**哈尔滨工业大学（威海）**

**Java语言实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称 | Java语言 | 课程编号 | SE33601 |
| 实验名称 | 实验4：异常处理及数据访问框架设计与实现 | | |
| 实验类型 | 验证与编程 | 实验学时 | 2学时 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 班号 | 21911103 | 学号 | 2191110314 | 姓名 | 刘浩然 |
| 开始 | 2020.10.30 | 截止 | 2020.11.06 | 成绩 |  |
| 批语 |  | | | | |

目录

[1 实验目的 1](#_Toc55551248)

[2 实验说明 1](#_Toc55551249)

[3 实验步骤 2](#_Toc55551250)

[步骤1： 用StarUML工具设计类图 2](#_Toc55551251)

[步骤2： 编写文本文件操作相关的类代码 2](#_Toc55551252)

[4 测试及结果 7](#_Toc55551253)

[4.1 能够增加一条数据的测试代码及证明截图 7](#_Toc55551254)

[4.2 学号唯一性的测试代码及证明截图 8](#_Toc55551255)

[4.3 手机号唯一性的测试代码及证明截图 8](#_Toc55551256)

[4.4 能够检索出所有学生的测试代码及证明截图 9](#_Toc55551257)

[4.5 能够根据学号检索的测试代码及证明截图 9](#_Toc55551258)

[4.6 能够根据姓名检索的测试代码及证明截图 9](#_Toc55551259)

[4.7 能够根据学号对学生进行更新的测试代码及证明截图 10](#_Toc55551260)

[4.8 能够根据学号删除某行信息的测试代码及证明截图 10](#_Toc55551261)

# 1 实验目的

本次实验有三个主要目的：

1. 掌握文件读写的基本流程并进行异常处理；
2. 掌握通过查询API方式，实现文件的读写；
3. 在理解实验1-3的基础上，综合应用面向对象相关知识，设计完成不同存储格式数据的增加(Create)、检索(Retrieve)、更新(Update)和删除(Delete)操作。

# 2 实验说明

某实验室学生(student)名单形式如下：

序号 姓名 性别 学号 手机号 备注

01 黄\*\* 男 150810124 15634323900 T4

02 纪\*\* 男 150811033 15634323901 T4

03 付\*\* 男 141080320 15634323902

04 梁\*\* 男 141080332 15634323904

05 董\*\* 男 140450228 15634323905

06 崔\*\* 男 170810202 15634323912

07 郭\*飞 男 160421116 15634323913

08 郭\*志 男 170910209 15634323921

09 宋\*用 男 171090221 15634323922

10 葛\*\* 女 140409105 15634323925

为了操作方便，该实验室准备采用软件对以上学生名单进行管理。对于以上内容的存储，除了纯文本格式文件存储，还准备采用Mysql数据库表存储方式和json格式进行存储。换言之，以上格式和内容需要存储成文件文件、数据库表和json三种格式，以方便不同的客户端调用其操作。

为方便设计类图和编程，给出本实验相关的中英文对照，如表2-1所示：

表2-1 本实验相关的名词中英文对照

|  |  |  |  |
| --- | --- | --- | --- |
| No | 中文名称 | 对应英文 | 备注 |
| 01 | 序号 | no | 唯一 |
| 02 | 姓名 | name |  |
| 03 | 性别 | gender |  |
| 04 | 学号 | student id | 唯一 |
| 05 | 手机号 | mobile phone number | 唯一 |
| 06 | 备注 | memo |  |
| 07 | 文本文件 | text file |  |
| 08 | 数据库文件 | database file |  |
| 09 | Json文件 | json file |  |

备注：未在本表列出，但在设计和编码过程中需要的标识符，请按照Java命名规范自行组织。

# 3 实验步骤

基于已完成的实验1—实验3的设计思路和面向对象相关知识，按照以下步骤完成本实验。

## 步骤1： 用StarUML工具设计类图

要求：绘制数据存储的相关业务类的类图，要求能够实现三种存储方式的各项操作，并且要求设计的类图必须满足开闭原则，不必绘制测试类。具体操作如下：

* 能够增加一条数据。
  + 注意序号（no）可以通过int型自增完成，但需要唯一。
  + 学号（student id）和手机号（mobile phone number）需要唯一，不能重复，因此需要进行唯一性检查。
* 能够检索数据
  + 能够检索出所有学生（建议用：findAll作为方法名）
  + 能够根据学号检索（建议用findByStudentId作为方法名）
  + 能够根据姓名检索（建议用findByStudentName作为方法名）
* 能够更新数据

能够根据学号对学生进行更新（建议采用update作为方法名）

* 能够删除数据

能够根据学号删除某行信息（建议采用delete作为方法名）

**请绘制整体类图（不必绘制测试类）：**

|  |
| --- |
|  |

## 步骤2： 编写文本文件操作相关的类代码

* 根据步骤1设计的类图，实现文本文件相关的代码，如果有超类或者接口，也需要实现并粘贴到代码表格。在代码中如涉及异常，请在相关方法中解决异常，不得抛出到调用类。

**Store类的代码：**

|  |
| --- |
| public interface Store {  public String create(Student[] stu, Student createStu, int cnt);  public String findAll();  public String findByStudentId(Student[] stu, String idToFind, int cnt);  public String findByStudentName(Student[] stu, String nameToFind, int cnt);  public String update(Student[] stu, Student updateStu, int cnt);  public String delete(Student[] stu, Student deleteStu, int cnt);  } |

**Text类的代码：**

|  |
| --- |
| import java.io.\*;  public class Text implements Store {  @Override  public String create(Student[] stu, Student createStu, int cnt) {  for (int i = 0; i < cnt; i++) {  if (stu[i].getStudentId().equals(createStu.getStudentId()))  return "Student Id already exists!";  if (stu[i].getMobilePhoneNumber().equals(createStu.getMobilePhoneNumber()))  return "Phone number already exists!";  }  // begin to append  BufferedWriter bw = null;  stu[cnt].setNo(createStu.getNo());  stu[cnt].setName(createStu.getName());  stu[cnt].setGender(createStu.getGender());  stu[cnt].setStudentId(createStu.getStudentId());  stu[cnt].setMobilePhoneNumber(createStu.getMobilePhoneNumber());  stu[cnt].setMemo(createStu.getMemo());  try {  bw = new BufferedWriter(  new OutputStreamWriter(new FileOutputStream("/Users/raptazure/experiment4/src/student.txt", true)));  } catch (FileNotFoundException e) {  e.printStackTrace();  }  try {  bw.newLine();  bw.write(  String.format("%02d", createStu.getNo()) + '\t' + createStu.getName() + '\t' + createStu.getGender() + '\t'  + createStu.getStudentId() + '\t' + createStu.getMobilePhoneNumber() + '\t' + createStu.getMemo() + '\t');  } catch (IOException e) {  e.printStackTrace();  } finally {  try {  bw.close();  } catch (IOException e) {  e.printStackTrace();  }  }  return "Create: success";  }  @Override  public String findAll() {  BufferedReader br = null;  try {  br = new BufferedReader(new FileReader("/Users/raptazure/experiment4/src/student.txt"));  String line;  while ((line = br.readLine()) != null)  System.out.println(line);  } catch (IOException e) {  } finally {  try {  br.close();  } catch (IOException e) {  e.printStackTrace();  }  }  return "Search: done";  }  @Override  public String findByStudentId(Student[] stu, String findStudentId, int cnt) {  for (int i = 0; i < cnt; i++) {  if (stu[i].getStudentId().equals(findStudentId)) {  System.out.print(String.format("%02d", stu[i].getNo()));  System.out.print('\t' + stu[i].getName());  System.out.print('\t' + stu[i].getGender());  System.out.print('\t' + stu[i].getStudentId());  System.out.print('\t' + stu[i].getMobilePhoneNumber());  System.out.print('\t' + stu[i].getMemo());  System.out.print(" ");  System.out.println('\r');  return "success";  }  }  return "Student not found :(";  }  @Override  public String findByStudentName(Student[] stu, String findStudentName, int cnt) {  for (int i = 0; i < cnt; i++) {  if (stu[i].getName().equals(findStudentName)) {  System.out.print(String.format("%02d", stu[i].getNo()));  System.out.print('\t' + stu[i].getName());  System.out.print('\t' + stu[i].getGender());  System.out.print('\t' + stu[i].getStudentId());  System.out.print('\t' + stu[i].getMobilePhoneNumber());  System.out.print('\t' + stu[i].getMemo());  System.out.print(" ");  System.out.println('\r');  return "success";  }  }  return "Student not found :(";  }  public void writeToFile(Student[] stu, int cnt) {  BufferedReader br = null;  BufferedWriter bw = null;  try {  br = new BufferedReader(new FileReader("/Users/raptazure/experiment4/src/student.txt"));  String line = br.readLine();  bw = new BufferedWriter(  new OutputStreamWriter(new FileOutputStream("/Users/raptazure/experiment4/src/student.txt", false)));  bw.write(line);  for (int i = 0; i < cnt; i++) {  if (!stu[i].getStudentId().equals(" ")) {  bw.newLine();  bw.write(String.format("%02d", stu[i].getNo()) + '\t' + stu[i].getName() + '\t' + stu[i].getGender() + '\t'  + stu[i].getStudentId() + '\t' + stu[i].getMobilePhoneNumber() + '\t' + stu[i].getMemo() + '\t');  }  }  } catch (IOException e) {  e.printStackTrace();  } finally {  try {  br.close();  bw.close();  } catch (IOException e) {  e.printStackTrace();  }  }  }  @Override  public String update(Student[] stu, Student updateStudent, int cnt) {  for (int i = 0; i < cnt; i++) {  if (stu[i].getStudentId().equals(updateStudent.getStudentId())) {  stu[i].setName(updateStudent.getName());  stu[i].setGender(updateStudent.getGender());  stu[i].setStudentId(updateStudent.getStudentId());  stu[i].setMobilePhoneNumber(updateStudent.getMobilePhoneNumber());  stu[i].setMemo(updateStudent.getMemo());  writeToFile(stu, cnt);  return "Update: success";  }  }  return "Student not found :(";  }  @Override  public String delete(Student[] stu, Student deleteStudent, int cnt) {  for (int i = 0; i < cnt; i++) {  if (stu[i].getStudentId().equals(deleteStudent.getStudentId())) {  stu[i].setName(" ");  stu[i].setGender(" ");  stu[i].setStudentId(" ");  stu[i].setMobilePhoneNumber(" ");  stu[i].setMemo(" ");  writeToFile(stu, cnt);  return "Delete: success";  }  }  return "Student not found :(";  }  } |

# 4 测试及结果

利用Junit编写测试方法，以测试文本文件操作类的各个操作，并分别将junit相关的测试代码和结果截图粘贴到对应表格，具体如下：

## 4.1 能够增加一条数据的测试代码及证明截图

* 测试代码（用Junit编写的测试方法）

|  |
| --- |
| @Test  public void test01\_CreateStudent() {  stuTest.setNo(cntTest + 1);  stuTest.setName("Coffee");  stuTest.setGender("Male");  stuTest.setStudentId("22222222");  stuTest.setMobilePhoneNumber("1008610");  stuTest.setMemo("w");  result = listTest.createInfo(stu, stuTest, cntTest);  System.out.println(result);  Assert.assertEquals("Create: success", result);  cntTest++;  } |

* 测试截图

|  |
| --- |
|  |

## 4.2 学号唯一性的测试代码及证明截图

* 测试代码（用Junit编写的测试方法）

|  |
| --- |
| @Test  public void test02\_OnlyOneStudentId() {  stuTest.setNo(cntTest + 1);  stuTest.setName("Jawa");  stuTest.setGender("Male");  stuTest.setStudentId("22222222");  stuTest.setMobilePhoneNumber("11012011");  stuTest.setMemo("w");  result = listTest.createInfo(stu, stuTest, cntTest);  System.out.println(result);  Assert.assertEquals("Student Id already exists!", result);  } |

* 测试截图

|  |
| --- |
|  |

## 4.3 手机号唯一性的测试代码及证明截图

* 测试代码（用Junit编写的测试方法）

|  |
| --- |
| @Test  public void test03\_OnlyOneMobilePhoneNumber() {  stuTest.setNo(cntTest + 1);  stuTest.setName("Doggy");  stuTest.setGender("Male");  stuTest.setStudentId("160410306");  stuTest.setMobilePhoneNumber("1008610");  stuTest.setMemo("w");  result = listTest.createInfo(stu, stuTest, cntTest);  System.out.println(result);  Assert.assertEquals("Phone number already exists!", result);  } |

* 测试截图

|  |
| --- |
|  |

## 4.4 能够检索出所有学生的测试代码及证明截图

* 测试代码（用Junit编写的测试方法）

|  |
| --- |
| @Test  public void test04\_FindAll() {  listTest.findAllInfo();  } |

* 测试截图

|  |
| --- |
|  |

## 4.5 能够根据学号检索的测试代码及证明截图

* 测试代码（用Junit编写的测试方法）

|  |
| --- |
| @Test  public void test05\_FindByStudentId() {  stuTest.setStudentId("22222222");  result = listTest.findByStudentIdInfo(stu, stuTest.getStudentId(), cntTest);  System.out.println(result);  Assert.assertEquals("success", result);  } |

* 测试截图

|  |
| --- |
|  |

## 4.6 能够根据姓名检索的测试代码及证明截图

* 测试代码（用Junit编写的测试方法）

|  |
| --- |
| @Test  public void test06\_FindByStudentName() {  stuTest.setName("Coffee");  result = listTest.findByStudentNameInfo(stu, stuTest.getName(), cntTest);  System.out.println(result);  Assert.assertEquals("success", result);  } |

* 测试截图

|  |
| --- |
|  |

## 4.7 能够根据学号对学生进行更新的测试代码及证明截图

* 测试代码（用Junit编写的测试方法）

|  |
| --- |
| @Test  public void test07\_Update() {  stuTest.setName("Coffee");  stuTest.setGender("Male");  stuTest.setStudentId("22222222");  stuTest.setMobilePhoneNumber("13562659875");  stuTest.setMemo("T7");  result = listTest.updateInfo(stu, stuTest, cntTest);  System.out.println(result);  Assert.assertEquals("Update: success", result);  } |

* 测试截图

|  |
| --- |
|  |

## 4.8 能够根据学号删除某行信息的测试代码及证明截图

* 测试代码（用Junit编写的测试方法）

|  |
| --- |
| @Test  public void test08\_Delete() {  stuTest.setStudentId("22222222");  result = listTest.deleteInfo(stu, stuTest, cntTest);  System.out.println(result);  Assert.assertEquals("Delete: success", result);  } |

* 测试截图

|  |
| --- |
|  |