## 1 类的继承

### 一、案例描述

1. 考核知识点

编号：029004001

名称：类的继承

1. 练习目标

* 了解类继承的意义和作用
* 掌握如何实现类的继承

1. 需求分析

在程序中，继承描述的是事物之间的所属关系，通过继承可以使多种事物之间形成一种关系体系。

为了让初学者熟悉类的继承，本案例将演示类的继承并编写测试类验证子类是否拥有父类的可继承成员。

1. 设计思路（实现原理）

1）设计两个类Student和Teacher。

2）抽取两个类共同的内容（如：吃饭、睡觉）封装到一个类Person中，各自特有的部分保留在各自类中。

3）让学生类继承Person类，老师类也继承Person。

4）编写测试类Example01，测试Student类和Teacher是否继承了Person类的成员。

### 二、案例实现

1、抽取父类Person，实现继承关系，代码如下：

class Person {

String name;//姓名

// 吃饭的功能

void eat() {

System.out.println("吃饭 ");

}

// 睡觉的功能

void sleep() {

System.out.println("睡觉");

}

}

class Student extends Person {

// 学号

int sid;

}

class Teacher extends Person {

// 工号

int tid;

// 教课的功能

void teach() {

System.out.println("老师教课");

}

}

2、定义测试类Example01，代码如下：

public class Example01{

public static void main(String[] args) {

Student s = new Student();

s.eat();

s.sleep();

System.out.println("----");

Teacher t = new Teacher();

t.eat();

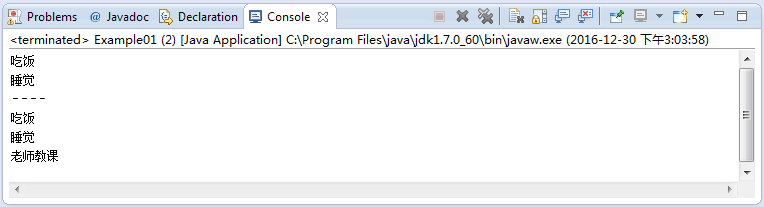
t.sleep();

t.teach();

}

}

运行结果如图4-1所示。



1. 运行结果

### 三、案例总结

1、在Java中，多个类可以继承一个父类，但是一个类不能直接继承多个类，一个类只能有一个直接父类。

2、父类是由子类不断抽取而来的，不断地抽取就形成了体系结构，这个结构称为继承体系结构。

3、子类在继承父类的时候，会自动拥有父类所有的成员。4、继承的好处是划分了类的层次性，实现了代码重用、扩展了程序功能。

## 2 接口的实现

### 一、案例描述

1. 考核知识点

编 号：029004006

名 称：接口

1. 练习目标

* 掌握如何实现接口

1. 需求分析

由于接口中的方法都是抽象方法，因此不能通过实例化对象的方式来调用接口中的方法。此时需要定义一个类，并使用implements关键字实现接口中所有的方法。为了上初学者掌握如何实现接口，本案例将设计一个类，使用关键字implements实现Inter接口中的方法。

1. 设计思路（实现原理）

1）设计名为Inter的接口，接口中定义一个初始值为20的num常量和method()抽象方法。

2）编写Inter接口的实现类InterImpl，使用implements关键字实现接口中的方法。

3）编写测试类Example02，测试类中对接口的实现类进行实例化，并调用接口中的方法。

### 二、案例实现

1、编写接口Inter，InterImpl类使用implements实现了接口，代码如下：

//定义一个接口

interface Inter {

int num = 20;

void method();

}

// 使用implements关键字实现接口

class InterImpl implements Inter {

void show() {

System.out.println(num);

}

public void method() {

System.out.println("InterImpl method");

}

}

2、定义测试类Example02，代码如下：

class Example02{

public static void main(String[] args) {

//通过实现类实例化

InterImpl ii = new InterImpl();

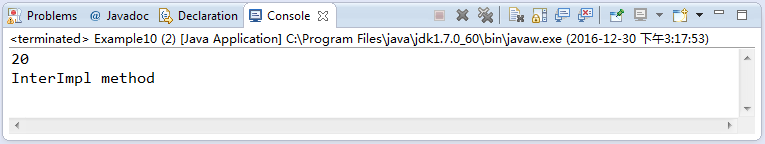
ii.show();

ii.method();

}

}

运行结果如图4-10所示。



1. 运行结果

### 三、案例总结

1、接口使用interface关键字来定义。

2、接口没有构造方法，接口中的变量缺省都是使用public static final修饰的，即全局常量，接口中的方法都是public abstract修饰的，即抽象方法。

3、一个类可以实现多个接口。

4、由于接口中的方法都是抽象方法，因此不能直接通过实例化对象的方式来调用接口中的方法，需要在类实现接口后借助类来间接实例化。

## 3 多态的作用

### 一、案例描述

1. 考核知识点

编 号：029004008

名 称：多态的概述

1. 练习目标

* 了解多态的概念、多态的好处和弊端
* 掌握如何实现多态

1. 需求分析

在Java中为了实现多态，允许使用一个父类类型的变量来引用一个子类类型的对象，根据被引用子类对象特征的不同，得到不同的运行结果。多态提高代码了的扩展性和可维护性。为了帮助初学者掌握如何实现多态，本案例将实现一个多态程序来演示如何实现多态。

1. 设计思路（实现原理）

1）编写Animal接口，接口中定义sleep()方法。

2）Cat类实现Animal接口的方法，并定义另一个方法catchMouse()。

3）编写测试类Example03，使Cat对象指向父类Animal的引用 ，并通过该引用调用sleep()方法。

### 二、案例实现

1、编写Animal接口及其实现类Cat。代码如下：

interface Animal {

void sleep();

}

class Cat implements Animal {

void catchMouse() {

System.out.println("抓老鼠");

}

public void sleep() {

System.out.println("睡觉");

}

}

2、定义测试类Example03，Cat对象指向Animal引用，并使用父类Animal引用调用sleep()方法。代码如下：

public class Example03 {

public static void main(String[] args) {

// 多态

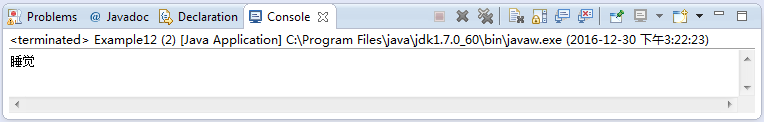
Animal animal = new Cat();

animal.sleep();

}

}

运行结果如图4-12所示。



1. 运行结果

### 三、案例总结

1、多态是指对象在不同时刻表现出来的不同状态，在Java中，允许使用一个父类类型的变量来引用一个子类类型的对象。

2、多态的前提：

(1) 要有继承关系。

(2) 要有方法重写。

(3) 要有父类引用指向子类对象。

3、多态的好处是提高代码的扩展性和可维护性。

4、多态的弊端是父类或者接口只能调用自己具备的成员，不能访问子类特有的成员。

## 4 匿名内部类

### 一、案例描述

1. 考核知识点

编 号：029004011

名 称：匿名内部类

1. 练习目标

* 了解什么是匿名内部类
* 掌握如何使用匿名内部类

1. 需求分析

在学习接口时，如果方法的参数被定义为一个接口类型，那么就需要定义一个类来实现接口。除此之外，还可以使用匿名内部类来实现接口。为了让初学者能更好地理解什么是匿名内部类，本案例将演示如何实现一个匿名内部类。

1. 设计思路（实现原理）

1）设计一个Inter接口，接口中定义show()和method()方法。

2）设计一个Outer类，在类的function()方法中使用匿名内部类实现Inter接口并实例化，并调用show()和method()方法。

3）最后编写一个测试类Example04，创建Outer的对象并调用function()方法。

### 二、案例实现

1、编写Inter接口及Outer类，使用匿名内部类实现接口并创建对象。代码如下：

interface Inter {

void show();

void method();

}

class Outer {

void function() {

Inter i = new Inter() {

public void show() {

System.out.println("show");

}

public void method() {

System.out.println("method");

}

};

i.show();

i.method();

}

}

2、定义测试类Example04。代码如下：

public class Example04 {

public static void main(String[] args) {

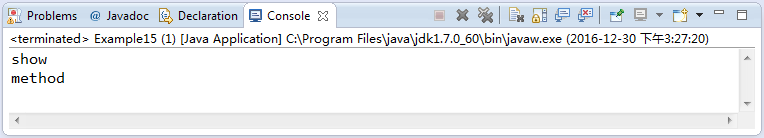
Outer o = new Outer();

o.function();

}

}

运行结果如图4-15所示。



1. 运行结果

### 三、案例总结

1、匿名内部类的格式，具体如下：

new 父类(参数列表) 或 父接口(){

//匿名内部类实现部分

}

2、匿名内部类是实现接口的一种简便写法，在程序中不一定非要使用匿名内部类。

3、什么时候使用匿名内部类：通常在使用方法是接口类型参数，并该接口中的方法不超过三个时，可以将匿名内部类作为参数传递。

4、内部类：

(1) 将一个类B定义在另一个类A的里面，那么B类就称为内部类(嵌套类)，内部类定义在成员位置上。

(2) 内部类可以直接访问外部类中的成员，包括私有成员，而外部类要访问内部类中的成员必须要建立内部类的对象。

## 5 try...catch语句

### 一、案例描述

1. 考核知识点

编 号：029004014

名 称：try...catch和finally

1. 练习目标

* 掌握如何使用try...catch语句进行异常处理

1. 需求分析

Java中提供了一种对异常进行处理的方式——异常捕获，异常捕获通常使用try…catch语句。为了让初学者熟悉try…catch语句，本案例将演示如何使用try...catch语句处理遍历时可能出现的数组角标越界异常。

1. 设计思路（实现原理）
2. 编写测试类Example05。
3. 在类Example05中定义名为arr的int类型一维数组，遍历数组输出每次遍历出的值，结束输出“over”。

3）使用try...catch语句处理遍历时可能出现的数组角标越界异常，如果出现异常输出“角标越界”。

### 二、案例实现

public class Example05 {

public static void main(String[] args) {

int[] arr = { 1, 2, 3, 4, 5 };

try {

for (int x = 0; x <= arr.length; x++) {

System.out.println(arr[x]);

}

System.out.println("程序继续向下执行...");

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("角标越界");

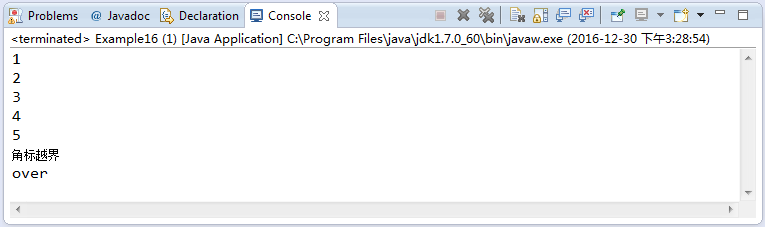
}

System.out.println("over");

}

}

运行结果如图4-16所示。



1. 运行结果

### 三、案例总结

1、try代码块中编写可能发生异常的Java语句，catch代码块中编写针对异常进行处理的代码。在try代码块中，发生异常语句后面的代码是不会被执行的。

2、可能发生多个异常时，可以使用多个try...catch或者用一个try多个catch。

3、 Throwable:是异常体系的超类，它包含两个子类，具体如下：

* Error 重大的错误，这个问题发生后，一般我们处理不了，一般不处理，而是要程序进行修正 。(内存溢出)
* Exception 可以处理的问题，这个问题需要我们编写针对性的代码进行处理。(异常处理)