Name: Yin Zhanpeng Student ID: [34742217](mailto:34742217@student.murdoch.edu.au)

1.Title: Student data handling.

Purpose: This program will do basic manipulation of the student data from the csv file and after changing the data it will output the new student data in a new csv file.

Author: Yin Zhanpeng Date: 27 March 2023

2. The program is supposed to read from a student csv file, add the necessary information from the student csv file to an ArrayList, have function that can remove student by ID, have function that can display all the information in the ArrayList, have function that can calculate the over marks and determine how many sore above and below the overall marks, have function that can display the student by Id, have function that can sort the ArrayList by student id in ascending order and finally be able to output the ArrayList in a csv file.

|  |  |
| --- | --- |
| Inheritance | I use Inheritance in creating my classes, I created two super class student class and unit class, and also created two sub class, Student\_Course class, Student\_Research class which inherit from the student super class. And Unit\_Course class, Research class which will inherit from the unit super class |
| Polymorphism | i use polymorphism in Java, which make use of the method overriding feature. For example, I define a method in a superclass, such as a reportGrade() method in a Student class, and then override that method in its subclasses, such as a StudentCourse or StudentResearch class. Each subclass implements the method differently, displaying different information depending on the specific type of student. This allows me to write more flexible and reusable code that can handle different types of students and their associated data. |
| Dynamic binding | I use the dynamic binding feature in Java to allow my code to output different information based on the type of student, using an ArrayList to hold objects of different student classes. For example, I define a reportGrade() method in a superclass, such as a Student class, and then override that method in its subclasses, such as a StudentCourse or StudentResearch class. Each subclass can implement the method differently, displaying different information depending on the specific type of student. When I call the reportGrade() method on an ArrayList of Student objects that includes instances of different student classes, the JVM will use dynamic binding to determine the appropriate implementation of the method to use for each object in the list, based on its actual type. This allows me to write more flexible and reusable code that can handle different types of students and their associated data. |
| Sorting algorithm | To sort my ArrayList of students in Java, I use a sorting algorithm, such as bubble sort. For example, I sort the students by their ID number in ascending order, by comparing the ID numbers of adjacent elements and swapping them if they are out of order. I implement the bubble sort algorithm in my code to traverse the ArrayList and sort the elements in place. This allows me to organize my data in a more meaningful and useful way, making it easier to find and access specific information about the students. |
| Handling CSV files | I handle my csv file by using the java.io package which give me classes such as File, (to get the directory of my file), FileWriter, (to write in the file in a new line instead overwriting it ), IOException,(which allow me to not crash my program if something wrong in the file),and PrintWriter(to write in a file) |

Bubble sort is different to the insertion sort algorithm, because bubble sort compares adjacent elements and swap them if they are not in order while insertion sort works by starting assuming the first item is sorted then compare to each items (from left to right)and inserting the new item to its proper position

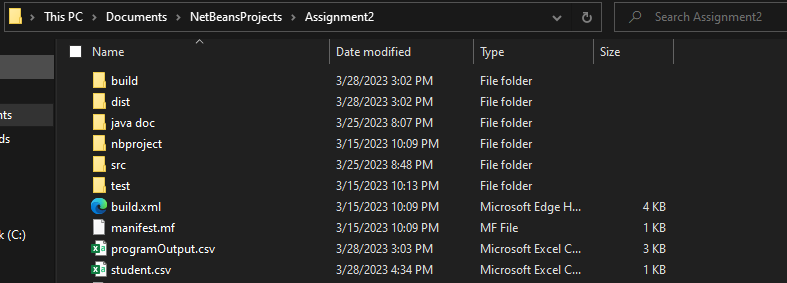
3. user guide

Step 1 load the program. Make sure all the program is loaded.

Text

Description automatically generated with medium confidence

Step 2 make sure that the student csv file is in the not same place as the rest of your program in the ide, the student csv file should be just in your assignment 2 folder see below for explanation



The programOutput.csv file will also be here it will be explain in the later steps

Graphical user interface, text, application, chat or text message

Description automatically generated

CORRECT

WRONG

STEP 3 click run.

Text

Description automatically generated

STEP 4 always start eith option 2 first as the program needs the data to do all the other options

Options 1 is to quit

Option 2 is to import the data from the student csv, make sure to enter the exact name of the file,”student.csv”

Text

Description automatically generated

Option 3 is to delete student by id

Text

Description automatically generated

Option 4 is to display all the infoamtion in the array list

Text

Description automatically generated

Option 5 is to calculate the average marks and number of students who get above and below it (course work student only)

Text

Description automatically generated

Option 6 is to report the grades of a student by id

Text

Description automatically generated

Option 7 is to sort the students by their id in accending order

Option 8 is to output the data in the array list to “programOutput.csv”

A screenshot of a computer

Description automatically generated with medium confidence

The output csv file is located here

Shape, rectangle

Description automatically generated

It should be the same place as your student.csv file

4. Pseudocode

**Class Student:**

**private String firstName**

**private String lastName**

**private long studentNumber**

**Constructor Student ():**

**this.firstName ← "Firstname"**

**this.lastName ← "LastName"**

**this.studentNumber ← 00**

**Constructor Student(firstName, lastName, studentNumber):**

**this.firstName ← firstName**

**this.lastName ← lastName**

**this.studentNumber ← studentNumber**

**Method reportGrade(sorted, OUTPUT\_FILENAME):**

**Print "There is no grade here"**

**Method equals(student2):**

**sameStudentNumber ← false**

**if (getStudentNumber() equals student2.getStudentNumber()):**

**sameStudentNumber ← true**

**return sameStudentNumber**

**Method setFirstName(firstName):**

**this.firstName ← firstName**

**Method setLastName(lastName):**

**this.lastName ← lastName**

**Method setStudentNumber(studentNumber):**

**this.studentNumber ← studentNumber**

**Method getFirstName():**

**return firstName**

**Method getLastName():**

**return lastName**

**Method getStudentNumber():**

**return studentNumber**

**Method getEnrolmentType():**

**return "no enrolment"**

**Method toString():**

**return firstName + "," + lastName + "," + studentNumber**

**class Student\_Course extends Student:**

**// define class variables**

**enrolmentType = "C"**

**studentCourseAssignment1**

**studentCourseAssignment2**

**studentCourseFinalExamination**

**unitID**

**levelOfUnit**

**// define constructor**

**function Student\_Course(studentCourseAssignment1, studentCourseAssignment2, studentCourseFinalExamination, firstName, lastName, studentNumber, unitID, levelOfUnit):**

**call super constructor with firstName, lastName, and studentNumber**

**set studentCourseAssignment1, studentCourseAssignment2, studentCourseFinalExamination, unitID, and levelOfUnit**

**// define getEnrolmentType method**

**function getEnrolmentType():**

**return enrolmentType**

**// define getOverallMarks method**

**function getOverallMarks():**

**// create new Unit\_Course object**

**uc = new Unit\_Course(studentCourseAssignment1, studentCourseAssignment2, studentCourseFinalExamination, enrolmentType, unitID, levelOfUnit)**

**// calculate overall mark**

**uc.overallMarkCalculation()**

**return uc.overallMark**

**// define reportGrade method**

**function reportGrade(sorted, OUTPUT\_FILENAME):**

**// create new Unit\_Course object**

**uc = new Unit\_Course(studentCourseAssignment1, studentCourseAssignment2, studentCourseFinalExamination, enrolmentType, unitID, levelOfUnit)**

**if sorted:**

**try:**

**// create PrintWriter object**

**outputFile = new File(OUTPUT\_FILENAME)**

**append = outputFile.exists() && outputFile.length() > 0**

**fw = new FileWriter(OUTPUT\_FILENAME, append)**

**pw = new PrintWriter(fw)**

**// write student grade information to file**

**pw.print("Enrolment Type: " + enrolmentType)**

**pw.print(" Student Name: " + super.getFirstName() + " " + super.getLastName())**

**pw.print(" Student Number: " + super.getStudentNumber())**

**uc.overallMarkCalculation()**

**uc.finalGrade = uc.finalGradeCalculation(uc.overallMark)**

**pw.print(" Student Overall Mark: " + " " + uc.overallMark)**

**pw.print(" Student Final Grade: " + " " + uc.finalGrade)**

**pw.print(" Student Unit ID: " + " " + uc.unitID)**

**pw.print(" Student Unit Level: " + " " + uc.levelOfUnit)**

**pw.println("")**

**except IOException ex:**

**// handle file error**

**print "file error"**

**else:**

**// write student grade information to console**

**print "Enrolment Type: " + enrolmentType**

**print "Student Name: " + super.getFirstName() + " " + super.getLastName()**

**print "Student Number: " + super.getStudentNumber()**

**uc.overallMarkCalculation()**

**uc.finalGrade = uc.finalGradeCalculation(uc.overallMark)**

**print "Student Overall Mark: " + " " + uc.overallMark**

**print "Student Final Grade: " + " " + uc.finalGrade**

**print "Student Unit ID: " + " " + uc.unitID**

**print "Student Unit Level: " + " " + uc.levelOfUnit**

**// define toString method**

**function toString():**

**return enrolmentType**

**Class Student\_Research extends Student:**

**private String enrolmentType = "R"**

**private int student\_ResearchProposalMark**

**private int student\_ResearchFinalDissertationMark**

**Constructor Student\_Research():**

**// No-args constructor**

**Constructor Student\_Research(student\_ResearchProposalMark, student\_ResearchFinalDissertationMark, firstName, lastName, studentNumber):**

**// Constructor with arguments**

**Call superclass constructor with firstName, lastName, and studentNumber as parameters**

**Set this.student\_ResearchProposalMark to student\_ResearchProposalMark**

**Set this.student\_ResearchFinalDissertationMark to student\_ResearchFinalDissertationMark**

**Method getEnrolmentType():**

**Return this.enrolmentType**

**Method reportGrade(sorted, OUTPUT\_FILENAME):**

**Create a new instance of Research with this.student\_ResearchProposalMark, this.student\_ResearchFinalDissertationMark, and this.enrolmentType as parameters**

**If sorted is true:**

**Try:**

**Create a new File object with OUTPUT\_FILENAME as parameter**

**Set boolean append to outputFile.exists() and outputFile.length() > 0**

**Create a new FileWriter with OUTPUT\_FILENAME and append as parameters**

**Create a new PrintWriter with the FileWriter as parameter**

**Write "Enrolment Type: " + enrolmentType to the PrintWriter**

**Write "Student Name: " + super.getFirstName() + " " + super.getLastName() to the PrintWriter**

**Write "Student Number: " + super.getStudentNumber() to the PrintWriter**

**Call r.overallMarkCalculation()**

**Set r.finalGrade to r.finalGradeCalculation(r.overallMark)**

**Write "Student Overall Mark: " + " " + r.overallMark to the PrintWriter**

**Write "Student Final Grade: " + " " + r.finalGrade to the PrintWriter**

**Write a new line to the PrintWriter**

**Catch IOException ex:**

**Print "file error"**

**Else:**

**Print "Enrolment Type: " + enrolmentType**

**Print "Student Name: " + super.getFirstName() + " " + super.getLastName()**

**Print "Student Number: " + super.getStudentNumber()**

**Call r.overallMarkCalculation()**

**Set r.finalGrade to r.finalGradeCalculation(r.overallMark)**

**Print "Student Overall Mark: " + " " + r.overallMark**

**Print "Student Final Grade: " + " " + r.finalGrade**

**Method toString():**

**Return this.enrolmentType**

**Class Unit:**

**private String enrolmentType**

**private int overallMark**

**private String finalGrade**

**Constructor Unit(enrolmentType):**

**this.enrolmentType ← enrolmentType**

**Method getEnrolmentType():**

**return enrolmentType**

**Method gradeReporting():**

**Print "NA"**

**Method overallMarkCalculation():**

**Helper.println("no marks to calculate")**

**Method finalGradeCalculation(overallMarks):**

**if overallMarks < 50:**

**return "N"**

**else if overallMarks >= 80:**

**return "HD"**

**else if overallMarks >= 70:**

**return "D"**

**else if overallMarks >= 60:**

**return "C"**

**else if overallMarks >= 50:**

**return "P"**

**return "error"**

**Class Research extends Unit:**

**private String enrolmentType ← "R"**

**private int proposalMark**

**private int finalDissertationMark**

**private int overallMark**

**private String finalGrade**

**Constructor Research(proposalMark, finalDissertationMark, enrolmentType):**

**super(enrolmentType)**

**this.proposalMark ← proposalMark**

**this.finalDissertationMark ← finalDissertationMark**

**Method overallMarkCalculation():**

**this.overallMark ← (int) (proposalMark \* 0.4 + finalDissertationMark \* 0.6)**

**Class Unit\_Course extends Unit:**

**private String unitID**

**private int levelOfUnit**

**private String enrolmentType ← "C"**

**private int assignment1 //max 100**

**private int assignment2 //max 100**

**private int finalExamination //max 100**

**private int overallMark**

**private String finalGrade**

**Constructor Unit\_Course(assignment1, assignment2, finalExamination, enrolmentType, unitID, levelOfUnit):**

**super(enrolmentType)**

**this.assignment1 ← assignment1**

**this.assignment2 ← assignment2**

**this.finalExamination ← finalExamination**

**this.unitID ← unitID**

**this.levelOfUnit ← levelOfUnit**

**Method overallMarkCalculation():**

**this.overallMark ← (int) (assignment1 \* 0.3 + assignment2 \* 0.3 + finalExamination \* 0.4)**

**Method toString():**

**return unitID + "," + levelOfUnit + "," + assignment1**

**+ "," + assignment2 + "," + finalExamination**

**+ "," + overallMark + "," + finalGrade**

**Client class**

**students ← [] # an empty list to store student data**

**OUTPUTFILE ← "programOutput.csv"**

**bubbleStored ← false**

**carrybit ← false**

**function run():**

**printMenuAndGetChoice()**

**function printMenuAndGetChoice():**

**carryOn ← True**

**while carryOn do**

**printMenu()**

**choice ← Helper.getString("please enter only options 1 to 8, 1 is to quit")**

**switch choice do**

**case "1":**

**carryOn ← false**

**Helper.println("Thank you, have a good day. bye bye ")**

**case "2":**

**addMarksInfomationFromCsv()**

**case "3":**

**removeInfomationById()**

**case "4":**

**outputAllDetailsOfStudents()**

**case "5":**

**getNumberOfStudentAboveAndBelowAverageCourseWork()**

**case "6":**

**checkArrayListByStudentId()**

**case "7":**

**sortingStudentsById()**

**case "8":**

**printSortedArrayListToCsv()**

**otherwise:**

**Helper.println("Invalid choice, Please only enter the number 1-8")**

**function addMarksInfomationFromCsv():**

**try:**

**fileName ← Helper.getString("Please enter the file name")**

**file ← File(fileName)**

**scanner ← Scanner(file)**

**while scanner.hasNextLine() do**

**line ← scanner.nextLine()**

**values ← line.split(",")**

**if values[0] equals to "C" then**

**student ← Student\_Course(values[1], values[2], values[3], values[4], values[5], values[6], values[7])**

**otherwise,**

**student ← Student(values[1], values[2], values[3], values[4], values[5], values[6])**

**students.append(student)**

**Helper.println("Data has been added to the ArrayList")**

**except FileNotFoundException:**

**Helper.println("File not found")**

**function removeInfomationById():**

**deletebyId()**

**function outputAllDetailsOfStudents():**

**for student in students do**

**student.reportGrade(carrybit, OUTPUTFILE)**

**function getNumberOfStudentAboveAndBelowAverageCourseWork():**

**averageMarks ← aveageMarksForCourseWorkStudent()**

**numberOfStudentGetAboveOrBelowCalculation(averageMarks)**

**function checkArrayListByStudentId():**

**getIdAndDisplay()**

**function sortingStudentsById():**

**bubbleSort()**

**function printSortedArrayListToCsv():**

**printToCsv()**

**function printToCsv():**

**if bubbleStored then**

**Helper.println("yes")**

**carrybit ← True**

**for student in students do**

**student.reportGrade(carrybit, OUTPUTFILE)**

**Helper.println("Arraylist have been printed to programOutput.csv")**

**carrybit ← False**

**otherwise,**

**Helper.println("Arraylist have not been sorted, please sort it before using this function")**

**function bubbleSort():**

**swap ← True**

**while swap do**

**swap ← False**

**for i in range(1, length(students)) do**

**studentId1 ← students[i - 1].getStudentNumber()**

**studentId2 ← students[i].getStudentNumber()**

**if studentId1 > studentId2 then**

**student1 ← students[i - 1]**

**student2 ← students[i]**

**students[i - 1] ← student2**

**students[i] ← student1**

**swap ← True**

**Helper.println("file have been sorted")**

**Function: getIdAndDisplay()**

**Input: None**

**Output: None**

**id <- Helper.getInt("Enter the ID")**

**for i <- 0 to students.size()-1**

**a. if id == students.get(i).getStudentNumber() then**

**i. students.get(i).reportGrade(carrybit, OUTPUTFILE)**

**ii. break**

**Helper.println("This ID does not exist")**

**Function: numberOfStudentGetAboveOrBelowCalculation(averageMarks)**

**Input: averageMarks (an integer representing the average marks)**

**Output: None**

**indexAbove <- 0**

**indexBelow <- 0**

**for i <- 0 to students.size()-1 a. if (students.get(i)).getEnrolmentType().equals("C") then**

**i. if ((Student\_Course) students.get(i)).getOverallMarks() >= averageMarks then**

**1. indexAbove <- indexAbove + 1**

**ii. else 1. indexBelow <- indexBelow + 1**

**print indexAbove + " number of students get above or equal to the average marks of : " + averageMarks**

**print indexBelow + " number of students get below the average marks of : " + averageMarks**

**Function: aveageMarksForCourseWorkStudent()**

**Input: None**

**Output: averageMarks (an integer representing the average marks)**

**averageMarks <- 0**

**index <- 0**

**for i <- 0 to students.size()-1 a. if (students.get(i)).getEnrolmentType().equals("C") then**

**i. averageMarks <- averageMarks + ((Student\_Course) students.get(i)).getOverallMarks()**

**ii. index <- index + 1**

**averageMarks <- averageMarks / index**

**return averageMarks**

**Function: deletebyId()**

**Input: None**

**Output: None**

**inputId <- Helper.getInt("Please enter the student ID")**

**carryOn <- true**

**for i <- 0 to students.size()-1 a. if inputId == students.get(i).getStudentNumber() then**

**i. index <- i**

**ii. deleteByIndex(index)**

**iii. carryOn <- false**

**if carryOn then**

**a. Helper.println("Student not found")**

**Function: deleteByIndex(index)**

**Input: index (an integer representing the index of the student to be deleted)**

**Output: None**

**if index >= 0 then**

**a. print "Do you want to delete student: " + students.get(index).getFirstName() + " with ID " + students.get(index).getStudentNumber() + "? (y/n) "**

**b. answer <- Helper.getString(" ").toLowerCase()**

**c. if answer equals "y" then**

**i. print "Student: " + students.get(index).getFirstName() + " with ID " + students.get(index).getStudentNumber() + " has been deleted."**

**ii. students.remove(index) d. else i. print "Deletion cancelled."**

**Function: addDataToArrayList()**

**Input: None**

**Output: None**

**inputFileName <- Helper.getString("Enter the file name")**

**fp <- new File(inputFileName)**

**file <- new Scanner(fp)**

**while file has next line do**

**a. line <- file.nextLine()**

**b. datas <- new ArrayList<String>()**

**c. datas.addAll(Arrays.asList(line.split(",")))**

**d. if datas.get(0).equals("C") then**

**i. studentsc <- new Student\_Course(Integer.parseInt(datas.get(1)), Integer.parseInt**

**FUNCTION addMarksInfomationFromCsv():**

**TRY: CALL addDataToArrayList()**

**CATCH FileNotFoundException:**

**OUTPUT "file not found"**

**FUNCTION removeInfomationById():**

**CALL deletebyId()**

**FUNCTION outputAllDetailsOfStudents():**

**FOR EACH student IN students DO:**

**CALL student.reportGrade(carrybit, OUTPUTFILE) OUTPUT "---------------------------------------------"**

**FUNCTION getNumberOfStudentAboveAndBelowAverageCourseWork():**

**TRY: CALL numberOfStudentGetAboveOrBelowCalculation(aveageMarksForCourseWorkStudent()) CATCH ArithmeticException: OUTPUT "Cant divide by 0"**

**FUNCTION checkArrayListByStudentId():**

**CALL getIdAndDisplay()**

**FUNCTION sortingStudentsById():**

**CALL bubbleSort() bubbleStored ← true**

**FUNCTION printSortedArrayListToCsv():**

**CALL printToCsv()**

**FUNCTION printMenu():**

**OUTPUT "1. Quit (exit the program)"**

**OUTPUT ""**

**OUTPUT ""**

**OUTPUT "2. Add (to the ArrayList) from csv. "**

**OUTPUT ""**

**OUTPUT ""**

**OUTPUT "3. Delete from arraylist by student ID. "**

**OUTPUT ""**

**OUTPUT ""**

**OUTPUT "4. Output all details currently held in the ArrayList. "**

**OUTPUT ""**

**OUTPUT ""**

**OUTPUT "5. Calculate overall marks for course work students. "**

**OUTPUT ""**

**OUTPUT ""**

**OUTPUT "6. Report grade by student ID. "**

**OUTPUT ""**

**OUTPUT ""**

**OUTPUT "7. Bubble sort the students in arraylist. "**

**OUTPUT ""**

**OUTPUT ""**

**OUTPUT "8. Output SORTED arraylist to csv. "**

**OUTPUT ""**

**OUTPUT ""**

UML Diagram

Graphical user interface, timeline

Description automatically generated

**(Refer to the ppt file for a clearer chart)**

Structure Diagram

Timeline

Description automatically generated

**(Refer to the ppt file for a clearer chart)**

5.limitations:

This program does only can handle two type of classes, course work and research class.

This program does not have any security to protect the data.

This program can only be run using an IDE and not an application base.

6. Testing:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Id | Test description | Data for test | Expected output | Actual output | Test Pass/Fail |
| 1 | Testing the equals method for the student class | Create two random student object both with different student number | It should return false |  | Pass |
| 2 | Testing the reportgrade in the student research class | Create a student research obj and fill up random info | It should return the the student’s infomations |  | Pass |
| 3 | Testing the marks calculation in the student course class | Create a student course obj and fill up the marks A1,A2&Exam with50,50,100 | It should return 70 |  | pass |
| 4 | Testing the final grade calculation in the unit class | Use the data 80,70,60,50,90 the calculation mehod | It should return HD,D,C,P,HD |  | PASS |
| 5 | Testing the research class the final grade calculation method to see if it can use the method in its parent class(unit) | Create a research class obj and pass in the marks of 50 and 100 and printing out the overall marks and the final grade | It should displat 80 and HD |  | pass |
| 6 | Testing the unit course class the final grade calculation method to see if it can use the method in its parent class(unit) | Create a research class obj and pass in the marks of 50 ,50 and 100 and printing out the overall marks and the final grade | It should displat 70 and D |  | pass |
| 7 | Testing the client class to see if the whole program works (option2) | Using the student csv file to test. | It should import the right file for (student.csv)and error for the rest |  | pass |
| 8 | Testing the client class to see if the whole program works (option3) | Using the student csv file to test. | It should give me the name of the student I want to delete and confirm with me again  It should also provide an error msg if no such student |  | pass |
| 9 | Testing the client class to see if the whole program works (option4) | Using the student csv file to test. | It should output all the student information stored in the array list and not including the ones that have been deleted |  | pass |
| 10 | Testing the client class to see if the whole program works (option5) | Using the student csv file to test. | It should return 61 as the avg marks and 4 above and 1 below |  | pass |
| 11 | Testing the client class to see if the whole program works (option6) | Using the student csv file to test. Using student 1 and 3 | It should display only 1 student and show an error msg if student is not found |  | pass |
| 12 | Testing the client class to see if the whole program works (option7) | Using the student csv file to test. | It should sort the students by student number and using option 4 to confirm |  | pass |
| 13 | Testing the client class to see if the whole program works (option8) | Using the student csv file to test. | It should output the sorted csv in a new file |  | pass |

7.source program listing

Student class

package assignment2

/\*\*

\* @author yin zhanpeng

\* @version 1.0 (current version number of program)

\* Assignment 2

\* this program have will read the student csv file and have functions that will provide a summary for the students

\*/

public class Student {

private String firstName;

private String

lastName;

private long studentNumber;

/\*\*

\* constructor

\*/

public Student() {

this.firstName = "Firstname";

this.lastName = "LastName";

this.studentNumber = 00;

}

/\*\*

\* constructor

\* @param firstName first name

\* @param lastName last name

\* @param studentNumber student number

\*/

public Student(String firstName, String lastName, long studentNumber) {

this.firstName = firstName;

this.lastName = lastName;

this.studentNumber = studentNumber;

}

/\*\*

\* display grade

\* @param sorted check if array list is sorted

\* @param OUTPUT\_FILENAME the file name which the array list will output

\*/

public void reportGrade(boolean sorted, String OUTPUT\_FILENAME) {

System.out.println("There is no grade here");

}

/\*\*

\* check if student A is equal to student B by student number

\* @param student2 the student you want to check with

\* @return boolean depending if the students are the same or not

\*/

public boolean equals(Student student2) {

boolean sameStudentNumber = false;

if (getStudentNumber() == student2.getStudentNumber()) {

sameStudentNumber = true;

}

return sameStudentNumber;

}

/\*\*

\* set the first name

\* @param firstName name you want to set

\*/

public void setFirstName(String firstName) {

this.firstName = firstName;

}

/\*\*

\* set last name

\* @param lastName the last name you want to set

\*/

public void setLastName(String lastName) {

this.lastName = lastName;

}

/\*\*

\* set student number

\* @param studentNumber the student number you want to set

\*/

public void setStudentNumber(long studentNumber) {

this.studentNumber = studentNumber;

}

/\*\*

\* get the first name

\* @return first name

\*/

public String getFirstName() {

return firstName;

}

/\*\*

\* get the last name

\* @return last name

\*/

public String getLastName() {

return lastName;

}

/\*\*

\* get the student number

\* @return student number

\*/

public long getStudentNumber() {

return studentNumber;

}

/\*\*

\* get the enrollment type

\* @return enrollment type

\*/

public String getEnrolmentType() {

return "no enrolment";

}

/\*\*

\* convert to string

\* @return first name, last name, student number

\*/

@Override

public String toString() {

return this.firstName + "," + this.lastName + "," + this.studentNumber;

}

//testing

/\*\*

\* testing

\* @param args args

\*/

public static void main(String[] args) {

Student student1 = new Student("john", "yip", 12);

Student student2 = new Student("tom", "neo", 2);

System.out.println(student1.equals(student2));

}

}

Student course class

package assignment2;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

/\*\*

\* @author yin zhanpeng

\* @version 1.0 (current version number of program)

\* Assignment 2

\* this program have will read the student csv file and have functions that will provide a summary for the students

\*/

public class Student\_Course extends Student {

private final String enrolmentType = "C";

private final int studentCourseAssignment1;

private final int studentCourseAssignment2;

private final int studentCourseFinalExamination;

private final String unitID;

private final int levelOfUnit;

/\*\*

\* Constructor

\* @param studentCourseAssignment1 Marks of assignment 1

\* @param studentCourseAssignment2 Marks of assignment 2

\* @param studentCourseFinalExamination Marks of final exam

\* @param firstName First name

\* @param lastName Last name

\* @param studentNumber Student number

\* @param unitID Student unit id

\* @param levelOfUnit Student level

\*/

public Student\_Course(int studentCourseAssignment1, int studentCourseAssignment2, int studentCourseFinalExamination, String firstName, String lastName, long studentNumber, String unitID, int levelOfUnit) {

super(firstName, lastName, studentNumber);

this.studentCourseAssignment1 = studentCourseAssignment1;

this.studentCourseAssignment2 = studentCourseAssignment2;

this.studentCourseFinalExamination = studentCourseFinalExamination;

this.unitID = unitID;

this.levelOfUnit = levelOfUnit;

}

/\*\*

\* returns the Enrollment type

\* @return Enrollment type

\*/

@Override

public String getEnrolmentType() {

return enrolmentType;

}

/\*\*

\* get the overall marks from the unit course

\* @return overall marks

\*/

public int getOverallMarks() {

Unit\_Course uc = new Unit\_Course(studentCourseAssignment1, studentCourseAssignment2, studentCourseFinalExamination, enrolmentType, unitID, levelOfUnit);

uc.overallMarkCalculation();

return uc.overallMark;

}

/\*\*

\* display the grades

\* @param sorted check if the array list is sorted

\* @param OUTPUT\_FILENAME the file which the array list will output

\*/

@Override

public void reportGrade(boolean sorted, String OUTPUT\_FILENAME) {

Unit\_Course uc = new Unit\_Course(studentCourseAssignment1, studentCourseAssignment2, studentCourseFinalExamination, enrolmentType, unitID, levelOfUnit);

if (sorted) {

try {

//PrintWriter pw = new PrintWriter(OUTPUT\_FILENAME);

File outputFile = new File(OUTPUT\_FILENAME);

boolean append = outputFile.exists() && outputFile.length() > 0;

FileWriter fw = new FileWriter(OUTPUT\_FILENAME, append);

try (PrintWriter pw = new PrintWriter(fw)) {

pw.print("Enrolment Type: " + enrolmentType);

pw.print(" Student Name: " + super.getFirstName() + " " + super.getLastName());

pw.print(" Student Number: " + super.getStudentNumber());

uc.overallMarkCalculation();

uc.finalGrade = uc.finalGradeCalculation(uc.overallMark);

pw.print(" Student Overall Mark: " + " " + uc.overallMark);

pw.print(" Student Final Grade: " + " " + uc.finalGrade);

pw.print(" Student Unit ID: " + " " + uc.unitID);

pw.print(" Student Unit Level: " + " " + uc.levelOfUnit);

pw.println("");

}

} catch (IOException ex) {

System.out.println("file error");

}

} else {

System.out.println("Enrolment Type: " + enrolmentType);

System.out.println("Student Name: " + super.getFirstName() + " " + super.getLastName());

System.out.println("Student Number: " + super.getStudentNumber());

uc.overallMarkCalculation();

uc.finalGrade = uc.finalGradeCalculation(uc.overallMark);

System.out.println("Student Overall Mark: " + " " + uc.overallMark);

System.out.println("Student Final Grade: " + " " + uc.finalGrade);

System.out.println("Student Unit ID: " + " " + uc.unitID);

System.out.println("Student Unit Level: " + " " + uc.levelOfUnit);

}

}

/\*\*

\* convert to string

\* @return enrollment type in string

\*/

@Override

public String toString() {

return this.enrolmentType;

}

/\*\*

\* testing program

\* @param args args

\*/

public static void main(String[] args) {

boolean test = true;

String OUTPUT\_FILENAME = "testing";

Student\_Course student1 = new Student\_Course(50, 50, 100, "john", "yip", 001,"ict110",10);

System.out.println(student1.unitID);

student1.reportGrade(test, OUTPUT\_FILENAME);

}

}

Student research class

package assignment2;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

/\*\*

\* @author yin zhanpeng

\* @version 1.0 (current version number of program)

\* Assignment 2

\* this program have will read the student csv file and have functions that will provide a summary for the students

\*/

public class Student\_Research extends Student {

private final String enrolmentType = "R";

private int student\_ResearchProposalMark;

private int student\_ResearchFinalDissertationMark;

/\*\*

\* constructor

\*/

public Student\_Research() {

}

/\*\*

\* constructor

\* @param student\_ResearchProposalMark proposal mark

\* @param student\_ResearchFinalDissertationMark final mark

\* @param firstName first name

\* @param lastName last name

\* @param studentNumber student number

\*/

public Student\_Research(int student\_ResearchProposalMark, int student\_ResearchFinalDissertationMark, String firstName, String lastName, long studentNumber) {

super(firstName, lastName, studentNumber);

this.student\_ResearchProposalMark = student\_ResearchProposalMark;

this.student\_ResearchFinalDissertationMark = student\_ResearchFinalDissertationMark;

}

/\*\*

\* get enrollment type

\* @return enrollment type

\*/

@Override

public String getEnrolmentType() {

return enrolmentType;

}

/\*\*

\* report the grades of the student

\* @param sorted check if the array list is sorted

\* @param OUTPUT\_FILENAME the file which the array list will output

\*/

@Override

public void reportGrade(boolean sorted, String OUTPUT\_FILENAME) {

Research r = new Research(student\_ResearchProposalMark, student\_ResearchFinalDissertationMark, enrolmentType);

if (sorted) {

try {

//PrintWriter pw = new PrintWriter(OUTPUT\_FILENAME);

File outputFile = new File(OUTPUT\_FILENAME);

boolean append = outputFile.exists() && outputFile.length() > 0;

FileWriter fw = new FileWriter(OUTPUT\_FILENAME, append);

try (PrintWriter pw = new PrintWriter(fw)) {

pw.print("Enrolment Type: " + enrolmentType);

pw.print(" Student Name: " + super.getFirstName() + " " + super.getLastName());

pw.print(" Student Number: " + super.getStudentNumber());

r.overallMarkCalculation();

r.finalGrade = r.finalGradeCalculation(r.overallMark);

pw.print(" Student Overall Mark: " + " " + r.overallMark);

pw.print(" Student Final Grade: " + " " + r.finalGrade);

pw.println("");

}

} catch (IOException ex) {

System.out.println("file error");

}

} else {

//System.out.println("Unit ID: " + r. );

System.out.println("Enrolment Type: " + enrolmentType);

System.out.println("Student Name: " + super.getFirstName() + " " + super.getLastName());

System.out.println("Student Number: " + super.getStudentNumber());

r.overallMarkCalculation();

r.finalGrade = r.finalGradeCalculation(r.overallMark);

System.out.println("Student Overall Mark: " + " " + r.overallMark);

System.out.println("Student Final Grade: " + " " + r.finalGrade);

}

}

/\*\*

\* convert enrollment type to string

\* @return enrollment type

\*/

@Override

public String toString() {

return this.enrolmentType;

}

/\*\*

\* tetsing

\* @param args args

\*/

public static void main(String[] args) {

boolean test = true;

String OUTPUT\_FILENAME = " ";

Student\_Research student = new Student\_Research(50, 100, "john", "yip", 001);

student.reportGrade(test, OUTPUT\_FILENAME);

}

}

Unit class

package assignment2;

/\*\*

\* @author yin zhanpeng

\* @version 1.0 (current version number of program)

\* Assignment 2

\* this program have will read the student csv file and have functions that will provide a summary for the students

\*/

public class Unit {

String enrolmentType; // C course work enrolment R research enrolment

int overallMark;

String finalGrade;

/\*\*

\* constructor

\* @param enrolmentType enrollment type

\*/

public Unit(String enrolmentType) {

this.enrolmentType = enrolmentType;

}

/\*\*

\* get the enrollment type

\* @return enrollment type

\*/

public String getEnrolmentType() {

return enrolmentType;

}

/\*\*

\* display the grades

\*/

public void gradeReporting() {

System.out.println("NA");

}

/\*\*

\* overall mark calculation

\*/

public void overallMarkCalculation() {

Helper.println("no marks to calculate");

}

/\*\*

\* final grade calculation

\* @param overallMarks overall marks

\* @return grades

\*/

public String finalGradeCalculation(int overallMarks ) {

if (overallMarks < 50) {

return "N";

} else if (overallMarks >= 80) {

return "HD";

} else if (overallMarks >= 70) {

return "D";

} else if (overallMarks >= 60) {

return "C";

} else if (overallMarks >= 50) {

return "P";

}

return "error";

}

}

Unit course class

package assignment2;

/\*\*

\* @author yin zhanpeng

\* @version 1.0 (current version number of program)

\* Assignment 2

\* this program have will read the student csv file and have functions that will provide a summary for the students

\*/

public class Unit\_Course extends Unit {

String unitID;

int levelOfUnit;

String enrolmentType = "C";

int assignment1; //max 100

int assignment2; //max 100

int finalExamination; //max 100

int overallMark;

String finalGrade;

/\*\*

\* constructor

\* @param assignment1 assignment 1 mark

\* @param assignment2 assignment 2 mark

\* @param finalExamination final examination mark

\* @param enrolmentType enrollment type

\* @param unitID unit id

\* @param levelOfUnit level of unit

\*/

public Unit\_Course(int assignment1, int assignment2, int finalExamination, String enrolmentType, String unitID, int levelOfUnit ) {

super(enrolmentType);

this.assignment1 = assignment1;

this.assignment2 = assignment2;

this.finalExamination = finalExamination;

this.unitID = unitID;

this.levelOfUnit = levelOfUnit;

}

/\*\*

\* overall mark calculation

\*/

@Override

public void overallMarkCalculation() {

overallMark = overallMark = (int) (assignment1 \* 0.3 + assignment2 \* 0.3 + finalExamination \* 0.4);

}

/\*\*

\* convert to string

\* @return unit id, level of unit, assignment 1, assignment 2, final exam, overall mark, final grade

\*/

public String toString() {

return this.unitID + "," + this.levelOfUnit + "," + this.assignment1

+ "," + this.assignment2 + "," + this.finalExamination

+ "," + this.overallMark + "," + this.finalGrade;

}

//testing

/\*\*

\* testing

\* @param args args

\*/

public static void main(String[] args) {

Unit\_Course uc = new Unit\_Course(50, 50, 100, "C", "ICT333", 3);

uc.finalGradeCalculation(uc.overallMark);

uc.overallMarkCalculation();

System.out.println(uc.overallMark);

System.out.println(uc.finalGrade);

}

}

Research class

package assignment2;

/\*\*

\* @author yin zhanpeng

\* @version 1.0 (current version number of program)

\* Assignment 2

\* this program have will read the student csv file and have functions that will provide a summary for the students

\*/

public class Research extends Unit {

String enrolmentType = "R";

int proposalMark;

int finalDissertationMark;

int overallMark;

String finalGrade;

/\*\*

\* constructor

\* @param enrolmentType enrollment type

\*/

public Research(String enrolmentType) {

super(enrolmentType);

}

/\*\*

\* constructor

\* @param proposalMark proposal mark

\* @param finalDissertationMark final mark

\* @param enrolmentType enrollment type

\*/

public Research(int proposalMark, int finalDissertationMark, String enrolmentType) {

super(enrolmentType);

this.proposalMark = proposalMark;

this.finalDissertationMark = finalDissertationMark;

}

/\*\*

\* coverall mark calculation

\*/

@Override

public void overallMarkCalculation() {

overallMark = (int) (proposalMark \* 0.4 + finalDissertationMark \* 0.6);

}

/\*\*

\* testing

\* @param args args

\*/

public static void main(String[] args) {

Research r = new Research(50, 100, "C");

r.overallMarkCalculation();

r.finalGrade = r.finalGradeCalculation(r.overallMark);

System.out.println(r.overallMark);

System.out.println(r.finalGrade);

}

}

Clients class

package assignment2;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.Scanner;

/\*\*

\* @author yin zhanpeng

\* @version 1.0 (current version number of program)

\* Assignment 2

\* this program have will read the student csv file and have functions that will provide a summary for the students

\*/

public class Client {

private static final ArrayList<Student> students = new ArrayList<>(); // will store the data from csv

//private static String fileName = "student.csv";

private static final String OUTPUTFILE = "programOutput.csv";

private static boolean bubbleStored = false;

private static boolean carrybit = false;

/\*\*

\* constructor

\*/

public Client() {

}

/\*\*

\* the main program

\*/

public static void run() {

printMenuAndGetChoice();

//Student\_Course Sc = new Student\_Course(50,10,100,"fname", "sname", 1 );

}

/\*\*

\* print the menu and get the choice of the user

\*/

public static void printMenuAndGetChoice() {

boolean carryOn = true;

while (carryOn) {

printMenu();

String choice = Helper.getString("please enter only options 1 to 8, 1 is to quit");

switch (choice) {

case "1" -> {

carryOn = false;

Helper.println("Thank you, have a good day. bye bye ");

}

case "2" ->

addMarksInfomationFromCsv();

case "3" ->

removeInfomationById();

case "4" ->

outputAllDetailsOfStudents();

case "5" ->

getNumberOfStudentAboveAndBelowAverageCourseWork();

case "6" ->

checkArrayListByStudentId();

case "7" ->

sortingStudentsById();

case "8" ->

printSortedArrayListToCsv();

default ->

System.out.println("INvalid choice, Please only enter the number 1-8");

}

}

}

/\*\*

\* display the arraylist to output csv file

\*/

public static void printToCsv() {

if (bubbleStored) {

Helper.println("yes");

carrybit = true;

for (Student s : students) {

s.reportGrade(carrybit, OUTPUTFILE);

}

Helper.println("Arraylist have been printed to programOutput.csv");

carrybit = false;

} else {

Helper.println("Arraylist have not been sorted, please sort it before using this function");

}

}

/\*\*

\* using bubble sort algo to sort the array list

\*/

public static void bubbleSort() {

boolean swap = true;

while (swap) {

swap = false;

for (int i = 1; i < students.size(); i++) {

long studentId1 = students.get(i - 1).getStudentNumber();

long studentId2 = students.get(i).getStudentNumber();

if (studentId1 > studentId2) {

Student student1 = students.get(i - 1);

Student student2 = students.get(i);

students.set(i - 1, student2);

students.set(i, student1);

swap = true;

}

}

}

Helper.println("file have been sorted");

}

/\*\*

\* find using student id and display the student

\*/

public static void getIdAndDisplay() {

int id = Helper.getInt("Enter the ID");

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

if (id == students.get(i).getStudentNumber()) {

students.get(i).reportGrade(carrybit, OUTPUTFILE);

break;

}

}

Helper.println("This ID does not exsit");

}

/\*\*

\* calculate the number of student below or above the average mark

\* @param averageMarks average mark

\*/

public static void numberOfStudentGetAboveOrBelowCalculation(int averageMarks) {

int indexAbove = 0;

int indexBelow = 0;

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

if ((students.get(i)).getEnrolmentType().equals("C")) {

if (((Student\_Course) students.get(i)).getOverallMarks() >= averageMarks) {

indexAbove++;

} else {

indexBelow++;

}

}

}

System.out.println(indexAbove + " number of student gets above or equal to the average marks of : " + averageMarks);

System.out.println(indexBelow + " number of student gets below the average marks of : " + averageMarks);

}

/\*\*

\* calculate the average mark

\* @return average mark

\*/

public static int aveageMarksForCourseWorkStudent() {

int averageMarks = 0;

int index = 0;

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

//students.get(i).reportGrade();

if ((students.get(i)).getEnrolmentType().equals("C")) {

//students.get(i).reportGrade();

averageMarks += ((Student\_Course) students.get(i)).getOverallMarks();

index++;

//students.get(i).reportGrade();

}

}

averageMarks = averageMarks / index;

return averageMarks;

}

/\*\*

\* delete the student by id

\*/

public static void deletebyId() {

int inputId = Helper.getInt("Please enter the student ID");

int index;

boolean carryOn = true;

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

if (inputId == (students.get(i).getStudentNumber())) {

index = i;

deleteByIndex(index);

carryOn = false;

}

}

if(carryOn){

Helper.println("Student not found");

}

}

/\*\*

\* confirm the the delete student by id

\* @param index the index of student in array list

\*/

public static void deleteByIndex(int index) {

if (index >= 0) {

System.out.print("Do you want to delete student: " + students.get(index).getFirstName() + " with ID " + students.get(index).getStudentNumber() + "? (y/n) ");

String answer = Helper.getString(" ").toLowerCase();

if (answer.equals("y")) {

System.out.println("Student: " + students.get(index).getFirstName() + " with ID " + students.get(index).getStudentNumber() + " has been deleted.");

students.remove(index);

} else {

System.out.println("Deletion cancelled.");

}

}

}

/\*\*

\* output the data to csv

\* @throws FileNotFoundException file not found

\*/

public static void addDataToArrayList() throws FileNotFoundException {

String inputFileName = Helper.getString("Enter the file name");

File fp = new File(inputFileName);

Scanner file = new Scanner(fp);

while (file.hasNextLine()) {

String line = file.nextLine(); // get the line

//Helper.println(line);

List<String> datas = new ArrayList<>();

datas.addAll(Arrays.asList(line.split(",")));

//System.out.println(datas);

//System.out.println(datas);

if (datas.get(0).equals("C")) {

//System.out.println(datas.get(0));

Student\_Course studentsc = new Student\_Course(Integer.parseInt(datas.get(1)), Integer.parseInt(datas.get(2))

, Integer.parseInt(datas.get(3)), datas.get(4), datas.get(5), Integer.parseInt(datas.get(6))

, datas.get(7), Integer.parseInt(datas.get(8)));

students.add(studentsc);

} else if (datas.get(0).equals("R")) {

//System.out.println(datas.get(0));

Student\_Research studentsr = new Student\_Research(Integer.parseInt(datas.get(1)), Integer.parseInt(datas.get(2))

, datas.get(3), datas.get(4), Integer.parseInt(datas.get(5)));

students.add(studentsr);

}

}

Helper.println("Data have been imported");

//(students.get(0)).reportGrade();

}

/\*\*

\* import the student data from csv file

\*/

public static void addMarksInfomationFromCsv() {

try {

addDataToArrayList();

} catch (FileNotFoundException ex) {

System.out.println("file not found");

}

}

/\*\*

\* remove the student by id

\*/

public static void removeInfomationById() {

deletebyId();

}

/\*\*

\* display all the student infomations in the array list

\*/

public static void outputAllDetailsOfStudents() {

//Helper.println("Testing");

for (Student s : students) {

s.reportGrade(carrybit, OUTPUTFILE);

Helper.println("---------------------------------------------");

}

}

/\*\*

\* get the student marks for course work student

\*/

public static void getNumberOfStudentAboveAndBelowAverageCourseWork() {

try{

numberOfStudentGetAboveOrBelowCalculation(aveageMarksForCourseWorkStudent());

}catch(java.lang.ArithmeticException ex){

Helper.println("Cant divide by 0");

}

}

/\*\*

\* get the student data from array list by id

\*/

public static void checkArrayListByStudentId() {

getIdAndDisplay();

}

/\*\*

\* sort the student by id in the array list

\*/

public static void sortingStudentsById() {

bubbleSort();

bubbleStored = true;

}

/\*\*

\* print array list to csv

\*/

public static void printSortedArrayListToCsv() {

printToCsv();

}

/\*\*

\* print menu

\*/

public static void printMenu() {

Helper.println("1. Quit (exit the program)");

System.out.println();

System.out.println();

Helper.println("""

2. Add (to the ArrayList) from csv. """);

System.out.println();

System.out.println();

Helper.println("""

3. Delete from arraylist by student ID. """);

System.out.println();

System.out.println();

Helper.println("4. Output all details currently held in the ArrayList. ");

System.out.println();

System.out.println();

Helper.println("""

5. Calculate overall marks for course work students. """);

System.out.println();

System.out.println();

Helper.println("""

6. Report grade by student ID. """);

System.out.println();

System.out.println();

Helper.println("""

7. Bubble sort the students in arraylist. """);

System.out.println();

System.out.println();

Helper.println("""

8. Output SORTED arraylist to csv. """);

System.out.println();

System.out.println();

}

}