## 1 定义学生类

### 一、案例描述

1. 考核知识点

编号：029003002

名称：类的定义

1. 练习目标

* 掌握类定义的方式
* 掌握如何在类中定义成员变量和成员方法

1. 需求分析

在面向对象的思想中最核心就是对象，在程序中创建对象的前提是需要定义一个类。为了让初学者掌握类的定义方式，本案例将设计一个表示学生的类，该类具有表示姓名的属性name和表示年龄的属性age，同时还具有表示说话行为的方法speak()，用于输出学生的姓名和年龄。

1. 设计思路（实现原理）

1）使用class关键字定义一个表示学生类型的类，类名为Student。

2）在Student类中定义两个成员变量name和age，分别用来表示姓名和年龄。其中，name的数据类型为String，变量age的数据类型为int。

3）在Student类中定义一个表示说话行为的speak()方法，用于输出学生的姓名和年龄。

### 二、案例实现

class Student{

String name;

int age;

void speak() {

System.out.println("我的名字是 "+name+"，今年 "+age+"岁");

}

}

### 三、案例总结

1、Java语言严格区分大小写，class和Class是不同的，在定义类时只能使用class关键字

2、在Student类中，成员变量name是String类型，String表示一个字符串，后面的章节会详细讲解

3、思考一下：自己定义一个手机(Phone)类，在类中定义品牌(brand)和价格(price)属性，定义打电话的call()方法，要如何定义？

代码如下所示

public class Phone {

String brand;

double price;

void call(){

System.out.println("hi,how are you doing");

}

}

## 2 同一对象被多个变量引用

### 一、案例描述

1. 考核知识点

编号：029003003

名称：对象创建与使用

1. 练习目标

* 掌握如何创建类的对象
* 掌握如何使用两个或者多个变量引用同一个实例对象。

1. 需求分析

在程序中，一个对象可能在多处使用，这样就可能需要有多个变量来引用这个对象。为了让初学者更好地掌握对象的创建和使用，本案例将基于案例3-1，创建三个学生对象，它们的引用变量分别是s1、s2和s3，首先分别使用s1和s2引用，为name和age赋值，然后调用speak()方法，最后将s2变量赋值给s3， s3也调用speak()方法。

1. 设计思路（实现原理）

1）编写Example02类。

2）在main()方法中，创建Student类的第一个对象，其引用变量为s1，使用s1调用name和age变量分别为它们赋值为“张三”和“19”，然后调用speak()方法。

3）创建Student类的第二个对象，其引用变量为s2，使用s2分别为name和age赋值为“李四”和“20”，然后调用speak()方法。

4）创建Student类的第三个对象，其引用变量为s3，将s2的值赋给s3，然后使用s3调用speak()方法。

### 二、案例实现

public class Example02 {

public static void main(String [] args) {

Student s1 = new Student();

s1.name="张三";

s1.age=19;

s1.speak();

Student s2 = new Student();

s2.name="李四";

s2.age=20;

s2.speak();

Student s3 = new Student();

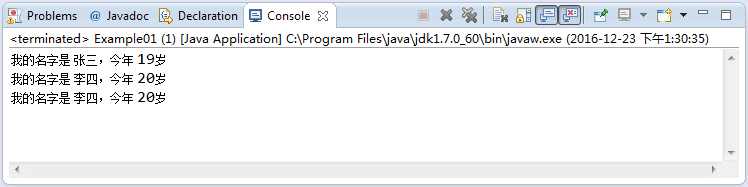
s3=s2;

s3.speak();

}

}

运行结果如图3-1所示。



1. 运行结果

### 三、案例总结

1、Student s3 = s2这句代码的作用是将s2引用的内存地址赋值给s3，换句话说，就是使变量s3和s2引用了同一个Student对象，因此s3.speak()方法和s2.speak()方法打印的结果相同。为了更加深刻地了解这句代码的含义，下面通过一张内存图来演示，具体如图3-2所示。



1. 内存图

2、可以使用两个或者多个变量引用同一个实例对象，只要通过其中一个变量对该对象的属性进行修改，使用其它引用变量访问时，访问的都是修改后的属性。

## 3 类的封装

### 一、案例描述

1. 考核知识点

编号：029003005

名称：类的封装

1. 练习目标

* 了解为什么要对类进行封装
* 掌握如何实现类的封装

1. 需求分析

在案例3-2中，s1对象的年龄是可以随便赋值的，如果将age的值赋值为-30，显然违背了事实。为了解决这类问题，我们需要对类进行封装，防止外界对类中的成员变量随意访问。为了让初学者更好地掌握类的封装，本案例将使用private关键字对成员变量name和age进行私有化，同时分别提供一个setName(String n)和setAge(int a)方法用于外界的访问，其中setAge(int a)中需要对age进行判断。

1. 设计思路（实现原理）
2. 编写测试类Example03，将属性age的值设为-30，演示不合理现象。
3. 对Student类进行修改，将name和age属性使用private修饰，然后定义getName()、setName(String n)、getAge()和setAge(int a)四个对外访问name和age的方法。
4. 在setAge(int a)方法中对传入的参数进行检查，如果输入值为负数，则打印出“设置的年龄不合法”，如果不为负数，才将其设置为age属性的值。

4）对Example03类进行修改，在main()方法中创建Student类的实例对象，通过调用对象的setName(String n)和setAge(int a)方法来设置的name属性和age属性值，并调用speak()方法。

### 二、案例实现

1、定义Example03类，代码如下所示：

public class Example03 {

public static void main(String[] args) {

Student s1 = new Student();

s1.name = "小新";

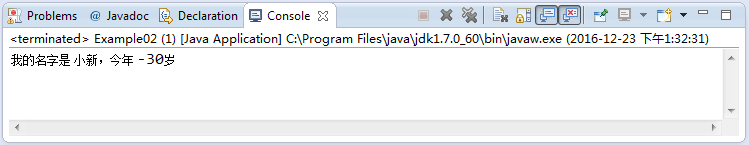
s1.age = -30;

s1.speak();

}

}

运行结果如图3-3所示。



1. 运行结果

从上图可以看出，当将age的值设置为-30后，程序不会报错，但却违背了现实。

2、对Student类进行封装，其代码如下：

class Student {

private String name = "张三";

private int age = 19;

public String getName() {

return name;

}

public void setName(String n) {

name = n;

}

public int getAge() {

return age;

}

public void setAge(int a) {

// 对传入的参数进行检查

if(a < 0){

System.out.println("设置的年龄不合法");

}else{

age = a;

}

}

void speak() {

System.out.println("我的名字是" + name + "，今年" + age + "岁");

}

}

public class Example03 {

public static void main(String[] args) {

Student s1 = new Student();

s1.setName("小新");

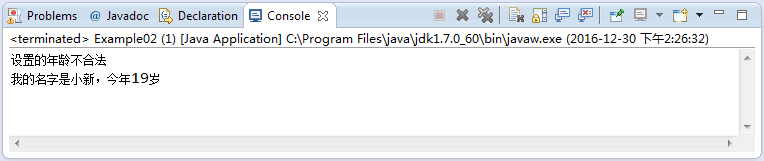
s1.setAge(-30);

s1.speak();

}

}

运行结果如图3-4所示。



1. 运行结果

### 三、案例总结

1、Student的name和age属性使用private关键字修饰为私有后，在Example02类中不能再使用s1.name和s1.age的方式访问这两个属性，只能通过public类型的setName(String n)和setAge(int a)方法进行访问。在上面的代码中，调用setAge(int a)方法时的传入参数为-30，由于参数小于0，会打印出“设置的年龄不合法”，并不会将负数赋值给age属性。由此可见，只要实现了封装就能对外界的访问进行控制，避免对私有变量随意修改而引发问题。

2、动手操作：定义一个Division类(除法)，类中定义两个int类型的私有成员变量dividend(被除数)和

divisor(除数)，默认值都为1。定义四个公有方法setDividend(int mDividend )、getDividend()、setDivisor(int mDivisor)和getDivisor()，用于对私有属性进行设置和访问。在setDivisor(int mDivisor)方法中对传入的参数进行检查，如果传入值为零，则打印“除数不能为零”，如果不为零，才将其设置为divisor属性的值。

定义Example03类，在类的main()方法中创建Division对象，分别调用setDividend(int mDividend)和setDivisor(int mDivisor)方法给两个私有属性dividend和divisor赋值，然后打印出dividend和divisor的结果。

(1)定义Division类，代码如下所示：

public class Division {

private int dividend = 1;

private int divisor = 1;

public void setDividend(int mDividend) {

dividend = mDividend;

}

public int getDividend() {

return dividend;

}

public void setDivisor(int mDivisor) {

if (mDivisor == 0) {

System.out.println("除数不能为零");

} else {

divisor = mDivisor;

}

}

public int getDivisor() {

return divisor;

}

}

(2)定义Example03类，代码如下所示：

public class Example03{

public static void main(String[] args) {

Division division = new Division();

division.setDividend(10);

division.setDivisor(0);

int dividend = division.getDividend();

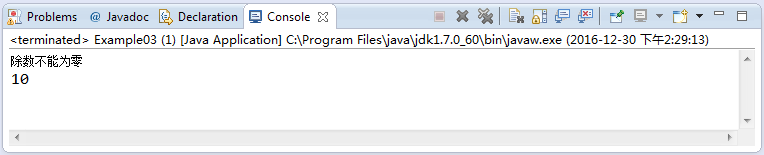
int divisor = division.getDivisor();

System.out.println(dividend / divisor);

}

}

运行结果如图3-5所示。



1. 运行结果

从运行结果可以看出，由于Division类实现了封装，在setDivisor(int mDivision)方法中对传入的值进行检查，从而避免程序中出现除数为0的错误。

## 4 定义有参的构造方法

### 一、案例描述

1. 考核知识点

编号：029003007

名称：构造方法的定义

1. 练习目标

* 掌握有参构造方法的定义方式

1. 需求分析

如果希望在创建对象的时候直接为其属性赋值，可以定义有参的构造方法。有参构造方法指的是在初始化对象时，接受外部传入的值并赋给对象的属性。为了让初学者掌握有参构造方法的用法，本案例将演示如何使用有参构造方法完成对象属性的初始化。

1. 设计思路（实现原理）
2. 定义一个Student类，该类有一个age属性，在类中定义一个有参数的构造方法，该参数用于为age属性赋值。
3. 编写一个测试类Example04，在main()方法中通过有参构造方法创建一个对象。
4. 打印该对象age属性的值。

### 二、案例实现

1、对Student类进行修改，代码如下所示：

class Student {

int age;

public Student(int mAge) {

age = mAge;

}

}

2、定义Example04类，代码如下所示：

public class Example04 {

public static void main(String[] args) {

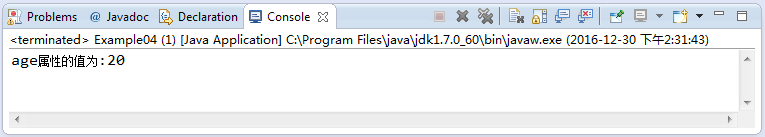
Student s1 = new Student(20);

System.out.println("age属性的值为:"+s1.age);

}

}

运行结果如图3-6所示。



1. 运行结果

### 三、案例总结

1、从运行结果可以看出，new Student(20)语句调用了有参的构造方法Student(int mAge)，动态地将20传递给了age属性。和普通方法一样，构造方法中同样可以接收多个参数，只要在使用new关键字创建对象时，传入数量相同和类型一致的参数，就可以自动地调用对应的构造方法。

2、思考一下：在Example04的main()方法中是否能够使用new Student()创建对象呢？答案是否定的，因为new Student()会调用无参的构造方法，而本案例的Student类中并没有定义无参的构造方法。

有些同学肯定会问，之前的Student类都没有定义无参的构造方法，却能使用new Student()创建对象，本案例为什么不行呢？这是因为一个类中如果没有定义构造方法，系统会默认为其分配一个方法体为空的无参构造方法，而一旦定义了构造方法，系统就不再提供默认的构造方法。本案例中由于我们定义了一个有参的构造方法，所以系统不会默认分配无参的构造方法，此时如果通过new Student()去调用无参的构造方法，程序就会发生错误。

## 5 静态变量

### 一、案例描述

1. 考核知识点

编号：029003011

名称：静态变量

1. 练习目标

* 了解静态变量的作用
* 掌握静态变量的定义和使用方式

1. 需求分析

当多个对象存储的数据相同时，可以使用静态变量的方式存储。例如，有一个Student类具有name、className属性，请根据该类创建出三个Student对象，并将这些对象的className值都设置为“三年级二班”。

1. 设计思路（实现原理）
2. 定义Student类，并在类中定义name和className属性。
3. 编写测试类Example05，在main()方法中创建三个学生对象，并分别为这些对象的name和className属性赋值，然后输出这些对象的name和className值。
4. 对Student类进行修改，将className定义为静态变量。
5. 修改测试类，在main()方法中使用Student.className = “三年级二班”语句为静态变量className进行赋值，然后输出这些对象的name和className值。

为了更好地理解Student类中静态变量className和Student实例对象的关系，下面通过一个图例进行演示，如图3-11所示：



1. 静态变量与实例对象的关系

### 二、案例实现

1、定义Student类，代码如下所示：

class Student{

String name;

String className;

}

2、定义Example05类，代码如下所示：

public class Example05 {

public static void main(String[] args) {

Student s1 = new Student();

s1.name = "张三";

s1.className = "三年级二班";

Student s2 = new Student();

s2.name = "李四";

s2.className = "三年级二班";

Student s3 = new Student();

s3.name = "王五";

s3.className = "三年级二班";

System.out.println("我的名字是" + s1.name + ",来自" + s1.className);

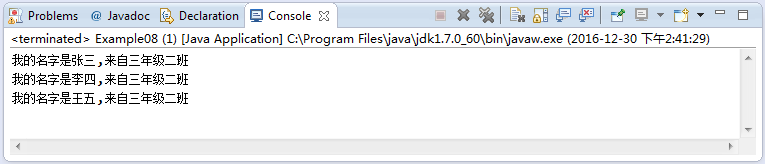
System.out.println("我的名字是" + s2.name + ",来自" + s2.className);

System.out.println("我的名字是" + s3.name + ",来自" + s3.className);

}

}

运行结果如图3-12所示。



1. 运行结果

3、对Student类进行修改，代码如下所示：

class Student {

String name;

static String className;

}

4、对Example05类进行修改，代码如下所示：

public class Example05 {

public static void main(String[] args) {

Student.className = "三年级二班";

Student s1 = new Student();

s1.name = "张三";

Student s2 = new Student();

s2.name = "李四";

Student s3 = new Student();

s3.name = "王五";

System.out.println("我的名字是" + s1.name + ",来自" + s1.className);

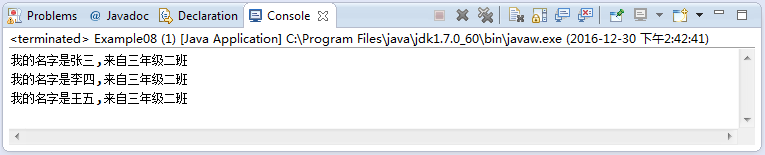
System.out.println("我的名字是" + s2.name + ",来自" + s2.className);

System.out.println("我的名字是" + s3.name + ",来自" + s3.className);

}

}

运行结果如图3-13所示。



1. 运行结果

### 三、案例总结

1、本案例中，三个Student对象的className属性值均为“三年级二班”，对于这样的相同数据，没有必要在每个对象中都开辟一块空间存储，完全可以在内存中只用一块空间存储，并被一个类的所有实例对象所共享。在Java中提供了一个static关键字，使用static关键字修饰的成员变量称为静态变量，静态变量能被该类所有实例对象共享。

2、静态变量可以使用“类名.静态方法名”的方式访问，也可以通过“对象引用变量.静态方法名”的方式访问，例如本例中的静态变量className，通过Student.className或者s2.className这两种方式访问都是可以的，不过更推荐使用前一种方式。

3、动手操作：定义一个Person类，在类中定义两个String类型的成员变量name和nationality(国籍)，变量nationality使用static关键字修饰为静态变量，其值为“中国”。

定义Example06类，在类的main()方法中创建Person类的两个实例对象，并打印出这两个对象的姓名和国籍。

(1)定义Person类，代码如下所示：

class Person {

String name;

static String nationality = "中国";

}

(2)定义Example06类，代码如下所示：

public class Example06{

public static void main(String[] args) {

Person lily = new Person();

lily.name = "丽丽";

Person lucy = new Person();

lucy.name = "露西";

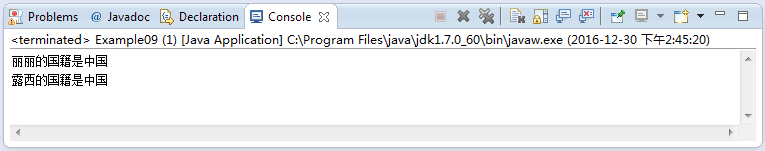
System.out.println(lily.name + "的国籍是" + lily.nationality);

System.out.println(lucy.name + "的国籍是" + lucy.nationality);

}

}

运行结果如图3-14所示。



1. 运行结果