Requirements and Specifications

The program is designed to view, manage, and process student data, handling both course work and research students. The program reads an existing .txt file and load an initial set of student data into the system. Thereafter, the program will allow users to add more student data from another .CSV file, remove student data from the system, sort, and display student information within the system. This program supports features such as calculation of the average overall mark of course work student and the amount of student above and below the average overall mark. Also, it sorts students by an ascending ID number, and exports the sorted list into a new .CSV file. The user can interact with the program by entering their commands in the console provided by the IDE.

Assumptions:

1. The initial data set must be in a .txt file.
2. Any additional data sets must be in a .CSV file.
3. The format for the data sets is as follows:
    - Course Work Students:
        - C, FirstName, LastName, ID, DOB, UnitID, AssignmentScore1, AssignmentScore2, LabScore1, … , LabScore12, ExamScore
    - Research Students:
        - R, FirstName, LastName, ID, DOB, ProposalScore, DissertationScore
4. The user is to follow the instructions given by the program as they use it to avoid errors or restarting from the main menu.

Inheritance

I have implemented Inheritance by using subclasses and “extends” in my code.

This allows me to use methods and objects from the main class in my subclasses.

In my code:

- “StudentCourse” and “StudentResearch” inherits from “Student”
- “UnitCourse” and “Research” inherits from “Unit”

Polymorphism

I have implemented Polymorphism by using method overriding in my code.

I override the method of the superclass with the method in the subclass, allowing me to add more specific methods and objects that are private to the respective subclasses.

In my code:

- “reportGrade” method from “StudentCourse” and “StudentResearch” overrides the method from their superclass “Student”

Dynamic Binding

I have implemented Dynamic Binding by utilizing polymorphism and inheritance in my code.

As previously established, “StudentCourse” and “StudentResearch” as subclasses of “Student”, and “reportGrade” method from “StudentCourse” and “StudentResearch” overrides the method from their superclass “Student”. Hence, when calling “reportGrade” on a “Student” reference, the appropriate subclass methods, either “StudentCourse” or “StudentResearch” is executed instead.

Example from my code:

```
public static void outputAllDetails(ArrayList<Student> studentData) {

   for (Student student : studentData) { student.reportGrade(); }

}
```

Sorting Algorithm

The sorting algorithm in my code would first have a for loop so go through the ArrayList and store each student's ID, assigning it is as "current". There is another instance called nextStudent that is used to compare to the "current" student. The nextStudent will have an index of 1 less than the "current" student that it is being compared to. Thereafter, by comparing the 2 IDs the one that is larger will be moved to the right. Thus, creating an ascending order of students.

Handling CSV File

I have implemented .CSV file handling twice in my code. Once to read a .csv file and the other to export an ArrayList as a .csv file. This is evident in my addStudent() and printSortedStudents() method. For the system to read the csv file I have implemented the use of a reader that would go through the data in the .csv file line by line. Afterwards, separating each instance with a "," to indicate that the respective data takes up a single index. Once done, each instance will be assigned to its respective object as indicated in "StudentCourse" and "StudentResearch". Similarly, in my printSortedStudent, instead of a reader, a writer is used. It writes and exports the sorted array into a .csv file using the format established in "StudentCourse" and "StudentResearch" classes for their respective students.

User Guide

Prerequites:

- Java Development Kit (JDK)
- Apache Netbeans IDE 19
- studentRecords.txt (for initial data set)
- additionalStudentRecords.csv (for adding students)
- Java source files:
    1. Client.java
    2. Student.java
    3. StudentCourse.java
    4. StudentResearch.java
    5. Unit.java
    6. UnitCourse.java
    7. Research.java

Accessing:

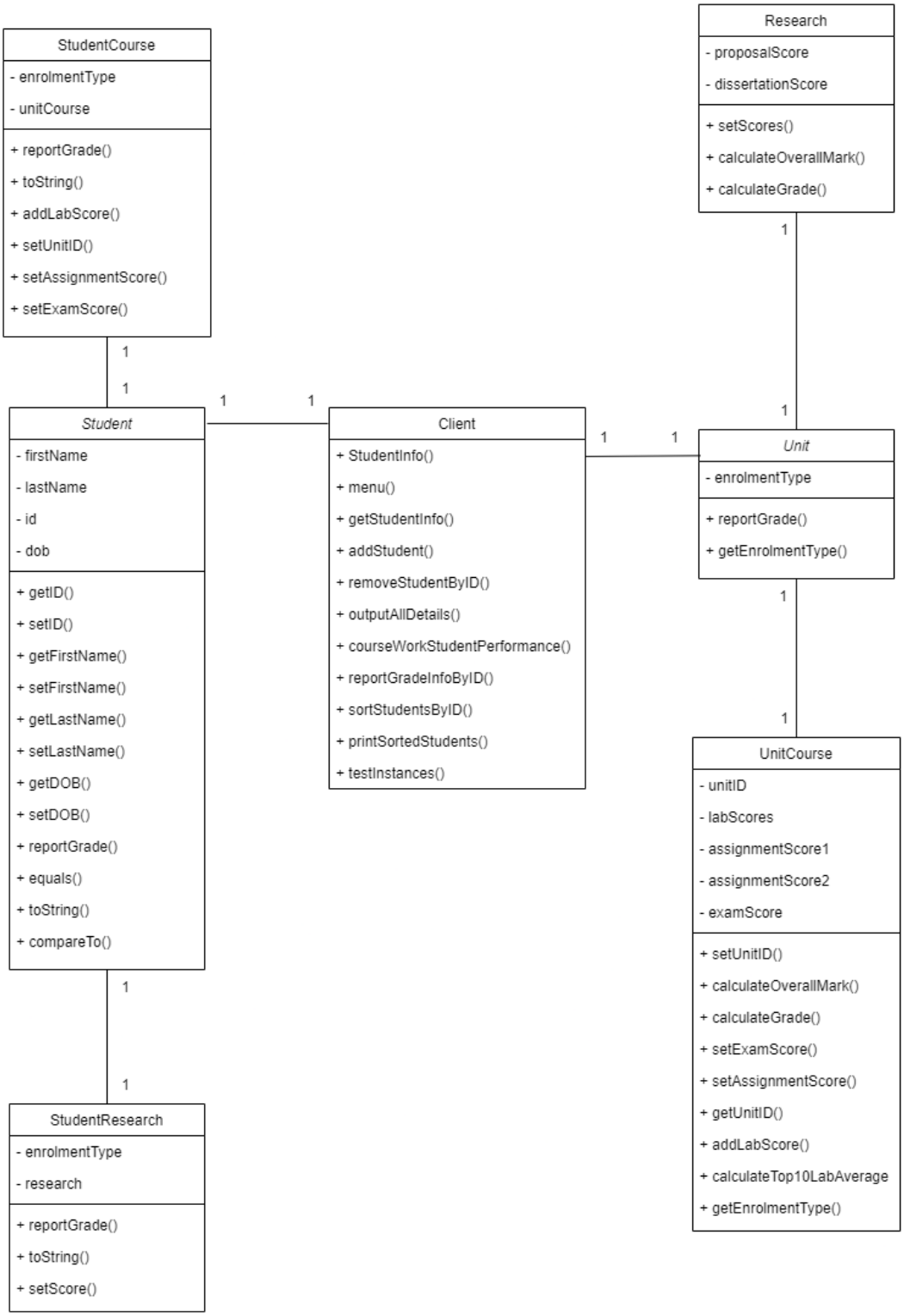
- Open Netbeans IDE 19 and create a new Java Project
- Place the java source files in the Java Project with Client as main class
- Place studentRecords.txt and additionalStudentRecords.csv in the Netbeans document folder

Running the Program:

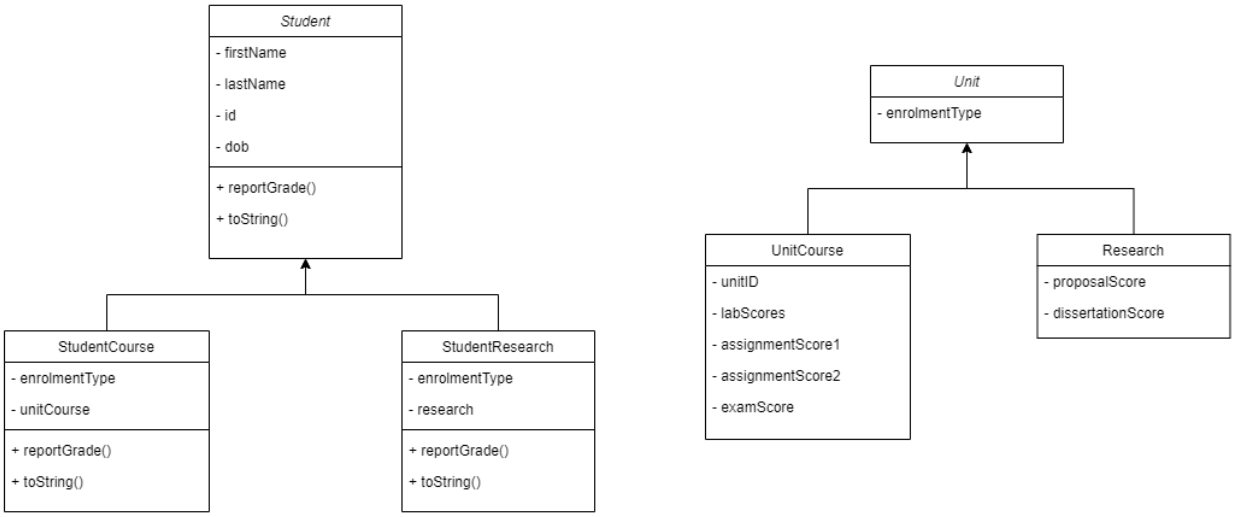
1. Once the source codes and data files are placed in their appropriate locations, run the program in Netbeans IDE 19
2. A console will appear and the user to interact with, and it will display the menu. Input the number (1-8) for the desired functions.
3. Menu Options:
    - 1. Quit the program
    - 2. Add students from another .CSV file
    - 3. Remove student by ID
    - 4. Display all student information
    - 5. Calculate the number of course work students above and below the average overall mark
    - 6. Search for student by ID and display information
    - 7. Sort the students in ascending ID number
    - 8. Export sorted list into a .CSV file

Structure/Design/Algorithm

Structure Diagram



UML Inheritance Diagram



Pseudocode:

<u>"Research" Class</u>

```
public class Research extends Unit
    private proposalScore
    private dissertationScore
    public Research()
        super(enrolmentType)
    setScore()
        this.proposalScore = proposalScore
        this.dissertationScore = dissertationScore
    calculateOverallMark()
        return proposalScore * 0.35 + dissertationScore * 0.65
    reportGrade()
        print(getScore())
```

<u>“Unit” Class</u>

```
public class Unit
    private String enrolmentType
    public Unit(String enrolmentType)
        this.enrolmentType = enrolmentType
    getEnrolmentType()
        return enrolmentType
    reportGrade()
        Print(“NA”)
```

<u>“UnitCourse” Class</u>

```
public class UnitCourse extends Unit
    private unitID
    private ArrayList<Integer> labScores
    private assignmentScore1, assignmentScore1, examScore
    public UnitCourse(String unitID)
        super(enrolmentType)
    setUnitID()
        unitID = unitID
    getUnitID()
        return unitID
    addLabScore()
        labScore++
    setAssignmentScore()
        assignmentScore1 = assignmentScore1
```

```
        assignmentScore2 = assignmentScore2
setExamScore()
        examScore = examScore
calculateTop10LabAverage()
        labScore++ / 10
calculateOverallMark()
        return 0.2 * assignmentScore1 + 0.2 * assignmentScore2 + 0.2 *
        top10LabAvg + 0.4 * examScore;
calculateGrade()
        if (overallMarks > 80)
                return “HD”
if (overallMarks > 70)
        return “D”
if (overallMarks > 60)
        return “C”
if (overallMarks > 50)
        return “P”
else
        return “N”
```

<u>“Student” Class</u>

```
private firstName, lastName, ID, dob
Student(sfirstName, slastName, sID, dob)
        firstName = sfirstname
        lastName = slastName,
        ID = sID
        dob = dob
getID()
        return id
setID()
        this.id = id
getFirstName()
        return firstName
setFirstName(firstName)
        this.firstName = firstName
getLastName()
        lastName
setLastName(lastName)
        this.lastName = lastName
getDOB()
        dob
setDOB(dob)
        this.dob = dob
reportGrade()
        print "There is no grade"
```

```
equals(Object same)
    if this is same
        return true
    if same is null
        return false
    if id equals same.id
        print "There is a duplicate student: " + firstName + " " + lastName
        return true
    return false

compareTo()
    return id.compareTo(other.id)
```

<u>“Student Course” Class</u>

```
public class StudentCourse extends Student
    private enrolment type = “C”
    private UnitCourse
    public Student(firstName, lastName, id, dob, unitID)
        super(firstName, lastName, id, dob)
    addLabScore()
        unitcourse.addLabScore()
        this.dissertationScore = dissertationScore
    setAssignmentScore()
        unitCourse.setAssignmentScore(assignmentScore1, assignmentScore2)
    setExamScore()
        unitCourse.setExamScore(examScore)
    setUnitID()
        unitCourse.setUnitID(unitID)
    getUnitCourse()
        return unitCourse
    reportGrade()
        print(enrolmentType, firstName, lastName, id, dob, unitID, overallMark,
        grade)
```

“Student Research” Class

```
public class StudentResearch extends Student
    private proposalScore
    private dissertationScore
    private enrolmentType

    StudentResearch(FirstName, LastName, ID, DOB)
        super(sFirstName, sLastName, sID, sDOB)
        enrolmentType = "R"
    setScore(proposalScore, dissertationScore)
        this.proposalScore = proposalScore
        this.dissertationScore = dissertationScore
    calculateOverallMark()
        return (proposalScore * 0.35) + (dissertationScore * 0.65)
    calculateGrade(overallMark)
        if overallMark >= 80
            return "HD"
        else if overallMark >= 70
            return "D"
        else if overallMark >= 60
            return "C"
        else if overallMark >= 50
            return "P"
        else
            return "N"
```

```
reportGrade()
        overallMark = research.calculateOverallMark()
        grade = research.calculateGrade(overallMark)
        print(enrolmentType + FirstName + LastName +  sID  + overallMark + grade)
```

<u>“Client” Class</u>

```
main()
        StudentInfo()
        studentData = new ArrayList<Student>
        getStudentData(studentData)
        menu(studentData)



getStudentData(studentData)
        read studentRecords.txt
        while there is a next line in the file
        read oneLine
        split oneLine into values
        if values[0] is "C"
        let student = new StudentCourse with values[1] to values[5]
        add lab scores from values[8] to values[19] to student
        set assignment scores as values[6] and values[7] to student
        set exam score from values[20] to student
        add student to studentData
        else if values[0] is "R"
        let student = new StudentResearch with values[1] to values[4]
        set proposal and dissertation scores as values[5] and values[6] to student
        student to studentData
```

```
menu(studentData)
    while true

        print("Menu:")
        print("1. Quit" )
        print("2. Add student records to new CSV file" )
        print("3. Remove student by ID" )
        print("4. Output all student details" )
        print("5. Calculate number of course work students that are above/below
        average")
        print("6. Display report grade of student by student ID")
        print("7. Sort students by ID")
        print("8. Output sorted list to CSV file")
        read choice
        if choice is 1
            print("Exiting program")
        else if choice is 2
            addStudent(studentData)
        else if choice is 3
            removeStudentByID(studentData)
        else if choice is 4
            outputAllDetails(studentData)
        else if choice is 5
            courseWorkStudentPerformance(studentData)
        else if choice is 6
            reportGradeByID(studentData)
```

```
        else if choice is 7
            sortStudentsByID(studentData)
        else if choice is 8
            printSortedStudents(studentData)

addStudent(studentData)
    print("Enter CSV file name")
    read fileName
    getStudentData(studentData) with filename
    add new student to studentData
    print("New student data added")

removeStudentByID(studentData)
    print("Enter student ID"
    read studentID
    search for student with studentID in studentData
    if student found
        print(student details)
    print("Confirm removal? (yes/no)")
    read confirmation
    if confirmation is "yes"
        remove student from studentData
        print("Student removed")
    else
        print("Removal cancelled")
```

```
outputAllDetails(studentData)
    for each student in studentData
    student.reportGrade()


courseWorkStudentPerformance(studentData)
    declare totalMarks = 0 and courseWorkStudentCount = 0
    for each student IN studentData
        if student is coursework student
            add student's overall mark to totalMarks
            totalMarks++
            courseWorkStudentCount++
    averageMark = totalMarks / courseWorkStudentCount
    declare aboveAverageCount = 0 and belowAverageCount = 0
    for each student in studentData
        if student is coursework student
            if student's overall mark > averageMark
                aboveAverageCount++
            else
                belowAverageCount++
    print(averageMark, aboveAverageCount, and belowAverageCount)
```

```
reportGradeByID(studentData)
    print("Enter student ID")
    read studentID
    search for student with studentID in studentData
    if student found
        student.reportGrade()
    else
        print("Student not found")

sortStudentsByID(studentData)
    for studentIndex from 1 to size of studentData
        set current = studentData[studentIndex]
        set nextStudent = studentIndex - 1
    while nextStudent >= 0 AND studentData[nextStudent].getID() > current.getID()
        shift studentData[nextStudent] to the right
        nextStudent --

printSortedStudents(studentData)
    set outFileName = "SortedStudentsList.csv"
    writer = new PrintWrtier(outFileName)
    foreach student in studentData
        write student details to file
    writer.close
```

```
StudentInfo()
        print(my information)
```

Limitations

1. The getStudentData method uses hardcoded file names which reduces flexibility. Changing file names would require code modification.
2. The ArrayList has a strict format which reduces flexibility. Incorrect placement of data will result in inaccurate data outputs.

Testing

Test Table

Test ID	Test description/justification	Actual data for this test	Expected output	Actual desk check result when desk check is carried out	Desk check outcome – Pass/Fail
1	Test loading of student data from file to ensure that the initial set of data is being read correctly	studentRecords.txt file with valid data	Student data should be loaded into the ArrayList	Student data is loaded correctly into the ArrayList	Pass
2	Test adding a new student from a CSV file to ensure the additional of the new student is possible and accurate	additionalStudentRecords.csv file with valid data	New student data should be added to the ArrayList	New student data is added correctly to the ArrayList	Pass
3	Test removal of student by ID to ensure that the student can found confirmed and removed from the ArrayList	Enter a valid student ID, e.g., "9512"	Student with ID "9512" should be removed from the ArrayList	Student with ID "9512" is removed from the ArrayList	Pass
4	Test sorting of students by ID to ensure it is in ascending order	Existing student data in the ArrayList	Student data should be sorted by ID in ascending order	Student data is sorted by ID in ascending order	Pass
5	Test reporting of grades by student ID to ensure that the student's information can be retrieve correctly	Enter a valid student ID, e.g., "9512"	The grade report for student with ID "9512" should be displayed	Grade report for student with ID "9512" is displayed	Pass
6	Test output of all students' information to ensure that the ArrayList's content are read and displayed correctly	Existing student data in the ArrayList	Student data in ArrayList should be displayed	Student data in ArrayList is correctly displayed	Pass
7	Testing of the calculation and display of number of course work students with overall marks above and below average marks	Existing student data in the ArrayList	The average marks and the number of students above and below the average mark should be displayed	Average mark and number of students are displayed correctly	Pass
8	Testing of export of sorted list of students to ensure that the export is possible and executed correctly	Sorted student data in the ArrayList	A file name "SortedStudentsList" should be exported into the user's computer	Student data is exported to "SortedStudentsList.csv" with the correct information	Pass

Program Table

Test ID	Test description/justification	Actual data for this test	Expected output	Actual program output when test is carried out	Test run outcome – Pass/Fail
1	Test loading of student data from file to ensure that the initial set of data is being read correctly	studentRecords.txt file with valid data	Student data should be loaded into the ArrayList	Student data is loaded correctly into the ArrayList	Pass
2	Test adding a new student from a CSV file to ensure the additional of the new student is possible and accurate	additionalStudentRecords.csv file with valid data	New student data should be added to the ArrayList	New student data is added correctly to the ArrayList	Pass
3	Test removal of student by ID to ensure that the student can found confirmed and removed from the ArrayList	Enter a valid student ID, e.g., "9512"	Student with ID "9512" should be removed from the ArrayList	Student with ID "9512" is removed from the ArrayList	Pass
4	Test sorting of students by ID to ensure it is in ascending order	Existing student data in the ArrayList	Student data should be sorted by ID in ascending order	Student data is sorted by ID in ascending order	Pass
5	Test reporting of grades by student ID to ensure that the student's information can be retrieve correctly	Enter a valid student ID, e.g., "9512"	The grade report for student with ID "9512" should be displayed	Grade report for student with ID "9512" is displayed	Pass
6	Test output of all students' information to ensure that the ArrayList's content are read and displayed correctly	Existing student data in the ArrayList	Student data in ArrayList should be displayed	Student data in ArrayList is correctly displayed	Pass
7	Testing of the calculation and display of number of course work students with overall marks above and below average marks	Existing student data in the ArrayList	The average marks and the number of students above and below the average mark should be displayed	Average mark and number of students are displayed correctly	Pass
8	Testing of export of sorted list of students to ensure that the export is possible and executed correctly	Sorted student data in the ArrayList	A file name "SortedStudentsList" should be exported into the user's computer	Student data is exported to "SortedStudentsList.csv" with the correct information	Pass

Test ID 1: Loading of Student Data

```
Output - ICT167 Assignment 2 (run) #2
run:
Styverson Ng
Kaplan Student Number: CT0372348
Murdoch Student Number: 35427675
ICT167
Siew Cheong Chong
ICT167A
Wednesday 4:15pm to 6:15pm
--------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
4
C Charles Leclerc, ID: 9512, Unit: ICT167, Mark: 81.1, Grade: HD
C Carlos Sainz, ID: 6548, Unit: ICT168, Mark: 43.92, Grade: N
C Max Verstappen, ID: 1485, Unit: ICT269, Mark: 70.58, Grade: D
C Sergio Perez, ID: 9587, Unit: ICT171, Mark: 50.88, Grade: P
C Lando Norris, ID: 2456, Unit: ICT367, Mark: 56.42, Grade: P
R Oscar Piastri, ID: 6582, Mark: 58.0, Grade: P
R Fernando Alonso, ID: 3214, Mark: 87.85, Grade: HD
R Daniel Riccardo, ID: 5968, Mark: 58.8, Grade: P
R Lewis Hamilton, ID: 7845, Mark: 48.95, Grade: N
R Sebastian Vettel, ID: 4862, Mark: 84.9, Grade: HD
------------------------------------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
1
Exiting program
BUILD SUCCESSFUL (total time: 9 seconds)
```

Test ID 2: Addition of New Student from New .CSV File

```
Output - ICT167 Assignment 2 (run) #2
8. Output sorted list to new CSV file
4
C Charles Leclerc, ID: 9512, Unit: ICT167, Mark: 81.1, Grade: HD
C Carlos Sainz, ID: 6548, Unit: ICT168, Mark: 43.92, Grade: N
C Max Verstappen, ID: 1485, Unit: ICT269, Mark: 70.58, Grade: D
C Sergio Perez, ID: 9587, Unit: ICT171, Mark: 50.88, Grade: P
C Lando Norris, ID: 2456, Unit: ICT367, Mark: 56.42, Grade: P
R Oscar Piastri, ID: 6582, Mark: 58.0, Grade: P
R Fernando Alonso, ID: 3214, Mark: 87.85, Grade: HD
R Daniel Riccardo, ID: 5968, Mark: 58.8, Grade: P
R Lewis Hamilton, ID: 7845, Mark: 48.95, Grade: N
R Sebastian Vettel, ID: 4862, Mark: 84.9, Grade: HD
-------------------------------------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
2
Enter the CSV file name to add new students: additionalStudentRecords.csv
Reading line: C,Yuki,Tsunoda,4848,6/8/2004,ICT231,60,60,14,14,15,12,13,14,18,16,18,19,11,12,65
Added Course Student: C Name: Yuki Tsunoda, DOB: 6/8/2004, ID: 4848
Reading line: R,Pierre,Gasly,8888,17/12/1999,20,25,,,,,,,,,,,,,
Added Research Student: R Name: Pierre Gasly, DOB: 17/12/1999, ID: 8888
New student data from additionalStudentRecords.csv has been added.
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
4
C Charles Leclerc, ID: 9512, Unit: ICT167, Mark: 81.1, Grade: HD
C Carlos Sainz, ID: 6548, Unit: ICT168, Mark: 43.92, Grade: N
C Max Verstappen, ID: 1485, Unit: ICT269, Mark: 70.58, Grade: D
C Sergio Perez, ID: 9587, Unit: ICT171, Mark: 50.88, Grade: P
C Lando Norris, ID: 2456, Unit: ICT367, Mark: 56.42, Grade: P
R Oscar Piastri, ID: 6582, Mark: 58.0, Grade: P
R Fernando Alonso, ID: 3214, Mark: 87.85, Grade: HD
R Daniel Riccardo, ID: 5968, Mark: 58.8, Grade: P
R Lewis Hamilton, ID: 7845, Mark: 48.95, Grade: N
R Sebastian Vettel, ID: 4862, Mark: 84.9, Grade: HD
C Yuki Tsunoda, ID: 4848, Unit: ICT231, Mark: 53.06, Grade: P
R Pierre Gasly, ID: 8888, Mark: 23.25, Grade: N
-------------------------------------------------------------
```

Test ID 3: Removal of Student

```
Output - ICT167 Assignment 2 (run)
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
4
C Charles Leclerc, ID: 9512, Unit: ICT167, Mark: 81.1, Grade: HD
C Carlos Sainz, ID: 6548, Unit: ICT168, Mark: 43.92, Grade: N
C Max Verstappen, ID: 1485, Unit: ICT269, Mark: 70.58, Grade: D
C Sergio Perez, ID: 9587, Unit: ICT171, Mark: 50.88, Grade: P
C Lando Norris, ID: 2456, Unit: ICT367, Mark: 56.42, Grade: P
R Oscar Piastri, ID: 6582, Mark: 58.0, Grade: P
R Fernando Alonso, ID: 3214, Mark: 87.85, Grade: HD
R Daniel Riccardo, ID: 5968, Mark: 58.8, Grade: P
R Lewis Hamilton, ID: 7845, Mark: 48.95, Grade: N
R Sebastian Vettel, ID: 4862, Mark: 84.9, Grade: HD
------------------------------------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
3
Enter the student ID to remove: 9512
Student found: C Name: Charles Leclerc, DOB: 05/02/1995, ID: 9512
Are you sure you want to remove this student? (yes/no): yes
Student has been removed.
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
4
C Carlos Sainz, ID: 6548, Unit: ICT168, Mark: 43.92, Grade: N
C Max Verstappen, ID: 1485, Unit: ICT269, Mark: 70.58, Grade: D
C Sergio Perez, ID: 9587, Unit: ICT171, Mark: 50.88, Grade: P
C Lando Norris, ID: 2456, Unit: ICT367, Mark: 56.42, Grade: P
R Oscar Piastri, ID: 6582, Mark: 58.0, Grade: P
R Fernando Alonso, ID: 3214, Mark: 87.85, Grade: HD
R Daniel Riccardo, ID: 5968, Mark: 58.8, Grade: P
R Lewis Hamilton, ID: 7845, Mark: 48.95, Grade: N
R Sebastian Vettel, ID: 4862, Mark: 84.9, Grade: HD
```

Test ID 4: Sort of Students

```
Output - ICT167 Assignment 2 (run)
--------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
4
C Charles Leclerc, ID: 9512, Unit: ICT167, Mark: 81.1, Grade: HD
C Carlos Sainz, ID: 6548, Unit: ICT168, Mark: 43.92, Grade: N
C Max Verstappen, ID: 1485, Unit: ICT269, Mark: 70.58, Grade: D
C Sergio Perez, ID: 9587, Unit: ICT171, Mark: 50.88, Grade: P
C Lando Norris, ID: 2456, Unit: ICT367, Mark: 56.42, Grade: P
R Oscar Piastri, ID: 6582, Mark: 58.0, Grade: P
R Fernando Alonso, ID: 3214, Mark: 87.85, Grade: HD
R Daniel Riccardo, ID: 5968, Mark: 58.8, Grade: P
R Lewis Hamilton, ID: 7845, Mark: 48.95, Grade: N
R Sebastian Vettel, ID: 4862, Mark: 84.9, Grade: HD
--------------------------------------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
7
Students sorted by ID is completed.
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
4
C Max Verstappen, ID: 1485, Unit: ICT269, Mark: 70.58, Grade: D
C Lando Norris, ID: 2456, Unit: ICT367, Mark: 56.42, Grade: P
R Fernando Alonso, ID: 3214, Mark: 87.85, Grade: HD
R Sebastian Vettel, ID: 4862, Mark: 84.9, Grade: HD
R Daniel Riccardo, ID: 5968, Mark: 58.8, Grade: P
C Carlos Sainz, ID: 6548, Unit: ICT168, Mark: 43.92, Grade: N
R Oscar Piastri, ID: 6582, Mark: 58.0, Grade: P
R Lewis Hamilton, ID: 7845, Mark: 48.95, Grade: N
C Charles Leclerc, ID: 9512, Unit: ICT167, Mark: 81.1, Grade: HD
C Sergio Perez, ID: 9587, Unit: ICT171, Mark: 50.88, Grade: P
```

Test ID 5: Grade Report Display

```
Output - ICT167 Assignment 2 (run)
run:
Styverson Ng
Kaplan Student Number: CT0372348
Murdoch Student Number: 35427675
ICT167
Siew Cheong Chong
ICT167A
Wednesday 4:15pm to 6:15pm
---------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
4
C Charles Leclerc, ID: 9512, Unit: ICT167, Mark: 81.1, Grade: HD
C Carlos Sainz, ID: 6548, Unit: ICT168, Mark: 43.92, Grade: N
C Max Verstappen, ID: 1485, Unit: ICT269, Mark: 70.58, Grade: D
C Sergio Perez, ID: 9587, Unit: ICT171, Mark: 50.88, Grade: P
C Lando Norris, ID: 2456, Unit: ICT367, Mark: 56.42, Grade: P
R Oscar Piastri, ID: 6582, Mark: 58.0, Grade: P
R Fernando Alonso, ID: 3214, Mark: 87.85, Grade: HD
R Daniel Riccardo, ID: 5968, Mark: 58.8, Grade: P
R Lewis Hamilton, ID: 7845, Mark: 48.95, Grade: N
R Sebastian Vettel, ID: 4862, Mark: 84.9, Grade: HD
-------------------------------------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
6
Enter the student ID to display information: 9512
C Charles Leclerc, ID: 9512, Unit: ICT167, Mark: 81.1, Grade: HD
```

Test ID 6: Display all Students' Information

```
Output - ICT167 Assignment 2 (run) #2
run:
Styverson Ng
Kaplan Student Number: CT0372348
Murdoch Student Number: 35427675
ICT167
Siew Cheong Chong
ICT167A
Wednesday 4:15pm to 6:15pm
--------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
4
C Charles Leclerc, ID: 9512, Unit: ICT167, Mark: 81.1, Grade: HD
C Carlos Sainz, ID: 6548, Unit: ICT168, Mark: 43.92, Grade: N
C Max Verstappen, ID: 1485, Unit: ICT269, Mark: 70.58, Grade: D
C Sergio Perez, ID: 9587, Unit: ICT171, Mark: 50.88, Grade: P
C Lando Norris, ID: 2456, Unit: ICT367, Mark: 56.42, Grade: P
R Oscar Piastri, ID: 6582, Mark: 58.0, Grade: P
R Fernando Alonso, ID: 3214, Mark: 87.85, Grade: HD
R Daniel Riccardo, ID: 5968, Mark: 58.8, Grade: P
R Lewis Hamilton, ID: 7845, Mark: 48.95, Grade: N
R Sebastian Vettel, ID: 4862, Mark: 84.9, Grade: HD
----------------------------------------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
1
Exiting program
BUILD SUCCESSFUL (total time: 9 seconds)
```

Test ID 7: Calculation of Number of Course Work Student Above/Below Average

```
Output - ICT167 Assignment 2 (run)
run:
Styverson Ng
Kaplan Student Number: CT0372348
Murdoch Student Number: 35427675
ICT167
Siew Cheong Chong
ICT167A
Wednesday 4:15pm to 6:15pm
--------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
5
Average Mark of Coursework Students: 60.58
Number of students above the average: 2
Number of students below the average: 3
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
1
Exiting program
BUILD SUCCESSFUL (total time: 5 seconds)
```

Test ID 8: Exporting of Sorted Student List's .CSV File Part 1

```
Output - ICT167 Assignment 2 (run)
run:
Styverson Ng
Kaplan Student Number: CT0372348
Murdoch Student Number: 35427675
ICT167
Siew Cheong Chong
ICT167A
Wednesday 4:15pm to 6:15pm
----------------------------------
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
7
Students sorted by ID is completed.
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
8
Sorted student data exported to SortedStudentsList.csv
Menu:
1. Quit
2. Add student records to ArrayList from new CSV file
3. Remove student by ID
4. Output all student details
5. Calculate number of course work students that are above/below average
6. Display report grade of student by student ID
7. Sort students by ID (Ascending Order)
8. Output sorted list to new CSV file
1
Exiting program
BUILD SUCCESSFUL (total time: 13 seconds)
```

Test ID 8: Exporting of Sorted Student List's .CSV File Part 2

Name	Date modified	Type	Size
build	7/29/2024 1:11 AM	File folder	
dist	7/29/2024 1:11 AM	File folder	
nbproject	7/29/2024 1:11 AM	File folder	
src	7/29/2024 1:15 AM	File folder	
test	7/29/2024 1:26 AM	File folder	
additionalStudentRecords	7/30/2024 4:43 AM	Microsoft Excel Comma Separated Values File	1 KB
build	7/29/2024 1:11 AM	Microsoft Edge HTML Document	4 KB
manifest.mf	7/29/2024 1:11 AM	MF File	1 KB
SortedStudentsList	7/31/2024 4:14 AM	Microsoft Excel Comma Separated Values File	1 KB
studentRecords	7/30/2024 5:56 AM	Text Document	1 KB