1.案例驱动模式

1.1案例驱动模式概述 (理解)

通过我们已掌握的知识点,先实现一个案例,然后找出这个案例中,存在的一些问题,在通过新知识点解决问题

1.2案例驱动模式的好处 (理解)

- 解决重复代码过多的冗余,提高代码的复用性
- 解决业务逻辑聚集紧密导致的可读性差,提高代码的可读性
- 解决代码可维护性差,提高代码的维护性

2.分类思想

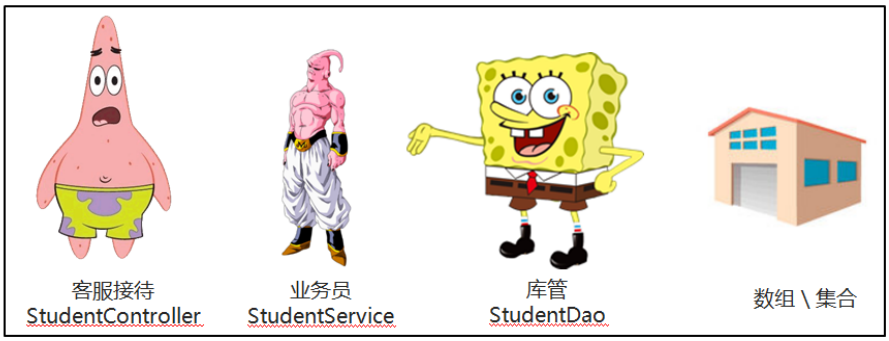
2.1分类思想概述 (理解)

分工协作,专人干专事

2.2黑马信息管理系统 (理解)

- Student类 标准学生类,封装键盘录入的学生信息(id , name , age , birthday)
- StudentDao类 Dao : (Data Access Object 缩写) 用于访问存储数据的数组或集合
- StudentService类 用来进行业务逻辑的处理(例如: 判断录入的id是否存在)
- StudentController类 和用户打交道(接收用户需求,采集用户信息,打印数据到控制台)

用户



3.分包思想

3.1分包思想概述 (理解)

如果将所有的类文件都放在同一个包下,不利于管理和后期维护,所以,对于不同功能的类文件,可以放在不同的包下进行管理

3.2包的概述 (记忆)

- 包

  本质上就是文件夹

- 创建包

  多级包之间使用 " . " 进行分割 多级包的定义规范：公司的网站地址翻转(去掉www) 比如：黑马程序员的网站址为www.itheima.com 后期我们所定义的包的结构就是：com.itheima.其他的包名

- 包的命名规则

  字母都是小写

3.3包的注意事项 (理解)

- package语句必须是程序的第一条可执行的代码
- package语句在一个java文件中只能有一个
- 如果没有package,默认表示无包名

3.4类与类之间的访问 (理解)

- 同一个包下的访问

  不需要导包，直接使用即可

- 不同包下的访问

  1.import 导包后访问

  2.通过全类名（包名 + 类名）访问

- 注意：import 、package 、class 三个关键字的摆放位置存在顺序关系

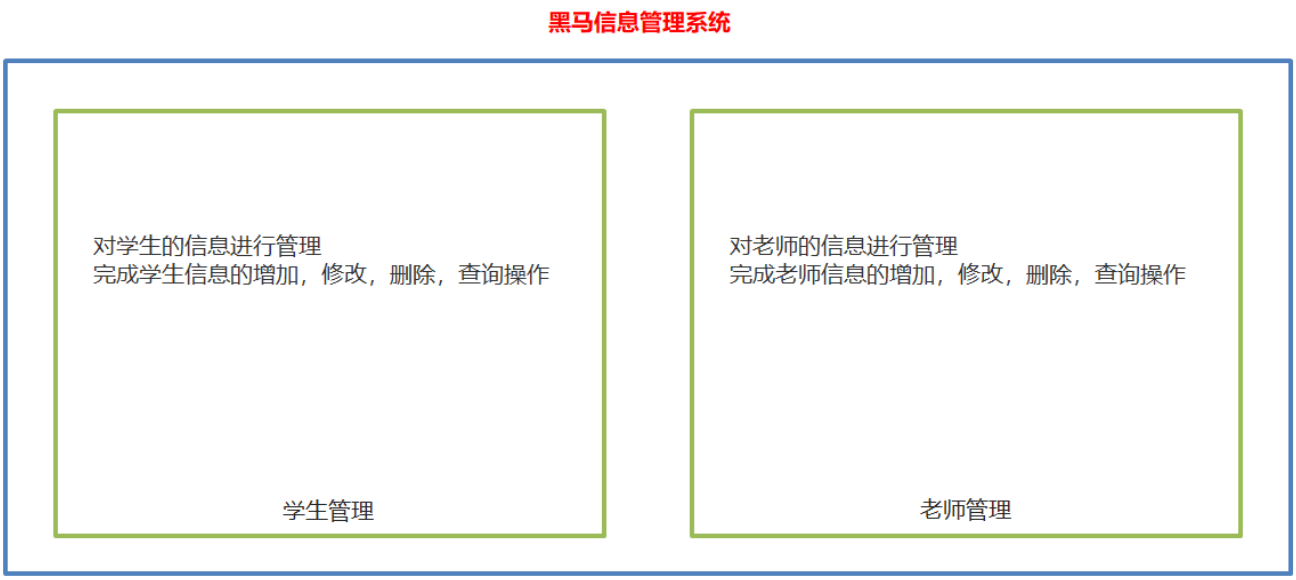
  package 必须是程序的第一条可执行的代码

  import 需要写在 package 下面

  class 需要在 import 下面

4.黑马信息管理系统

4.1系统介绍 (理解)



4.2学生管理系统 (应用)

4.2.1需求说明

- 添加学生: 键盘录入学生信息(id，name，age，birthday)

  使用数组存储学生信息,要求学生的id不能重复
- 删除学生: 键盘录入要删除学生的id值,将该学生从数组中移除,如果录入的id在数组中不存在,需要重新录入
- 修改学生: 键盘录入要修改学生的id值和修改后的学生信息

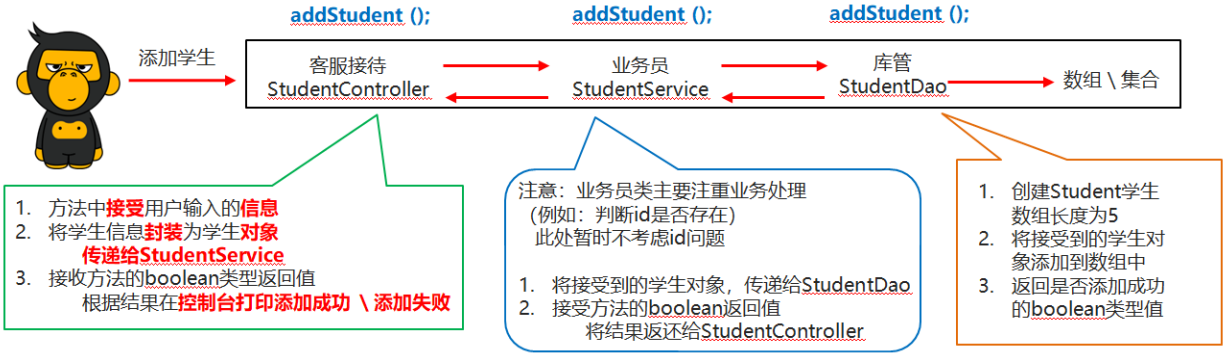
  将数组中该学生的信息修改,如果录入的id在数组中不存在,需要重新录入
- 查询学生: 将数组中存储的所有学生的信息输出到控制台

4.2.2实现步骤

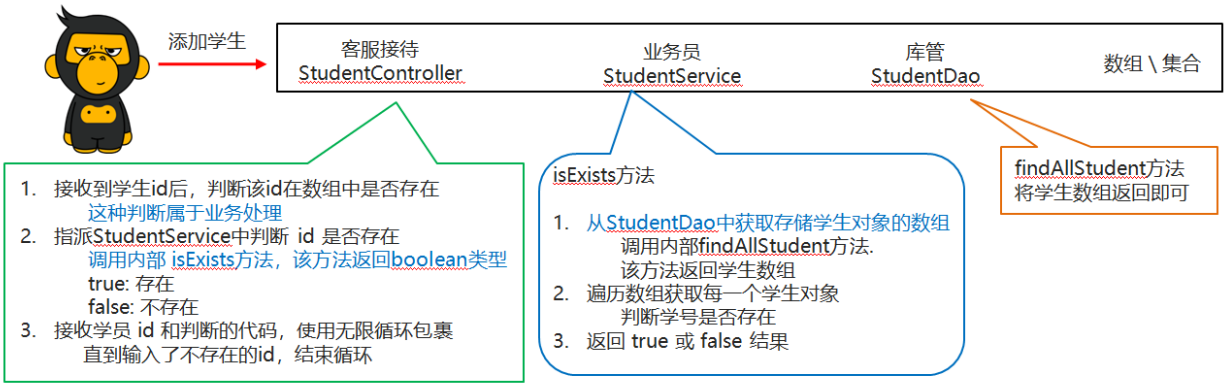
- 环境搭建实现步骤

包	存储的类	作用
com.itheima.edu.info.manager.domain	Student.java	封装学生信息
com.itheima.edu.info.manager.dao	StudentDao.java	访问存储数据的数组，进行赠删改查（库管）
com.itheima.edu.info.manager.service	StudentService.java	业务的逻辑处理（业务员）
com.itheima.edu.info.manager.controller	StudentController.java	和用户打交道（客服接待）
com.itheima.edu.info.manager.entry	InfoManagerEntry.java	程序的入口类，提供一个main方法

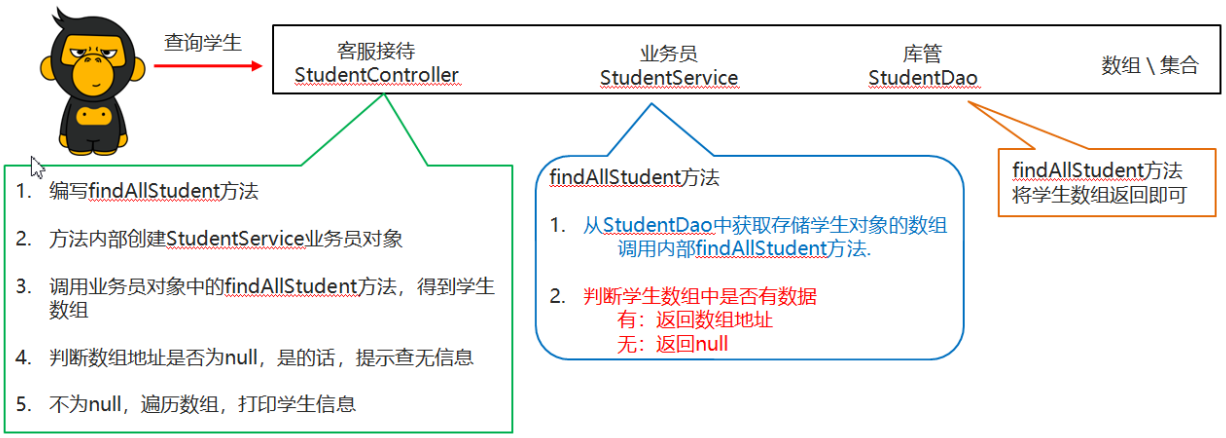
- 菜单搭建实现步骤
  - 需求
    - 黑马管理系统菜单搭建
    - 学生管理系统菜单搭建
  - 实现步骤
    1. 展示欢迎页面,用输出语句完成主界面的编写
    2. 获取用户的选择,用Scanner实现键盘录入数据
    3. 根据用户的选择执行对应的操作,用switch语句完成操作的选择
- 添加功能实现步骤



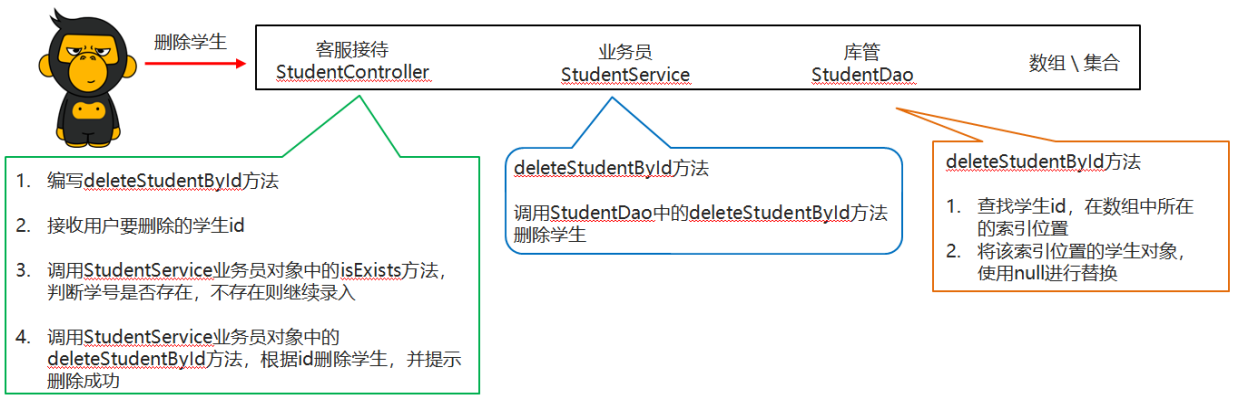
- 添加功能优化:判断id是否存在



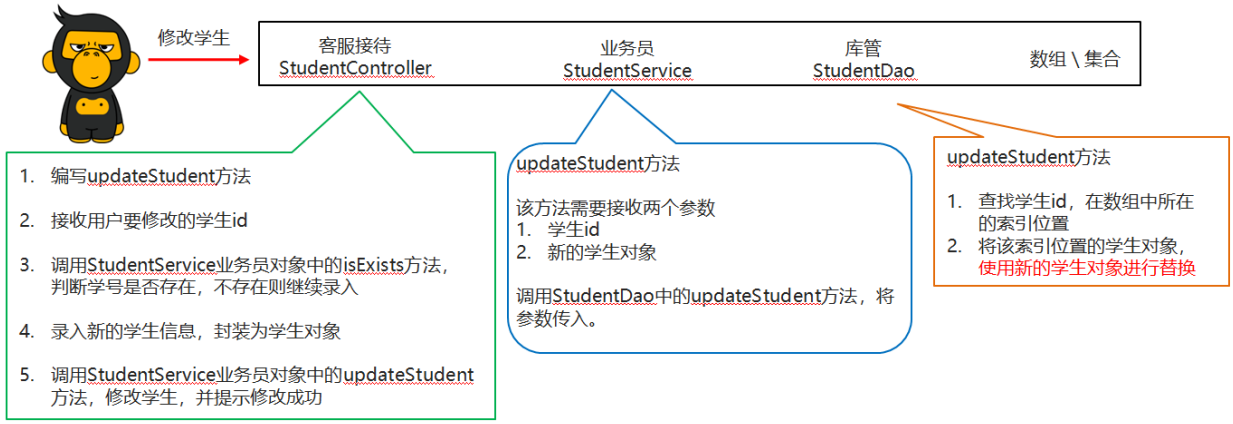
- 查询功能实现步骤



- 删除功能实现步骤



- 修改功能实现步骤



- 系统优化
  - 把updateStudent和deleteStudentById中录入学生id代码抽取到一个方法(inputStudentId)中
    该方法的主要作用就是录入学生的id，方法的返回值为String类型
- 把addStudent和updateStudent中录入学生信息的代码抽取到一个方法(inputStudentInfo)中
  该方法的主要作用就是录入学生的信息，并封装为学生对象，方法的返回值为Student类型

4.2.3代码实现

学生类

```
public class Student {
    private String id;
    private String name;
    private String age;
    private String birthday;
    String address;

    public Student() {
    }

    public Student(String id, String name, String age, String birthday) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.birthday = birthday;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    public String getAge() {
        return age;
    }

    public void setAge(String age) {
        this.age = age;
    }

    public String getBirthday() {
        return birthday;
    }

    public void setBirthday(String birthday) {
        this.birthday = birthday;
    }

}
```

程序入口InfoManagerEntry类

```
public class InfoManagerEntry {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (true) {
            // 主菜单搭建
            System.out.println("--------欢迎来到黑马信息管理系统--------");
            System.out.println("请输入您的选择：1.学生管理  2.老师管理  3.退出");
            String choice = sc.next();
            switch (choice) {
                case "1":
                    // System.out.println("学生管理");
                    // 开启学生管理系统
                    StudentController studentController = new StudentController();
                    studentController.start();
                    break;
                case "2":
                    System.out.println("老师管理");
                    TeacherController teacherController = new TeacherController();
                    teacherController.start();
                    break;
                case "3":
                    System.out.println("感谢您的使用");
                    // 退出当前正在运行的JVM虚拟机
                    System.exit(0);
                    break;
                default:
                    System.out.println("您的输入有误，请重新输入");
                    break;
            }
```

```
        }
    }
}
```

StudentController类

```
public class StudentController {
    // 业务员对象
    private StudentService studentService = new StudentService();

    private Scanner sc = new Scanner(System.in);

    // 开启学生管理系统，并展示学生管理系统菜单
    public void start() {
        //Scanner sc = new Scanner(System.in);
        studentLoop:
        while (true) {
            System.out.println("--------欢迎来到 <学生> 管理系统--------");
            System.out.println("请输入您的选择：1.添加学生  2.删除学生  3.修改学生  4.查看学生  5.退
出");
            String choice = sc.next();
            switch (choice) {
                case "1":
                    // System.out.println("添加");
                    addStudent();
                    break;
                case "2":
                    // System.out.println("删除");
                    deleteStudentById();
                    break;
                case "3":
                    // System.out.println("修改");
                    updateStudent();
                    break;
                case "4":
                    // System.out.println("查询");
                    findAllStudent();
                    break;
                case "5":
                    System.out.println("感谢您使用学生管理系统，再见!");
                    break studentLoop;
                default:
                    System.out.println("您的输入有误，请重新输入");
                    break;
            }
        }
    }

    // 修改学生方法
    public void updateStudent() {
        String updateId = inputStudentId();
        Student newStu = inputStudentInfo(updateId);
        studentService.updateStudent(updateId, newStu);
```

```
        System.out.println("修改成功!");
    }

    // 删除学生方法
    public void deleteStudentById() {
        String delId = inputStudentId();
        // 3. 调用业务员中的deleteStudentById根据id，删除学生
        studentService.deleteStudentById(delId);
        // 4. 提示删除成功
        System.out.println("删除成功!");
    }

    // 查看学生方法
    public void findAllStudent() {
        // 1. 调用业务员中的获取方法，得到学生的对象数组
        Student[] stus = studentService.findAllStudent();
        // 2. 判断数组的内存地址，是否为null
        if (stus == null) {
            System.out.println("查无信息，请添加后重试");
            return;
        }
        // 3. 遍历数组，获取学生信息并打印在控制台
        System.out.println("学号\t\t姓名\t年龄\t生日");
        for (int i = 0; i < stus.length; i++) {
            Student stu = stus[i];
            if (stu != null) {
                System.out.println(stu.getId() + "\t" + stu.getName() + "\t" + stu.getAge() +
"\t\t" + stu.getBirthday());
            }
        }
    }

    // 添加学生方法
    public void addStudent() {
        // StudentService studentService = new StudentService();
        // 1. 键盘接收学生信息
        String id;
        while (true) {
            System.out.println("请输入学生id:");
            id = sc.next();
            boolean flag = studentService.isExists(id);
            if (flag) {
                System.out.println("学号已被占用，请重新输入");
            } else {
                break;
            }
        }

        Student stu = inputStudentInfo(id);

        // 3. 将学生对象,传递给StudentService(业务员)中的addStudent方法

        boolean result = studentService.addStudent(stu);
```

```
        // 4. 根据返回的boolean类型结果，在控制台打印成功\失败
        if (result) {
            System.out.println("添加成功");
        } else {
            System.out.println("添加失败");
        }
    }

    // 键盘录入学生id
    public String inputStudentId() {
        String id;
        while (true) {
            System.out.println("请输入学生id:");
            id = sc.next();
            boolean exists = studentService.isExists(id);
            if (!exists) {
                System.out.println("您输入的id不存在，请重新输入:");
            } else {
                break;
            }
        }
        return id;
    }

    // 键盘录入学生信息
    public Student inputStudentInfo(String id) {
        System.out.println("请输入学生姓名:");
        String name = sc.next();
        System.out.println("请输入学生年龄:");
        String age = sc.next();
        System.out.println("请输入学生生日:");
        String birthday = sc.next();
        Student stu = new Student();
        stu.setId(id);
        stu.setName(name);
        stu.setAge(age);
        stu.setBirthday(birthday);
        return stu;
    }
}
```

StudentService类

```
public class StudentService {
    // 创建StudentDao (库管)
    private StudentDao studentDao = new StudentDao();
    // 添加学生方法
    public boolean addStudent(Student stu) {
        // 2. 将学生对象，传递给StudentDao 库管中的addStudent方法
        // 3. 将返回的boolean类型结果，返还给StudentController
        return studentDao.addStudent(stu);
    }
    // 判断学号是否存在方法
```

```
    public boolean isExists(String id) {
        Student[] stus = studentDao.findAllStudent();
        // 假设id在数组中不存在
        boolean exists = false;
        // 遍历数组，获取每一个学生对象，准备进行判断
        for (int i = 0; i < stus.length; i++) {
            Student student = stus[i];
            if(student != null && student.getId().equals(id)){
                exists = true;
                break;
            }
        }

        return exists;
    }
    // 查看学生方法
    public Student[] findAllStudent() {
        // 1. 调用库管对象的findAllStudent获取学生对象数组
        Student[] allStudent = studentDao.findAllStudent();
        // 2. 判断数组中是否有学生信息（有：返回地址， 没有：返回null）
        // 思路：数组中只要存在一个不是null的元素，那就代表有学生信息
        boolean flag = false;
        for (int i = 0; i < allStudent.length; i++) {
            Student stu = allStudent[i];
            if(stu != null){
                flag = true;
                break;
            }
        }

        if(flag){
            // 有信息
            return allStudent;
        }else{
            // 没有信息
            return null;
        }

    }

    public void deleteStudentById(String delId) {
        studentDao.deleteStudentById(delId);
    }

    public void updateStudent(String updateId, Student newStu) {
        studentDao.updateStudent(updateId, newStu);
    }
}
```

StudentDao类

```
public class StudentDao {
    // 创建学生对象数组
```

```
    private static Student[] stus = new Student[5];
    // 添加学生方法
    public boolean addStudent(Student stu) {

        // 2. 添加学生到数组
        //2.1 定义变量index为-1，假设数组已经全部存满，没有null的元素
        int index = -1;
        //2.2 遍历数组取出每一个元素，判断是否是null
        for (int i = 0; i < stus.length; i++) {
            Student student = stus[i];
            if(student == null){
                index = i;
                //2.3 如果为null，让index变量记录当前索引位置，并使用break结束循环遍历
                break;
            }
        }

        // 3. 返回是否添加成功的boolean类型状态
        if(index == -1){
            // 装满了
            return false;
        }else{
            // 没有装满，正常添加，返回true
            stus[index] = stu;
            return true;
        }
    }
    // 查看学生方法
    public Student[] findAllStudent() {
        return stus;
    }

    public void deleteStudentById(String delId) {
        // 1. 查找id在容器中所在的索引位置
        int index = getIndex(delId);
        // 2. 将该索引位置,使用null元素进行覆盖
        stus[index] = null;
    }

    public int getIndex(String id){
        int index = -1;
        for (int i = 0; i < stus.length; i++) {
            Student stu = stus[i];
            if(stu != null && stu.getId().equals(id)){
                index = i;
                break;
            }
        }
        return index;
    }

    public void updateStudent(String updateId, Student newStu) {

        // 1. 查找updateId，在容器中的索引位置
```

```
            int index = getIndex(updateId);
            // 2. 将该索引位置，使用新的学生对象替换
            stus[index] = newStu;
        }
    }
```

4.3老师管理系统 (应用)

4.3.1需求说明

- 添加老师: 通过键盘录入老师信息(id，name，age，birthday)

  使用数组存储老师信息,要求老师的id不能重复
- 删除老师: 通过键盘录入要删除老师的id值,将该老师从数组中移除,如果录入的id在数组中不存在,需要重新录入
- 修改老师: 通过键盘录入要修改老师的id值和修改后的老师信息

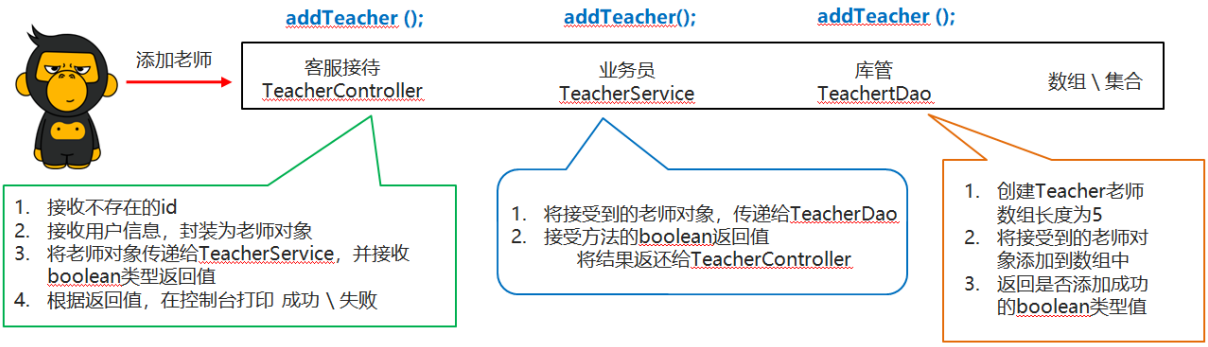
  将数组中该老师的信息修改,如果录入的id在数组中不存在,需要重新录入
- 查询老师: 将数组中存储的所有老师的信息输出到控制台

4.3.2实现步骤

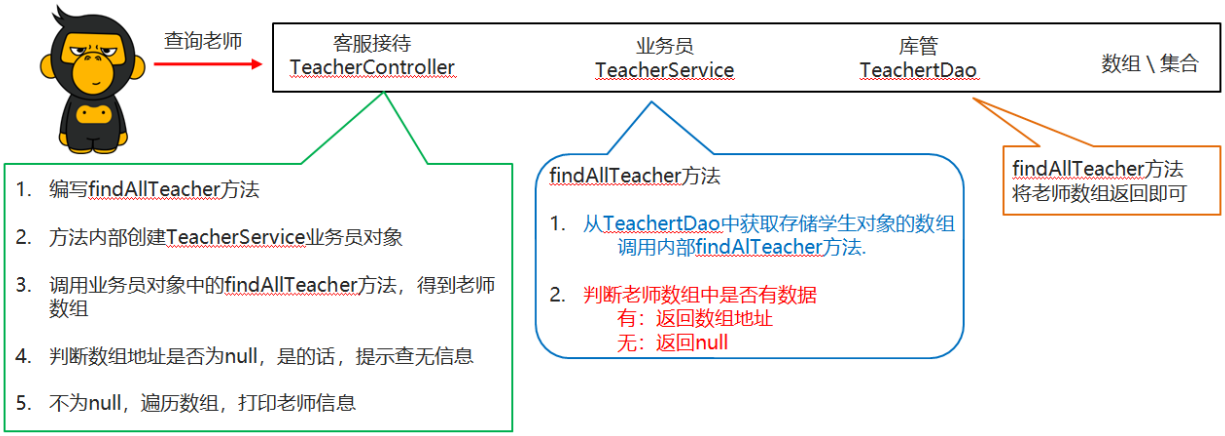
- 环境搭建实现步骤

包	存储的类	作用
com.itheima.edu.info.manager.domain	Student.java Teacher.java	封装学生信息 封装老师信息
com.itheima.edu.info.manager.dao	StudentDao.java TeacherDao.java	访问存储数据的数组,进行赠删改查（库管）
com.itheima.edu.info.manager.service	StudentService.java TeacherService.java	业务的逻辑处理（业务员）
com.itheima.edu.info.manager.controller	StudentController.java TeacherController.java	和用户打交道（客服接待）
com.itheima.edu.info.manager.entry	InfoManagerEntry.java	程序的入口类,提供一个main方法

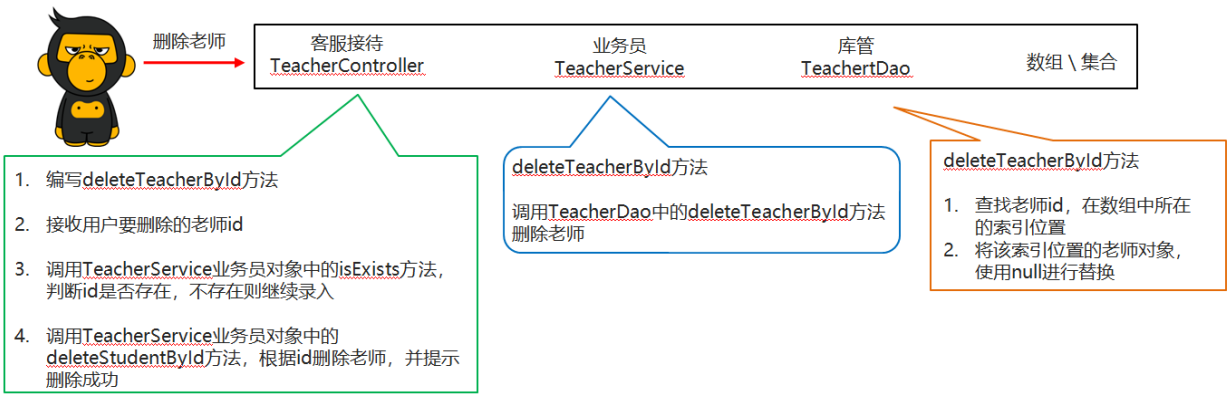
- 菜单搭建实现步骤
  1. 展示欢迎页面,用输出语句完成主界面的编写
  2. 获取用户的选择,用Scanner实现键盘录入数据
  3. 根据用户的选择执行对应的操作,用switch语句完成操作的选择
- 添加功能实现步骤



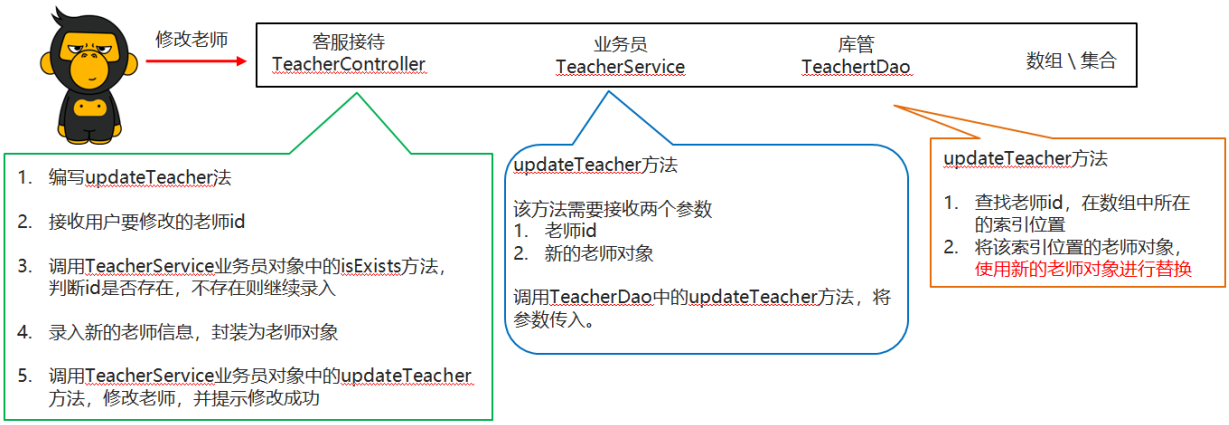
- 查询功能实现步骤



- 删除功能实现步骤



- 修改功能实现步骤



- 系统优化

- 把updateTeacher和deleteTeacherById中录入老师id代码抽取到一个方法(inputTeacherId)中
  该方法的主要作用就是录入老师的id,方法的返回值为String类型
- 把addTeacher和updateTeacher中录入老师信息的代码抽取到一个方法(inputTeacherInfo)中
  该方法的主要作用就是录入老师的信息,并封装为老师对象,方法的返回值为Teacher类型

4.3.3代码实现

老师类

```
public class Teacher extends Person{
     private String id;
    private String name;
    private String age;
    private String birthday;
    String address;

    public Teacher() {
    }

    public Teacher(String id, String name, String age, String birthday) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.birthday = birthday;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAge() {
        return age;
    }

    public void setAge(String age) {
        this.age = age;
    }

    public String getBirthday() {
        return birthday;
    }

    }
```

```
    public void setBirthday(String birthday) {
        this.birthday = birthday;
    }
}
```

TeacherController类

```
public class TeacherController {

    private Scanner sc = new Scanner(System.in);
    private TeacherService teacherService = new TeacherService();

    public void start() {

        teacherLoop:
        while (true) {
            System.out.println("--------欢迎来到 <老师> 管理系统--------");
            System.out.println("请输入您的选择: 1.添加老师  2.删除老师  3.修改老师  4.查看老师  5.退
出");
            String choice = sc.next();
            switch (choice) {
                case "1":
                    // System.out.println("添加老师");
                    addTeacher();
                    break;
                case "2":
                    // System.out.println("删除老师");
                    deleteTeacherById();
                    break;
                case "3":
                    // System.out.println("修改老师");
                    updateTeacher();
                    break;
                case "4":
                    // System.out.println("查看老师");
                    findAllTeacher();
                    break;
                case "5":
                    System.out.println("感谢您使用老师管理系统，再见!");
                    break teacherLoop;
                default:
                    System.out.println("您的输入有误，请重新输入");
                    break;
            }
        }

    }

    public void updateTeacher() {
        String id = inputTeacherId();

        Teacher newTeacher = inputTeacherInfo(id);
```

```
        // 调用业务员的修改方法
        teacherService.updateTeacher(id,newTeacher);
        System.out.println("修改成功");
    }

    public void deleteTeacherById() {

        String id = inputTeacherId();

        // 2. 调用业务员中的删除方法，根据id，删除老师
        teacherService.deleteTeacherById(id);

        // 3. 提示删除成功
        System.out.println("删除成功");


    }

    public void findAllTeacher() {
        // 1. 从业务员中，获取老师对象数组
        Teacher[] teachers = teacherService.findAllTeacher();

        // 2. 判断数组中是否有元素
        if (teachers == null) {
            System.out.println("查无信息，请添加后重试");
            return;
        }

        // 3. 遍历数组，取出元素，并打印在控制台
        System.out.println("学号\t\t姓名\t年龄\t生日");
        for (int i = 0; i < teachers.length; i++) {
            Teacher t = teachers[i];
            if (t != null) {
                System.out.println(t.getId() + "\t" + t.getName() + "\t" + t.getAge() + "\t\t" +
t.getBirthday());
            }
        }
    }

    public void addTeacher() {
        String id;
        while (true) {
            // 1. 接收不存在的老师id
            System.out.println("请输入老师id:");
            id = sc.next();
            // 2. 判断id是否存在
            boolean exists = teacherService.isExists(id);

            if (exists) {
                System.out.println("id已被占用，请重新输入:");
            } else {

                break;
```

```
            }
        }

        Teacher t = inputTeacherInfo(id);

        // 5. 将封装好的老师对象，传递给TeacherService继续完成添加操作
        boolean result = teacherService.addTeacher(t);

        if (result) {
            System.out.println("添加成功");
        } else {
            System.out.println("添加失败");
        }
    }

    // 录入老师id
    public String inputTeacherId(){
        String id;
        while(true){
            System.out.println("请输入id");
            id = sc.next();
            boolean exists = teacherService.isExists(id);
            if(!exists){
                System.out.println("您输入的id不存在，请重新输入:");
            }else{
                break;
            }
        }
        return id;
    }

    // 录入老师信息，封装为老师对象
    public Teacher inputTeacherInfo(String id){
        System.out.println("请输入老师姓名:");
        String name = sc.next();
        System.out.println("请输入老师年龄:");
        String age = sc.next();
        System.out.println("请输入老师生日:");
        String birthday = sc.next();

        Teacher t = new Teacher();
        t.setId(id);
        t.setName(name);
        t.setAge(age);
        t.setBirthday(birthday);

        return t;
    }
}
```

TeacherService类

```
public class TeacherService {
```

```
    private TeacherDao teacherDao = new TeacherDao();

    public boolean addTeacher(Teacher t) {
        return teacherDao.addTeacher(t);
    }

    public boolean isExists(String id) {
        // 1. 获取库管对象中的数组
        Teacher[] teachers = teacherDao.findAllTeacher();

        boolean exists = false;

        // 2. 遍历数组，取出每一个元素，进行判断
        for (int i = 0; i < teachers.length; i++) {
            Teacher teacher = teachers[i];
            if(teacher != null && teacher.getId().equals(id)){
                exists = true;
                break;
            }
        }

        return exists;
    }

    public Teacher[] findAllTeacher() {
        Teacher[] allTeacher = teacherDao.findAllTeacher();

        boolean flag = false;

        for (int i = 0; i < allTeacher.length; i++) {
            Teacher t = allTeacher[i];
            if(t != null){
                flag = true;
                break;
            }
        }

        if(flag){
            return allTeacher;
        }else{
            return null;
        }

    }

    public void deleteTeacherById(String id) {
        teacherDao.deleteTeacherById(id);
    }

    public void updateTeacher(String id, Teacher newTeacher) {
        teacherDao.updateTeacher(id,newTeacher);

    }
```

```
}
```

TeacherDao类

```
public class TeacherDao {

    private static Teacher[] teachers = new Teacher[5];

    public boolean addTeacher(Teacher t) {
        int index = -1;
        for (int i = 0; i < teachers.length; i++) {
            Teacher teacher = teachers[i];
            if(teacher == null){
                index = i;
                break;
            }
        }

        if(index == -1){
            return false;
        }else{
            teachers[index] = t;
            return true;
        }

    }

    public Teacher[] findAllTeacher() {
        return teachers;
    }

    public void deleteTeacherById(String id) {
        // 1. 查询id在数组中的索引位置
        int index = getIndex(id);
        // 2. 将该索引位置的元素，使用null进行替换
        teachers[index] = null;
    }

    public int getIndex(String id){
        int index = -1;
        for (int i = 0; i < teachers.length; i++) {
            Teacher t = teachers[i];
            if(t != null && t.getId().equals(id)){
                index = i;
                break;
            }
        }

        return index;
    }

    public void updateTeacher(String id, Teacher newTeacher) {
        int index = getIndex(id);
```

```
            teachers[index] = newTeacher;
        }
    }
```

5.static关键字

5.1static关键字概述 (理解)

static 关键字是静态的意思,是Java中的一个修饰符,可以修饰成员方法,成员变量

5.2static修饰的特点 (记忆)

- 被类的所有对象共享

  是我们判断是否使用静态关键字的条件
- 随着类的加载而加载，优先于对象存在

  对象需要类被加载后，才能创建
- 可以通过类名调用

  也可以通过对象名调用

5.3static关键字注意事项 (理解)

- 静态方法只能访问静态的成员
- 非静态方法可以访问静态的成员，也可以访问非静态的成员
- 静态方法中是没有this关键字