**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 | 2191110323 | 姓名 | 张凯文 |
| 开始 | 2020.10.30 | 截止 | 2020.11.06 | 成绩 |  |
| 批语 | 电子版材料也截止到2020.11.06日20:00 | | | | |

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 4](#_Toc54947991)

[4.1 能够增加一条数据的测试代码及证明截图 4](#_Toc54947992)

[4.2 学号唯一性的测试代码及证明截图 4](#_Toc54947993)

[4.3 手机号唯一性的测试代码及证明截图 5](#_Toc54947994)

[4.4 能够检索出所有学生的测试代码及证明截图 5](#_Toc54947995)

[4.5 能够根据学号检索的测试代码及证明截图 6](#_Toc54947996)

[4.6 能够根据姓名检索的测试代码及证明截图 6](#_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 7](#_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 7](#_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）
* 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。

**Data类的代码：**

|  |
| --- |
| public abstract class Data {  private int no;  private String gender;  private String name;  private String student\_id;  private String memo;  private String mobile\_phone\_number;  public abstract void Create();  public abstract void Update();  public abstract void Delete();  public abstract void findAll();  public abstract void findByStudentId();  public abstract void findByStudentName();  public String getGender() {  return gender;  }  public void setGender(String gender) {  this.gender = gender;  }  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 getStudent\_id() {  return student\_id;  }  public void setStudent\_id(String student\_id) {  this.student\_id = student\_id;  }  public String getMemo() {  return memo;  }  public void setMemo(String memo) {  this.memo = memo;  }  public String getMobile\_phone\_number() {  return mobile\_phone\_number;  }  public void setMobile\_phone\_number(String mobile\_phone\_number) {  this.mobile\_phone\_number = mobile\_phone\_number;  }  } |

**Text类的代码：**

|  |
| --- |
| import java.io.\*;  import java.util.Scanner;  public class Text extends Data {  public void Create() {  try {  int n;  String gen;  String nam;  String stu;  String mem;  String mob;  Scanner sc = new Scanner(System.in);  n = sc.nextInt();  nam = sc.next();  gen = sc.next();  stu = sc.next();  mob = sc.next();  mem = sc.next();  BufferedReader br = new BufferedReader(new FileReader("D:\\MyProjects\\Java\\JavaProgram\\Exp04\\Data.txt"));  String line;  if ((line = br.readLine()) == null) {  setNo(n);  setName(nam);  setGender(gen);  setStudent\_id(stu);  setMobile\_phone\_number(mob);  setMemo(mem);  } else {  while ((line = br.readLine()) != null) {  if (!(line.contains(stu)) && !(line.contains(mob))&&line.indexOf(n)==-1) {  setNo(n);  setName(nam);  setGender(gen);  setStudent\_id(stu);  setMobile\_phone\_number(mob);  setMemo(mem);  }  else{  System.out.println("唯一性测试不通过");  return;  }  }  try {  FileWriter fw = new FileWriter("D:\\MyProjects\\Java\\JavaProgram\\Exp04\\Data.txt", true);  BufferedWriter bw = new BufferedWriter(fw);  bw.append(getNo() + "\t" + getName() + "\t" + getGender() + "\t" + getStudent\_id() + "\t" + getMobile\_phone\_number() + "\t" + getMemo() + "\r\n");  bw.flush();  bw.close();  fw.close();  } catch (Exception e) {  System.out.println(e);  }  }catch (Exception e){  System.out.println(e);  }  }  public void Update() {  Delete();  Create();  findAll();  }  public void Delete() {  try{  System.out.println("请输入要删除对象学号");  Scanner sc = new Scanner(System.in);  String id = sc.next();  File inFile = new File("D:\\MyProjects\\Java\\JavaProgram\\Exp04\\Data.txt");  File outFile = new File("D:\\MyProjects\\Java\\JavaProgram\\Exp04\\Data\_new.txt");  BufferedReader br = null;  String readedLine;  BufferedWriter bw = null;  FileWriter fw = new FileWriter(outFile);  bw = new BufferedWriter(fw);  if (!outFile.exists()) {  outFile.createNewFile();  }  br = new BufferedReader(new FileReader(inFile));  int idx = 0;  while ((readedLine = br.readLine()) != null) {  if (readedLine.contains(id)) {  continue;  }  bw.write(readedLine + "\n");  if (idx++ == 100) {  bw.flush();  idx = 0;  }  }  inFile.delete();  outFile.renameTo(inFile);  bw.flush();  } catch (Exception e) {  e.printStackTrace();  } finally {  try {  if (br != null) {  br.close();  }  } catch (IOException e) {  e.printStackTrace();  }  try {  if (bw != null) {  bw.close();  }  } catch (IOException e) {  e.printStackTrace();  }  }  }  public void findAll() {  try {  System.out.print("序号" + "\t" + "姓名" + "\t" + "性别" + "\t" + "学号" + "\t" + "手机号" + "\t" + "备注" + "\n");  BufferedReader br = new BufferedReader(new FileReader("D:\\MyProjects\\Java\\JavaProgram\\Exp04\\Data.txt"));  String line;  while ((line = br.readLine()) != null) {  System.out.println(line);  }  br.close();  } catch (Exception e) {  System.out.println(e);  }  }  public void findByStudentName() {  String n;  int flag = 0;  System.out.println("请输入要查询学生姓名：");  Scanner sc = new Scanner(System.in);  n = sc.next();  try {  BufferedReader br = new BufferedReader(new FileReader("D:\\MyProjects\\Java\\JavaProgram\\Exp04\\Data.txt"));  String line;  while ((line = br.readLine()) != null)  if (line.contains(n)) {  System.out.println(line);  flag = 1;  }  if (flag == 0) {  System.out.println("未发现此名称学生");  }  br.close();  }catch (Exception e){  System.out.println(e);  }  }  public void findByStudentId(){  Scanner sc = new Scanner(System.in);  String id;  int flag = 0;  System.out.println("请输入要查询学生id：");  id = sc.next();  try {  BufferedReader br = new BufferedReader(new FileReader("D:\\MyProjects\\Java\\JavaProgram\\Exp04\\Data.txt"));  String line;  while ((line = br.readLine()) != null) {  if(line.contains(id)) {  System.out.println(line);  flag = 1;  }  }  if(flag==0)  {  System.out.println("未发现此名称学生");  }  br.close();  }catch (Exception e){  System.out.println(e);  }  }  } |

# 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 create() {  Text tx = new Text(11,"张三","男","2191110323","13485997985","菜鸡");  tx.Create();  } |

* 测试截图

|  |
| --- |
| 测试前：    测试图：    结果文件： |

## 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 create() {  Text tx = new Text(11,"张三","男","2191110323","13485997985","菜鸡");  //这次在Create中输入与文件中相同学号信息  tx.Create();  } |

* 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 create() {  Text tx = new Text(11,"张三","男","2191110323","13485997985","菜鸡");  //这次在Create中输入与文件中相同手机号信息  tx.Create();  } |

* 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）

|  |
| --- |
| public class TextTest {  @Test  public void findAll() {  Text tx = new Text();  tx.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）

|  |
| --- |
| public class TextTest {  @Test  public void findByStudentId() {  Text tx = new Text();  tx.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）

|  |
| --- |
| public class TextTest {  @Test  public void findByStudentName() {  Text tx = new Text();  tx.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）

|  |
| --- |
| public class TextTest {  @Test  public void update() {  Text tx = new Text();  tx.Update();  }  } |

* 测试截图

|  |
| --- |
| 更改前：    更改后： |

## 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）

|  |
| --- |
| public class TextTest {  @Test  public void delete() {  Text tx = new Text();  tx.Delete();  }  } |

* 测试截图

|  |
| --- |
| 删除前：    删除后： |