package project;  
  
import org.apache.poi.ss.usermodel.\*;  
import org.apache.poi.xssf.usermodel.XSSFRow;  
import org.apache.poi.xssf.usermodel.XSSFSheet;  
import org.apache.poi.xssf.usermodel.XSSFWorkbook;  
import java.io.\*;  
import java.text.NumberFormat;  
import java.util.\*;  
  
public class project1 {  
  
 *//两个数据源* private final List<ArrayList> dataSourceXlsx=new ArrayList();  
 private final List<ArrayList> dataSourceTxt=new ArrayList();  
  
  
 *//处理xlsx数据源* public void parseXlsx(String fileName) throws IOException {  
 *// 指定excel文件，创建缓存输入流* BufferedInputStream inputStream = new BufferedInputStream(new FileInputStream(fileName));  
  
 *// 直接传入输入流即可，此时excel就已经解析了* XSSFWorkbook workbook = new XSSFWorkbook(inputStream);  
  
 *// 选择要处理的sheet名称* XSSFSheet sheet = workbook.getSheetAt(0);  
  
 ArrayList arrayList = null;  
 int j;  
 *// 迭代遍历sheet剩余的每一行,除了第一行* for (int rowNum = 1; rowNum < sheet.getPhysicalNumberOfRows(); rowNum++) {  
  
 arrayList = new ArrayList();  
  
 XSSFRow row = sheet.getRow(rowNum);  
 j = 0;  
  
 *//这里有个小bug，就是数据集里存在空列时会导致读取中断，这里并未处理，只是把空列删除了而已* while (row.getCell(j) != null) {  
  
 switch (row.getCell(j).getCellType()) {  
 case *STRING*:  
 *// 注：有可能读取到的是空的格子。这里一开始忘记判断了，导致老是出错，NullPointerException* if (row.getCell(j) != null) {  
 arrayList.add(row.getCell(j++).getStringCellValue());  
 break;  
 }  
 case *NUMERIC*:  
 if (row.getCell(j) != null) {  
 NumberFormat nf = NumberFormat.*getInstance*();  
 *//去除小数点后的0，并转化为字符串* arrayList.add(nf.format(row.getCell(j++).getNumericCellValue()));  
 }  
 break;  
 }  
 if(j==14){  
 j++;  
 }  
 if ((row.getCell(j) == null && row.getCell(j + 1) != null)||(j==15&&row.getCell(j) == null)) {  
 arrayList.add("");  
 j++;  
 }  
 }  
 dataSourceXlsx.add(arrayList);  
 workbook.close();  
 inputStream.close();  
 }  
 }  
  
 *//处理txt数据源* public void parseTxt(String fileName) throws IOException {  
 FileReader reader = new FileReader(fileName);  
 BufferedReader br = new BufferedReader(reader);  
 String line;  
  
 while((line=br.readLine())!=null){  
 *//这里的正则容易出错，会将末尾的空字符串直接丢弃，加上-1限制就不会了* String []str=line.split(",",-1);  
 *//将string[]转化为List* List li=Arrays.*asList*(str);  
 *//两个List是不同的，需要再进行转化* ArrayList list=new ArrayList(li);  
 *//移除C10的空字符串* list.remove(14);  
 this.dataSourceTxt.add(list);  
 }  
 }  
  
 */\*\*\*  
 \* 代码出错会导致无法进入debug  
 \* 可通过打多个断点的方式来进行调式  
 \* 还有，判断String相等不能使用==，==比较的是地址，要用equal才是比较内容  
 \*/  
  
 /\*\*\*  
 \* 根据传递的参数来决定来对txt或xlsx  
 \* 根据 ID 或 Name 去重  
 \* 去重策略是删除第一个数据  
 \*/* public void delByNameOrId(String dataSource,String choice){  
  
 *//map用来记录重复的数据的下标* Map<Integer,ArrayList> map=new HashMap<>();  
 List<ArrayList> list=null;  
 int j=0;  
  
 if(choice.equals("Id")){  
 j=0;  
 }else{  
 j=1;  
 }  
  
 */\*\*\*  
 \* 对两个数据源要区别对待  
 \* xlsx的Id是double，Name是string  
 \* txt的Id和Name都是string  
 \*/* if(dataSource.equals("xlsx")){  
 list=dataSourceXlsx;  
 switch (choice){  
 case "Id":  
 *//遍历找出重复的数据的下标* for(int i=0;i<list.size()-1;i++){  
 String b= list.get(i).get(j).toString();  
 String c= list.get(i+1).get(j).toString();  
 if(b.equals(c)){  
 map.put(i,list.get(i));  
 }  
 }  
 break;  
 default:  
 *//遍历找出重复的数据的下标* for(int i=0;i<list.size()-1;i++){  
 if(list.get(i).get(j).equals(list.get(i+1).get(j))){  
 map.put(i,list.get(i));  
 }  
 }  
  
 }  
 }  
 else{  
 list=dataSourceTxt;  
 for(int i=0;i<list.size()-1;i++){  
 if(list.get(i).get(j).equals(list.get(i+1).get(j))){  
 map.put(i,list.get(i));  
 }  
 }  
  
 }  
  
 *//迭代删除重复的数据* Set<Integer> keySet=map.keySet();  
 Iterator<Integer>it=keySet.iterator();  
 while(it.hasNext()){  
 Integer key=it.next();  
 list.remove(list.get(key));  
  
 }  
  
 }  
  
 *//对表中的空值进行处理，置为0* public void nullHandle(){  
  
 ArrayList al=null;  
 for(int i=0;i<dataSourceTxt.size();i++){  
 al=dataSourceTxt.get(i);  
 for(int j=0;j<al.size();j++){  
 if(al.get(j).equals("")){  
 al.set(j,"null");  
 }  
 }  
 }  
 int a=0;  
 }  
  
 public void createXlsx() throws IOException {  
 Workbook wb = new XSSFWorkbook();  
 CreationHelper createHelper = wb.getCreationHelper();  
 Sheet sheet = wb.createSheet("new sheet");  
  
 Row row=null;  
 for(int i=0;i<dataSourceTxt.size();i++){  
 row = sheet.createRow(i);  
 ArrayList list=dataSourceTxt.get(i);  
 for(int j=0;j<list.size();j++){  
 row.createCell(j).setCellValue(list.get(j).toString());  
 }  
 }  
  
 FileOutputStream fileOut = new FileOutputStream("workbook03.xlsx");  
 wb.write(fileOut);  
 fileOut.close();  
 }  
  
 */\*\*\*  
 \* 格式化Height:  
 \* 将单位定为m  
 \*/* public void formatHeight(){  
  
 for(int i=1;i<dataSourceTxt.size();i++){  
 ArrayList arrayList= dataSourceTxt.get(i);  
 Double temp=Double.*parseDouble*(arrayList.get(4).toString());  
 if(temp>10){  
 arrayList.set(4,String.*valueOf*(temp/100));  
 }  
 }  
  
 }  
  
 */\*\*\*  
 \* 以txt数据源(有学号)为基准，合并两个数据源  
 \*/* public void mergeData() throws IOException {  
  
 *//遍历dataSourceXlsx，对dataSourceTxt数据进行补充完善* dataSourceXlsx.forEach(e->  
 {  
  
 int length=dataSourceTxt.get(0).size();  
 ArrayList arrayList=null;  
 boolean flag=true;  
  
 for(int i=0;i<dataSourceTxt.size();i++){  
 if(dataSourceTxt.get(i).get(1).equals(e.get(1))){  
 arrayList=dataSourceTxt.get(i);  
 flag=false;  
 for(int j=0;j<length;j++){  
 *//txt的column为空，则用xlsx对应的column填充* if(arrayList.get(j).equals("")){  
 arrayList.set(j,e.get(j).toString());  
 }  
 }  
 }  
 }  
 if(flag){  
 dataSourceTxt.add(e);  
 }  
 }  
 );  
  
 int length=dataSourceTxt.size();  
 for(int i=length-1;i>99;i--){  
 dataSourceTxt.remove(i);  
 }  
 }  
  
 *//格式化ID，使之都带有前缀202\** public void formatID(){  
 for(int i=1;i<dataSourceTxt.size();i++){  
 ArrayList list=dataSourceTxt.get(i);  
 if(!list.get(0).toString().contains("202")){  
 list.set(0,"2020"+list.get(0));  
 }  
 }  
 }  
  
 *//格式化性别，男性和女性分别male和female* public void formatGender(){  
 for(int i=1;i<dataSourceTxt.size();i++){  
 ArrayList list=dataSourceTxt.get(i);  
 String gender=list.get(3).toString();  
 switch (gender){  
 case "boy":  
 list.set(3,"male");  
 break;  
 case "girl":  
 list.set(3,"female");  
 }  
 }  
 }  
  
 public void formatData(){  
 *//处理空数据* this.nullHandle();  
  
 *//统一Gender格式* this.formatGender();  
  
 *//统一Height格式* this.formatHeight();  
  
 *//统一ID格式* this.formatID();  
 }  
  
 *//对数据进行去重处理* public void del(){  
 this.delByNameOrId("xlsx","Id");  
 this.delByNameOrId("xlsx","Name");  
 this.delByNameOrId("txt","Id");  
 this.delByNameOrId("txt","Name");  
 }  
  
 *//1.统计学生中家乡在Beijing的所有课程的平均成绩* public void count1(){  
  
 ArrayList<ArrayList> bjerAl =new ArrayList<>();  
 ArrayList<Double> totalScores = new ArrayList();  
 double totalScore=0;  
 int count=0;  
 ArrayList tempAl=null;  
  
 for(int i=0;i<dataSourceTxt.size();i++){  
 tempAl=dataSourceTxt.get(i);  
 if(tempAl.get(2).toString().equals("Beijing")){  
 bjerAl.add(tempAl);  
 count++;  
 }  
 }  
 *//初始化各科总分集合* for(int i=5;i<15;i++){  
 totalScores.add(0.00);  
 }  
  
 *//计算家在北京的学生各科的总分* for(int k=5;k<15;k++){  
 for(int j=0;j<bjerAl.size();j++) {  
 double temp=totalScores.get(k-5);  
 tempAl=bjerAl.get(j);  
 if(k==14){  
 switch (tempAl.get(k).toString()){  
 case "bad":  
 totalScores.set(k-5,temp+25);  
 break;  
 case "general":  
 totalScores.set(k-5,temp+50);  
 break;  
 case "good":  
 totalScores.set(k-5,temp+75);  
 break;  
 case "excellent":  
 totalScores.set(k-5,temp+100);  
 break;  
 }  
 }else{  
 totalScores.set(k-5,temp+Double.*parseDouble*(tempAl.get(k).toString()));  
 }  
 }  
 }  
  
 ArrayList<Double> averages = new ArrayList<>();  
 for(int i=0;i<totalScores.size();i++){  
 averages.add(totalScores.get(i)/count);  
 }  
  
 System.*out*.println("1.学生中家乡在Beijing的所有课程的平均成绩(C1-C10)分别为：");  
 averages.forEach(e->  
 {  
 Double temp=Double.*parseDouble*(String.*format*("%.2f",e));  
 System.*out*.print(" "+temp+" ");  
 });  
 System.*out*.println();  
 System.*out*.println();  
 }  
  
 *//2.统计学生中家乡在广州，课程1在80分以上，且课程9在9分以上的男同学的数量* public void count2(){  
 int count=0;  
 ArrayList e=null;  
  
 for(int i=0;i<dataSourceTxt.size();i++){  
 e=dataSourceTxt.get(i);  
 boolean bl=e.get(2).equals("Guangzhou")&&e.get(3).equals("male")&&Integer.*parseInt*(e.get(5).toString())>80&&Integer.*parseInt*(e.get(5).toString())>9;  
 if(bl) {  
 count++;  
 }  
 }  
 System.*out*.println("2.学生中家乡在广州，课程1在80分以上，且课程9在9分以上的男同学的数量的为:"+count);  
 System.*out*.println();  
 }  
  
 *//3.比较广州和上海两地女生的平均体能测试成绩，哪个地区的更强些  
 //bad,general,good,excellent的值分别定为：25,50,75,100* public void count3(){  
 int countSH=0;  
 int countGZ=0;  
 ArrayList al=null;  
  
 for(int i=1;i<dataSourceTxt.size();i++){  
 al=dataSourceTxt.get(i);  
 if(al.get(3).toString().equals("female")){  
 switch (al.get(3).toString()){  
 case "Guangzhou":  
 switch (al.get(14).toString()){  
 case "bad":countGZ+=25;  
 break;  
 case "general":countGZ+=50;  
 break;  
 case "good":countGZ+=75;  
 break;  
 case "excellent":countGZ+=100;  
 break;  
 }  
 break;  
  
 case "Shanghai":  
 switch (al.get(14).toString()){  
 case "bad":countSH+=25;  
 break;  
 case "general":countSH+=50;  
 break;  
 case "good":countSH+=75;  
 break;  
 case "excellent":countSH+=100;  
 break;  
 }  
 }  
 }  
 }  
  
 System.*out*.print("3.广州和上海两地女生的平均体能测试成绩,更强的地区是：");  
 if(countGZ>countSH){  
 System.*out*.println("广州");  
 }  
 else  
 {  
 System.*out*.println("上海");  
 }  
 System.*out*.println();  
 }  
  
 *//4.求学习成绩和体能测试成绩，两者的相关性* public void count4() {  
 ArrayList<Double> correlation = new ArrayList<>();  
 ArrayList<ArrayList<Double>> zscoreAl = this.Zscore();  
 double total = 0;  
 ArrayList<Double> temp = null;  
 ArrayList<Double> gymAl = zscoreAl.get(zscoreAl.size() - 1);  
 for (int i = 0; i < zscoreAl.size() - 1; i++) {  
 total = 0;  
 temp = zscoreAl.get(i);  
 for (int j = 0; j < temp.size(); j++) {  
 double a = temp.get(j);  
 double b = gymAl.get(j);  
 total += a \* b;  
 }  
 correlation.add(total);  
 }  
 System.*out*.println("4.各学科(C1-C9)的成绩和体能测试成绩的相关性分别为：");  
 correlation.forEach(e ->  
 {  
 *//保留小数点后5位小数* Double te=Double.*parseDouble*(String.*format*("%.5f",e));  
 System.*out*.print(" "+te + "\t");  
 });  
 System.*out*.println();  
 }  
  
 *//计算所有科目的平均值，打包成集合返回* public ArrayList<Double> Average(){  
 List<ArrayList> arrayLists=dataSourceTxt;  
 double total=0;  
 ArrayList al=null;  
 ArrayList averages=new ArrayList<Double>();  
  
 for(int j=5;j<15;j++){  
 total=0;  
 for(int i=1;i<arrayLists.size();i++){  
 al=arrayLists.get(i);  
 if(j==14){  
 switch (al.get(14).toString()){  
 case "bad":  
 total+=25;  
 break;  
 case "general":  
 total+=50;  
 break;  
 case "good":  
 total+=75;  
 break;  
 case "excellent":  
 total+=100;  
 break;  
 }  
 }  
 else if(!al.get(j).toString().equals("null")){  
 total+=Integer.*parseInt*(al.get(j).toString());  
 }  
 }  
 Double temp=Double.*parseDouble*(String.*format*("%.2f",total/dataSourceTxt.size()));  
 averages.add(temp);  
 }  
 return averages;  
 }  
  
 *//将相同科目的成绩归纳到一个ArrayList，再将所有科目归纳到一个ArrayList* public ArrayList<ArrayList<Double>> subjectsToArray(){  
  
 ArrayList<Double> averages=this.Average();  
 ArrayList<ArrayList<Double>> allSubjectAl = new ArrayList<>();  
 ArrayList subject=null;  
 ArrayList tempAl=null;  
  
 for(int j=5;j<15;j++){  
 subject=new ArrayList();  
 for(int i=1;i<dataSourceTxt.size();i++){  
 tempAl=dataSourceTxt.get(i);  
 if(j==14){  
 int a=0;  
 switch(tempAl.get(j).toString()){  
 case "bad":  
 subject.add(25);  
 break;  
 case "general":  
 subject.add(50);  
 break;  
 case "good":  
 subject.add(75);  
 break;  
 case "excellent":  
 subject.add(100);  
 break;  
 default:  
 subject.add(averages.get(j-5));  
 }  
 }else {  
 *//值为null则用平均值来代替，否则后面无法计算* if(tempAl.get(j).equals("null")){  
 subject.add(averages.get(j-5));  
 }else{  
 subject.add(tempAl.get(j));  
 }  
 }  
 }  
 allSubjectAl.add(subject);  
 }  
 return allSubjectAl;  
 }  
  
 *//求协方差* public ArrayList<Double> Covariance(){  
 ArrayList<ArrayList<Double>> subjectsAl=this.subjectsToArray();  
 ArrayList<Double> averages=this.Average();  
 double total=0;  
 ArrayList<Double> covariances=new ArrayList();  
  
 for(int i=0;i<averages.size();i++){  
 ArrayList subject=subjectsAl.get(i);  
 for(int j=0;j<subject.size();j++){  
 if(!subject.get(j).toString().equals("null")){  
 total+=Math.*pow*(Double.*parseDouble*(subject.get(j).toString())-Double.*parseDouble*(averages.get(i).toString()),2);  
 }  
 }  
 covariances.add(total/(subject.size()-1));  
 }  
 return covariances;  
 }  
  
 *//求标准差* public ArrayList<Double> Std(){  
 ArrayList<Double> stdAl=new ArrayList<>();  
 ArrayList<Double> covariance=this.Covariance();  
 for(int i=0;i<covariance.size();i++){  
 stdAl.add(Math.*sqrt*(covariance.get(i)));  
 }  
 return stdAl;  
 }  
  
 *//求Z-score规范化* public ArrayList<ArrayList<Double>> Zscore(){  
 ArrayList<ArrayList<Double>> zscoreAl = new ArrayList<>();  
 ArrayList<Double> averageAl=this.Average();  
 ArrayList<Double> std=this.Std();  
 ArrayList<ArrayList<Double>> al=this.subjectsToArray();  
 ArrayList<Double> temp;  
 ArrayList<Double> tempzscore=null;  
  
 for(int i=0;i<al.size();i++){  
 temp=al.get(i);  
 tempzscore=new ArrayList<>();  
 for(int j=0;j<temp.size();j++){  
*// if(i==al.size()-1){  
// int a=0;  
// }* double a=Double.*parseDouble*(String.*valueOf*(temp.get(j)));  
 double b=averageAl.get(i);  
 double c=std.get(i);  
 tempzscore.add((a-b)/c);  
 }  
 zscoreAl.add(tempzscore);  
 }  
 return zscoreAl;  
 }  
  
 public static void main(String[] args) throws IOException {  
  
 *//读取数据源* project1 project=new project1();  
 project.parseXlsx("D:\\Java\\JavaProject\\Homework\\src\\main\\resources\\1.xlsx");  
 project.parseTxt("D:\\Java\\JavaProject\\Homework\\src\\main\\resources\\2.txt");  
  
 *//对数据进行去重处理* project.del();  
  
 *//合并数据* project.mergeData();  
 int b=0;  
 *//处理空数据* project.nullHandle();  
  
 *//对数据进行统一格式处理，包括：统一Height格式、ID格式、gender格式* project.formatData();  
  
 *//创建新xlsx表* project.createXlsx();  
  
 *//1.统计学生中家乡在Beijing的所有课程的平均成绩。* project.count1();  
  
 *//2.学生中家乡在广州，课程1在80分以上，且课程9在9分以上的男同学的数量。* project.count2();  
  
 *//3.比较广州和上海两地女生的平均体能测试成绩，哪个地区的更强些？* project.count3();  
  
 *//4.学习成绩和体能测试成绩，两者的相关性是多少？（九门课的成绩分别与体能成绩计算相关性）* project.count4();  
  
 }  
 }