。

分析：

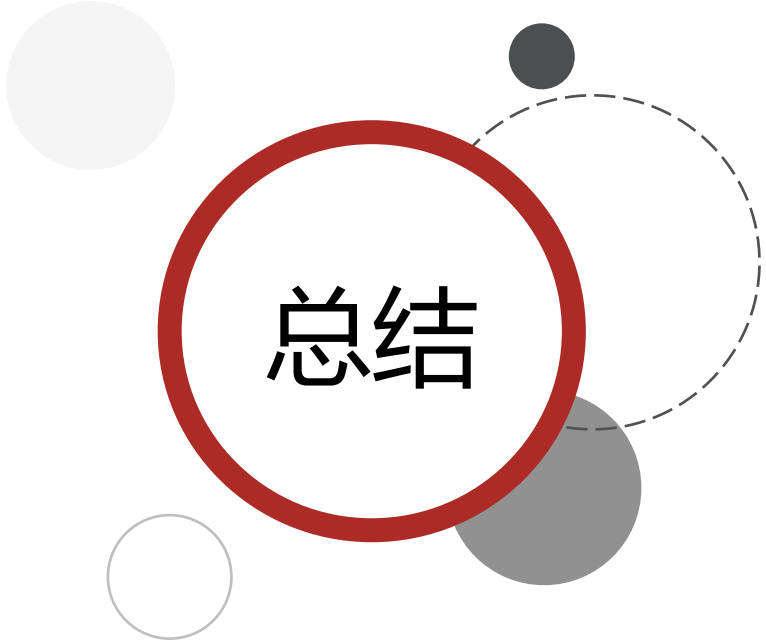


模板方法我们是建议使用final修饰的，这样会更专业，那么为什么呢？

答：模板方法是给子类直接使用的，不是让子类重写的，

一旦子类重写了模板方法，则模板方法就失效了，因此，加上final后可以

防止子类重写了模板方法，这样更安全、专业。



1、模板方法模式解决了什么问题？

- **提高了代码的复用性**
- **模板方法已经定义了通用结构，模板方法不能确定的部分定义成抽象方法，交给子类实现，因此，使用者只需要关心自己需要实现的功能即可。**

目录

Contents

什么是多态？

- 指对象可以有多种形态。

多态的常见形式

```
父类类型 对象名称 = new 子类构造器;
```

多态中成员访问特点

- 方法调用：编译看左边，运行看右边。
- 变量调用：编译看左边，运行也看左边。（**注意**）

多态的前提

- 有继承/实现关系；有父类引用指向子类对象；有方法重写（**多态侧重行为多态**）。

目录

Contents

优势

- 在多态形式下，右边对象可以实现解耦合，便于扩展和维护。

```
Animal a = new Dog();
a.run(); // 后续业务行为随对象而变，后续代码无需修改
```

- 定义方法的时候，使用父类型作为参数，该方法就可以接收这父类的一切子类对象，体现出多态的扩展性与便利。

多态下会产生的一个问题:

- 多态下不能使用子类的独有功能

目录

Contents

自动类型转换（从子到父）：

```
Animal c = new Cat();
```

强制类型转换（从父到子）

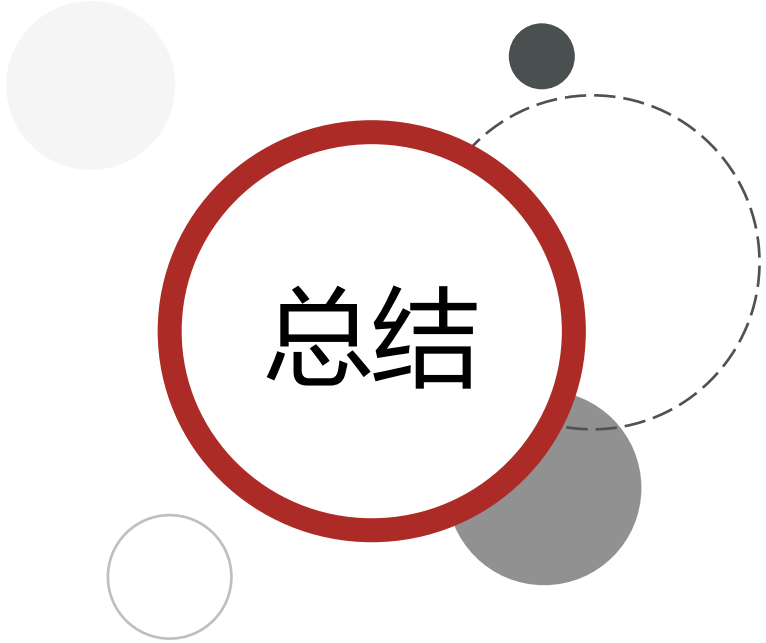
- **从父到子（** 必须进行强制类型转换,否则报错）：子类 对象变量 = (子类)父类类型的变量
- 作用：可以解决多态下的劣势，可以实现调用子类独有的功能。
- 注意： 有继承/实现关系的类就可以在编译阶段进行强制类型转换；但是，如果转型后的类型和对象真实对象的类型不是同一种类型，那么在运行代码时，就会出现ClassCastException

```
Animal c = new Cat();
Dog d = (Dog)c; // 出现异常 ClassCastException
```

Java建议强转转换前使用instanceof判断当前对象的真实类型，再进行强制转换

变量名 **instanceof 真实**类型

判断关键字左边的变量指向的对象的真实类型，是否是右边的类型或者是其子类类型，是则返回true，反之。



1. 引用数据类型的类型转换，有几种方式？
   - **自动类型转换、强制类型转换。**
2. 强制类型转换能解决什么问题？
   - **可以转换成真正的子类类型，从而调用子类独有功能。**
3. 强制类型转换需要注意什么？
   - **有继承关系/实现的2个类型就可以进行强制转换，编译无问题。**
   - **运行时，如果发现强制转换后的类型不是对象真实类型则报错（ClassCastException）**
4. 强制类型转换前最好做什么事情，如何进行？
   - **使用instanceof判断当前对象的真实类型，再进行强制转换**
   - **对象变量名 instanceof 真实类型**

目录

Contents

案例 模拟开发一款动物表演类的游戏

需求：

- 模拟开发一款动物表演类的游戏

分析

① 定义一个USB的接口（申明USB设备的规范必须是：可以接入和拔出）。

② 提供2个USB实现类代表鼠标和键盘，让其实现USB接口，并分别定义独有功能。

③ 创建电脑对象，创建2个USB实现类对象，分别安装到电脑中并触发功能的执行。

传智教育旗下高端IT教育品牌