模块2

对象和类

本模块主要介绍面向对象的程序设计思想，并介绍如何利用 Java 语言来描述现实世界对象的特性、实现继承关系，并明确在 Java 语言中所有类都是继承自 java.lang.Object 类的事实。

2.1 对　　象

2.1.1 对象简介

世间万物皆对象，既包括可以被感官感知的实物，又包括思想、感觉或行为所及的概念。世界是由物质组成的，这里的物质是一个抽象概念。什么是物质？小到一种化学元素、一个细胞，大到一个生命体，如植物、动物等，都可以称为物质。这个物质并不仅仅指其中的一种或几种事物，而是把宇宙间的万事万物都包含进去了，而它所包含的其中的某一种类，如汽车等，都是它的一个具体体现，就等同于面向对象编程语言中的对象。

从软件角度看，对象是将特性（数据）和行为（功能）捆绑在一起的软件结构或模块，这两部分合起来表示实际（物理或概念）对象的抽象，如图 2-1 所示为对象在概念上的阐述。

图 2-1　捆绑了特性（数据）和行为（功能）的软件对象

2.1.2 对象的特性

每个对象都有自身的特性和行为，描述自己是什么或具有什么功能，从而与其他对象区别开来。

车在人们的生活中随处可见，如图 2-2 所示。如何利用面向对象技术来描述这些不同的车呢？

图 2-2 几种车的图片

通常用这样一些数据来描述车：颜色、车型、品牌、价格、变速箱、速度、燃料的经济性和制动性能等，如图 2-3 所示。

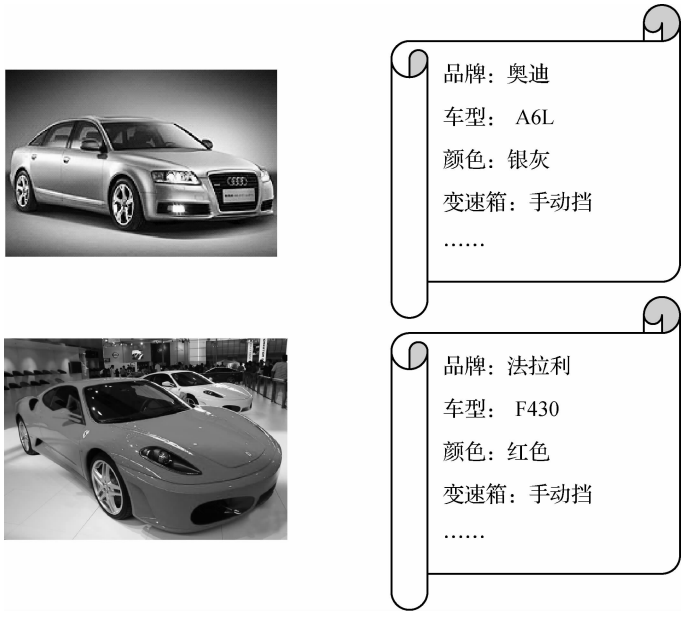

图 2-3 “轿车”对象和“跑车”对象

用于描述对象的数据元素称为属性，对象的属性值定义了对象的特性。例如，上述描述对象车的数据，即颜色、车型、品牌等，都是车的属性。一个给定的属性值可能很简单（如车的品牌可以表示成一个简单的字符串），也可能很复杂。

2.1.3 对象的行为

汽车的速度会时快时慢，计算机中存储这些数据的对象必须随之改变，即对象内的数据会改变。汽车的颜色改变了，汽车类的对象也必须改变。这种对象内数据的变化是对象的行为。对象的基本行为包括以下几种。

(1)把数据送入对象并存储起来。

(2)改变对象内的数据。

(3)对数据进行运算。

(4)从对象中送出数据。

例如,汽车打九五折后的售价为多少? 此时,可对对象 a 内的价格数据进行运算,如图 2-4 所示。

这里是要将对象 a 中的价格数据(35 万元)与折扣(0.95)相乘,进而算出其售价金额(33.25 万元)。然而在面向对象的程序设计概念中,必须将其解释为把“询问售价金额”送入对象 a,在对象内部进行相乘运算,然后把售价“33.25 万元”输出。其中,“33.25 万元”是在对象 a 接收到外界的消息之后才作出的反应,至于它的反应过程,即乘法运算则是在对象 a 内部完成的。这就好比一个灯泡,当电流进入灯泡中,此灯泡就会发光,这是人们的日常生活经验。例如,当你到自动售票机上买票时,只需把钱投入售票机,经过一些处理之后,售票机就会有反应,即把票吐出来,如图 2-5 所示。

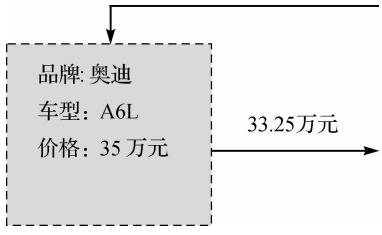

图 2-4　汽车对象的行为

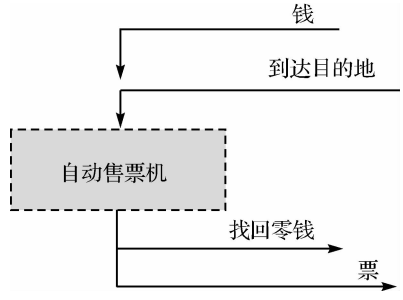

图 2-5　自动售票机的售票过程

上述实物对象(如自动售票机)接到消息(message)时,其内部产生运作,并输出运作的结果。同样地,程序内的对象接收到消息时,其内部也会对数据进行运算,并输出运算结果。

对象对消息会产生反应,但并非对任何消息都有反应。例如,灯泡只会对电流有所反应——发光;自动售票机必须投入钱才会有反应——吐出票。这就好比一只猫,它对身边的一个馒头可能不会有反应,但对身旁的老鼠将会出现强烈的反应。当使用灯泡或自动售票机时,能轻易学会其使用方法,即知道输入什么消息,也很清楚它们的反应。类似地,在面向对象程序设计中,也能轻易学会对象的使用方法,即知道输入什么消息并了解其反应。所以使用程序中的对象,就像使用灯泡一样简单方便。

2.2 类

2.2.1 类简介

在面向对象程序设计中,将现实中的多个对象共有的特性和行为抽象为一个类。

对象的特性在类中表示为成员变量,称为类的属性。例如,汽车的颜色、车型、品牌等特性可使用类的属性来描述。

对象的行为在类中表示为成员方法,用来操作成员变量,称为类的方法。例如,汽车的起动、

停止、加速、减速等行为可使用类的方法来描述，同时设计类时也应考虑类与类之间的关系。

类是拥有相同特性和行为的若干对象的集合。

在图 2-3 讨论的情境中，轿车、跑车等均为车类的对象，这些对象和类的关系如图 2-6 所示。

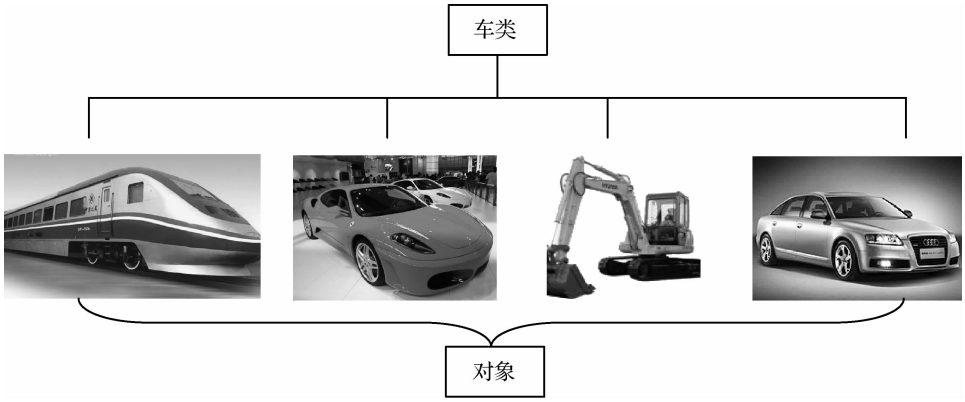

图 2-6　车的对象和类

2.2.2　类和对象的区别

类是对若干对象抽象后得出的一个模板，该模板包含了此类对象都拥有的特性和行为。而对象是一个现实存在的实体，每个对象都是所在类的一个实例。

从程序设计语言的角度可将类理解为一种数据类型，而对象则是具有这种类型的变量。

类和对象的对比如表 2-1 所示。

表 2-1　类和对象的对比

序　　号	类	对　　象
1	人	一位年轻的教授
		一位和蔼的医生
2	动物	一只呱呱叫的青蛙
		一只喵喵叫的小猫
3	汽车	一辆红色宝马轿车
		一辆黑色奔驰轿车

2.2.3　类的定义

1. 类的定义

类定义的一般语法格式如下。

```
[类修饰符]  class 类名 [extends 基类名][implements 接口名]{
            ……//成员变量声明
            ……//成员方法声明
            }
```

相关说明如下。

- class、extends 和 implements 都是 Java 语言的关键字，分别是关于类、继承和接口的内容。
- 关键字 class 表示创建了一个类。
- 关键字 extends 表示该类继承了某一父类。
- 关键字 implements 表示该类实现了某些接口。

类的命名规范为：类名首字母大写，其他字母小写；如果由多个单词组成，那么一般每个单词的首字母大写。

【例 2-1】 汽车类的定义。

```
public class Automotive{
        ……//成员变量声明
        ……//成员方法声明
        }
```

2. 成员变量声明

成员变量声明的一般语法格式如下。

```
[修饰符] 类型符 成员变量名[=初始值];
```

说明：成员变量的类型可以是 Java 语言中的任意数据类型，也可以是简单类型，还可以是类、接口、数组等复合类型。一个类中的成员变量名是唯一的。

【例 2-2】 汽车类中成员变量定义了汽车有车牌号、品牌、颜色和型号等属性。

```
public class Automotive{
                int number;             //车牌号
                String brand;           //品牌
                String color;           //颜色
                String model            //型号
                ……                      //成员方法声明
}
```

提示：成员变量与局部变量两者的声明格式不同，方法里的局部变量不能用修饰符来修饰。

3. 成员方法声明

成员方法声明的一般语法格式如下。

```
[修饰符] 类型符 成员方法名(参数列表)[throws 异常名 1,异常名 2……]{
    ……//方法语句主体
}
```

相关说明如下。

- 类型符是方法返回值的数据类型。如果方法有返回值，则在方法体内必须使用关键字 return 返回值；如果方法无返回值，则类型符必须为 void。
- 参数列表是一组变量声明，参数由圆括号内的逗号隔开；参数作为方法主体中的局部变量，而参数的值在调用方法时进行传递。
- 在方法体内，语句、表达式和方法调用可以共存。

【例 2-3】 汽车类中成员方法的定义：每辆汽车都有加速、刹车、倒车等行为。

```
public class Automotive{
……
void speedUp(){
        System.out.println("汽车加速");     //汽车加速方法
    }
void stop(){
        System.out.println("汽车刹车");     //汽车刹车方法
    }
void back(){
        System.out.println("汽车倒车");     //汽车倒车方法
    }
}
```

4. 对象的创建、使用和释放

(1)对象的创建。

对象的创建格式如下。

[修饰符] 类名 对象名 [=new 类名(实际参数列表)];

对象的创建包括声明、实例化和生成 3 部分。

- 声明是指定义一个类的对象，但不分配任何内存空间，而只是分配一个引用空间，对象的实际数据所在的内存地址尚未分配，如“Automotive car1;”。
- 实例化是利用运算符 new 为对象分配内存空间，它调用对象的构造方法并返回一个引用；一个类的不同对象分别存储在不同的内存空间里，如“Automotive car2=new Automotive();”。
- 生成是指执行构造方法，进行初始化；根据不同参数调用不同的构造方法，如“Automotive car3=new Automotive(num,br,cl,ml);”。

(2)对象的使用。

对象的使用格式如下。

对象名.变量名;

或

对象名.方法名([参数列表]);

通过运算符“.”可以实现对对象中的变量和方法的调用，可以通过设置访问权限来限制其他对象对它们的访问，如“car1.color;”和“ car1.stop();”等。

(3)对象的释放。

对象的释放格式如下。

对象名=null;

如“car1=null;”等。

5. 构造方法

构造方法是在创建类的对象时必须调用的一个方法，它的名字与类名相同，不具有任何

返回值，负责对象的初始化工作，如给成员变量赋值等。

构造方法声明的一般语法格式如下。

```
[修饰符] 构造方法名(参数列表)[throws 异常名1,异常名2,……]{
    ……//构造方法语句主体
}
```

相关说明如下。

- 构造方法和类应具有相同的名字。
- 一个类可以有多个构造方法，调用时根据不同的参数，执行不同的方法。
- 构造方法可以有0个或多个参数。
- 构造方法没有返回值。
- 构造方法总是和new运算符一起使用。
- 若类中没有定义构造方法，则Java虚拟机会提供一个默认的构造方法，此方法不带参数，主体不含任何语句。

【例2-4】 定义汽车类的两个构造方法。

```
public class Automotive{
……
Automotive(){
    number=0;
    brand=" ";
    color=" ";
    model=" ";
}                                    //汽车构造方法1,为属性赋值
Automotive(int num,String br,String cl,String ml){
    number=num;
    brand=br;
    color=cl;
    model=ml;
}                                    //汽车构造方法2,为属性赋值
    ……
}
```

构造方法重载是指一个类可以有多个构造方法，调用时可以根据不同的参数，执行不同的方法。

【课堂实训2-1】 汽车类的实现与测试

参考程序代码如下。

```
public class Automotive{
    int number;                      //车牌号属性
    String brand;                    //品牌属性
    String color;                    //颜色属性
```

```
    String model;                                    //型号属性
Automotive(){
    number=0;
    brand=" ";
    color=" ";
    model=" ";
}                                                    //汽车构造方法 1,为属性赋值
Automotive(int num,String br,String cl,String ml){
number=num;
brand=br;
color=cl;
model=ml;
}                                                    //汽车构造方法 2,为属性赋值
void speedUp(){
    System.out.println(brand +"汽车加速");
}                                                    //汽车加速方法
void stop(){
    System.out.println(brand +"汽车刹车");
}                                                    //汽车刹车方法
void back(){
    System.out.println(brand +"汽车倒车");
}                                                    //汽车倒车方法
public static void main(String args[]){
      Automotive car1=new Automotive();
      car1.speedUp();
      Automotive car2=new Automotive(74100,"奔驰","红色","E470");
      car2.speedUp();
    }
}
```

程序的运行结果如图 2-7 所示。

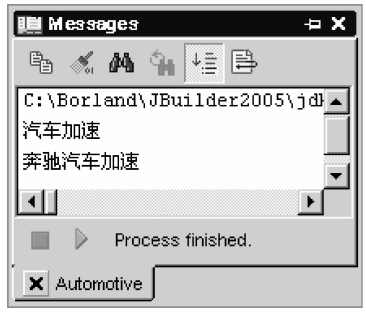

图 2-7　调用不同构造方法的运行结果

在该程序中 Automotive 类定义了两个构造方法，分别是 Automotive()和 Automotive(int num,String br,String cl,String ml)，当初始化对象时，系统根据参数个数和类型的不同决定调用哪个构造方法。显然，car1 调用构造方法 Automotive()，而 car2 调用构造方法 Automotive(int num,String br,String cl,String ml)，从而得到两辆不同的汽车。

2.3 继　　承

2.3.1 认识继承

继承是现实世界存在的一种现象，如图 2-8 所示是现实世界中人类的继承关系。

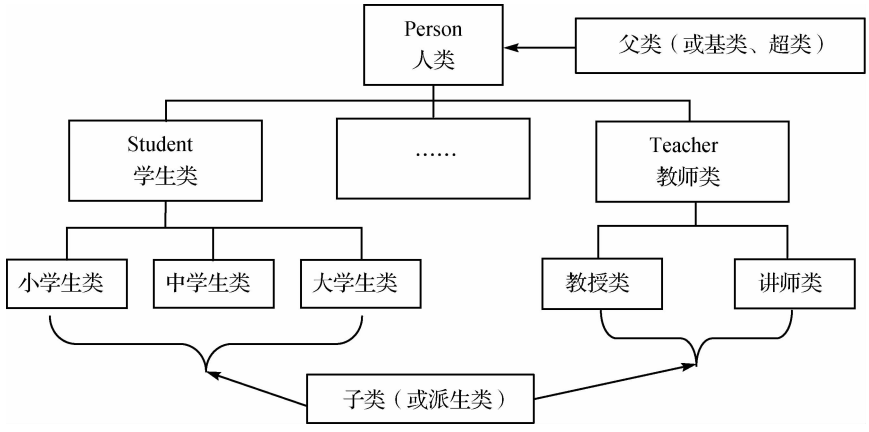

图 2-8　现实世界中人类的继承关系

从图 2-8 中可以看出，学生类和教师类都具有人的行为和特征，这是因为他们都继承了人类的所有行为和特征。此时，如果把人类称为父类(或基类、超类)，则学生类和教师类都是人类的子类。再从小一点的视角来看，小学生、中学生和大学生都具有学生的行为和特征，如果把学生类称为父类，则把小学生类、中学生类和大学生类称为学生类的子类。

2.3.2 实现继承

在面向对象编程中，如果有两个类，它们有部分相同的变量和方法，那么可以创建一个拥有两个类相同的变量和方法的父类，而这两个类就是父类的子类，它们可以直接继承父类中的所有变量和方法。在定义这两个类时，只需编写各自与父类不同的特性和行为。继承减少了编码量，提高了代码的可重用性。

当需要处理人类和学生类信息时，可以分别定义人类和学生类，如图 2-9 所示。

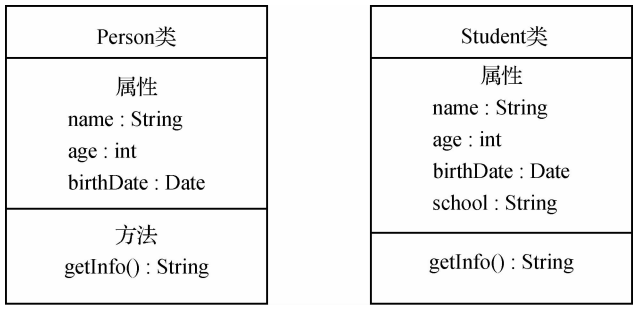

图 2-9　人类和学生类的类定义

从图 2-9 中可以发现，学生类除了自己特有的属性 school 外，其他属性和方法与人类的属性和方法完全相同，代码具有一定的重复性，这时可以将共同的属性和方法放在一个人类中，而学生类作为子类去继承父类，即继承人类的全部属性和方法，再编写自己与父类不同的属性或方法，从而得到具有继承关系的类图，如图 2-10 所示。

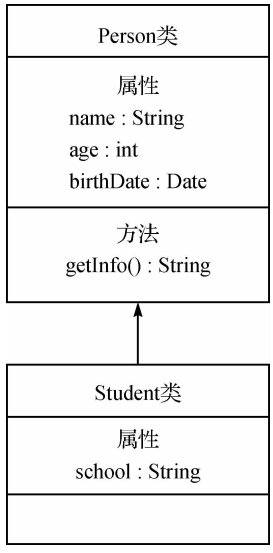

图 2-10　人类及其学生类子类

Java 语言中继承的实现主要有以下步骤。

(1)确定父类。确定父类的属性和方法，子类可从父类继承非私有属性和方法，构造方法不能被继承。

(2)定义子类。子类继承父类是通过 extends 关键字来实现的，其一般语法格式如下。

[类修饰符]　class 子类名 extends 父类名

(3)实现子类功能。与一般类相同，子类的方法与父类的方法同名时不能继承父类的该方法。

【例 2-5】 使用继承，创建名为 Person 的父类和名为 Student 的子类。

```
public class Person{
    String name;
```

```
    int age;
    Date birthDate;
    public String getInfo(){
    ……
    }
}
public class Student extends Person{
    String school;
}
```

关于继承的几点说明如下。

- Java 语言只支持单继承，不允许多重继承，即一个子类只能有一个父类，一个父类可以有多个子类。
- 继承具有传递性。子类沿继承路径向上继承所有父类的有关属性和方法。
- Object 类是所有 Java 类的最高层父类。如果在一个类的声明中没有指明这个类的直接父类，Java 语言则认为其为 Object 类的直接子类，如图 2-11 所示。

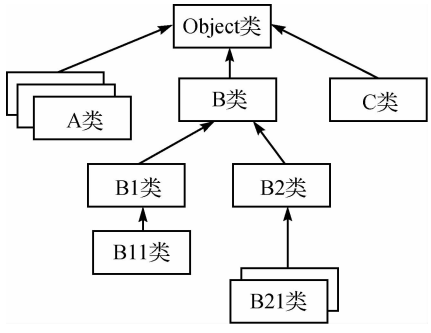

图 2-11　Java 语言中类的继承关系

2.3.3　子类的构造方法

构造方法用于初始化特定类型的对象并分配内存，构造方法名称与类名相同。创建对象时会自动调用构造方法。类似地，子类构造方法的名称也与子类名相同，创建子类的对象时将调用此构造方法。

【例 2-6】　编写学生子类的构造方法。

```
import java.util.Date;
class Person{
String name;
int age;
Date birthDate;
Person(){                                    //父类构造方法
    System.out.println("执行了人类的构造方法!");
}
```

```
public String getInfo(){
    return name+":"+age+","+birthDate;
}
}
class Student extends Person{
String school;
Student(){                                    //子类构造方法
System.out.println("执行了学生类的构造方法!");
}
}
public class PersonTest{
  public static void main(String[] args){
        Student s=new Student();
  }
}
```

程序的运行结果如图 2-12 所示。

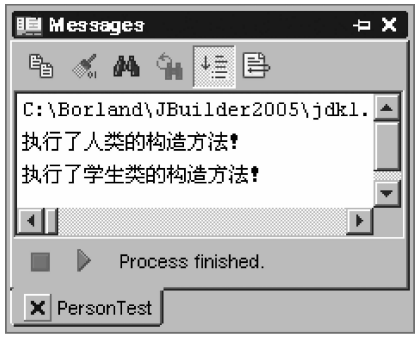

图 2-12 【例 2-6】的程序运行结果

从图 2-12 中可以看出，创建子类(Student)对象时，首先调用父类(Person)的构造方法，然后才调用子类的构造方法。

子类的构造方法中除了可以隐式调用父类的默认构造方法外，还可以通过关键字 super 来调用父类的默认构造方法或带参数的构造方法，同时，super 语句必须是子类构造方法中的第一个可执行语句。

【例 2-7】 在学生子类的构造方法中，调用带参数的父类构造方法。

```
import java.util.Date;
class Person{
String name;
int age;
Date birthDate;
Person(){
    System.out.println("执行了人类的构造方法!");
```

```
}
Person(String name,int age){
    this.name=name;
    this.age=age;
}
public String getInfo(){
    return name+":"+age+"岁";
}
}
class Student extends Person{
String school;
    Student(){
    super("瑞瑞",9);        //调用父类中带参数的构造方法
    System.out.println("执行了学生类的构造方法!");
}
}
public class PersonTest{
    public static void main(String[] args){
        Student s=new Student();
        System.out.println(s.getInfo());
}
}
```

程序的运行结果如图 2-13 所示。

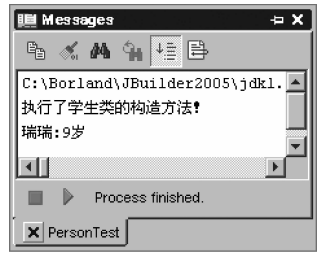

图 2-13　【例 2-7】的程序运行结果

在涉及构造方法的继承时，有以下两点需要说明。

- 对于每个可能成为父类的类，若需要编写有参数的构造方法，则必须提供一个无参数的构造方法。
- 若在子类中没有显式调用父类的构造方法，则会默认调用父类默认的构造方法。每个子类可以显式调用父类的一个构造方法，并且这个调用语句应该是该构造方法的第一个可执行语句。

【课堂实训 2-2】 灯与管状灯

在日常生活中，灯有瓦数、开关之分，可以通过开灯、关灯行为来改变灯的状态。管状灯是灯的一种，除具有灯的所有特性外，还有长度、颜色等特性。编程打印输出管状灯的相关信息。

1. 定义灯类

```
class Light{
    //两个成员变量
    private int watts;              //用于存放灯的瓦数
    private boolean indicator;  //用于存放灯的开或关的状态
    //两个构造器方法
    Light(int watts){
        this.watts=watts;                //用于创建具有 watts 的对象
    }
    Light(int watts,boolean indicator){
        this.watts=watts;
        this.indicator=indicator;
    }        //用于创建具有 watts,开关状态为 indicator 的对象
    //3 个成员方法
    public void switchOn(){
        indicator=true;
    }      //开灯
    public void switchOff(){
        indicator=false;
    }    //关灯
    public void printInfo(){  //输出灯的瓦数信息和开关状态
        if(indicator){
            System.out.println("灯的瓦数是:"+ watts+",灯开着!");
        }
        else{
            System.out.println("灯的瓦数是:"+ watts+",灯关着!");
    }
  }
}
```

2. 定义管状灯类

```
class TubeLight extends Light{       //继承 Light 类
    //两个成员变量
    private int tubeLength;          //用于存放灯管的长度
```

```
    private String color;              //用于存放灯管的颜色
    //构造器方法
    //创建具有 watts,灯管长度为 tubeLength,颜色为 color 的对象
    TubeLight(int watts,int tubeLength,String color){
        super(watts);
        this.tubeLength=tubeLength;
        this.color=color;
    }
    //成员方法
    //打印输出灯的相关信息,包括瓦数、开关信息、长度及颜色
    public void printInfo(){
        if(indicator){
        System.out.println("灯的瓦数是:" + watts +",灯的长度:"+tubeLength +
",灯的颜色:"+ color +",灯开着!");
        }
        else{
        System.out.println("灯的瓦数是:"+ watts +",灯的长度:"+tubeLength +
",灯的颜色:"+ color +",灯关着!");
        }
    }
}
```

3. 创建一个管状灯的实例

打印输出该灯的相关信息,该灯瓦数为 32,长度为 50,白色灯光,状态为开。

```
public class TubeLightTest{
    public static void main(String[] args){
        TubeLight tl=new TubeLight(32,50,"白色灯光");
        tl.switchOn();
        tl.printInfo();
    }
}
```

4. 运行结果

程序的运行结果如图 2-14 所示。

图 2-14　【课堂实训 2-2】的程序运行结果

2.4 小　　结

本模块介绍了如何将现实世界中的对象抽象化，建立基于面向对象的模型，以及模型在Java 语言中的编码实现，其中重点介绍了以下内容。

(1)如何理解面向对象编程技术中的类和对象。

(2)如何定义类的成员变量和成员方法。

(3)如何编写构造方法。

(4)如何理解和实现类的继承。

(5)如何编写子类的构造方法。