

**School of Engineering and Information Technology**

**ASSIGNMENT COVER SHEET**

* Please complete and insert this form as the first page of EACH electronic assignment.
* Submit the assignment with attached coversheet electronically as per the instructions in the Assignment Question sheet.
* Please make sure you keep a copy of the assignment.

**Student Details**

|  |  |  |  |
| --- | --- | --- | --- |
| Surname | Nguyen Thanh | Given name | Son |
| Student Number | **34468582** | Email | **Thanhh.sonnguyen@gmail.com** |

**Assignment details**

|  |  |  |  |
| --- | --- | --- | --- |
| Unit name | **Principles of Computer Science** | Unit Code | **ICT167** |
| Unit Coordinator |  | Tutor/Tutorial time | Professor Kevin Wong |
| Due date/time | **Midnight, Saturday – 2 April 2022** | Submission date | **Midnight, Saturday – 2 April 2022** |
| Assignment title |  | | |
| Other information |  | | |

***All forms of plagiarism, cheating and unauthorized collusion are regarded seriously by the University and could result in penalties including failure in the unit and possible exclusion from the University. If in doubt, please contact the Unit Coordinator.***

**Student’s Declaration**

Please double click on all the check boxes.

Except where I have indicated, the work I am submitting in this assignment is my own work and has not been submitted for assessment in another unit.



This submission complies with Murdoch University policies regarding plagiarism, cheating and collusion.



I have retained a copy of this assignment for my own records.



**Required External Documentation for each question:**

This is Son Thanh Nguyen's personal assignment about creating a program for loading students infomation from a **CSV file** (Students.csv) and then display the menu created. The client program will allow entry of these data for several different student into an ArrayList and then perform some analysis and queries. The program is completely finished and submitted on 2 April 2022 within a zip file name ASM2\_Klay\_34468582.

The program is supposed to use an ArrayList to store N student objects and in this case the program goes through the trial with 10 student objects. The students information is stored in a Students.csv file with 5 common students and 5 major students. The program is operated by the program user having to enter the following information: 1 Quit (exit the program), 2 add (to the ArrayList) all the marks information about a common year student or a student in the major by reading it from another CSV file, 3 using StudentID to remove the specified student and relevant information from the ArrayList, 4 output all information of student held in the ArrayList., 5 display how many common year students obtained an overall mark equal to or above the average overall mark and how many obtained an overall mark below the average overall mark, 6 use StudentID then report the grade information (using reportGrade) of the student with that number, 7 sort the ArrayList of the student objects into ascending order of the students’ numbers (IDs), and output the sorted array, 8 output the sorted ArrayList from (7) to a CSV file whether the student is common year (C) or student in the major (M).

The version for the app running is IntelliJ IDEA 2021.1.1 (Community Edition) with JDK 16 version 16.0.1

When user runs the program, the menu of 8 selections will be displayed and a “Please select your choice” message will ask you to choose a choice from the menu. If you select option 1, the program termintes immediately and requires you restart the program to show the menu option again. With option 2, you will add the csv file information into ArrayList and user will decide whether to remove the particular student from the ArrayList by providing StudentID for option 3. Input option 4 to output all student details in the ArrayList and option 5 will separate common students with overall grade above the average or below the average and count them. Searching the specific student in the ArrayList by StudentID and display the “Not Found” message when the StudentID is not matched for option 6. At option 7, the program will sort the StudentID in the ArrayList into ascending order and display the sorted one and output the sorted result into OutPut.csv file with option 8.

▪ **Inheritance**: With this program, inheritance is the most important concept of code reusability, this means a Student\_Common class, a Student\_Major class (both are the children from the parent Student class) and Unit\_Common class, Unit\_Major class (both are the children from the parent Unit class). For example, Student\_Common class can use Student class constructor parameter values by calling super(a, b, c) from Student class.

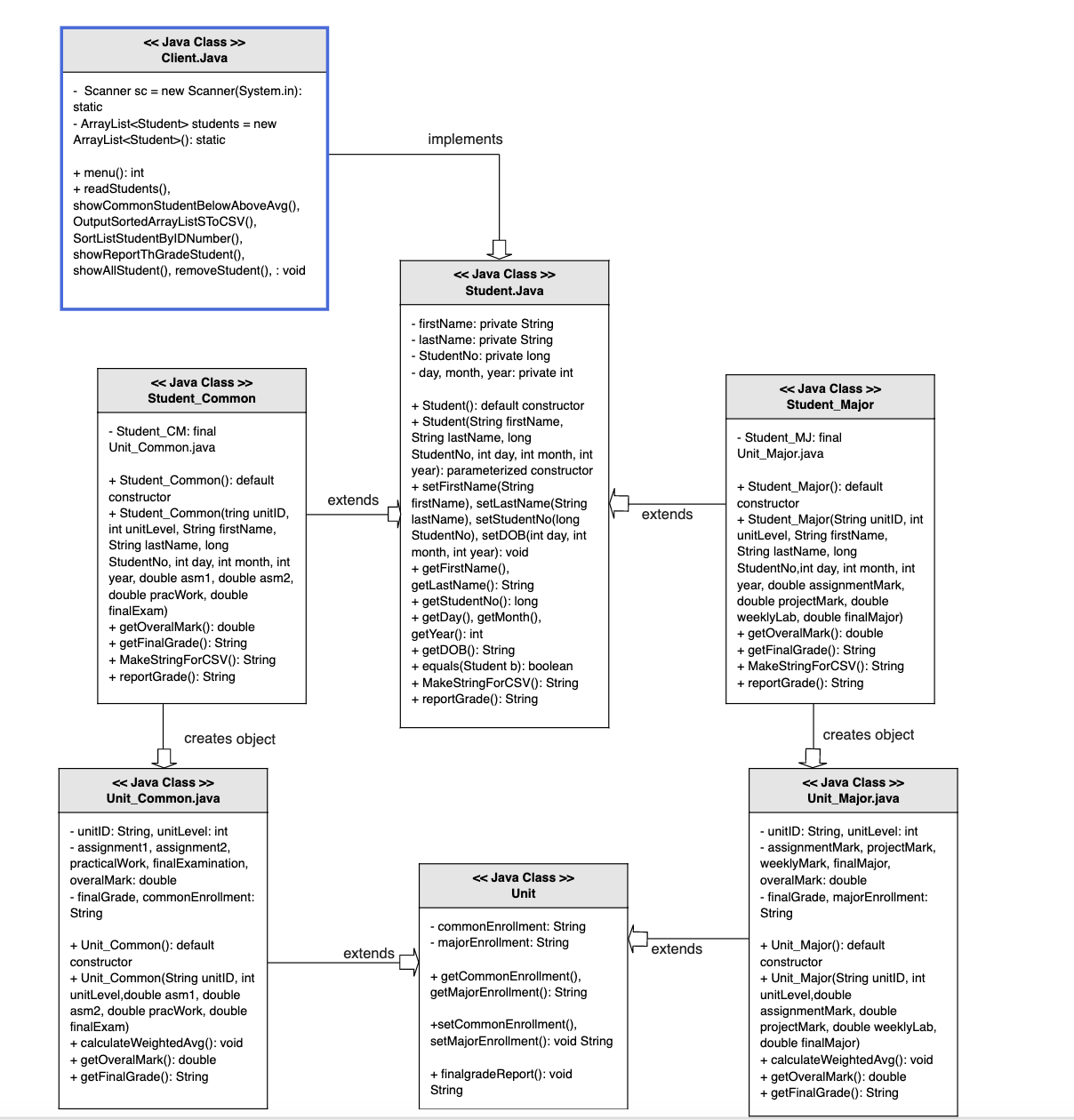
▪ **Polymorphism**: The concept of polymorphism means that the method can have many forms and can operate a single action by different ways. The reportGrade() method is the good example to demonstrate method overriding. When the Object of the Student\_Major class is created, with the reportGrade() method from the Student\_Major class when called will override the parent Student class that has the same name but different form. With the reportGrade() method when be called with the reference Object in Student\_Major class, the method will inherit Student class reportGrade() method values as well as override it.

▪ **Dynamic binding**: Dynamic Binding is the term of runtime polymorphism in Java. The call to an overridden method is resolved dynamically at runtime rather than at build time in this approach. Method Overriding can be used to produce Runtime polymorphism.

▪ **Sorting algorithm**: Sorting refers to a set of techniques for rearranging the places of elements in an array so that they are all in ascending or descending order. Bubble sort uses many swaps. The algorithm keeps looping around the array, changing out elements that aren't in the right place. Swap the values of the two items if the current element in the inner loop is smaller than the next element.

▪ **Handling CSV files**: The Comma-Separated Values (CSV) file stands for Comma-Separated Values. It's a simple plain-text file format for storing tabular data in columns in simple text forms like spreadsheets or databases, and splitting it using a separator. The most common separator used to split the data is commas (,). There are 4 ways to read CSV file: Java Scanner class, Java Bufferred Reader class, Java String.split() method and openCSV API. This program uses Java Bufferred Reader class with the syntax BufferredReader name = new BufferredReader(new FileReader(fileName)).

**UML diagram:**



The Structure UML Class Diagram above demonstrates the relationship between superclasses and subclasses in Object Oriented Design with inheritance. Unit\_Common and Unit\_Major inherit Unit attributes and the same way goes to Student\_Common and Student\_Major with Student parent class. In addition, Student\_Common and Student\_Major also create objects from Unit\_Common() and Unit\_Major() and assign those objects with Student\_CM and Student\_MJ variables in order to use the parameter values in the object created constructor.

**Student.java pseudocode:**

**Problem 1**: Create a default constructor with default local variables and parameterized constructor with approriate values pass in. Acessors and mutators are implemented to get and set the value of the attribute. An equal method is required to check both name and StudentID. A reportGrade method to display the student details (name, DOB, mark, final grade). Finally, MakeStringForCSV method is created to add student information into the csv file later in use.

**Solution**: For the default constructor, use this keywork to set the instance variable with quotation String for the name and day, month, year, StudentID are defined by interger data type with 0.

The parameterized constructor has all the required parameter values and matched with the instance variables of the Student class by this keywork.

For the equal method, if the parameter object Student b equals this keywork instance variable of the Student class will return true or else return false.

**Algorithms**:

Student() {

Name = “”

Day = month = day = StudentID = 0

}

Student(name, DOB, StudentID) {

This.name = name

This.DOB = DOB

This.StudentID = StudentID

}

Equals( Student b) {

If ( b = = this )

Return true;

Else {

Return false;

}

**Problem 2:** MakeStringCSV will have String data type and return StudentID, Name, DOB and then this will be called to read student information in csv file.

**Solution**: Create a String method name MakeStringCSV and return firstName, StudentNo and getDOB() method.

**Algorithms:**

String MakeStringCSV() {

return firstName + "," + lastName + "," + StudentNo + "," + getDOB()

}

**Problem 3:** Display student details with reportGrade() method

**Solution**: Use acessors to get all the information needed to display the message.

**Algorithms:**

String reportGrade()

{

Show = getFirstName() + getLastName() + getStudentNo() + getDOB() ;

return Show;

}

**Student\_Common.java:**

**Problem 1:** Create an Student\_CM variable with Unit\_Common class and final keywork.

**Solution**: To specify this and only class can call and use the object and cannot be overriden by any subclasses. Use Student\_Common constructor and assign the Student\_CM to the Unit\_Common() constructor with values within Student\_Common constructor.

**Algorithms:**

**Final** Unit\_Common **Student\_CM**;

Student\_Common() {

super();

Student\_CM = new Unit\_Common();

Student\_Common() {

super( firstName, lastName, StudentNo, day, month, year);

Student\_CM = new Unit\_Common(unitID, unitLevel, asm1, asm2, pracWork, finalExam;

**Problem 2:** Output the Student\_Common details method

**Solution:** By overriding from the Student class (parent class), Student\_Common class can display more details about Unit Level, Assignment 1, Assignment 2, Practical Work, Final Examination, Weighted Average, Grade.

**Algorithms:**

reportGrade() {

showGrade = super.reportGrade();

showGrade += Student\_CM.unitID + (Student\_CM.unitLevel==1?"C":"M") + Student\_CM.assignment1 +

Student\_CM.assignment2 +

Student\_CM.practicalWork +

Student\_CM.finalExamination +

getOveralMark()+

getFinalGrade;

return showGrade;

}

**Problem 2:** Create MakeStringCSV method to and output the Student Common

into the TestResult.csv

**Solution:** By overriding from the Student class (parent class), Student\_Common class can display more details about Unit Level, Assignment 1, Assignment 2, Practical Work, Final Examination, Weighted Average, Grade into the CSV file

**Algorithms:**

String MakeStringCSV() {

return Student\_CM.Unit Level + super.MakeStringCSV() + Student\_CM.firstName + Student\_CM.lastName + Student\_CM.StudentNo + Student\_CM.getDOB() + Student\_CM.Assignment1 + Student\_CM.Assignment2 + Student\_CM.PracticalWork + Student\_CM.finalExamination + Student\_CM.getOverrallMark() + Student\_CM.getfinalGrade()

}

**Student\_Major.java:**

**Problem 1:** Create an Student\_MJ variable with Unit\_Major and final keywork.

**Solution**: To specify this and only class can call and use the object and cannot be overriden by any subclasses. Use Student\_Major constructor and assign the Student\_MJ to the Unit\_Major() constructor with values within Student\_Major constructor.

**Algorithms:**

**Final** Unit\_Major **Student\_MJ**;

Student\_Major() {

super();

Student\_MJ = new Unit\_Major();

Student\_Major() {

super( firstName, lastName, StudentNo, day, month, year);

Student\_MJ = new Unit\_Major(unitID, unitLevel, assignmentMark, projectMark, weeklyLab, finalMajor)

**Problem 2:** Output the Student\_Major details method

**Solution:** By overriding from the Student class (parent class), Student\_Major class can display more details about Unit Level, Assignment, Project, Weekly Lab, Final Examination, Weighted Average, Grade.

**Algorithms:**

reportGrade() {

showGrade = super.reportGrade();

showGrade += Student\_CM.unitID + (Student\_MJ.unitLevel<1?"C":"M") + Student\_MJ.assignment1 +

Student\_MJ.assignment2 +

Student\_MJ.practicalWork +

Student\_MJ.finalExamination +

getOveralMark()+

getFinalGrade;

return showGrade;

}

**Problem 2:** Create MakeStringCSV method to and output the Student Major

into the TestResult.csv

**Solution:** By overriding from the Student class (parent class), Student\_Major class can display more details about Unit Level, Assignment, Project, Weekly Lab, Final Examination, Weighted Average, Grade into the CSV file.

**Algorithms:**

String MakeStringCSV() {

return Student\_MJ.Unit Level + super.MakeStringCSV() + Student\_MJ.firstName + Student\_MJ.lastName + Student\_MJ.StudentNo + Student\_MJ.getDOB() + Student\_MJ.Assignment1 + Student\_MJ.Assignment2 + Student\_MJ.PracticalWork + Student\_MJ.finalExamination + Student\_MJ.getOverrallMark() + Student\_MJ.getfinalGrade()

}

**Unit.java:**

**Problem:** This class is created in order to get the common enrollment and major enroll and display “NA” message for finalGrade method if there is no grade.

**Solution:** Use accessor and mutator methods to get “C” for common enrollment and “M” for major enrollment and display “NA” for finalGrade without any grade exists.

**Algorithms:**

public String getCommonEnrollment() {

return commonEnrollment;

}

public String getMajorEnrollment() {

return majorEnrollment;

}

public void setCommonEnrollment(){

this.commonEnrollment = "C";

}

public void setMajorEnrollment() {

this.majorEnrollment = "M";

}

public String finalgradeReport() {

return "NA";

}

**Unit\_Common.java:**

There are two constructors within this class, a default constructor and parameterized constructor with set of parameter values are matched up with Common Student.

**Problem**: To get the overrall mark and final grade of the Common Student, calculating the average mark is required. After the average mark is calculated, we will output final grade in characters.

**Solution**: Calculate overrall mark by adding all of the mark from Assignment 1, 2 (take 60%), practical work (15%) and final examination mark (25%). Then, a method to return the overrall mark and a method to display final grade.

**Algorithms:**

calculateWeightedAvg() {

overrallMark += (Assignment 1 \* 0.3) + (Assignment \* 0.3) + (practicalWork \* 0.15) + (finalExamination \* 0.25)

if

overralMark > = 80.0

display finalGrade “HD”

overralMark > = 70.0

display finalGrade “D”

overralMark > = 60.0

display finalGrade “C”

overralMark > = 50.0

display finalGrade “P”

overralMark < 50.0

display finalGrade “N”

else

display finalGrade “NA”

getOverralMark() {

calculateWeightedAvg()

return overralMakr;

}

getfinalGrade() {

calculateWeightedAvg()

return finalGrade

}

**Unit\_Major.java:**

There are two constructors within this class, a default constructor and parameterized constructor with set of parameter values are matched up with Major Student.

**Problem**: To get the overrall mark and final grade of the Major Student, calculating the average mark is required. After the average mark is calculated, we will output final grade in grade characters.

**Solution**: Calculate overrall mark by adding all of the mark from Assignment (take 15%), Project (35%), weekly lab mark (10%) and final examination major mark (40%). Then, a method to return the overrall mark and a method to display final grade.

**Algorithms:**

calculateWeightedAvg() {

overrallMark += (Assignment \* 0.15) + (Project \* 0.35) + (Weekly Lab \* 0.10) + (FinalMajor \* 0.40)

if

overralMark > = 80.0

display finalGrade “HD”

overralMark > = 70.0

display finalGrade “D”

overralMark > = 60.0

display finalGrade “C”

overralMark > = 50.0

display finalGrade “P”

overralMark < 50.0

display finalGrade “N”

else

display finalGrade “NA”

getOverralMark() {

calculateWeightedAvg()

return overralMark;

}

getfinalGrade() {

calculateWeightedAvg()

return finalGrade

}

**Client.java:**

The Client class requires to import BufferredReader, BufferredWriter that scans text, buffering huge reads at a moment for performance, and saving what isn't needed right away in memory for later use. Also, FileNotFoundException, File, Scanner, ArrayList are imported into the class to implement the program in order to run.

**Problem**: Create a static Scanner object “sc” and ArrayList<Students> object “students”, a menu() option to display option from 1 to 8, a readStudents() method to read student information from Students.CSV file, a showCommonStudentBelowAboveAvg() to count common student below or above the average, a OutputSortedArrayListSToCSV() method to output the sorted CSV, a SortListStudentByIDNumber() method to sort the CSV file by StudentID, a showAllStudent() method to display all student details in csv file and finally a removeStudent() to remove a specific student in the CSV by StudentID. A main method will have a switch case and pass all of the method into each case, then run the program.

**Solution:** About readStudents() method, we will use the BufferredReader object “br” and pass the FileReader into BufferredReader object “br” with Students.csv file. After this we will split each String value index in the Array by the comma “,”. Create a variable “st” for each Student in the ArrayList. We will parse each value index from “st” variable into approriate data types since csv file will recognize it as defaul String data type. Next, we assign “st” variable to Student\_Common or Student\_Major Object with approriate values above. Finally, we add all of the students in the ArrayList. This method is under try catch exception.

**Algorithms:**

readStudents() {

try { BufferedReader br = new BufferedReader(new FileReader("Students.csv"))

String line;

String[] values = line.split(“,”)

Student st;

Name = values[0]

StudentNo = Long.parseLong(values[1])

Day = Int.parseLong(values[2])

month = Int.parseLong(values[3])

year = Int.parseLong(values[4])

if values[0] = = ("C")

asm1 = Double.parseDouble(values[5])

asm2 = Double.parseDouble(values[6])

practicalWork = Double.parseDouble(values[7])

finalExam = Double.parseDouble(values[8])

st = new Student\_Common(Name,StudentNo,Day,Month,Year,asm1,asm2,pracWork,finalExam)

else

asm = Double.parseDouble(values[5])

project = Double.parseDouble(values[6])

weeklyLab = Double.parseDouble(values[7])

finalMajorExam = Double.parseDouble(values[8])

st = new Student\_Major(Name,StudentNo,Day,Month,Year,asm,project,weeklyLA ,finalMajorExam)

} catch FileNotFound

Catch IOException

**Problem:** the showCommonStudentBelowAboveAvg() will initialise CountBelowAVG and CountAboveAVG with int datatype. Then we will use for-each loop, in the ArrayList Student data type each student in students ArrayList variable, if each student in the Array equal Student\_Common class, we will count the overrall mark by using getOverralMark() method and count the above and below grades and pass it into CountBelowAVG and CountAboveAVG.

**Algorithms:**

showCommonStudentBelowAboveAvg() {

int CountBelowAVG = 0

int CountAboveAVG = 0

for (Student student : students)

if student.getclass() equals Student\_Common class

st = (Student\_Common) student

if getoverralMark() > 50.0

CountAboveAVG++

Else

CountBelowAVG++

**Problem:** the SortListStudentByIDNumber() is created to sort the StudentID by ascending order. I use Bubble Sort algorithm for this method and this takes the array as the input to be sorted, student at index i, and the size of the array. The second step is to create an outer for loop which will iterate over each element of the array. The third step is to create an inner for loop as shown on line 7. This for loop starts from the first element of the array till the second last index, (size - i - 1). The largest element for that iteration reaches the end each time one index less than the last is scanned as at the end of each iteration. If the element on the left is greater than the element on the right, this is checked. If that's the case, the two items are switched.

**Algorithms:**

SortListStudentByIDNumber() {

For int = 0; I < students.size() – i – 1; i++

For int j = i + 1; j < students.size(); j++

If students.get(i).getStudentID() > students.get(j).getStudentID()

Var temp = students.get(i)

Students.set(i, students.get(j))

Students.set(j, temp)

**Problem:**  the OutputSortedArrayListSToCSV() method will use File library that is already imported and create csv file name TestResult.csv and use FileWrite variable “fw” to write data into the csv file and then use try catch to handle Input Output exception. Finally output the TestResult.csv file with sorted arraylist Students.csv.

**Algorithms**:

OutputSortedArrayListSToCSV() {

File file = new File(TestResult.csv)

FileWriter fw;

Try

Fw = new FileWriter(file)

Bw = new BufferredWriter(fw)

For each Student student : students {

Bw.write(student.MakeStringForCSV())

Bw.newLine;

Bw.close

Fw.close

Catch IOExcetpion

**Problem**: the removeStudent() and showReportThGradeStudent() method swill prompt StudentID and use for each loop to determine each student in the ArrayList is matched with the StudentID, then remove that student out of the ArrayList and show the student details.

**Algorithms**:

removeStudent() {

prompt StudentID

Assign StudentRemove = new Student()

Int index = 0;

For Student student : students

Student.getID() = StudentID

StudentRemove.remove(index)

Index++

}

showReportThGradeStudent() {

prompt StudentID

Assign st = new Student()

Int index = 0;

For Student student : students

Student.getID() = StudentID

St = student;

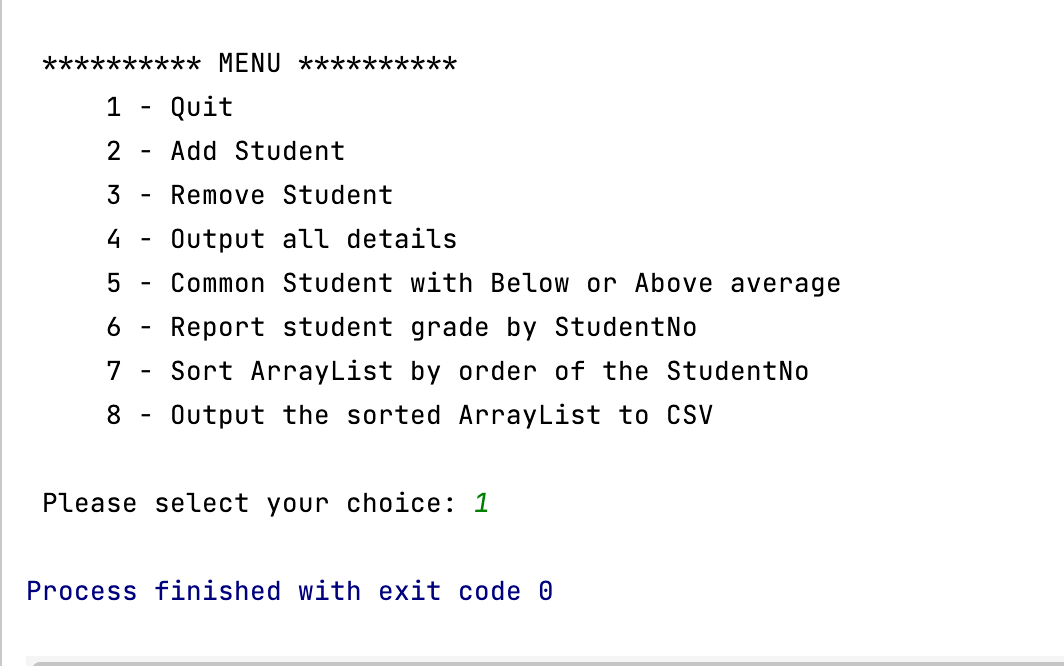
Index++

}

**Limitations**:

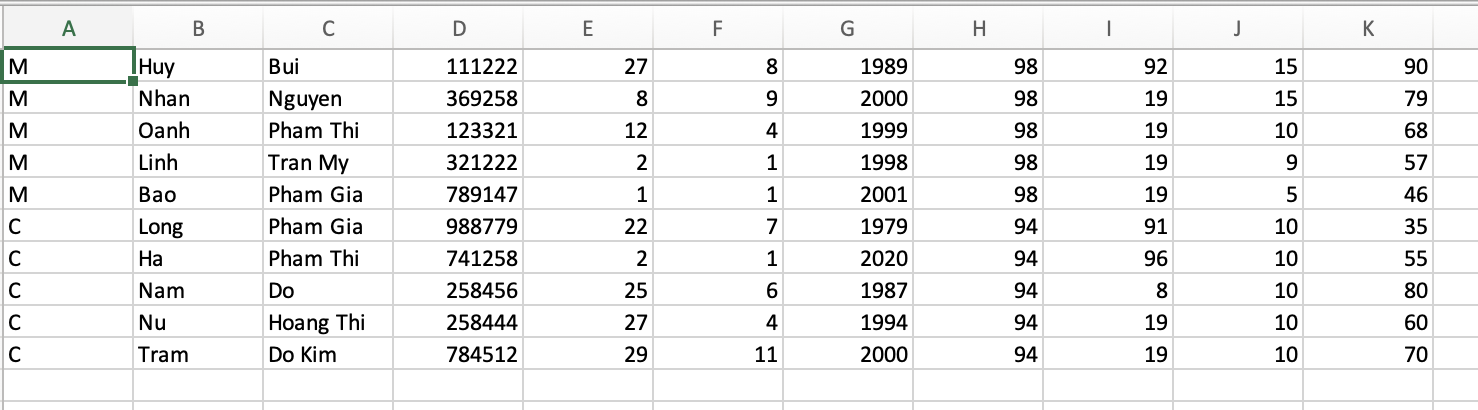
**Testing**:   
Before releasing a software, programmers can perform a desk check to ensure that the code and algorithm logic are correct. This enables them to detect problems that may prohibit a program from functioning properly.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Testid | Test description/justification– what is the test for and whythis particular test | Actualdata forthis test | Expectedoutput | Actualprogramoutputwhen test iscarried out | Test runoutcome –Pass/Fail |
| 1 | If you want to exit the program, enter 1 from the keyboard | Select option 1 by pressing number 1 | Terminate the program | Terminate the program | Pass |
| 2 | Input file Students.csv into project .check data have been into ArrayList yet? | When calling readStudents() with case 2 function will proceed to load data into ArrayList<Student> students and use case 4 to print data just import,for print to check we was read by method “readStudents()” | Data import same:  Reading components include and enter number 4 from keyboard to print  All students were read by method “readStudents()” | Data import successfully  for check data we call method “showAllStudent()” to print all students | Pass |
| 3 | Input Id Student and Remove this student from ArrayList | Input Id from keyboard with  Select option 3:  + Call funcion:  removeStudent()  and input Id Number: 258444  if Id input exists in ArrayList then will be input confirm again if comfirn Y or Yes. it’s deleted outside ArrayList and print:  First Name  Last Name:  Student Code ID:  If choose N or other input. It isn’t deleted. |  | If Id input Id: 258444  Check Id number exist in ArrayList .  + Show confirm and input yes. And choose option 3 if you want to check ArrayList again | Pass |
| 4 | How many student Common have AVG below and above in ArrayList | Input from keyboard with  Select option 5:  Call function  +showCommonStudentBelowAboveAvg();  + This function:   * Check is Student Common * Count Below AVG and Abover AVG.   This out Count number student common AVG below and AVG above. |  | print Count of Students have AVG below 50 and students have AVG above AVG.  + Create 2 variable to increase if its true with case | Pass |
| 5 | Show Report Grade Student by Id. | Input Id from keyboard with Select option 6:  +Find StudentId match with Student in ArrayList;  + Get data into variable was reated | Show on display : | Print student data in ArrayList by method reportGrade()  + Overide method reportGrade() for each type Student with the correct data fields. | Pass |
| 6 | Sort to List Student By ID Number using Bubble Sort | Input Id from keyboard with Select option 7:  +Swap list if Id of Studenti > Id of Studentj  After sort we can print all student to check. |  | Call function SortListStudentByIDNumber() when we choose option 7:  +Using function showAllStudent() to check again  + Using bubble to sort ArrayList | Pass |
| 7 | Output Sorted ArrayListS To CSV | Input from keyboard with  Select option 8:  Call function  +OutputSortedArrayListSToCSV ();  + Generate data from ArrayList into file “TestResult.csv” |  | The data was inserted to TestResult.csv:  + Custom method MakeStringForCSV() to export data into TestResult.csv | Pass |

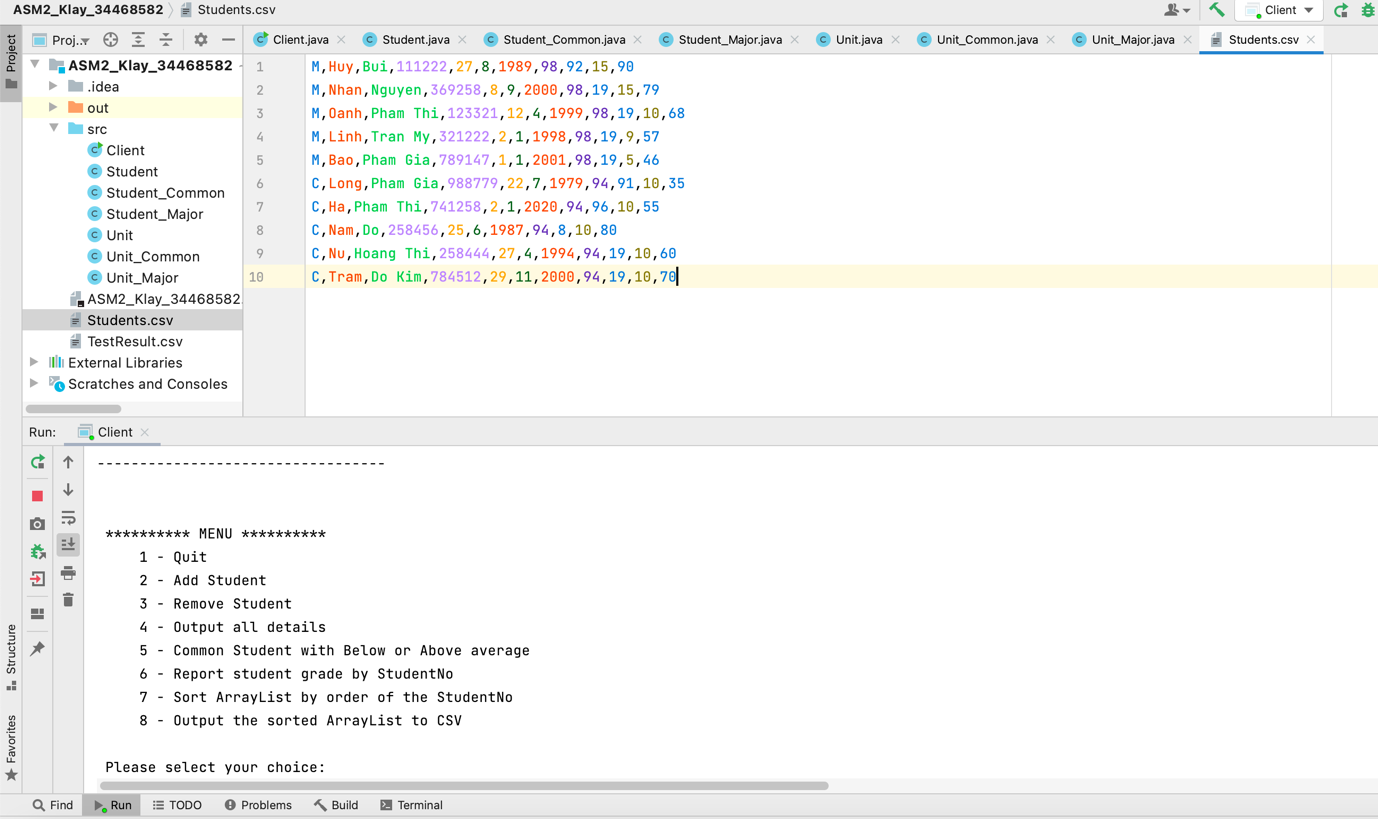
Test 1: System exit the program

Test 2: Input file Students.csv into the program and check data if it has been added into ArrayList yet?

+ Data need import to project



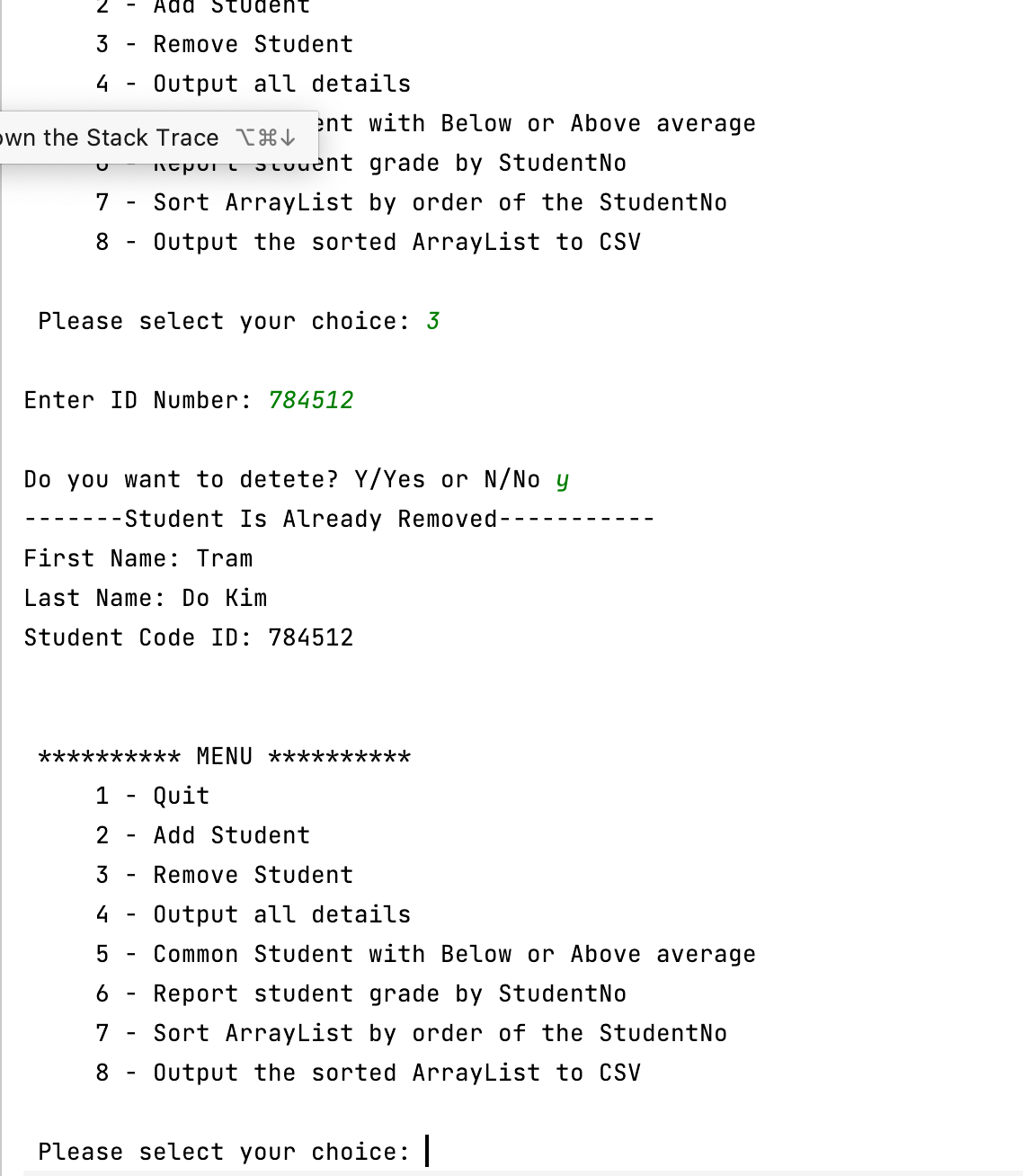
+ Data was imported to project



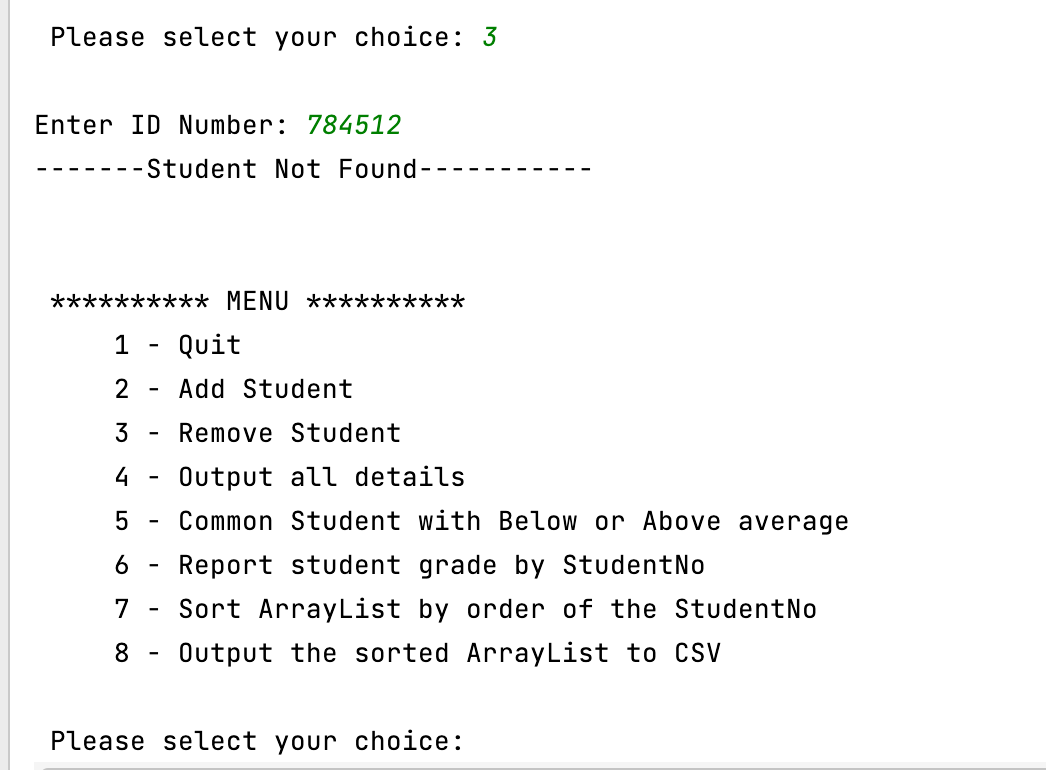
Test 3: Input Id Student and Remove this student from ArrayList

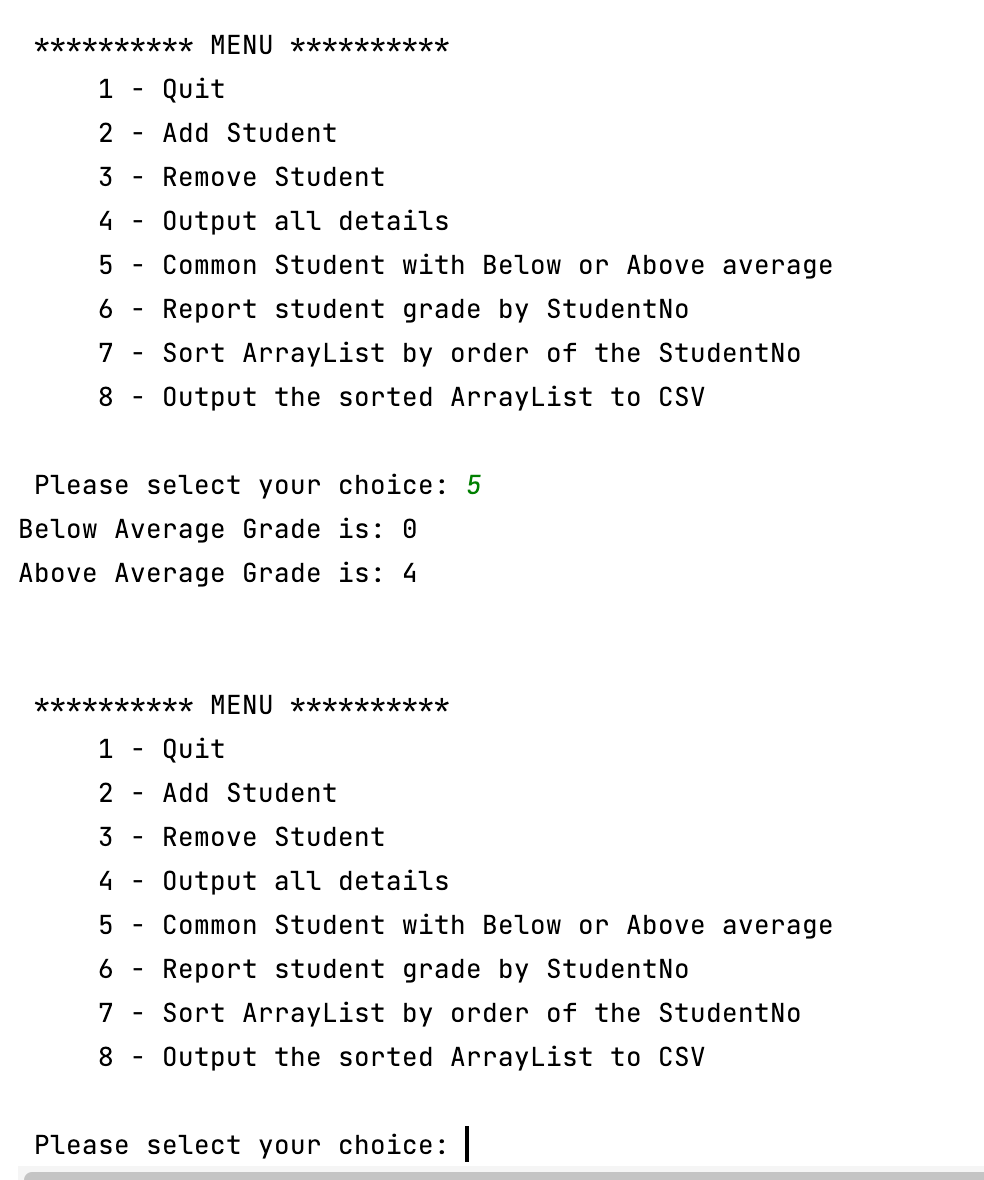
Exist Id 784512:

+ student was Deteled and print information student



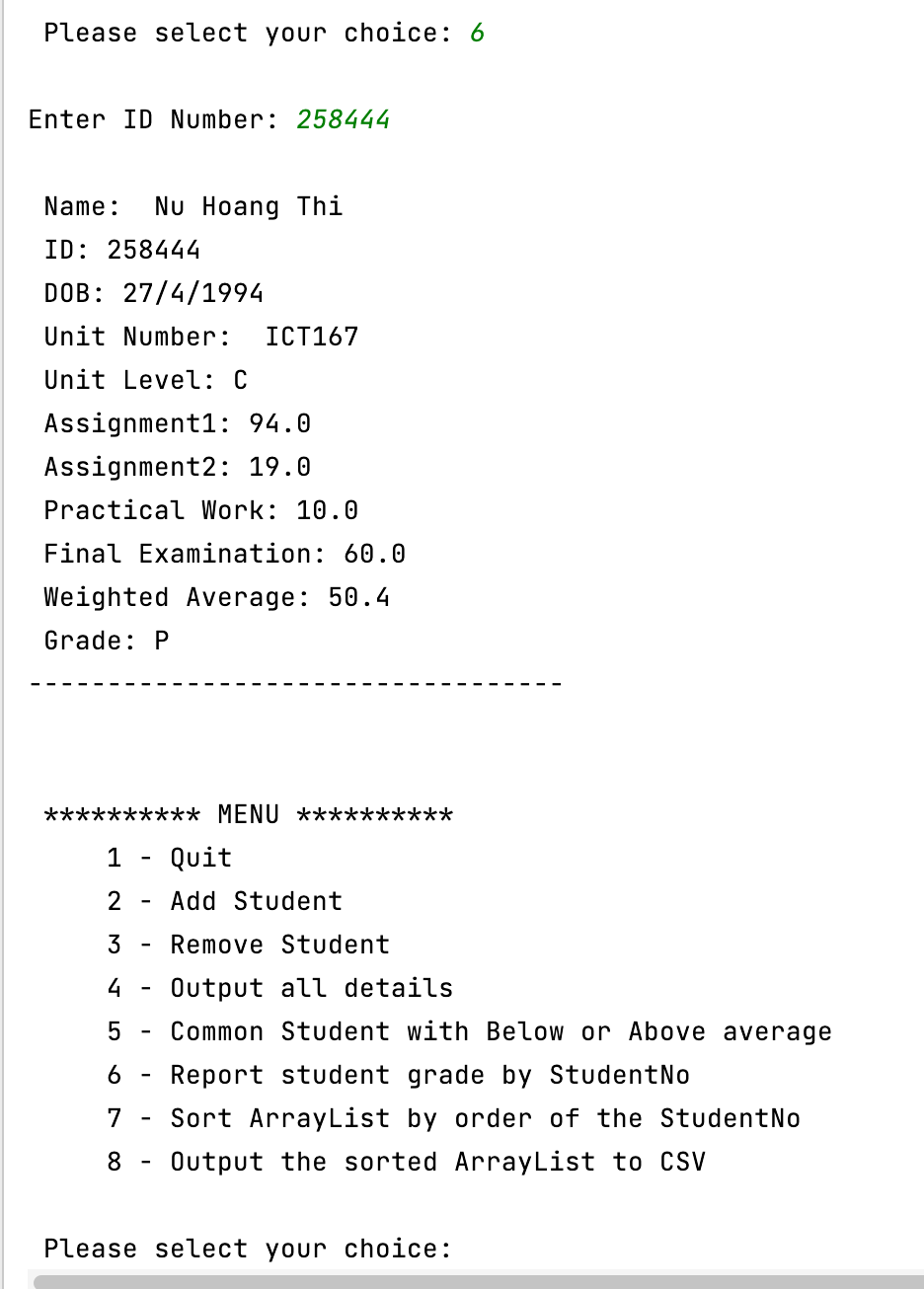
+ Check again ArrayList if studentID (784512) is deleted in the ArrayList:



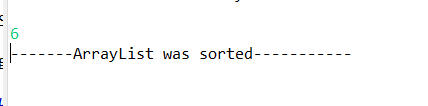
Test 4: How many students Common have AVG below and above in the ArrayList.

Test 5: Show Report Grade Student by Id.

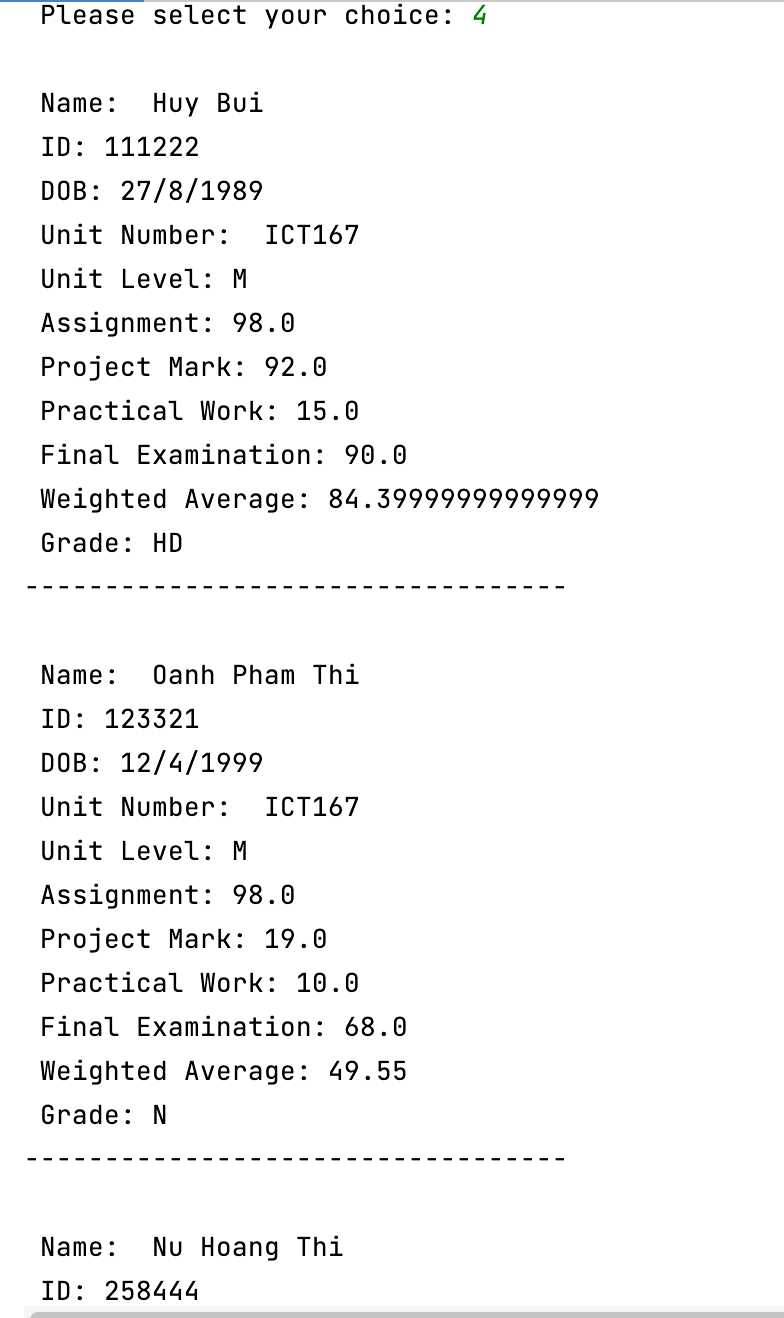
Input Id: 258444



Test 6: Sort to List Student By ID Number using Bubble Sort



+ Check again by show all students



Task 7: Output Sorted ArrayList To CSV:

