**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 | 1911103 | 学号 | 2191110315 | 姓名 | 刘梓铭 |
| 开始 | 2020.10.30 | 截止 | 2020.11.06 | 成绩 |  |
| 批语 |  | | | | |

目录

[1 实验目的 1](#_Toc54947986)

[2 实验说明 1](#_Toc54947987)

[3 实验步骤 2](#_Toc54947988)

[步骤1： 用staruml设计业务类 2](#_Toc54947989)

[步骤2： 编写文本文件操作相关的类代码 3](#_Toc54947990)

[4 测试及结果 9](#_Toc54947991)

[4.1 能够增加一条数据的测试代码及证明截图 9](#_Toc54947992)

[4.2 学号唯一性的测试代码及证明截图 10](#_Toc54947993)

[4.3 手机号唯一性的测试代码及证明截图 10](#_Toc54947994)

[4.4 能够检索出所有学生的测试代码及证明截图 10](#_Toc54947995)

[4.5 能够根据学号检索的测试代码及证明截图 11](#_Toc54947996)

[4.6 能够根据姓名检索的测试代码及证明截图 11](#_Toc54947997)

[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 12](#_Toc54947998)

[4.8 能够根据学号删除某行信息的测试代码及证明截图 13](#_Toc54947999)

# 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）//姓名有可能不唯一
* 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。

**List类的代码：**

|  |
| --- |
| package studentList;  public class List {  public static int count = 0;  public Storage store;  public static Student[] Stu = new Student[100];  public void setStoreType(Storage s){  this.store = s;  }  } |

**Student类的代码：**

|  |
| --- |
| package studentList;  public class Student {  private int no;  private String name;  private String gender;  private String studentId;  private String mobilePhoneNumber;  private String memo;  public int getNo() {  return no;  }  public void setNo(int no) {  this.no = no;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getGender() {  return gender;  }  public void setGender(String gender) {  this.gender = gender;  }  public String getStudentId() {  return studentId;  }  public void setStudentId(String studentId) {  this.studentId = studentId;  }  public String getMobilePhoneNumber() {  return mobilePhoneNumber;  }  public void setMobilePhoneNumber(String mobilePhoneNumber) {  this.mobilePhoneNumber = mobilePhoneNumber;  }  public String getMemo() {  return memo;  }  public void setMemo(String memo) {  this.memo = memo;  }  } |

**Storage类的代码：**

|  |
| --- |
| package studentList;  import java.io.IOException;  public interface Storage {  public abstract void add() throws IOException;  public abstract void findAll() throws IOException;  public abstract void findByStudentId();  public abstract void findByStudentName();  public abstract void update() throws IOException;  public abstract void delete() throws IOException;  } |

**Text类的代码：**

|  |
| --- |
| package studentList;  import java.io.\*;  import java.util.NoSuchElementException;  import java.util.Scanner;  public class Text implements Storage {  @Override  public void add() throws IOException {  Student stu = new Student();  System.out.println("请依次输入学生姓名 性别 学号 电话号码 备注（选填）");  Scanner scanner\_ = new Scanner(System.in); //读取键盘输入  String line = scanner\_.nextLine();  Scanner scanner = new Scanner(line); //将读取的一行信息通过以下方式分解  stu.setNo(List.count + 1);  stu.setName(scanner.next());  stu.setGender(scanner.next());  String id = scanner.next();  String number = scanner.next();  for(int i = 0;i<List.count;i++){  if(List.Stu[i].getStudentId().equals(id)){  System.out.println("该学号已存在，请重新输入：");  Scanner scanner\_id = new Scanner(System.in);  id = scanner\_id.nextLine();  i = 0;  }  if(List.Stu[i].getMobilePhoneNumber().equals(number)){  System.out.println("该电话号码已存在，请重新输入：");  Scanner scanner\_number = new Scanner(System.in);  number = scanner\_number.nextLine();  i = 0;  }  }  stu.setStudentId(id);  stu.setMobilePhoneNumber(number);  try {  stu.setMemo(scanner.next()); //备注（可以不填）  } catch (NoSuchElementException e) //无内容错误，即没有备注  {  stu.setMemo(" ");  }  scanner.close();  List.Stu[List.count] = stu;  List.count++;  BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("data.txt", true)));  bw.newLine();  bw.write(String.format("%02d",stu.getNo())+'\t'+stu.getName()+'\t'+stu.getGender()+'\t'+stu.getStudentId()+'\t'+stu.getMobilePhoneNumber()+'\t'+stu.getMemo());  bw.close();  }  @Override  public void findAll() throws IOException {  BufferedReader br = null;  try {  br = new BufferedReader(new FileReader("data.txt"));  } catch (FileNotFoundException e) {  e.printStackTrace();  }  String line;  while((line = br.readLine()) != null) {  System.out.println(line);  }  br.close();  }  @Override  public void findByStudentId() {  int flag = 0;  System.out.println("请输入您要寻找的学生学号：");  Scanner scanner\_id = new Scanner(System.in);  String id\_ = scanner\_id.nextLine();  for(int i = 0;i<List.count;i++){  if(List.Stu[i].getStudentId().equals(id\_)) {  System.out.println("您所找的学生基本信息为：");  System.out.println(String.format("%02d",List.Stu[i].getNo()) + '\t' + List.Stu[i].getName() + '\t' + List.Stu[i].getGender() + '\t' + List.Stu[i].getStudentId() + '\t' + List.Stu[i].getMobilePhoneNumber() + '\t' + List.Stu[i].getMemo());  flag = 1;  }  }  if(flag==0)  System.out.println("未找到学号为"+id\_+"的学生");  }  @Override  public void findByStudentName() {  int flag = 0;  System.out.println("请输入您要寻找的学生姓名：");  Scanner scanner = new Scanner(System.in);  String name\_ = scanner.nextLine();  for(int i = 0;i<List.count;i++){  if(List.Stu[i].getName().equals(name\_)) {  if(flag == 0)  System.out.println("您所找的学生基本信息为：");  System.out.println(String.format("%02d",List.Stu[i].getNo()) + '\t' + List.Stu[i].getName() + '\t' + List.Stu[i].getGender() + '\t' + List.Stu[i].getStudentId() + '\t' + List.Stu[i].getMobilePhoneNumber() + '\t' + List.Stu[i].getMemo());  flag = 1;  }  }  if(flag==0)  System.out.println("未找到姓名为"+name\_+"的学生");  }  @Override  public void update() throws IOException {  int flag = 0;  System.out.println("请输入您要更新的学生的学号：");  Scanner scanner\_id = new Scanner(System.in);  String id\_ = scanner\_id.nextLine();  for(int i = 0;i<List.count;i++){  if(List.Stu[i].getStudentId().equals(id\_)) {  System.out.println("请依次输入更新后的学生姓名 性别 电话号码 备注（选填）");  Scanner scanner\_ = new Scanner(System.in); //读取键盘输入  String line = scanner\_.nextLine();  Scanner scanner = new Scanner(line); //将读取的一行信息通过以下方式分解  List.Stu[i].setName(scanner.next());  List.Stu[i].setGender(scanner.next());  List.Stu[i].setMobilePhoneNumber(scanner.next());  try {  List.Stu[i].setMemo(scanner.next()); //备注（可以不填）  } catch (NoSuchElementException e) //无内容错误，即没有备注  {  List.Stu[i].setMemo(" ");  }  scanner.close();  System.out.println("已将学号为"+"\_id"+"的学生信息更新为：");  System.out.println(String.format("%02d",List.Stu[i].getNo()) + '\t' + List.Stu[i].getName() + '\t' + List.Stu[i].getGender() + '\t' + List.Stu[i].getStudentId() + '\t' + List.Stu[i].getMobilePhoneNumber() + '\t' + List.Stu[i].getMemo());  BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("data.txt", false)));  for(int j = 0;j<List.count;j++){  if(List.Stu[j] != null){  bw.write(String.format("%02d",List.Stu[j].getNo())+'\t'+List.Stu[j].getName()+'\t'+List.Stu[j].getGender()+'\t'+List.Stu[j].getStudentId()+'\t'+List.Stu[j].getMobilePhoneNumber()+'\t'+List.Stu[j].getMemo());  if(j!=List.count-1)  bw.newLine();  }  }  flag = 1;  bw.close();  }  }  if(flag==0)  System.out.println("未找到学号为"+id\_+"的学生");  }  @Override  public void delete() throws IOException {  int flag = 0;  System.out.println("请输入您要删除的学生的学号：");  Scanner scanner\_id = new Scanner(System.in);  String id\_ = scanner\_id.nextLine();  for(int i = 0;i<List.count;i++){  if(List.Stu[i].getStudentId().equals(id\_)) {  System.out.println("已将该条信息删除：");  System.out.println(String.format("%02d",List.Stu[i].getNo()) + '\t' + List.Stu[i].getName() + '\t' + List.Stu[i].getGender() + '\t' + List.Stu[i].getStudentId() + '\t' + List.Stu[i].getMobilePhoneNumber() + '\t' + List.Stu[i].getMemo());  List.Stu[i] = null;  BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("data.txt", false)));  for(int j = 0;j<List.count;j++){  if(List.Stu[j] != null){  bw.write(String.format("%02d",List.Stu[j].getNo())+'\t'+List.Stu[j].getName()+'\t'+List.Stu[j].getGender()+'\t'+List.Stu[j].getStudentId()+'\t'+List.Stu[j].getMobilePhoneNumber()+'\t'+List.Stu[j].getMemo());  if(j!=List.count-2)  bw.newLine();  }  }  flag = 1;  bw.close();  }  }  if(flag==0)  System.out.println("未找到学号为"+id\_+"的学生");  }  } |

# 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 test\_add() throws IOException {  List a = new List();  a.setStoreType(new Text());  traverseList();  a.store.add();  } |

* 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 test\_add() throws IOException {  List a = new List();  a.setStoreType(new Text());  traverseList();  a.store.add();  } |

* 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 test\_add() throws IOException {  List a = new List();  a.setStoreType(new Text());  traverseList();  a.store.add();  } |

* 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 test\_findAll() throws IOException {  List a = new List();  a.setStoreType(new Text());  traverseList();  a.store.findAll();  } |

* 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 test\_findByStudentId() throws IOException {  List a = new List();  a.setStoreType(new Text());  traverseList();  a.store.findByStudentId();  } |

* 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 test\_findByStudentName() throws IOException {  List a = new List();  a.setStoreType(new Text());  traverseList();  a.store.findByStudentName();  } |

* 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 test\_update() throws IOException {  List a = new List();  a.setStoreType(new Text());  traverseList();  a.store.update();  } |

* 测试截图

|  |
| --- |
| Junit结果：    运行前：    运行后： |

## 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 test\_delet() throws IOException {  List a = new List();  a.setStoreType(new Text());  traverseList();  a.store.delete();  } |

* 测试截图

|  |
| --- |
| Junit结果：    运行前：    运行后： |