day04 【String类、综合练习】

今日内容

- String类的常用方法
- 面向对象综合练习

教学目标

- [ ] 能够了解String类常用方法的功能及使用
- [ ] 能够编写主菜单
- [ ] 能够定义Person类并应用模板模式
- [ ] 能够定义子类Student类并添加特有成员
- [ ] 能够定义子类Teacher类并添加特有成员
- [ ] 能够理解继承在案例中的使用
- [ ] 能够理解模板模式在案例中的使用
- [ ] 能够定义并使用打印Person的静态方法
- [ ] 能够定义并使用打印ArrayList的静态方法
- [ ] 能够理解静态成员和静态方法在案例中的使用

第一章 String类常用方法

1.1 concat

- 方法原型：public String concat (String str)
- 功能：将当前字符串与参数字符串进行"拼接"，返回一个新字符串。

```
String s1 = "Hello";
String s2 = "World";
String result = s1.concat(s2);
System.out.println("result = " + result);
System.out.println("s1 = " + s1);
System.out.println("s2 = " + s2);
```

concat的作用和+符号的不同：

1). concat只能拼接String类型，不能拼接其它类型。

+符号可以拼接任何类型。

2). concat的效率要高于+符号。

1.2 contains

- 方法原型：public boolean contains (CharSequence s)

  CharSequence是一个接口，String类是它的实现类。

- 功能：判断参数字符串在当前字符串中是否存在(区分大小写)。存在，返回true，否则，返回false。

```
String s = "我爱Java，我爱学习！";
System.out.println("字符串中是否包含Java：" + s.contains("Java"));//true
System.out.println("字符串中是否包含java：" + s.contains("java"))//false
```

1.3 endsWith

- 方法原型：public boolean endsWith(String suffix)
- 功能：测试此字符串是否以指定的后缀结尾(区分大小写)。

```
String name = "Test.java";
System.out.println("判断name是否以java结尾：" + name.endsWith("java"));//true
System.out.println("判断name是否以Java结尾：" + name.endsWith("Java"));//false
```

1.4 startsWith

- 方法原型：public boolean startsWith(String prefix)
- 功能：测试此字符串是否以指定的前缀开始(区分大小写)

```
String name = "我爱Java";
System.out.println("字符串是否以‘我’开头：" + name.startsWith("我"));//true
```

1.5 indexOf

- 方法原型：public int indexOf(String str)
- 功能：返回指定子字符串第一次出现的字符串内的索引。如果不包含，则返回-1.

```
String str = "我爱Java，我爱学习！";
System.out.println("在字符串中，Java第一次出现的位置：" +
str.indexOf("Java"));//2
System.out.println("在字符串中，java第一次出现的位置：" +
str.indexOf("java"));//-1
```

1.6 lastIndexOf

- 方法原型：public int lastIndexOf(String str)
- 功能：返回指定子字符串最后一次出现的字符串中的索引。 如果不包含，则返回-1.

```
String str = "我爱Java，我爱学习！";
System.out.println("在字符串中，'我'最后一次出现的位置：" +
str.lastIndexOf("我"));//7
```

1.7 replace

- 方法原型：public String replace(CharSequence target,CharSequence replacement)
- 功能：将与字面目标序列匹配的字符串的每个子字符串替换为指定的文字替换序列。 替换从字符串开始到结束，例如，在字符串“aaa”中用“b”替换“aa”将导致“ba”而不是“ab”。

```
String str = "我爱吃红烧鲤鱼，我太想吃红烧鲤鱼了！";
System.out.println("将'红烧鲤鱼'替换为'咸水鸡'：" + str.replace("红烧鲤鱼","咸水
鸡"));
System.out.println("原字符串：" + str);
```

1.8 substring

- 方法原型：public String substring(int beginIndex)：将当前字符串从beginIndex开始截取到末尾。
- 方法原型：public String substring(int beginIndex, int endIndex)：将当前字符串从beginIndex开始截取到endIndex - 1处。
- 功能：截取字符串，并将截取后的字符串返回。原字符串不变。

```
String str = "我爱Java";
System.out.println("截取'Java'：" + str.substring(2));//Java
System.out.println("截取'我爱'：" + str.substring(0,2))//我爱
```

1.9 toCharArray

- 方法原型：public char[] toCharArray()
- 功能：将当前字符串转换为char[]数组。

```
String str = "身无彩凤双飞翼";
char[] chArray = str.toCharArray();
System.out.println(chArray);
```

1.10 toLowerCase

- 方法原型： public String toLowerCase()
- 功能：将当前字符串中的所有英文字符转换为小写，并返回一个转换后的新字符串，原字符串不变。

```
String str = "我爱Java";
System.out.println("转换为小写：" + str.toLowerCase());//我爱java
System.out.println("原字符串：" + str);//我爱Java
```

1.11 toUpperCase

- 方法原型：public String toUpperCase()
- 功能：将当前字符串中的所有英文字符转换为大写，并返回一个转换后的新字符串，原字符串不变。

```
String str = "我爱Java";
System.out.println("转换为大写：" + str.toUpperCase());//我爱JAVA
System.out.println("原字符串：" + str);//我爱Java
```

1.12 trim

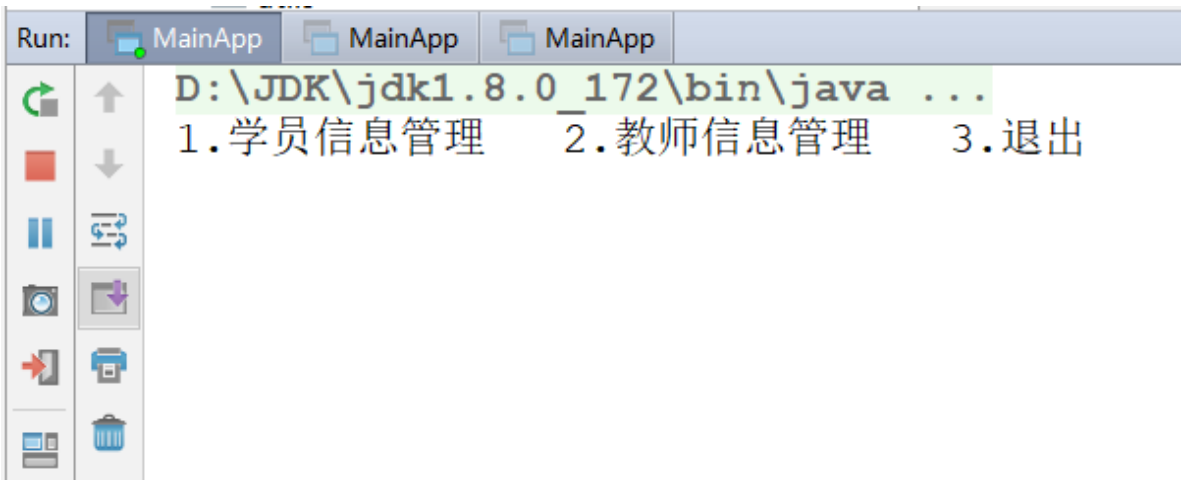
- 方法原型：public String trim()
- 功能：去掉当前字符串的前后空格，并返回一个新字符串，原字符串不变。

```
String str = "  ad  min     ";
System.out.println("去掉前后空格后|" + str.trim() + "|");//去掉前后空格后|ad
min|
System.out.println("原字符串|" + str + "|");//原字符串|  ad  min     |
```

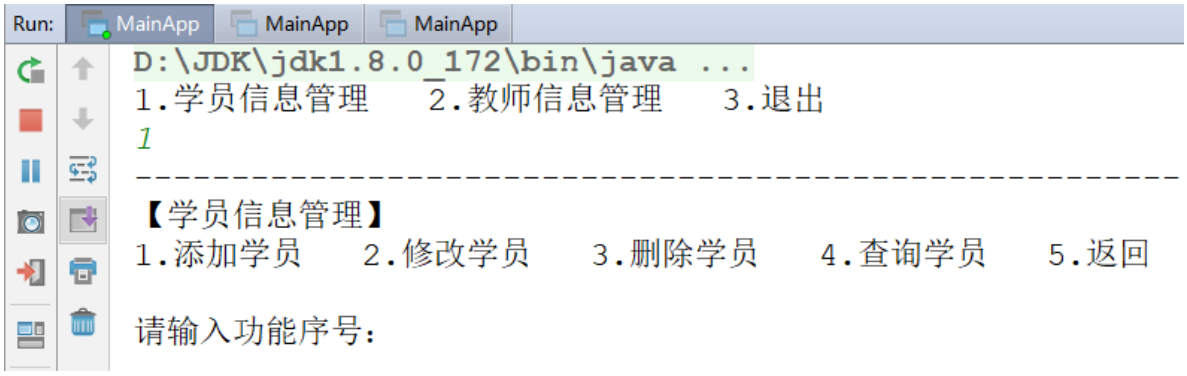
第二章 综合案例-案例演示

2.1 程序启动

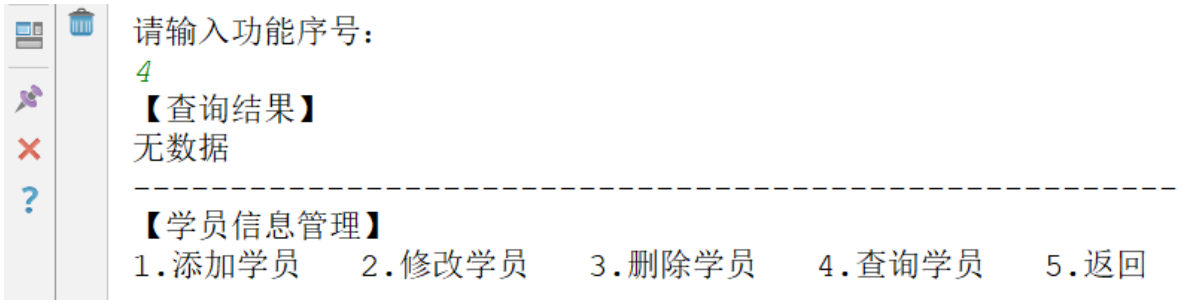
运行com.itheima.main.MainApp类，启动程序：



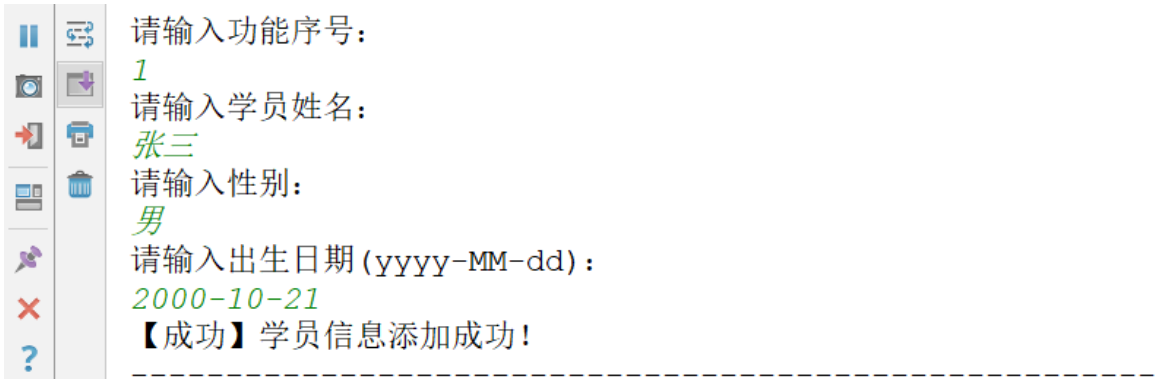
2.2 测试学员信息管理模块



2.3 测试【4.查询学员】

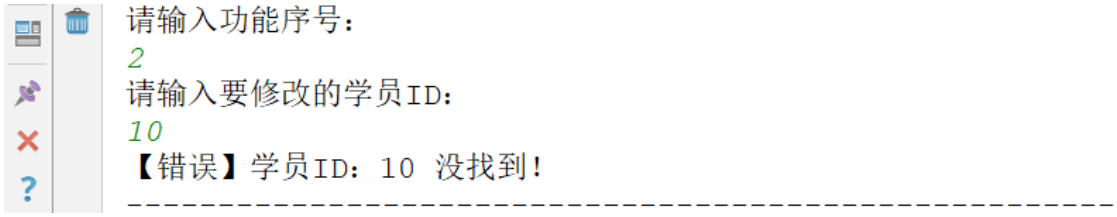


2.4 测试【1.添加学员】



2.5 测试【2.修改学员】

- 输入不存在的编号:



- 输入存在的编号:

```
请输入功能序号：
2
请输入要修改的学员ID：
1
【查询结果】要修改的学员信息：
********************************************************************************
编号      姓名      性别      生日          年龄      描述
1         张三      男        2000-10-21    18        我是一名：学生 我的工作是：学习Java
********************************************************************************
请输入新姓名(保留原值输入0)：
王五
请输入新性别(保留原值输入0)：
0
请输入新出生日期(yyyy-MM-dd)(保留原值输入0)：
2000-10-10
【成功】学员信息修改成功！
--------------------------------------------------
```

2.6 测试【3.删除学员】

- 输入不存在的编号：

```
请输入功能序号：
3
请输入要删除的学员ID：
10
【错误】学员ID：10 未找到！
--------------------------------------------------
```

- 输入存在的编号，但取消操作：

```
请输入功能序号：
3
请输入要删除的学员ID：
1
【查询结果】要删除的学员信息：
********************************************************************************
编号      姓名      性别      生日          年龄      描述
1         王五      男        2000-10-10    18        我是一名：学生 我的工作是：学习Java
********************************************************************************
【确认】您确定要删除这条信息吗(y/n)？
n
【取消】操作被取消！
--------------------------------------------------
```

- 输入存在的编号，执行删除：

```
请输入功能序号：
3
请输入要删除的学员ID：
1
【查询结果】要删除的学员信息：
********************************************************************************
编号      姓名      性别      生日          年龄      描述
1         王五      男        2000-10-10    18        我是一名：学生 我的工作是：学习Java
********************************************************************************
【确认】您确定要删除这条信息吗(y/n)？
y
【成功】数据已被删除！
--------------------------------------------------
```

第三章 综合案例-类设计

3.1 父类Person(抽象)

- 成员属性：
  - id(编号)
  - name(姓名)
  - sex(性别)

    - birthday(生日)
    - age(年龄-由生日计算得出)
- 构造方法：
    - 无参构造
    - 全参构造
- 成员方法：
    - toString()
- 抽象方法：
    - getType()：由各子类实现，返回各自的"类型"字符串。
    - getWork()：由各子类实现，返回各自的"工作"字符串。

3.2 子类Student

- 构造方法
    - 无参构造
    - 全参构造(super调用父类全参构造)
- 重写抽象方法
    - 重写getType()
    - 重写getWork()

3.3 子类Teacher

- 构造方法
    - 无参构造
    - 全参构造(super调用父类全参构造)
- 重写抽象方法
    - 重写getType()
    - 重写getWork()

3.4 工具类Utils类

- 全局变量
    - 学员ID值(添加学员信息时，编号由此ID加1生成)
    - 教师ID值(添加教师信息时，编号由此ID加1生成)
- 全局方法
    - 根据生日计算年龄的方法
    - 打印一个Person对象的方法；
    - 打印一个ArrayList集合的方法；

3.5 启动类

- 定义启动类：MainApp启动程序。

第四章 综合案例-类制作

4.1 父类Person(抽象)

```
public abstract class Person {
    private int id;//编号
    private String name;//姓名
    private String sex;//性别
    private String birthday;//出生日期
    private int age;//年龄--通过出生日期换算
    //构造方法
    public Person() {
    }
    public Person(int id,String name, String sex, String birthday)  {
        this.id = id;
        this.name = name;
        this.sex = sex;
        this.birthday = birthday;
    }
    //getter/setter
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getSex() {
        return sex;
    }
    public void setSex(String sex) {
        this.sex = sex;
    }
    public String getBirthday() {
        return birthday;
    }
    public void setBirthday(String birthday) {
        this.birthday = birthday;
    }
    public int getAge()  {
        //通过生日计算年龄
        age = Utils.birthdayToAge(this.getBirthday());
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    //重写toString，同时作为模板
    @Override
    public String toString() {
        return  id + "\t\t" +
                name + "\t\t" +
```

```
                sex + "\t\t" +
                birthday + "\t" +
                this.getAge() + "\t\t" +
                " 我是一名：" + getType() + " 我的工作是：" + getWork();
    }
    //模板用到的两个方法，由子类重写
    public abstract String getWork();
    public abstract String getType();
}
```

4.2 子类Student

```
public class Student extends Person {
    public Student() {
    }
    public Student(int id, String name, String sex, String birthday) {
        super(id, name, sex, birthday);
    }
    @Override
    public String getWork() {
        return "学习Java";
    }
    @Override
    public String getType() {
        return "学生";
    }
}
```

4.3 子类Teacher

```
public class Teacher extends Person {
    public Teacher() {
    }
    public Teacher(int id, String name, String sex, String birthday) {
        super(id, name, sex, birthday);
    }
    @Override
    public String getWork() {
        return "讲课";
    }
    @Override
    public String getType() {
        return "老师";
    }
}
```

4.4 工具类Utils类

```
public class Utils {
```

```
    public static int stuId ;//学员ID的初始值
    public static int teaId ;//教师ID的初始值
    static {
        stuId = 0;
        teaId = 0;
        //后期可以改为从文件/数据库读取初始值
    }
    public static int birthdayToAge(String birthday)  {
        Date birthDate = null;
        try {//异常处理代码，后面讲
            birthDate = new SimpleDateFormat("yyyy-MM-dd").parse(birthday);
        } catch (ParseException e) {
            e.printStackTrace();
            return -1;
        }
        //获取当前系统时间
        Calendar cal = Calendar.getInstance();
        //如果出生日期大于当前时间，则返回-1
        if (cal.before(birthDate)) {
            return -1;
        }
        //取出系统当前时间的年、月、日部分
        int yearNow = cal.get(Calendar.YEAR);
        int monthNow = cal.get(Calendar.MONTH);
        int dayOfMonthNow = cal.get(Calendar.DAY_OF_MONTH);

        //将日期设置为出生日期
        cal.setTime(birthDate);
        //取出出生日期的年、月、日部分
        int yearBirth = cal.get(Calendar.YEAR);
        int monthBirth = cal.get(Calendar.MONTH);
        int dayOfMonthBirth = cal.get(Calendar.DAY_OF_MONTH);
        //当前年份与出生年份相减，初步计算年龄
        int age = yearNow - yearBirth;
        //当前月份与出生日期的月份相比，如果月份小于出生月份，则年龄减1，表示不满多少周岁
        if (monthNow <= monthBirth) {
            //如果月份相等，在比较日期，如果当前日，小于出生日，也减1，表示不满多少周岁
            if (monthNow == monthBirth) {
                if (dayOfMonthNow < dayOfMonthBirth) age--;
            }else{
                age--;
            }
        }
        return age;

    }
    //打印ArrayList的方法
    public static void printPersonList(ArrayList personList) {

System.out.println("************************************************************");
        System.out.println("编号\t\t姓名\t\t性别\t\t生日\t\t\t年龄\t\t描述");
        for (int i = 0; i < personList.size(); i++) {
            Object p = personList.get(i);
            System.out.println(personList.get(i));
        }

System.out.println("************************************************************");
    }
```

```
    //打印Person的方法
    public static void printPerson(Person person) {

 System.out.println("************************************************************");
        System.out.println("编号\t\t姓名\t\t性别\t\t生日\t\t\t年龄\t\t描述");
        System.out.println(person );

 System.out.println("************************************************************");
    }
}
```

第五章 综合案例-启动类实现

5.1 主菜单

```
public class MainApp {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        //学生集合
        ArrayList<Student> stuList = new ArrayList<>();
        //教师集合
        ArrayList<Teacher> teaList = new ArrayList<>();
        //主菜单
        while (true) {
            System.out.println("1.学员信息管理    2.教师信息管理    3.退出");
            int op = sc.nextInt();
            switch (op) {
                case 1:
                    studentManage(stuList,sc);
                    break;
                case 2:
                    teacherManage(teaList,sc);
                    break;
                case 3:
                    System.out.println("谢谢使用，拜拜！！");
                    System.exit(0);
                default:
                    System.out.println("你的输入有误，请重新输入！");
                    break;
            }
        }
    }
    //教师信息管理
    private static void teacherManage(ArrayList<Teacher> teaList,Scanner sc) {

    }

    //学员信息管理
    private static void studentManage(ArrayList<Student> stuList,Scanner sc) {

    }
}
```

5.2 学员信息管理二级菜单

```
public class MainApp {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        //学生集合
        ArrayList<Student> stuList = new ArrayList<>();
        //教师集合
        ArrayList<Teacher> teaList = new ArrayList<>();
        //主菜单
        while (true) {
            System.out.println("1.学员信息管理    2.教师信息管理    3.退出");
            int op = sc.nextInt();
            switch (op) {
                case 1:
                    studentManage(stuList,sc);
                    break;
                case 2:
                    teacherManage(teaList,sc);
                    break;
                case 3:
                    System.out.println("谢谢使用，拜拜！！");
                    System.exit(0);
                default:
                    System.out.println("你的输入有误，请重新输入！");
                    break;
            }
        }
    }
    //教师信息管理
    private static void teacherManage(ArrayList<Teacher> teaList,Scanner sc) {

    }

    //学员信息管理
    private static void studentManage(ArrayList<Student> stuList,Scanner sc) {

        //二级菜单
        while (true) {
            System.out.println("----------------------------------------------
------");
            System.out.println("【学员信息管理】");
            System.out.println("1.添加学员    2.修改学员    3.删除学员    4.查询学员
5.返回");
            System.out.println();
            System.out.println("请输入功能序号：");
            int op = sc.nextInt();
            switch (op) {
                case 1:
                    addStudent(stuList, sc);
                    break;
                case 2:
                    updateStudent(stuList, sc);
                    break;
                case 3:
                    deleteStudent(stuList, sc);
```

```
                    break;
                case 4:
                    selectAll(stuList, sc);
                    break;
                case 5:
                    return;//结束方法
                default:
                    System.out.println("你的输入有误，请重新输入！");
                    break;
            }
        }
    }
    //添加学员
    private static void addStudent(ArrayList<Student> stuList, Scanner sc) {

    }
    //修改学员
    private static void updateStudent(ArrayList<Student> stuList, Scanner sc) {

    }
    //删除学员
    private static void deleteStudent(ArrayList<Student> stuList, Scanner sc) {

    }
    //查询所有学员
    private static void selectAll(ArrayList<Student> stuList, Scanner sc) {

    }
}
```

5.3 查询所有学员

```
//查询所有学员
private static void selectAll(ArrayList<Student> stuList, Scanner sc) {
    System.out.println("【查询结果】");
    if (stuList.size() == 0) {
        System.out.println("无数据");
        return;
    }
    Utils.printPersonList(stuList);//调用工具类打印
}
```

5.4 添加学员

```
//添加学员
private static void addStudent(ArrayList<Student> stuList, Scanner sc) {
    System.out.println("请输入学员姓名：");
    String name = sc.next();
    System.out.println("请输入性别：");
    String sex = sc.next();
    System.out.println("请输入出生日期(yyyy-MM-dd)：");
    String birthday = sc.next();

    stuList.add(new Student(++Utils.stuId,name,sex,birthday));

    System.out.println("【成功】学员信息添加成功！");
}
```

5.5 修改学员

```
//修改学员
private static void updateStudent(ArrayList<Student> stuList, Scanner sc) {
    System.out.println("请输入要修改的学员ID：");
    int stuId = sc.nextInt();
    //查询
    for (int i = 0; i < stuList.size(); i++) {
        Student stu = stuList.get(i);
        if (stu.getId() == stuId) {
            System.out.println("【查询结果】要修改的学员信息：");
            //打印
            Utils.printPerson(stu);
            //执行修改
            System.out.println("请输入新姓名(保留原值输入0)：");
            String newName = sc.next();
            System.out.println("请输入新性别(保留原值输入0)：");
            String newSex = sc.next();
            System.out.println("请输入新出生日期(yyyy-MM-dd)(保留原值输入0)：");
            String newBirthday = sc.next();

            if (!"0".equals(newName)) {
                stu.setName(newName);
            }
            if (!"0".equals(newSex)) {
                stu.setSex(newSex);
            }
            if (!"0".equals(newBirthday)) {
                stu.setBirthday(newBirthday);
            }
            System.out.println("【成功】学员信息修改成功！");
            return;
        }

    }
    System.out.println("【错误】学员ID：" + stuId + " 没找到！");
}
```

5.6 删除学员

```
//删除学员
private static void deleteStudent(ArrayList<Student> stuList, Scanner sc) {
    System.out.println("请输入要删除的学员ID：");
    int stuId = sc.nextInt();
    //查询
    for (int i = 0; i < stuList.size(); i++) {
        Student stu = stuList.get(i);
        if (stu.getId() == stuId) {
            System.out.println("【查询结果】要删除的学员信息：");
            Utils.printPerson(stu);
            System.out.println("【确认】您确定要删除这条信息吗(y/n)？");
            String str = sc.next();
            if ("y".equals(str)) {
                stuList.remove(i);
                System.out.println("【成功】数据已被删除！");
                return;//结束方法
            }else{
                System.out.println("【取消】操作被取消！");
                return;
            }

        }

    }
    System.out.println("【错误】学员ID：" + stuId + " 未找到！");
}
```

第六章 课堂练习

6.1参考学员管理实现教师管理模块