Title: ICT167 Assignment 2 Documentation

Contents

[1. This documentation is authored by Yang Fengfan (35342323) on April 1, 2024. 1](#_Toc163268230)

[2. Requirements/Specification 2](#_Toc163268231)

[Assumptions and Conditions 2](#_Toc163268232)

[Inheritance 2](#_Toc163268233)

[Polymorphism 2](#_Toc163268234)

[Dynamic binding 2](#_Toc163268235)

[Sorting algorithm 2](#_Toc163268236)

[Handling CSV files 2](#_Toc163268237)

[3. User Guide: 2](#_Toc163268238)

[4. Structure/Design/Algorithm: 3](#_Toc163268239)

[Student class design 3](#_Toc163268240)

[UML class diagram for Student class 3](#_Toc163268241)

[Client program design 3](#_Toc163268242)

[Client program Structure Diagram 3](#_Toc163268243)

[Pseudocode of Client program: 3](#_Toc163268244)

[5. Limitations 3](#_Toc163268245)

[6. Testing 3](#_Toc163268246)

[Test Table 3](#_Toc163268247)

[Results of Program Testing 6](#_Toc163268248)

1. This documentation is authored by Yang Fengfan (35342323) on April 1, 2024.

It is for a Java with Ant application named ICTAssignment2\_3, which is submitted in a zip archive named 35342323.zip. The Client program file name is Client.java, and the Student and Unit classes are in Student.java. Client.java and Student.java are in the 35342323/ICTAssignment2\_3/src/ictassignment2

A screenshot of a computer

Description automatically generated

The CSV files containing the data for Student objects are in the 35342323/ICTAssignment2\_3/ folder

A screenshot of a computer

Description automatically generated

The program prompts for user to select options from a menu, with functionality to add or update student records, display various processed data and output files to a directory based on user input.

# 2. Requirements/Specification

The project is a Java application to manage student records stored in CSV files, featuring a base Student class with private attributes for the student's first name, last name, and ID, alongside essential constructors, getters/setters, a placeholder reportGrade method, and an equals method for ID comparison. Derived classes, Student\_Course and Student\_Research, extend Student, each incorporating specific attributes and overriding reportGrade to display relevant information. The Unit base class, differentiated by enrolment type (C for coursework, R for research), and its subclasses, Unit\_Course and Research, contain attributes and methods for handling course details and grading. The application uses an ArrayList to manage a collection of student objects, supporting functionalities such as sorting by student ID and file I/O operations for data management. The Client program offers functions to add or update entries for Student Records via loading CSV files into the program, display specific or aggregated data, and interact with the user through a menu-driven interface.

## Assumptions and Conditions

The system assumes that a student's identity is uniquely determined by their student number. This is necessary for the equals method in the Student class to work as it compares students based on their student numbers to identify equality.

We also assume a student cannot be both a Coursework student and a Research student. Each student object has one-and-only-one enrolment type, either coursework (C) or research (R), and this attribute determines how their grades are calculated and reported.

Third, we assume that when the data for Coursework and Research students are in the same CSV file, the columns for coursework and research students after the common fields (type, first name, last name, student number) are for different data. For coursework students (C), the following columns are for: unit ID, level, marks for two assignments, and a final exam mark. For research students (R), the columns are for: marks for the proposal and marks for the dissertation.Inheritance

## Inheritance

Inheritance is implemented when defining Student\_Course and Student\_Research classes that extend the Student base class. Similarly, Unit\_Course and Research extend the Unit base class. This structure allows derived classes to inherit properties and methods from their base classes, promoting code reuse and organization. For example, since enrolmentType is private in the Unit class and inherited by Unit\_Course and Research, we use a public getter method to access this field outside of these classes.

// Base class: Unit

class Unit {

private String enrolmentType;

public Unit(String enrolmentType) {

this.enrolmentType = enrolmentType;

}

// Method to report final grade

public void reportFinalGrade() {

System.out.println("NA");

}

}

// Derived class: Unit\_Course

class Unit\_Course extends Unit {

private String unitID;

private int level;

private double assignment1Mark;

private double assignment2Mark;

private double finalExamMark;

public Unit\_Course(String enrolmentType, String unitID, int level, double assignment1Mark, double assignment2Mark, double finalExamMark) {

super(enrolmentType);

this.unitID = unitID;

)

## Polymorphism

Polymorphism is utilized through method overriding (e.g., the reportGrade method in Student\_Course and Student\_Research overrides the method in the Student class). This allows objects of different subclasses to be treated as objects of a superclass and behave differently based on their actual class. There is effective polymorphism in the context of calling reportGrade on a Student reference, which could point to either a Student\_Course or Student\_Research object, demonstrating the principle of "one interface, multiple methods."

The reportFinalGrade() method is overrided in both Unit\_Course and Research to provide specific implementations for reporting the final grade, which is an excellent use of polymorphism. The calculateOverallMark() and calculateFinalGrade() methods in both derived classes tailor the calculation logic to the needs of coursework and research units, respectively.

// Derived class: Student\_Course

class Student\_Course extends Student

{

private Unit\_Course unitCourse; // Reference to a Unit\_Course object

public Student\_Course(String firstName, String lastName, long studentNumber, Unit\_Course unitCourse) {

super(firstName, lastName, studentNumber);

this.unitCourse = unitCourse; // Initialize the Unit\_Course object

}

public Unit\_Course getUnitCourse(){

return this.unitCourse;

}

public String getUnitID()

{

return unitCourse.getUnitID();

}

@Override

public void reportGrade()

{

// Use unitCourse to get the Unit ID, Overall Mark, and Final Grade

System.out.println("C, " + getFirstName() + " " + getLastName() + ", Student Number: " + getStudentNumber() +

", Unit ID: " + unitCourse.getUnitID() + ", Overall Mark: " + unitCourse.calculateOverallMark() +

", Final Grade: " + unitCourse.calculateFinalGrade());

}

## Dynamic binding

Dynamic binding occurs in the overridden reportGrade method. When you call this method on a Student object, Java determines at runtime which specific version of the method to execute based on the object's actual class type. Dynamic binding ensures that the correct method is called for each Student object at runtime, depending on whether it's an instance of Student\_Course or Student\_Research.

## Sorting algorithm

An insertion sort algorithm is implemented within the sortStudentsByID in Client.java. The insertion sort is a simple sorting algorithm that builds the final sorted array (or list) one item at a time.

1. Start from the second element: Assume the first element is part of the sorted list.
2. For each item in the array (or ArrayList), compare it backward with elements in the sorted part (elements before it).
3. Insert the item in the correct position in the sorted part by shifting the larger elements to the right.
4. Repeat until the whole list is sorted.

public static void sortStudentsByID() {

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

Student key = students.get(i);

int j = i - 1;

while (j >= 0 && students.get(j).getStudentNumber() > key.getStudentNumber()) {

students.set(j + 1, students.get(j));

j = j - 1;

}

students.set(j + 1, key);

}

}

## Handling CSV files

CSV file handling is done through the loadStudentsFromCSV and outputSortedStudentsToCSV methods. The program utilizes the BufferedReader and FileWriter classes to read from and write to CSV files, respectively, parsing or formatting the data as needed.

# 3. User Guide:

I am using Apache NetBeans IDE 21. To run the application, Open Project ICTAssignment1 folder and Right-Click Run. Preloaded data of Array N = 10 is in students.csv

# 4. Structure/Design/Algorithm:

## Student class design

At its foundation, the Student base class abstracts the universal traits of students, including their names and a unique student number, encapsulating these fundamental attributes to ensure data integrity and provide a unified interface for student operations. This class is extended by two specialized subclasses: Student\_Course and Student\_Research, each representing students involved in distinct academic tracks—coursework and research, respectively. These subclasses not only inherit the foundational properties from Student but also integrate specific academic attributes through associations with Unit\_Course and Research classes. The Unit class acts as an abstract base for these associations, delineating an enrolment type that signifies the academic path (C for coursework and R for research) and serving as a placeholder for methods to report and calculate grades, encapsulating the concept of an academic unit within the system. Derived from Unit, the Unit\_Course and Research classes embody the specifics of coursework and research endeavors. The Unit\_Course class captures the essence of coursework through attributes such as unit ID, academic level, and marks for assignments and exams, providing methods to calculate and report the overall mark and final grade based on predefined criteria. Conversely, the Research class focuses on research-centric evaluations, storing marks for proposals and dissertations, and implementing tailored calculations for overall marks and grades.

## UML class diagram for Student class

A screenshot of a computer

Description automatically generated

## Client program design

The Client program operates as the central hub for managing student data within an educational system, encapsulating a comprehensive suite of functionalities to load, manage, and report student records from a CSV file. The program begins by loading student data through loadStudentsFromCSV, a method that reads from a specified CSV file, dynamically distinguishing between coursework (C) and research (R) students based on the enrolment type and handling existing records by updating marks or adding new student entries as necessary.

A user-driven menu, presented in displayMenu, offers options to interact with the system, including adding new student marks from a CSV file, removing students by ID, displaying all student records, analyzing coursework students' marks against the average, reporting individual student grades, sorting students by their ID in ascending order through an insertion sort algorithm, and finally, exporting the sorted list back to a CSV file.

Key functionalities like analyzeCourseWorkStudents and reportStudentGrade leverage polymorphism and dynamic binding to calculate and report grades specific to the type of student (coursework or research), utilizing overridden methods in Unit\_Course and Research classes.

## Client program Structure Diagram

A diagram of a company

Description automatically generated

## Pseudocode of Client program:

Class Client

Declare students as ArrayList of Student

Declare scanner as Scanner

Method main

Create an instance of Client named app

Call app's startApplication method

EndMethod

Method startApplication

Initialize scanner

Display welcome message

Call loadStudentsFromCSV method with "students.csv" as argument

Call displayMenu method

EndMethod

Method loadStudentsFromCSV(fileName as String)

Try

Open fileName for reading into a BufferedReader br

While br has next line

Read line and split by "," into data array

If data length is not sufficient, continue to next line

Extract enrolmentType, firstName, lastName, studentNumber from data

Find if student exists using findStudentByNumber method

If enrolmentType is "C"

Extract course-related data

If student exists and is a Student\_Course

Update marks

Else

Create new Student\_Course and add to students

Else If enrolmentType is "R"

Extract research-related data

If student exists and is a Student\_Research

Update marks

Else

Create new Student\_Research and add to students

EndIf

EndWhile

Close br

Catch IOException or NumberFormatException

Display error message

EndTry

EndMethod

Method findStudentByNumber(studentNumber as Long) as Student

For each student in students

If student's number matches studentNumber

Return student

EndIf

EndFor

Return null

EndMethod

Method displayMenu

Declare choice as Integer

Repeat

Display menu options

Set choice to scanner input

Call chooseMenuOption method with choice

Until choice is 6

EndMethod

Method chooseMenuOption(choice as Integer)

Switch choice

Case 1: Exit program

Case 2: Call addStudentFromCSV method

Case 3: Call removeStudent method

Case 4: Call displayAllStudents method

Case 5: Call analyzeCourseWorkStudents method

Case 6: Call reportStudentGrade method

Case 7: Call sortStudentsByID method

Case 8: Call outputSortedStudentsToCSV method

Default: Display "Invalid choice. Please try again."

EndSwitch

EndMethod

Method addStudentFromCSV

Prompt for CSV file name

Read fileName from scanner

Call loadStudentsFromCSV with fileName

EndMethod

Method removeStudent

Prompt for student number

Read studentNumber from scanner

Remove student from students if found, else display not found message

EndMethod

Method displayAllStudents

For each student in students

Call student's reportGrade method

EndFor

EndMethod

Method analyzeCourseWorkStudents

Initialize total, sumOverallMarks to 0

Calculate total and sumOverallMarks for coursework students

Calculate averageOverallMark

Determine counts of students above and below average

Display analysis results

EndMethod

Method reportStudentGrade

Prompt for student number

Read studentNumber from scanner

Find student and call their reportGrade method, else display not found message

EndMethod

Method sortStudentsByID

Perform insertion sort on students based on studentNumber

EndMethod

Method outputSortedStudentsToCSV

Try

Open "sorted\_students.csv" for writing

For each student in students

Write student details to file

EndFor

Close file

Display success message

Catch IOException

Display error message

EndTry

EndMethod

EndClass

## 5. Limitations

The program does not ask user to confirm again before removing records.

It also does not have a method for checking whether the ArrayList is sorted before outputting to sorted\_students.csv

# 6. Testing

## Desk Checking Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test ID | Test description/justification | Actual data for this test | Expected output | Actual desk check result | Desk check outcome |
| 1 | Loading Data from initial CSV file:  Test the program's ability to add Student objects to an empty ArrayList by loading Data from a CSV file with valid data. Ensure that students with both C (coursework) and R (research) enrolment types are correctly loaded, parsed, and added to the students list. | Enter your choice: 4  Data from students.csv:  enrolmentType,firstName,lastName,000000  C,Alan,Turing,100001,CS101,0,0,0,0  R,Ada,Lovelace,200001,0,0  C,Grace,Hopper,100002,CS102,0,0,0,0  R,Donald,Knuth,200002,0,0  C,Margaret,Hamilton,100003,CS103,0,0,0,0  R,Claude,Shannon,200003,0,0  C,Tim,Berners-Lee,100004,WEB101,0,0,0,0  R,Edsger,Dijkstra,200004,0,0  C,Linus,Torvalds,100005,OS102,0,0,0,0  R,Richard,Hamming,200005,0,0 | C, Alan Turing, Student Number: 100001, Unit ID: CS101, Overall Mark: 0.0, Final Grade: N  R, Ada Lovelace, Student Number: 200001, Overall Mark: 0.0, Final Grade: N  C, Grace Hopper, Student Number: 100002, Unit ID: CS102, Overall Mark: 0.0, Final Grade: N  R, Donald Knuth, Student Number: 200002, Overall Mark: 0.0, Final Grade: N  C, Margaret Hamilton, Student Number: 100003, Unit ID: CS103, Overall Mark: 0.0, Final Grade: N  R, Claude Shannon, Student Number: 200003, Overall Mark: 0.0, Final Grade: N  C, Tim Berners-Lee, Student Number: 100004, Unit ID: WEB101, Overall Mark: 0.0, Final Grade: N  R, Edsger Dijkstra, Student Number: 200004, Overall Mark: 0.0, Final Grade: N  C, Linus Torvalds, Student Number: 100005, Unit ID: OS102, Overall Mark: 0.0, Final Grade: N  R, Richard Hamming, Student Number: 200005, Overall Mark: 0.0, Final Grade: N | C, Alan Turing, Student Number: 100001, Unit ID: CS101, Overall Mark: 0.0, Final Grade: N  R, Ada Lovelace, Student Number: 200001, Overall Mark: 0.0, Final Grade: N  C, Grace Hopper, Student Number: 100002, Unit ID: CS102, Overall Mark: 0.0, Final Grade: N  R, Donald Knuth, Student Number: 200002, Overall Mark: 0.0, Final Grade: N  C, Margaret Hamilton, Student Number: 100003, Unit ID: CS103, Overall Mark: 0.0, Final Grade: N  R, Claude Shannon, Student Number: 200003, Overall Mark: 0.0, Final Grade: N  C, Tim Berners-Lee, Student Number: 100004, Unit ID: WEB101, Overall Mark: 0.0, Final Grade: N  R, Edsger Dijkstra, Student Number: 200004, Overall Mark: 0.0, Final Grade: N  C, Linus Torvalds, Student Number: 100005, Unit ID: OS102, Overall Mark: 0.0, Final Grade: N  R, Richard Hamming, Student Number: 200005, Overall Mark: 0.0, Final Grade: N | Pass |
| 2 | Updating Grades of Existing Students from a CSV File: Ensure existing students are updated as intended. | Enter your choice: 2  Enter CSV file name: addgrades.csv  Enter your choice: 4  Data from addgrades.csv:  enrolmentType,firstName,lastName,000000  C,Alan,Turing,100001,CS101,1,75,65,70  C,Grace,Hopper,100002,CS102,2,84,78,81  C,Margaret,Hamilton,100003,CS103,3,92,95,93  C,Tim,Berners-Lee,100004,WEB101,3,50,60,55  C,Linus,Torvalds,100005,OS102,2,100,98,99  R,Ada,Lovelace,200001,60,65  R,Donald,Knuth,200002,70,72  R,Claude,Shannon,200003,85,87  R,Edsger,Dijkstra,200004,40,45  R,Richard,Hamming,200005,95,94 | C, Alan Turing, Student Number: 100001, Unit ID: CS101, Overall Mark: 70.0, Final Grade: D  R, Ada Lovelace, Student Number: 200001, Overall Mark: 63.25, Final Grade: C  C, Grace Hopper, Student Number: 100002, Unit ID: CS102, Overall Mark: 81.0, Final Grade: HD  R, Donald Knuth, Student Number: 200002, Overall Mark: 71.30000000000001, Final Grade: D  C, Margaret Hamilton, Student Number: 100003, Unit ID: CS103, Overall Mark: 93.25, Final Grade: HD  R, Claude Shannon, Student Number: 200003, Overall Mark: 86.3, Final Grade: HD  C, Tim Berners-Lee, Student Number: 100004, Unit ID: WEB101, Overall Mark: 55.0, Final Grade: P  R, Edsger Dijkstra, Student Number: 200004, Overall Mark: 43.25, Final Grade: N  C, Linus Torvalds, Student Number: 100005, Unit ID: OS102, Overall Mark: 99.0, Final Grade: HD  R, Richard Hamming, Student Number: 200005, Overall Mark: 94.35, Final Grade: HD | C, Alan Turing, Student Number: 100001, Unit ID: CS101, Overall Mark: 70.0, Final Grade: D  R, Ada Lovelace, Student Number: 200001, Overall Mark: 63.25, Final Grade: C  C, Grace Hopper, Student Number: 100002, Unit ID: CS102, Overall Mark: 81.0, Final Grade: HD  R, Donald Knuth, Student Number: 200002, Overall Mark: 71.30000000000001, Final Grade: D  C, Margaret Hamilton, Student Number: 100003, Unit ID: CS103, Overall Mark: 93.25, Final Grade: HD  R, Claude Shannon, Student Number: 200003, Overall Mark: 86.3, Final Grade: HD  C, Tim Berners-Lee, Student Number: 100004, Unit ID: WEB101, Overall Mark: 55.0, Final Grade: P  R, Edsger Dijkstra, Student Number: 200004, Overall Mark: 43.25, Final Grade: N  C, Linus Torvalds, Student Number: 100005, Unit ID: OS102, Overall Mark: 99.0, Final Grade: HD  R, Richard Hamming, Student Number: 200005, Overall Mark: 94.35, Final Grade: HD | Pass |
| 3(a) | Sorting Array by StudentID: After sorting, ensure students are listed in ascending order by their ID.  Done after Test 2 | Enter your choice: 7  Enter your choice: 4 | Students sorted by ID.  C, Alan Turing, Student Number: 100001, Unit ID: CS101, Overall Mark: 70.0, Final Grade: D  C, Grace Hopper, Student Number: 100002, Unit ID: CS102, Overall Mark: 81.0, Final Grade: HD  C, Margaret Hamilton, Student Number: 100003, Unit ID: CS103, Overall Mark: 93.25, Final Grade: HD  C, Tim Berners-Lee, Student Number: 100004, Unit ID: WEB101, Overall Mark: 55.0, Final Grade: P  C, Linus Torvalds, Student Number: 100005, Unit ID: OS102, Overall Mark: 99.0, Final Grade: HD  R, Ada Lovelace, Student Number: 200001, Overall Mark: 63.25, Final Grade: C  R, Donald Knuth, Student Number: 200002, Overall Mark: 71.30000000000001, Final Grade: D  R, Claude Shannon, Student Number: 200003, Overall Mark: 86.3, Final Grade: HD  R, Edsger Dijkstra, Student Number: 200004, Overall Mark: 43.25, Final Grade: N  R, Richard Hamming, Student Number: 200005, Overall Mark: 94.35, Final Grade: HD | Students sorted by ID.  C, Alan Turing, Student Number: 100001, Unit ID: CS101, Overall Mark: 70.0, Final Grade: D  C, Grace Hopper, Student Number: 100002, Unit ID: CS102, Overall Mark: 81.0, Final Grade: HD  C, Margaret Hamilton, Student Number: 100003, Unit ID: CS103, Overall Mark: 93.25, Final Grade: HD  C, Tim Berners-Lee, Student Number: 100004, Unit ID: WEB101, Overall Mark: 55.0, Final Grade: P  C, Linus Torvalds, Student Number: 100005, Unit ID: OS102, Overall Mark: 99.0, Final Grade: HD  R, Ada Lovelace, Student Number: 200001, Overall Mark: 63.25, Final Grade: C  R, Donald Knuth, Student Number: 200002, Overall Mark: 71.30000000000001, Final Grade: D  R, Claude Shannon, Student Number: 200003, Overall Mark: 86.3, Final Grade: HD  R, Edsger Dijkstra, Student Number: 200004, Overall Mark: 43.25, Final Grade: N  R, Richard Hamming, Student Number: 200005, Overall Mark: 94.35, Final Grade: HD | Pass |
| 3(b) | Outputting Sorted List to CSV  File Creation: Check if the file sorted\_students.csv is correctly created and contains all students in the sorted order. | Enter your choice: 8  Open sorted\_students.csv in | In sorted\_students.csv:  C,Alan,Turing,100001,CS101  C,Grace,Hopper,100002,CS102  C,Margaret,Hamilton,100003,CS103  C,Tim,Berners-Lee,100004,WEB101  C,Linus,Torvalds,100005,OS102  R,Ada,Lovelace,200001,N/A  R,Donald,Knuth,200002,N/A  R,Claude,Shannon,200003,N/A  R,Edsger,Dijkstra,200004,N/A  R,Richard,Hamming,200005,N/A | In sorted\_students.csv:  C,Alan,Turing,100001,CS101  C,Grace,Hopper,100002,CS102  C,Margaret,Hamilton,100003,CS103  C,Tim,Berners-Lee,100004,WEB101  C,Linus,Torvalds,100005,OS102  R,Ada,Lovelace,200001,N/A  R,Donald,Knuth,200002,N/A  R,Claude,Shannon,200003,N/A  R,Edsger,Dijkstra,200004,N/A  R,Richard,Hamming,200005,N/A | Pass |
| 4 | Menu Navigation: Verify that the menu prompts reappear after an action is completed, when any option except option 1 is selected | Enter your choice: 2  Enter CSV file name: addgrades.csv | Menu:  1. Quit  2. Add student marks information from CSV file  3. Remove student  4. Output all student details  5. Analyze course work student marks  6. Report grade for a student  7. Sort student list by ID  8. Output sorted student list to CSV  Enter your choice: | Menu:  1. Quit  2. Add student marks information from CSV file  3. Remove student  4. Output all student details  5. Analyze course work student marks  6. Report grade for a student  7. Sort student list by ID  8. Output sorted student list to CSV  Enter your choice: | Pass |
| 5 | Input Validation: Adding Students from CSV file that doesn’t exist i.e. file not found. Ensure proper error message pops up when the specified CSV file doesn't exist or can't be opened, and program does not terminate. | Enter your choice: 2  Enter CSV file name: addgradess.csv | Error reading from CSV file: addgradess.csv (The system cannot find the file specified) | Error reading from CSV file: addgradess.csv (The system cannot find the file specified) | Pass |
| 6 | Reporting Student Grades for  Existing Student: Confirm that the grade report for a student by ID displays correctly for both coursework and research students. | Enter your choice: 6  Enter student number to report grade: 200003  Enter your choice: 6  Enter student number to report grade: 100002 | R, Claude Shannon, Student Number: 200003, Overall Mark: 86.3, Final Grade: HD  C, Grace Hopper, Student Number: 100002, Unit ID: CS102, Overall Mark: 81.0, Final Grade: HD | R, Claude Shannon, Student Number: 200003, Overall Mark: 86.3, Final Grade: HD  C, Grace Hopper, Student Number: 100002, Unit ID: CS102, Overall Mark: 81.0, Final Grade: HD | Pass |
| 7 | Reporting Student Grades for  non-existing Student: Ensure the program handles the case where a student ID does not exist | Enter your choice: 6  Enter student number to report grade: 300003 | Student with ID 300003 not found. | Student with ID 300003 not found. | Pass |
| 8 | Analyzing Coursework Students: Verify that the average calculation is correct and that the program correctly identifies students above and below the average. | Enter your choice: 5 | Number of course work students above or equal to average overall mark: 3  Number of course work students below average overall mark: 2 | Number of course work students above or equal to average overall mark: 3  Number of course work students below average overall mark: 2 | Pass |
| 9 | Removing Students  Valid Removal: Test removing a student by ID that exists in the list, ensuring the student is correctly removed.  (Must be done after Test 2, and preceding Test 11) | Enter your choice: 3  Enter student number to remove: 100004  Enter your choice: 3  Enter student number to remove: 100005 | Student removed: Tim Berners-Lee (ID: 100004)  Student removed: Linus Torvalds (ID: 100005) | Student removed: Tim Berners-Lee (ID: 100004)  Student removed: Linus Torvalds (ID: 100005) | Pass |
| 10 | Removing Students  invalid Removal: Test removing a student whose ID does not exist in the list, ensuring program returns appropriate error message but keeps running. | Enter your choice: 3  Enter student number to remove: 2000001 | Student with ID 2000001 not found. | Student with ID 2000001 not found. | Pass |
| 11 | Check whether analysingCourseworkstudents returns updated values after *removing students* by studentID  (Must be done after Test 9) | Enter your choice: 5 | Number of course work students above or equal to average overall mark: 1  Number of course work students below average overall mark: 2 | Number of course work students above or equal to average overall mark: 1  Number of course work students below average overall mark: 2 | Pass |
| 12 | After *removing students*, Check whether options 7 & 8 Sorting Array by StudentID and Outputting Sorted List to CSV reflect the updated ArrayList in the output CSV file. | Enter your choice: 7  Students sorted by ID.  Enter your choice: 8  Sorted students exported to sorted\_students.csv | In sorted\_students-12.csv:  C,Alan,Turing,100001,CS101  C,Grace,Hopper,100002,CS102  C,Margaret,Hamilton,100003,CS103  R,Donald,Knuth,200002,N/A  R,Claude,Shannon,200003,N/A  R,Edsger,Dijkstra,200004,N/A  R,Richard,Hamming,200005,N/A | In sorted\_students-12.csv:  C,Alan,Turing,100001,CS101  C,Grace,Hopper,100002,CS102  C,Margaret,Hamilton,100003,CS103  R,Donald,Knuth,200002,N/A  R,Claude,Shannon,200003,N/A  R,Edsger,Dijkstra,200004,N/A  R,Richard,Hamming,200005,N/A | Pass |
| 13 | Verify that the exit option is triggered only by selection of 1 from menu, not by accidental invalid input. | Enter your choice: 100003 | Invalid choice. Please enter a number from 1 to 8. | Invalid choice. Please enter a number from 1 to 8. | Pass |
| 14 | Verify that the exit option terminates the program. | Enter your choice: 1 |  |  | Pass |

## Results of Program Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Test description/justification | Actual program output | Test run outcome |
| 1 | Loading Data from initial CSV file.  Test the program's ability to add Student objects to an empty ArrayList by loading Data from a CSV file with valid data. |  | Pass |
| 2 | Updating Grades of Existing Students Only from a CSV file.  Ensure existing students are updated as intended. |  | Pass |
| 3(a) | Sorting Array by StudentID: After sorting, ensure students are listed in ascending order by their ID. |  | Pass |
| 3(b) | Outputting Sorted List to CSV  File Creation: Check if the file sorted\_students.csv is correctly created and contains all students in the sorted order. |  |  |
| 4 | Menu Navigation: Verify that the menu prompts reappear after an action is completed, when any option except option 1 is selected. |  | Pass |
| 5 | Input Validation: Adding Students from CSV file that doesn’t exist i.e. file not found. Ensure proper error message pops up when the specified CSV file doesn't exist or can't be opened, and program does not terminate. |  | Pass |
| 6 | Reporting Student Grades for  Existing Student: Confirm that the grade report for a student by ID displays correctly for both coursework and research students. |  | Pass |
| 7 | Reporting Student Grades for  non-existing Student: Ensure the program handles the case where a student ID does not exist |  | Pass |
| 8 | Analyzing Coursework Students: Verify that the average calculation is correct and that the program correctly identifies students above and below the average. |  | Pass |
| 9 | Removing Students  Valid Removal: When removing a student by ID that exists in the list, ensure the student is correctly removed. |  | Pass |
| 10 | Removing Students  invalid Removal: Test removing a student whose ID does not exist in the list, ensuring program returns appropriate error message but keeps running. |  | Pass |
| 11 | Check whether analysingCourseworkstudents returns updated values after removing students by studentID |  | Pass |
| 12 | After *removing students*, Check whether options 7 & 8 Sorting Array by StudentID and Outputting Sorted List to CSV reflect the updated ArrayList in the output CSV file. |  | Pass |
| 13 | Verify that the exit option is triggered only by selection of 1 from menu, not by accidental invalid input. |  | Pass |
| 14 | Verify that the exit option terminates the program. |  | Fail |