

 未标题-1

**《Java编程基础课程设计》课程设计报告**

**题 目 ：** 研究生初试录取程序

**专 业 班 级 ：** 软件1702

**学 生 姓 名 ：** 刘文博

**学 号 ：** 201716040224

**指 导 教 师 ：** 赵玉娟

**课程设计时间：** 2019.6.3—2019.6.14

**软件工程 专业Java编程基础课程设计任务书**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学生姓名** | **刘文博** | **专业班级** | | 软件1702 | **学号** | | **201716040224** |
| **题 目** | 研究生初试录取程序 | | | | | | |
| **课题性质** | 工程设计 | | **课题来源** | | | 自拟课题 | |
| **指导教师** | **赵玉娟** | | **同组姓名** | | | 无 | |
| **主要内容** | 设计完成一个研究生初试录取程序。使用Java语言，与存储在Hsqldb或mysql等数据库中的数据进行交互，恰当地使用集合对数据相应的操作，通过控制台或图形界面拟实现以下功能：  （1）编程统计初试合格的人数，并按总分由高到低的顺序输出合格考生的信息。  （2）程序运行时从原始数据文件中读取以下信息：考生姓名，准考证号，报考专业，是否应届生，4门课程（政治、数学、外语、专业基础课）成绩；  （3）输入录取的总分要求，各课程的最低分数要求。  （4）输出过线考生的姓名、准考证号、报考专业、是否应届生、4门课程（政治、数学、外语、专业基础课）成绩及总分，并保存在另一个文件中。 | | | | | | |
| **任务要求** | 能够将课本上的理论知识和实际问题有机的结合起来，掌握分析和解决实际工程问题的方法和能力。能够按照设计题目要求进行方案设计和实现；在设计实现中能够体现良好的身心素质和团队；在答辩过程中要求具有良好的表达能力，专业的描述方法，并能进行良好的沟通和交流，撰写课程设计报告，包含需求分析、概要设计、运行环境、开发工具、详细设计、调试分析、测试结果等内容，并对关键部分给出UML图说明，文字、段落和图表格式规范。 | | | | | | |
| **参考文献** | [1]《Java8程序设计及实验》. 刘继承等. 清华大学出版社，2018.  [2] Java从入门到精通（第5版）.[明日科技](https://book.jd.com/writer/%E6%98%8E%E6%97%A5%E7%A7%91%E6%8A%80_1.html). 清华大学出版社，2019  [3]《Java核心技术卷I:基础知识》(第10版)(英文影印版)，(美)Cay S. Horstmann、Gary Cornell，人民邮电出版社，2016  [4] 《Core Java Volume II: Advanced Features》（10th Edition）. Cay S. Horstmann, Gary Cornell. Prentice Hall, 2016. | | | | | | |
| **审查意见** | **教研室主任签字： 2019年 5月 31日** | | | | | | |

1 需求分析

研究生考试课程为4门，其中数学、外语、政治为统一命题，而专业基础课则根据不同的专业由招生学校自行命题。国家对初试录取分数有总分要求（如某一年要求四门课总分应达到310分），另外还有对每门课的最低分数要求（如总分为100的试卷最低应达到40分，总分为150的试卷最低应达到65分）。

基本要求：

（1）编程统计初试合格的人数，并按总分由高到低的顺序输出合格考生的信息。

（2）程序运行时从原始数据文件中读取以下信息：考生姓名，准考证号，报考专业，是否应届生，4门课程（政治、数学、外语、专业基础课）成绩；

（3）输入录取的总分要求，各课程的最低分数要求。

（4）输出过线考生的姓名、准考证号、报考专业、是否应届生、4门课程（政治、数学、外语、专业基础课）成绩及总分，并保存在另一个文件中。

根据题目要求，该项目应该有学生类，教师类，分数类等，为了实现面相对象的继承特性，可以将学生与老师继承于Person这一父类，展现继承特性。本项目基本要求需要实现对数据的筛选，对数据的排序，对文件的操作，作为扩展，也应当使用上数据库的技术，从数据库中读取信息，并且可以将修改后的信息同步到数据库中，也可以进行拓展，对不同的数据，例如不同的成绩，分别进行排序，可以让老师更好地分析录取的学生情况。

2 概要设计

2.1基础类的定义

Person类：集成最基本的人的要素，姓名，年龄，性别。

Score类：包含几门课程的成绩以及分数线。

Student类：继承于person类，并且添加了自己的独特要素，学生ID，是否应届，报考专业，并且与score类是组合关系。

Teacher类：继承于person类，并且添加了独特元素教师号，登录密码，以及研究方向。

2.2数据的保存

对于初始数据，从数据库中读取，用ArrayList来保存数据，再根据用户的选择进行筛选排序等操作。

2.3数据的操作

对于排序，选择TreeSet进行排序，重写Comparable接口实现。

3 运行环境

操作系统：Windows 10 8G内存

4 开发工具和编程语言

编程语言：JAVA

IDE：Eclipse

5 详细设计

Person类：

**public** **class** Person {

**private** String name;

**private** **int** age;

**private** String sex;

**public** Person() {}

**public** Person(String name, **int** age, String sex) {

**super**();

**this**.name = name;

**this**.age = age;

**this**.sex = sex;

}

**public** Person(String name, String sex) {

**super**();

**this**.name = name;

**this**.sex = sex;

}

}

Student类：

**public** **class** Student **extends** Person {

**private** String ID;

**private** Score score ;

**private** String current;

**private** String applyingmajor;

**public** Student(String name, **int** age, String sex, String iD, Score score, String current,String applyingmajor) {

**super**(name, age, sex);

ID = iD;

**this**.score = score;

**this**.current = current;

**this**.applyingmajor = applyingmajor;

}

**public** Student(String iD, Score score, String current) {

**super**();

ID = iD;

**this**.score = score;

**this**.current = current;

}

**public** Student() {

**super**();

}

}

Teacher类：

**public** **class** Teacher **extends** Person {

**private** String course;

**private** String id ;

**private** String pwd;

**public** Teacher(String name, String sex, String course, String id, String pwd) {

**super**(name, sex);

**this**.course = course;

**this**.id = id;

**this**.pwd = pwd;

}

**public** Teacher() {

**super**();

}

}

对于数据的读取，写一个工具类来实现：DbRead

**public** **class** DbRead {

**static** ArrayList<Student> *stuarr* = **new** ArrayList<Student>();

**static** ArrayList<Teacher> *teaarr* = **new** ArrayList<Teacher>();

**public** **static** ArrayList<Student> readstuinfo() {

Connection conn = JDBCutil.*getMysqlConn*();

PreparedStatement pst = **null**;

ResultSet rs = **null**;

**try** {

String sql = "select \* from stu\_info where 1=1";

pst = conn.prepareStatement(sql);

rs = pst.executeQuery();

**while**(rs.next()) {

Score score = **new** Score(rs.getInt(5), rs.getInt(6), rs.getInt(7), rs.getInt(8));

Student stu = **new** Student(rs.getString(2), rs.getInt(4), rs.getString(3), rs.getString(1), score, rs.getString(10),rs.getString(11));

*stuarr*.add(stu);

}

**return** *stuarr*;

} **catch** (SQLException e) {

e.printStackTrace();

**return** **null**;

}**finally** {

JDBCutil.*close*(rs, conn);

JDBCutil.*close*(pst);

}

}

**public** **static** ArrayList<Teacher> readteacherinfo(){

Connection conn = JDBCutil.*getMysqlConn*();

PreparedStatement pst = **null**;

ResultSet rs = **null**;

**try** {

String sql = "select \* from teacher\_info where 1=1";

pst = conn.prepareStatement(sql);

rs = pst.executeQuery();

**while**(rs.next()) {

Teacher tea = **new** Teacher(rs.getString(2), rs.getString(3), rs.getString(4), rs.getString(1), rs.getString(5));

*teaarr*.add(tea);

}

**return** *teaarr*;

} **catch** (SQLException e) {

e.printStackTrace();

**return** **null**;

}**finally** {

JDBCutil.*close*(rs, conn);

JDBCutil.*close*(pst);

}

}

}

对于数据的操作，同样适用一个工具类来实现：DataKit

**public** **class** DataKit {

**static** Properties *pros* = **null**;

**static** Score *score* = **null**;

**static** {

*pros* = **new** Properties();

**try** {

*pros*.load(Thread.*currentThread*().getContextClassLoader().getResourceAsStream("scoreline\_cofg.properties"));

*score* = **new** Score(Integer.*parseInt*(*pros*.getProperty("mathscoresline")),Integer.*parseInt*(*pros*.getProperty("politicsscoresline")),

Integer.*parseInt*(*pros*.getProperty("foreignlanguagescoresline")),Integer.*parseInt*(*pros*.getProperty("professionalbasiccoursescoresline")),

Integer.*parseInt*(*pros*.getProperty("totalscoresline")));

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **static** **void** changescoreline(**int** mathscoresline, **int** politicsscoresline, **int** foreignlanguagescoresline,

**int** professionalbasiccoursescoresline, **int** totalscoresline) {

**if**(mathscoresline>0)

*score*.setMathscoresline(mathscoresline);

**if**(mathscoresline>0)

*score*.setPoliticsscoresline(politicsscoresline);

**if**(mathscoresline>0)

*score*.setForeignlanguagescoresline(foreignlanguagescoresline);

**if**(mathscoresline>0)

*score*.setProfessionalbasiccoursescoresline(professionalbasiccoursescoresline);

**if**(mathscoresline>0)

*score*.setTotalscoresline(totalscoresline);

}

**public** **static** Set<Student> selectstu(ArrayList<Student> stuarr,String type) {

Set<Student> stuset = **new** TreeSet<Student>(MyComparator.*getcomparator*(type));

**for** (**int** i = 0; i < stuarr.size(); i++) {

**if**(stuarr.get(i).getScore().getTotalscores() >= *score*.getTotalscoresline()

&& stuarr.get(i).getScore().getMathscores() >= *score*.getMathscoresline()

&& stuarr.get(i).getScore().getPoliticsscores() >= *score*.getPoliticsscoresline()

&& stuarr.get(i).getScore().getForeignlanguagescores() >= *score*.getForeignlanguagescoresline()

&& stuarr.get(i).getScore().getProfessionalbasiccoursescores() >= *score*.getProfessionalbasiccoursescoresline()

) {

stuset.add(stuarr.get(i));

}

}

**return** stuset;

}

**public** **static** Set<Student> selectallstu(ArrayList<Student> stuarr) {

Set<Student> stuset = **new** TreeSet<Student>(MyComparator.*getcomparator*("all"));

**for** (**int** i = 0; i < stuarr.size(); i++) {

stuset.add(stuarr.get(i));

}

**return** stuset;

}

}

手动实现比较接口，用来满足自己对各种分数进行排序的需求：MyComparator

**public** **class** MyComparator {

**public** **static** Comparator<Student> getcomparator(String type){

**if**(type.equals("math")) {

Comparator<Student> com = **new** Comparator<Student>() {

@Override

**public** **int** compare(Student o1, Student o2) {

**int** num = o1.getScore().getMathscores() >= o2.getScore().getMathscores() ? -1:1;

**return** num;

}

};

**return** com;

}**else** **if**(type.equals("politics")) {

Comparator<Student> com = **new** Comparator<Student>() {

@Override

**public** **int** compare(Student o1, Student o2) {

**int** num = o1.getScore().getPoliticsscores() >= o2.getScore().getPoliticsscores() ? -1:1;

**return** num;

}

};

**return** com;

}**else** **if**(type.equals("foreignlanguage")) {

Comparator<Student> com = **new** Comparator<Student>() {

@Override

**public** **int** compare(Student o1, Student o2) {

**int** num = o1.getScore().getForeignlanguagescores() >= o2.getScore().getForeignlanguagescores() ? -1:1;

**return** num;

}

};

**return** com;

}**else** **if**(type.equals("professionalbasiccourses")) {

Comparator<Student> com = **new** Comparator<Student>() {

@Override

**public** **int** compare(Student o1, Student o2) {

**int** num = o1.getScore().getProfessionalbasiccoursescores() >= o2.getScore().getProfessionalbasiccoursescores() ? -1:1;

**return** num;

}

};

**return** com;

}**else** **if**(type.equals("all")) {

Comparator<Student> com = **new** Comparator<Student>() {

@Override

**public** **int** compare(Student o1, Student o2) {

**int** num = o1.getID().compareTo(o2.getID()) ;

**return** num;

}

};

**return** com;

}**else** {

Comparator<Student> com = **new** Comparator<Student>() {

@Override

**public** **int** compare(Student o1, Student o2) {

**int** num = o1.getScore().getTotalscores() >= o2.getScore().getTotalscores() ? -1:1;

**return** num;

}

};

**return** com;

}

}

**public** **static** Comparator<Student> getcomparator(){

Comparator<Student> com = **new** Comparator<Student>() {

@Override

**public** **int** compare(Student o1, Student o2) {

**int** num = o1.getScore().getTotalscores() >= o2.getScore().getTotalscores() ? -1:1;

**return** num;

}

};

**return** com;

}

}

对于数据的筛选，通过选择函数来实现，通过TreeSet容器，实现数据的有序化：

**public** **static** Set<Student> selectstu(ArrayList<Student> stuarr,String type) {

Set<Student> stuset = **new** TreeSet<Student>(MyComparator.*getcomparator*(type));

**for** (**int** i = 0; i < stuarr.size(); i++) {

**if**(stuarr.get(i).getScore().getTotalscores() >= *score*.getTotalscoresline()

&& stuarr.get(i).getScore().getMathscores() >= *score*.getMathscoresline()

&& stuarr.get(i).getScore().getPoliticsscores() >= *score*.getPoliticsscoresline()

&& stuarr.get(i).getScore().getForeignlanguagescores() >= *score*.getForeignlanguagescoresline()

&& stuarr.get(i).getScore().getProfessionalbasiccoursescores() >= *score*.getProfessionalbasiccoursescoresline()

) {

stuset.add(stuarr.get(i));

}

}

**return** stuset;

}

6 调试分析

在通过实现重写Comparator函数的时候，会出现需要排序的哪一种分数的学生，只会被插入一个，不会有重复的，这显然不符合实际情况，经过调试排查，发现Comparator函数的返回值为0的时候，会出现没有重复元素插入的情况，所以经过改进，取消了返回值为0这种情况，结果显示正常。

**public** **static** Comparator<Student> getcomparator(String type){

**if**(type.equals("math")) {

Comparator<Student> com = **new** Comparator<Student>() {

@Override

**public** **int** compare(Student o1, Student o2) {

**int** num = o1.getScore().getMathscores() >= o2.getScore().getMathscores() ? -1:1;

**return** num;

}

};

**return** com;

}

在图形界面操作的时候，当上层界面使用完毕点击关闭的时候，下层的工作界面也会跟着关闭，导致程序的异常退出，在搜索后知道了可以让上层界面自行销毁，从而不影响下层界面的正常继续使用dispose函数可以销毁当前的Frame。

7 测试结果

初始化数据库：

图1 学生信息

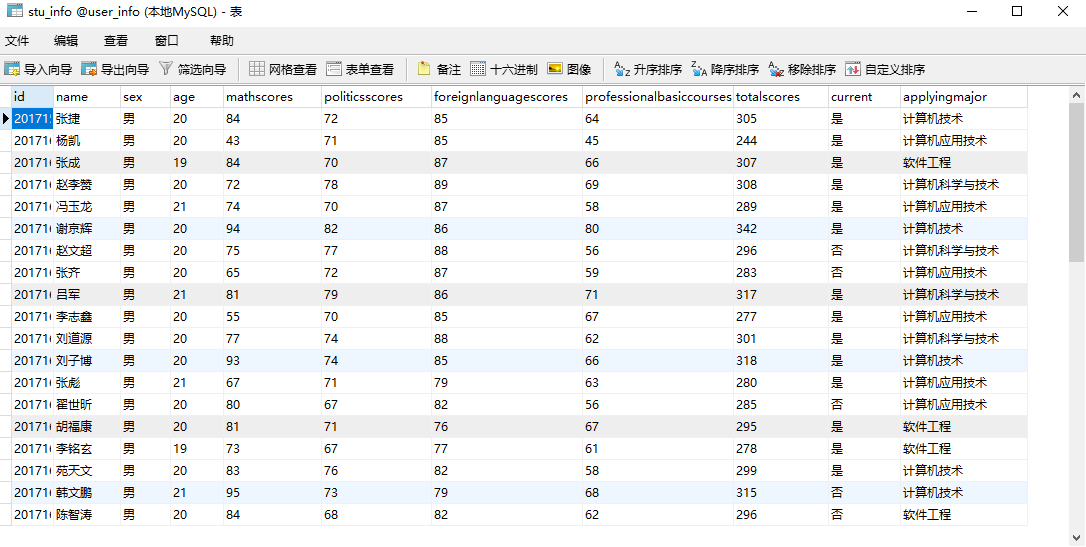


图2 教师信息

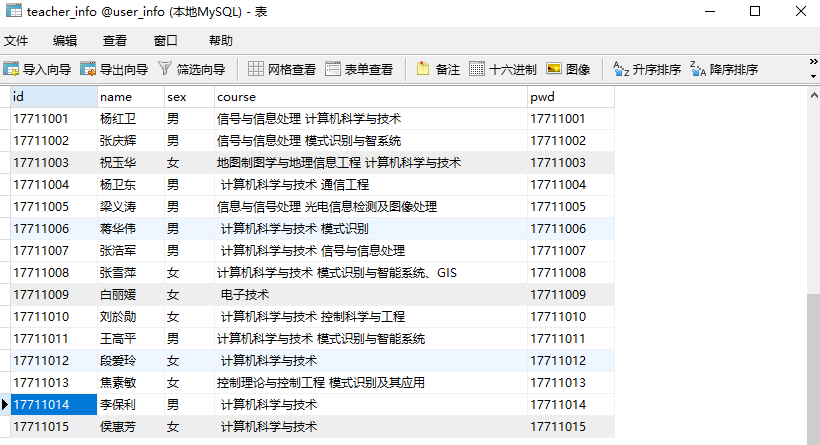


图3 教师登录

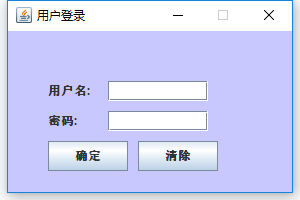


图4 登录失败



图5 登录成功



图6 操作界面

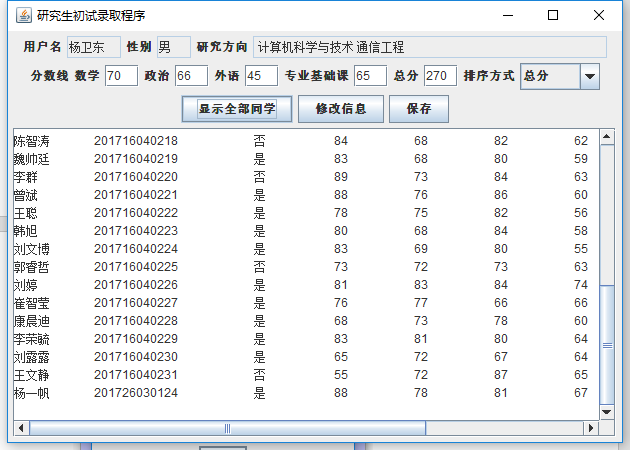


图7 设置成绩

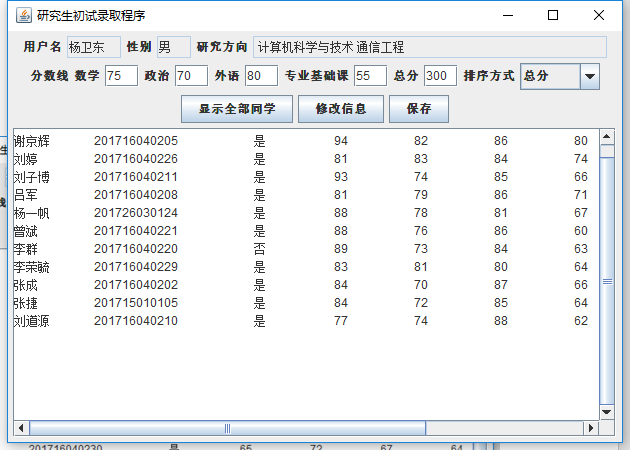


图8 按数学成绩排序

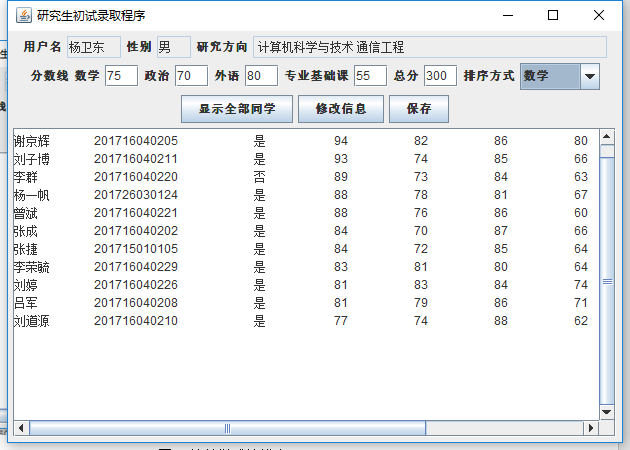


图9 按总成绩排序



图10 修改信息1

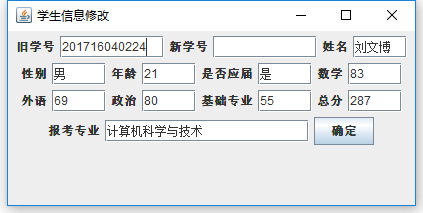


图11 修改信息2

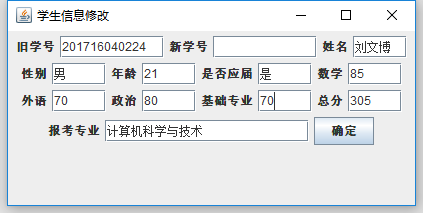


图12 修改信息3

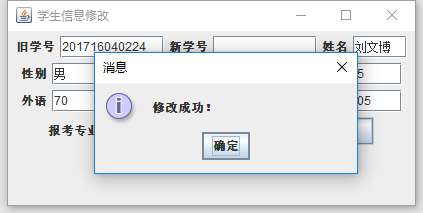


图13 修改信息4



图14 保存信息1

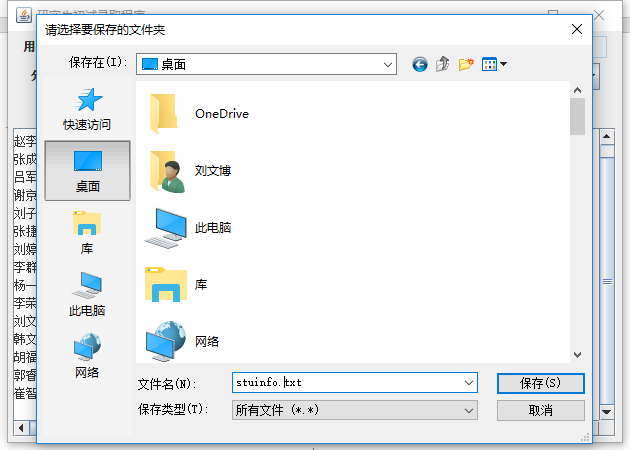
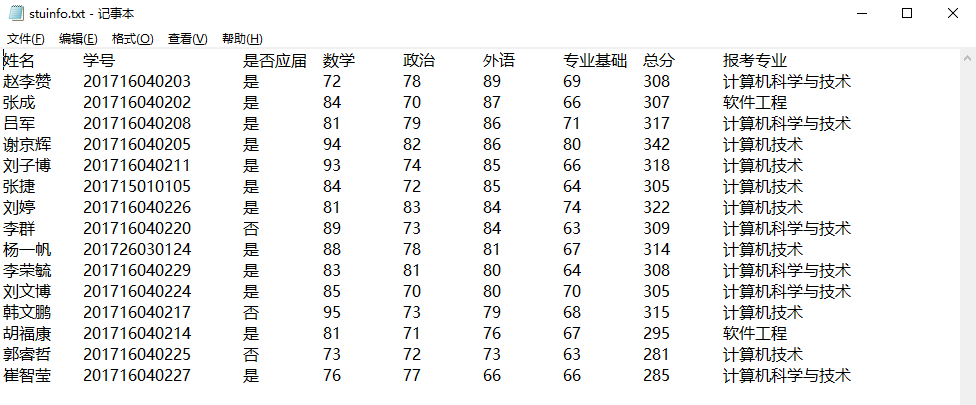


图15 保存信息2



参考文献

在“课程设计报告”的最后应附上所参考的相关文献，参考文献数目〉=5。

参考文献格式如下：([1][2][3]书籍 [4] 文章例)

1. [美]Paul Deitel, Harvey Deitel著, 张引等译. C++大学教程（第九版）[M].北京: 电子工业出版社, 2016.
2. 郭炜. 新标准C++程序设计 [M]. 北京: 高等教育出版社, 2016.
3. 谭浩强. C++面向对象程序设计（第2版）[M]. 北京: 清华大学出版社, 2014.
4. 宣华, 付小龙, 刘玉玲,等. 学生成绩管理系统的研究与创新[J]. 实验技术与管理, 2011, 28(2):101-103.

**心得体会**

使用泛型容器储存Javabean对象可以很方便地管理大小，减少了内存的开销，并可以方便的调用，但也要注意访问不要越界。

图形界面的监视器的添加，不同的组件也有不同的监视器，需要区别对待，不要混淆。

SQL语句中，有些地方是需要带有空格的，没有空格会报错，错误还不好找，容易浪费很多不必要的时间。字符串类型还需要使用单引号引起来