第 3 章　接口与内部类

接口是什么？接口有什么作用？如何使用接口？这些都是本章需要解决的问题。本章将详细地讲述接口的概念，并以实战结合的方式，学习这些抽象概念。本章还会介绍内部类的相关知识，包括内部类的种类和内部类的使用环境等等。

3.1　接口

接口同继承、多态一样，都是 Java 程序语言的特色。它贯穿了整个 Java 程序开发，是对继承的很好补充，其原因下面会详细的讲述。

3.1.1　接口概念的引入

为什么在买 USB 电脑鼠标的时候，不需要问电脑配件的商家，USB 鼠标是什么型号的？也不需要询问是满足什么要求？原因就是 USB 接口是统一的、固定不变的一种型号，是一种规范。所有的厂家都会按照这个规范，来制造 USB 接口的鼠标。这个规范说明制作该 USB 类型的鼠标应该做些什么，但并不说明如何做。

而 Java 程序设计中的接口，也是一种规范。这个接口定义了类应该做什么？但不关心如何做？即接口中只有方法名，没有方法体。

3.1.2　接口的概念

从专业的角度讲，接口只是说明类应该做什么，但并不指定应该如何去做。在实际开发过程中，通过类来实现接口。接口只有方法名没有方法体，实现接口就是让其既有方法名又有方法体。下面就举个有关接口的模型。

```
接口
 {
    应该作的事情一
    {.....}
    应该作的事情二
    {.....}
    应该作的事情三
    {.....}
 }
```

这个例子只是声明了要做什么事情，但没有说明如何做，需要一个类去实现它，将它的方法体完善。

3.1.3 接口的声明

接口的声明很简单，使用关键字“Interface”来声明。接口的形式跟类很相似，但要记住接口是接口，类是类，两者不能混为一谈。接口是要求类如何做的一套规范。下面将举一个实例，来演示如何声明接口。

```
///创建一个学校的接口
///在这个接口中，创建了很多个方法，并且这些方法没有方法体
public interface school
{
    void setschoolname();
    void setclassname();
    void setstudentname();
    void setstudentcode();
    void setstudentsexy();
    void setstudentbirthday();
    void setfamilyaddress();
    String getschoolname();
    String getclassname();
    String getstudentname();
    String getstudentcode();
    String getstudentsexy();
    String getstudentbirthday();
    String getfamilyaddress();}
```

上面的实例，演示了如何声明一个接口，可以看出整个接口中，只有几个设置器和访问器的方法名称，并没有真正实现方法。另外还有一点，接口的声明必须是“Public”的，否则没有任何意义。

3.1.4 接口的实现

接口的用处就是让类通过实现它，来执行一定的功能。下面通过实例演示接口的实现功能，在看实例之前，先看看这个实例的流程，如图 3.1 所示。

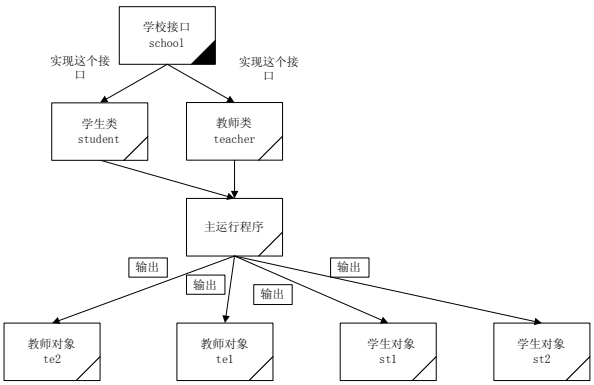

图 3.1　有关接口的一个实例

首先看看学校接口的设计。

```
///声明一个学校接口，来告诉程序需要做些什么
```

```
///其中包括了很多方法。但是都没有去实现。即都没有函数体
interface school
{
    void setschoolname(String schoolname);
    void setclassname(String schoolclassname);
    void setname(String name);
    void setcode(String code);
    void setsexy(String sexy);
    void setbirthday(String birthday);
    void setfamilyaddress(String familyaddress);
    String getschoolname();
    String getclassname();
    String getname();
    String getcode();
    String getsexy();
    String getbirthday();
    String getfamilyaddress();
}
```

下面再来设计学生类，其代码如下：

```
///创建一个类，让它实现学校这个接口
///通过设置器来设置各个参数
///通过访问器来获得对象的参数
///通过 tostring 方法来让对象以字符串形式输出。
class student implements school
{
    private String schoolname;
    private String classname;
    private String studentname;
    private String studentcode;
    private String studentsexy;
    private String studentbirthday;
    private String familyaddress;
    public void setschoolname(String schoolname)
    {
        this.schoolname=schoolname;
    }
    public void setclassname(String classname)
    {
    this.classname=classname;
    }
    public void setname(String studentname)
    {
        this.studentname=studentname;
    }
    public void setcode(String studentcode)
    {
        this.studentcode=studentcode;
    }
    public void setsexy(String studentsexy)
    {
        this.studentsexy=studentsexy;
    }
    public void setbirthday(String studentbirthday)
```

```
    {
        this.studentbirthday=studentbirthday;
    }
    public void setfamilyaddress(String familyaddress)
    {
        this.familyaddress=familyaddress;
    }
    public String getschoolname()
    {
        return schoolname;
    }
    public String getclassname()
    {
        return classname;
    }
    public String getname()
    {
        return studentname;
    }
    public String getcode()
    {
        return studentcode;
    }
    public String getsexy()
    {
        return studentsexy;
    }
    public String getbirthday()
    {
        return studentbirthday;
    }
    public String getfamilyaddress()
    {
        return familyaddress;
    }
    public String tostring()
    {
        String infor="学校名称："+schoolname+" "+"班级名称："+classname+" "+"学生姓名："
+studentname+" "+"学号："+studentcode+" "+"性别："+studentsexy+" "+"出生年月："
+studentbirthday+" "+"家庭地址："+familyaddress;
        return infor;
    }
}
```

下面再来设计教师类，其代码如下所示。

```
///让教师类实现学校这个接口
///设置器来设置各个参数
///通过访问器来获得各个参数
///通过 tostring 方法来让对象以字符串形式输出
class teacher implements school
{
    private String schoolname;
    private String classname;
    private String teachername;
```

```
private String teachercode;
private String teachersexy;
private String teacherbirthday;
private String familyaddress;
public void setschoolname(String schoolname)
{
    this.schoolname=schoolname;
}
public void setclassname(String classname)
{
    this.classname=classname;
}
public void setname(String teachername)
{
    this.teachername=teachername;
}
public void setcode(String teachercode)
{
    this.teachercode=teachercode;
}
public void setsexy(String teachersexy)
{
    this.teachersexy=teachersexy;
}
public void setbirthday(String teacherbirthday)
{
    this.teacherbirthday=teacherbirthday;
}
public void setfamilyaddress(String familyaddress)
{
    this.familyaddress=familyaddress;
}
public String getschoolname()
{
    return schoolname;
}
public String getclassname()
{
    return classname;
}
public String getname()
{
    return teachername;
}
public String getcode()
{
    return teachercode;
}
public String getsexy()
{
    return teachersexy;
}
public String getbirthday()
```

```
    {
        return teacherbirthday;
    }
    public String getfamilyaddress()
    {
        return familyaddress;
    }
    public String tostring()
    {
        String  infor="学校名称："+schoolname+"  "+"班级名称："+classname+"  "+"教师姓名："
"+teachername+"  "+"教师工号："+teachercode+"  "+"性别："+teachersexy+"  "+"出生年月："
"+teacherbirthday+"  "+"家庭地址："+familyaddress;
        return infor;
    }
}
```

主运行程序的代码如下所示。

```
///主运行函数
///从学生类中创建出几个对象
///从教师类中创建出几个对象
//通过设置器设置各个参数，在对象内部隐藏着访问器，来访问这些参数
///以字符串形式输出这些对象的字符串
public class schooltest
{
    public static void main(String[] args)
    {
        student st1=new student();
        student st2=new student();
        teacher te1=new teacher();
        teacher te2=new teacher();
        st1.setschoolname("重庆大学");
        st1.setclassname("计算机二班");
        st1.setname("王浩");
        st1.setcode("951034");
        st1.setsexy("男");
        st1.setbirthday("1975-07-21");
        st1.setfamilyaddress("上海市浦东新区");
        st2.setschoolname("重庆大学");
        st2.setclassname("计算机三班");
        st2.setname("赵丽");
        st2.setcode("951068");
        st2.setsexy("女");
        st2.setbirthday("1975-10-09");
        st2.setfamilyaddress("北京海淀区");
        te1.setschoolname("四川大学");
        te1.setclassname("计算机二班");
        te1.setname("孙敏");
        te1.setcode("00123");
        te1.setsexy("女");
        te1.setbirthday("1968-04-20");
        te1.setfamilyaddress("重庆市沙坪坝区");
        te2.setschoolname("四川大学");
        te2.setclassname("机械系三班");
        te2.setname("赵为民");
```

```
        te2.setcode("11233");
        te2.setsexy("男");
        te2.setbirthday("1961-02-13");
        te2.setfamilyaddress("成都市区");
        System.out.println(st1.tostring());
        System.out.println(st2.tostring());
        System.out.println(te1.tostring());
        System.out.println(te2.tostring());
    }
}
}
```

运行结果

学校名称：重庆大学　班级名称：计算机二班　学生姓名：王浩　学号：951034　性别：男
　出生年月：1975-07-21　家庭地址：上海市浦东新区
学校名称：重庆大学　班级名称：计算机三班　学生姓名：赵丽　学号：951068　性别：女
　出生年月：1975-10-09　家庭地址：北京海淀区
学校名称：四川大学　班级名称：计算机二班　教师姓名：孙敏　教师工号：00123　性别
：女　出生年月：1968-04-20　家庭地址：重庆市沙坪坝区
学校名称：四川大学　班级名称：机械系三班　教师姓名：赵为民　教师工号：11233　性
别：男　出生年月：1961-02-13　家庭地址：成都市区

举这个例子的目的就是了解接口的用处。在这个程序段中，将接口作为一种规范，当要提取一个学生类时，就用学生类来实现学校这个接口。当要提取一个教师类时，教师类也实现学校这个接口。

接口的用处在于让整个程序段中相同类型的类有一个统一的规范，这样看到接口的定义，就知道要实现它的类的功能。在类实现接口时，需要注意以下几点：

- 声明类需要实现指定的接口。
- 提供接口中所有方法的定义。
- 实现接口的类时，其访问控制符必须全部是“public”的。

3.1.5　接口的多重实现

前面提到过接口能够补充继承的不足，现在讲解如何补充。继承必须是单继承的，即一个类继承另一个类后，那这个类就不能继承其他类。而接口则无所谓，一个类可以实现一个接口，也可以同时实现另一个接口。使用接口为编程提供了很大的方便，可以把上面的程序段修改一下。为了能更好的理解这个程序，先看看程序的流程，如图 3.2 所示。

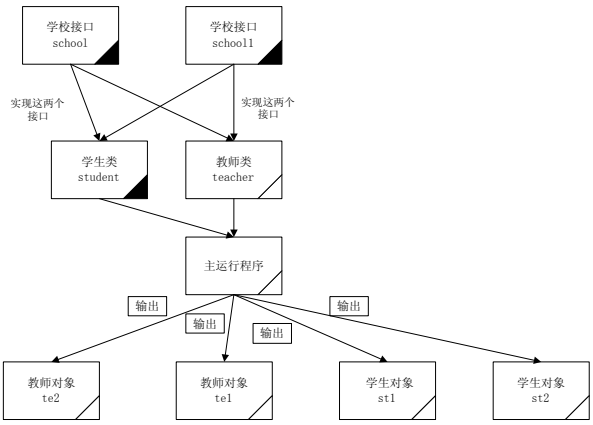

图 3.2　多重实现的流程图

首先是第一个学校接口的设计，代码如下所示。

```
interface school                                          //创建一个接口
{
    void setschoolname(String schoolname);
    void setclassname(String schoolclassname);
    void setname(String name);
    void setcode(String code);
    void setsexy(String sexy);
    void setbirthday(String birthday);
    void setfamilyaddress(String familyaddress);
}
```

第二个学校接口的设计，代码如下所示。

```
interface school1                                         //创建另一个接口
{
    String getschoolname();
    String getclassname();
    String getname();
    String getcode();
    String getsexy();
    String getbirthday();
    String getfamilyaddress();
}
```

下面学习如何设计学生类，详细代码如下所示。

```
///创建一个类，让它实现学校这两个接口
///通过设置器来设置各个参数
///通过访问器来获得对象的参数
///通过 tostring 方法来让对象以字符串形式输出。
class student implements school,school1                   //一个类实现两个不同的接口
{
    private String schoolname;
    private String classname;
    private String studentname;
    private String studentcode;
    private String studentsexy;
    private String studentbirthday;
    private String familyaddress;
    public void setschoolname(String schoolname)
    {
        this.schoolname=schoolname;
    }
    public void setclassname(String classname)
    {
        this.classname=classname;
    }
    public void setname(String studentname)
    {
        this.studentname=studentname;
    }
    public void setcode(String studentcode)
    {
        this.studentcode=studentcode;
```

```
    }
    public void setsexy(String studentsexy)
    {
        this.studentsexy=studentsexy;
    }
    public void setbirthday(String studentbirthday)
    {
        this.studentbirthday=studentbirthday;
    }
    public void setfamilyaddress(String familyaddress)
    {
        this.familyaddress=familyaddress;
    }
    public String getschoolname()
    {
        return schoolname;
    }
    public String getclassname()
    {
        return classname;
    }
    public String getname()
    {
        return studentname;
    }
    public String getcode()
    {
        return studentcode;
    }
    public String getsexy()
    {
        return studentsexy;
    }
    public String getbirthday()
    {
        return studentbirthday;
    }
    public String getfamilyaddress()
    {
        return familyaddress;
    }
    public String tostring()
    {
        String  infor="学校名称："+schoolname+"  "+"班级名称："+classname+"  "+"学生姓名："
"+studentname+"  "+"学号："+studentcode+"  "+"性别："+studentsexy+"  "+"出生年月："
"+studentbirthday+"  "+"家庭地址："+familyaddress;
        return infor;
    }
}
```

设计教师类的代码如下所示。

```
//让教师类实现学校这两个接口
///设置器来设置各个参数
///通过访问器来获得各个参数
```

```
///通过 tostring 方法来让对象以字符串形式输出
class teacher implements school,school1                    //一个类实现两个不同的接口
{
    private String schoolname;
    private String classname;
    private String teachername;
    private String teachercode;
    private String teachersexy;
    private String teacherbirthday;
    private String familyaddress;
    public void setschoolname(String schoolname)
    {
        this.schoolname=schoolname;
    }
    public void setclassname(String classname)
    {
        this.classname=classname;
    }
    public void setname(String teachername)
    {
        this.teachername=teachername;
    }
    public void setcode(String teachercode)
    {
        this.teachercode=teachercode;
    }
    public void setsexy(String teachersexy)
    {
        this.teachersexy=teachersexy;
    }
    public void setbirthday(String teacherbirthday)
    {
        this.teacherbirthday=teacherbirthday;
    }
    public void setfamilyaddress(String familyaddress)
    {
        this.familyaddress=familyaddress;
    }
    public String getschoolname()
    {
        return schoolname;
    }
    public String getclassname()
    {
        return classname;
    }
    public String getname()
    {
        return teachername;
    }
    public String getcode()
    {
        return teachercode;
```

```
    }
    public String getsexy()
    {
        return teachersexy;
    }
    public String getbirthday()
    {
        return teacherbirthday;
    }
    public String getfamilyaddress()
    {
        return familyaddress;
    }
    public String tostring()
    {
        String  infor="学校名称："+schoolname+"  "+"班级名称："+classname+"  "+"教师姓名："+teachername+"  "+"教师工号："+teachercode+"  "+"性别："+teachersexy+"  "+"出生年月："+teacherbirthday+"  "+"家庭地址："+familyaddress;
        return infor;
    }
}
```

主运行程序将上面设计的类，通过对象的形式输出。其详细代码如下所示。

```
///主运行函数
///从学生类中创建出几个对象
///从教师类中创建出几个对象
//通过设置器设置各个参数，在对象内部隐藏着访问器，来访问这些参数
///以字符串形式输出这些对象的字符串
public class schooltest
{
    public static void main(String[] args)
    {
        student st1=new student();
        student st2=new student();
        teacher te1=new teacher();
        teacher te2=new teacher();
        st1.setschoolname("重庆大学");
        st1.setclassname("计算机二班");
        st1.setname("王浩");
        st1.setcode("951034");
        st1.setsexy("男");
        st1.setbirthday("1975-07-21");
        st1.setfamilyaddress("上海市浦东新区");
        st2.setschoolname("重庆大学");
        st2.setclassname("计算机三班");
        st2.setname("赵丽");
        st2.setcode("951068");
        st2.setsexy("女");
        st2.setbirthday("1975-10-09");
        st2.setfamilyaddress("北京海淀区");
        te1.setschoolname("四川大学");
        te1.setclassname("计算机二班");
        te1.setname("孙敏");
        te1.setcode("00123");
```

```
            te1.setsexy("女");
            te1.setbirthday("1968-04-20");
            te1.setfamilyaddress("重庆市沙坪坝区");
            te2.setschoolname("四川大学");
            te2.setclassname("机械系三班");
            te2.setname("赵为民");
            te2.setcode("11233");
            te2.setsexy("男");
            te2.setbirthday("1961-02-13");
            te2.setfamilyaddress("成都市区");
            System.out.println(st1.tostring());
            System.out.println(st2.tostring());
            System.out.println(te1.tostring());
            System.out.println(te2.tostring());
        }
}
运行结果
学校名称：重庆大学　班级名称：计算机二班　学生姓名：王浩　学号：951034　性别：男
  出生年月：1975-07-21　家庭地址：上海市浦东新区
学校名称：重庆大学　班级名称：计算机三班　学生姓名：赵丽　学号：951068　性别：女
  出生年月：1975-10-09　家庭地址：北京海淀区
学校名称：四川大学　班级名称：计算机二班　教师姓名：孙敏　教师工号：00123　性别
：女　出生年月：1968-04-20　家庭地址：重庆市沙坪坝区
学校名称：四川大学　班级名称：机械系三班　教师姓名：赵为民　教师工号：11233　性
别：男　出生年月：1961-02-13　家庭地址：成都市区
```

在上面的程序段中，将一个接口分成了两个。然后用一个类同时实现两个接口，运行结果依然不变。如果是继承就不允许这样，因为一个类只能继承一个类，不能继承多个类，这就是接口的多重实现。

3.1.6　接口的属性

接口不是一个类，正因为其不是一个类，所以不能使用关键字“new”生成一个接口的实例。虽然这样，还是可以声明一个接口变量，如：“school sc”。

如果要生成一个接口的实例，可以让接口变量，指向一个已经实现了此接口的类的对象，如下面的例子。

```
School sc=new student();
```

另外，在接口中，不能声明实例字段及静态方法，但可以声明常量。其实接口不一定要有方法，也可以全部是常量。这个在后面的章节中，随着应用的加深，读者会看到和体会到。

3.1.7　接口的继承

接口从某些方面具有类的一些特性，如有方法、有属性，那么是否像类一样可以继承？回答是肯定的。接口的继承和类的继承一样，也是用关键字“extends”来实现，下面先看一个有关接口继承的实例。实例的流程如图 3.3 所示。

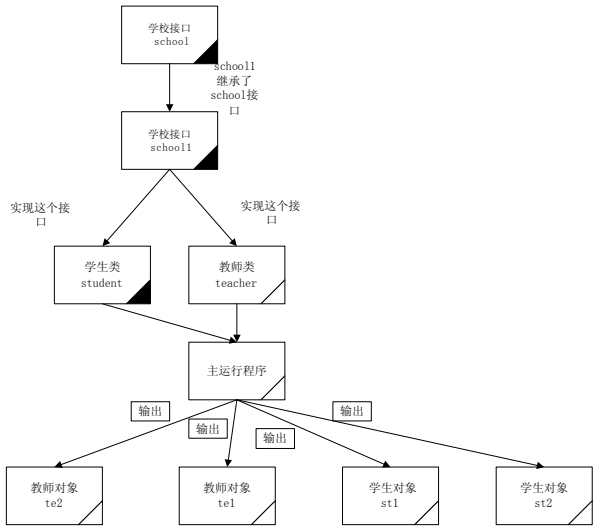

图 3.3　接口继承的流程图

首先看看学校接口的设计，代码如下所示。

```
///创建一个接口
interface school
{
    void setschoolname(String schoolname);
    void setclassname(String schoolclassname);
    String getschoolname();
    String getclassname();
}
```

设计另一个学校接口，代码如下所示。

```
///这个接口继承了上面那个接口
interface school1 extends school
{
    void setname(String name);
    void setcode(String code);
    void setsexy(String sexy);
    void setbirthday(String birthday);
    void setfamilyaddress(String familyaddress);
    String getname();
    String getcode();
    String getsexy();
    String getbirthday();
    String getfamilyaddress();
}
```

设计学生类，代码如下所示。

```
///创建一个类，让它实现学校这个接口
///通过设置器来设置各个参数
///通过访问器来获得对象的参数
///通过 tostring 方法来让对象以字符串形式输出。
class student implements school1                    //学生类实现子接口
{
    private String schoolname;
    private String classname;
    private String studentname;
    private String studentcode;
```

```
private String studentsexy;
private String studentbirthday;
private String familyaddress;
public void setschoolname(String schoolname)
{
    this.schoolname=schoolname;
}
public void setclassname(String classname)
{
    this.classname=classname;
}
public void setname(String studentname)
{
    this.studentname=studentname;
}
public void setcode(String studentcode)
{
    this.studentcode=studentcode;
}
public void setsexy(String studentsexy)
{
    this.studentsexy=studentsexy;
}
public void setbirthday(String studentbirthday)
{
    this.studentbirthday=studentbirthday;
}
public void setfamilyaddress(String familyaddress)
{
    this.familyaddress=familyaddress;
}
public String getschoolname()
{
    return schoolname;
}
public String getclassname()
{
    return classname;
}
public String getname()
{
    return studentname;
}
public String getcode()
{
    return studentcode;
}
public String getsexy()
{
    return studentsexy;
}
public String getbirthday()
{
```

```
        return studentbirthday;
    }
    public String getfamilyaddress()
    {
        return familyaddress;
    }
    public String tostring()
    {
        String  infor="学校名称："+schoolname+"  "+"班级名称："+classname+"  "+"学生姓名："+studentname+"  "+"学号："+studentcode+"  "+"性别："+studentsexy+"  "+"出生年月："+studentbirthday+"  "+"家庭地址："+familyaddress;
        return infor;
    }
}
```

设计教师类，详细代码如下。

```
///创建一个类，让它实现学校这个接口
///通过设置器来设置各个参数
///通过访问器来获得对象的参数
///通过 tostring 方法来让对象以字符串形式输出。
class teacher implements school1
{
    private String schoolname;
    private String classname;
    private String teachername;
    private String teachercode;
    private String teachersexy;
    private String teacherbirthday;
    private String familyaddress;
    public void setschoolname(String schoolname)
    {
        this.schoolname=schoolname;
    }
    public void setclassname(String classname)
    {
        this.classname=classname;
    }
    public void setname(String teachername)
    {
        this.teachername=teachername;
    }
    public void setcode(String teachercode)
    {
        this.teachercode=teachercode;
    }
    public void setsexy(String teachersexy)
    {
        this.teachersexy=teachersexy;
    }
    public void setbirthday(String teacherbirthday)
    {
        this.teacherbirthday=teacherbirthday;
    }
    public void setfamilyaddress(String familyaddress)
```

```
    {
        this.familyaddress=familyaddress;
    }
    public String getschoolname()
    {
        return schoolname;
    }
    public String getclassname()
    {
        return classname;
    }
    public String getname()
    {
        return teachername;
    }
    public String getcode()
    {
        return teachercode;
    }
    public String getsexy()
    {
        return teachersexy;
    }
    public String getbirthday()
    {
        return teacherbirthday;
    }
    public String getfamilyaddress()
    {
        return familyaddress;
    }
    public String tostring()
    {
        String  infor="学校名称："+schoolname+"  "+"班级名称："+classname+"  "+"教师姓名："
+teachername+"  "+"教师工号："+teachercode+"  "+"性别："+teachersexy+"  "+"出生年月："
+teacherbirthday+"  "+"家庭地址："+familyaddress;
        return infor;
    }
}
```

主运行类的代码如下所示。

```
///主运行函数
///从学生类中创建出几个对象
///从教师类中创建出几个对象
//通过设置器设置各个参数，在对象内部隐藏着访问器，来访问这些参数
///以字符串形式输出这些对象的字符串
public class schooltest
{
    public static void main(String[] args)
    {
        student st1=new student();
        student st2=new student();
        teacher te1=new teacher();
        teacher te2=new teacher();
```

```
        st1.setschoolname("重庆大学");
        st1.setclassname("计算机二班");
        st1.setname("王浩");
        st1.setcode("951034");
        st1.setsexy("男");
        st1.setbirthday("1975-07-21");
        st1.setfamilyaddress("上海市浦东新区");
        st2.setschoolname("重庆大学");
        st2.setclassname("计算机三班");
        st2.setname("赵丽");
        st2.setcode("951068");
        st2.setsexy("女");
        st2.setbirthday("1975-10-09");
        st2.setfamilyaddress("北京海淀区");
        te1.setschoolname("四川大学");
        te1.setclassname("计算机二班");
        te1.setname("孙敏");
        te1.setcode("00123");
        te1.setsexy("女");
        te1.setbirthday("1968-04-20");
        te1.setfamilyaddress("重庆市沙坪坝区");
        te2.setschoolname("四川大学");
        te2.setclassname("机械系三班");
        te2.setname("赵为民");
        te2.setcode("11233");
        te2.setsexy("男");
        te2.setbirthday("1961-02-13");
        te2.setfamilyaddress("成都市区");
        System.out.println(st1.tostring());
        System.out.println(st2.tostring());
        System.out.println(te1.tostring());
        System.out.println(te2.tostring());
    }
}
运行结果
学校名称：重庆大学  班级名称：计算机二班  学生姓名：王浩  学号：951034  性别：男
  出生年月：1975-07-21  家庭地址：上海市浦东新区
学校名称：重庆大学  班级名称：计算机三班  学生姓名：赵丽  学号：951068  性别：女
  出生年月：1975-10-09  家庭地址：北京海淀区
学校名称：四川大学  班级名称：计算机二班  教师姓名：孙敏  教师工号：00123  性别
：女  出生年月：1968-04-20  家庭地址：重庆市沙坪坝区
学校名称：四川大学  班级名称：机械系三班  教师姓名：赵为民  教师工号：11233  性
别：男  出生年月：1961-02-13  家庭地址：成都市区
```

3.1.8 接口意义

接口不仅仅是一种规范，还是一种编程的思路。接口的所有方法和属性，都代表了后面将要设计的类的基本思路，这些方法就代表着这个程序的需求，所以掌握好接口，对学好 Java 程序开发非常关键。

3.2 内部类

内部类就是在一个类的内部再创建一个类。下面介绍如何使用内部类编写程序代码，并了解内部类在编写代码的过程中，为程序员提供了哪些方便和优点。

内部类究竟有什么好处：

- 内部类的对象能够访问创建它的对象的所有方法和属性，包括私有数据。
- 对于同一个外包中的其他类来说，内部类是隐形的。
- 匿名内部类可以很方便的定义回调。
- 使用内部类可以很方便的编写事件驱动的程序。

下面将这些特点贯穿整节，通过实例来讲述。

3.2.1 使用内部类来访问对象

下面将举个有关内部类的实例，在分析这个实例之前，先了解这个实例的流程，如图3.4 所示。

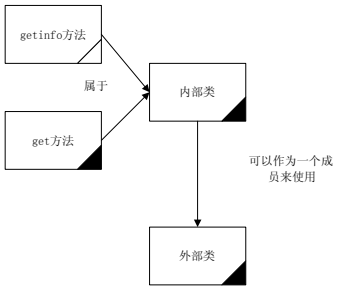

图 3.4 内部类访问对象的实例

首先看看如何设计学生类，代码如下所示。

```
///创建一个学生类
///在学生类中，创建姓名，学号，性别等参数
///使用设置器来设置参数
///使用访问器来获取参数值
public class students
{
    private String name;
    private String code;
    private String sexy;
    private String birthday;
    private String address;
    public void setname(String name)
    {
        this.name=name;
    }
    public void setcode(String code)
    {
        this.code=code;
    }
    public void setsexy(String sexy)
```

```
    {
        this.sexy=sexy;
    }
    public void setbirthday(String birthday)
    {
        this.birthday=birthday;
    }
    public void setaddress(String address)
    {
        this.address=address;
    }
    public String getname()
    {
        return name;
    }
    public String getcode()
    {
        return code;
    }
    public String getsexy()
    {
        return sexy;
    }
    public String getbirthday()
    {
        return birthday;
    }
    public String getaddress()
    {
        return address;
    }
    public String tostring()
    {
        String infor="学生姓名："+name+"  "+"学号："+code+"  "+"性别："+sexy+"  "+"出生年月："
"+birthday+"  "+"家庭地址："+address;
        return infor;
    }
    public void setstudentcourse(String[] courses)
    {
        new course(courses);
    }

}
```

再设计一个内部类，代码如下所示。

```
///内部类的创建，把内部类作为外部类的一个成员
private class course
{
    private   String[] courses;
    private int coursenum;
///内部类的构造器
    public course(String[] course)
    {
        courses=course;
```

```
            coursenum=course.length;
            getinfo();
        }
///获得课程数组中的课程
        private void get()
        {
            for(int i=0;i<coursenum;i++)
            {
                System.out.print("    "+courses[i]);
            }
        }
///按字符串形式输出
        void getinfo()
        {
            System.out.println("学生姓名："+students.this.name+"学生学号："+students.this.code+"一
共选择了："+coursenum+"门科，分别是：");
            get();
        }
}
}
```

在主运行方法内部实现输出，详细代码如下所示。

```
///在主运行方法中，通过学生类的方法来访问学生类的内部类 courses
public static void main(String[] args)
    {
        String[] courses={"语文","数学","英语","化学"};
        students st=new students();
        st.setname("王浩");
        st.setcode("200123");
        st.setsexy("男");
        st.setaddress("北京海淀区");
        System.out.println(st.tostring());
        st.setstudentcourse(courses);
    }
```

其运行结果如下：

```
运行结果
学生姓名：王浩   学号：200123   性别：男   出生年月：null   家庭地址：北京海淀区
学生姓名：王浩学生学号：200123 一共选择了：4 门科，分别是：
  语文   数学   英语   化学
```

类的访问控制符在前面章节中讲过，只能有“public”和“default”两种。那为什么在这里的内部类会出现 private 呢？

作为一个单独的类，的确只能有“public”和“default”两种访问控制符，但是作为内部类，就可以使用“private”控制符。当内部类设置为“private”，包含此内部类的外部类的方法才可以访问它。在这个程序段里，“students”类中的方法可以访问这个内部类。

内部类如何创建对象呢？其实可以像一般的类一样，直接使用“new”关键字来创建。在上面的实例中，使用一个方法来创建内部类的实例对象。

在上面的程序段中，内部类的构造器中包含了这个内部类要实现的所有方法。这里注意，当类创建出对象之前，首先会访问构造器，会运行构造器中的所有方法。这样，就相当于直接访问了内部类的私有方法，这种方法可以将对象中所有方法一并实现。

3.2.2 局部内部类

本小节通过对比局部变量，来学习局部内部类。局部变量就是在某个类的方法中定义的变量，它的作用范围就在这个方法体内。同样局部内部类就是在类的方法中定义的一个内部类，它的作用范围也在这个方法体内。把上面的实例修改一下，学习局部内部类的使用。这个实例的流程，如图 3.5 所示。

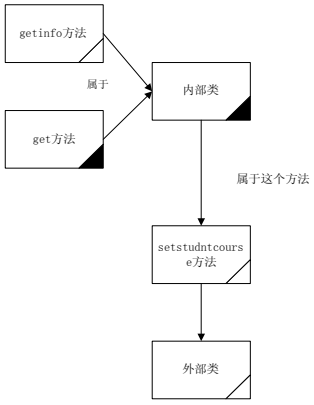

图 3.5　局部内部类的实例流程图

首先设计一个学生类，代码如下所示。

```
///这是一个学生类
///通过设置器来设置对象参数
///通过访问器来获得对象参数
///通过 tostring 方法让对象以字符串形式输出。
public class students
{
    private String name;
    private String code;
    private String sexy;
    private String birthday;
    private String address;
    public void setname(String name)
    {
        this.name=name;
    }
    public void setcode(String code)
    {
        this.code=code;
    }
    public void setsexy(String sexy)
    {
        this.sexy=sexy;
    }
    public void setbirthday(String birthday)
    {
        this.birthday=birthday;
    }
    public void setaddress(String address)
   {
```

```
        this.address=address;
    }

    public String getname()
    {
        return name;
    }
    public String getcode()
    {
        return code;
    }
    public String getsexy()
    {
        return sexy;
    }
    public String getbirthday()
    {
        return birthday;
    }
    public String getaddress()
    {
        return address;
    }
    public String tostring()
    {
        String infor="学生姓名："+name+"  "+"学号："+code+"  "+"性别："+sexy+"  "+"出生年月："
"+birthday+"  "+"家庭地址："+address;
        return infor;
    }
}
```

在前面设计的外部类的方法中，设计一个内部类，代码如下所示。

```
///将课程类作为外部类中的一个方法中的成员
    public void setstudentcourse(String[] courses)
    {
        class course
        {
            private   String[] courses;
            private int coursenum;
            public course(String[] course)
            {
                courses=course;
                coursenum=course.length;
                getinfo();
    }
    private void get()
    {
        for(int i=0;i<coursenum;i++)
        {
            System.out.print("   "+courses[i]);
        }
    }
    private void getinfo()
    {
```

```
            System.out.println("学生姓名："+students.this.name+"学生学号："+students.this.code+"一
共选择了："+coursenum+"门科，分别是：");
            get();
            }
        }
///创建了一个内部类的对象，随着包含这个内部类的外部方法一起运行
        new course(courses);
}
```

在主运行程序中，实现对象的输出，代码如下所示。

```
///在主运行方法中，使用学生类对象的方法，来访问局部类 courses
        public static void main(String[] args)
        {
        String[] courses={"语文","数学","英语","化学"};
        students st=new students();
        st.setname("王浩");
        st.setcode("200123");
        st.setsexy("男");
        st.setaddress("北京海淀区");
        System.out.println(st.tostring());
        st.setstudentcourse(courses);
        }
}
运行结果
学生姓名：王浩　学号：200123　性别：男　出生年月：null　家庭地址：北京海淀区
学生姓名：王浩学生学号：200123 一共选择了：4 门科，分别是：
  语文　数学　英语　化学
```

这个程序从上一节的实例程序修改过来，把内部类放到了“students”类的一个方法体内，这样的内部类就是局部内部类。

局部内部类是定义在外部类的方法中，与局部变量类似，在局部内部类前不加修饰符“public”和“private”，其范围为定义它的代码块。局部内部类不仅可以访问外部类实例变量，还可以访问外部类的局部常量，但要求外部类的局部变量是“final”的。其实，以上做法相当于是为内部类添加了一个属性，这个属性就是外部类的“final”局部变量。在类外不可直接访问局部内部类，以保证局部内部类对外是不可见的，只有在方法中才能调用其局部内部类。

3.2.3　静态内部类

当一个内部类不需要引用它的外部类的方法、属性值时，可以将这个类设置为“static”，这就是静态内部类。既然是静态的，包含它的类要引用它时，就可以不必创建对象，直接引用。在静态内部类中只能访问外部类的静态成员。构造静态内部类对象，不再需要构造外部类对象。

3.2.4　匿名内部类

在编写程序代码时，不一定要给内部类取一个名字，可以直接以对象名来代替。在图形化编程的事件监控器代码中，会大量使用匿名内部类，这样可以大大的简化代码的编写，并

增强了代码的可读性。

3.3 常见疑难解答

3.3.1 匿名类如何在程序中使用

答：匿名类是一种特殊的局部内部类，用来继承一个类或者实现一个接口。匿名内部类不能定义构造方法匿名内部类。在编译的时候由系统自动起名 Out$1.class。如果一个对象编译时的类型是接口，那么其运行的类型是实现这个接口的类，因为匿名内部类无构造方法，所以其使用范围非常的有限。

3.3.2 接口与继承有什么区别

答：接口在本质上就是一个特殊的类。在语法上跟继承有着很大的差别。

- 属性：接口中的所有属性都是公开静态常量，继承则无所谓。
- 方法：接口中所有方法都是公开抽象方法，继承中所有的方法不一定都是抽象。
- 接口方法：接口没有构造器，继承有构造器。