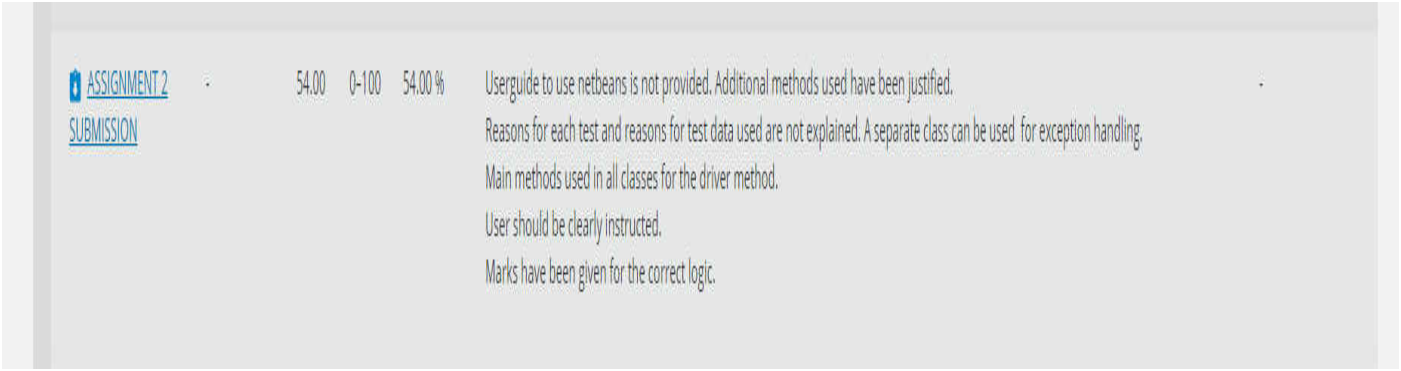
ICT167 Principles of Computer Science

Assignment 2

Jin Cherng Chong

33170193

Murdoch University



ASSIGNMENT 2 SUBMISSION	-	54.00	0-100	54.00 %	Userguide to use netbeans is not provided. Additional methods used have been justified. Reasons for each test and reasons for test data used are not explained. A separate class can be used for exception handling. Main methods used in all classes for the driver method. User should be clearly instructed. Marks have been given for the correct logic.	-

Table of content

Introduction

This documentation explains my ICT167 Assignment 2 program. The files that are referenced throughout the documentation are- Client.java, CourseWorkStudent.java, ResearchStudent.java and Student.java, student.txt, courseWorkMark.txt, and researchStudentMark.txt

The Client.java contains the code for the client program and the Student.java contains the base class representing students. Both CourseWorkStudent class and ResearchStudent class are subclasses of the base class Students. Therefore, the attributes and methods found in the Student.java file are inherited by both the CourseWorkStudent.java and ResearchStudent.java file. This documentation is for version: 0.1 which is the most up to date version as of 23/10/2020. This program is called student marks and information tracker. it can be utilised in a school where you have students and the students have assessments to complete. This program is meant to be used for only one unit per student. The files student.txt, courseWorkMark.txt, and researchStudentMark.txt contains input read in by the program.

Requirements/Specification

This student marks and information tracker program inputs the student information by reading from student.txt file and storing it into an arrayList of student objects. The option 2 menu is in in charge of reading either the courseWorkMark.txt or the researchStudentMark.txt which consist of the marks obtained by the students and storing it in the relevant student object. The program begins with asking the client whether they are dealing with researchStudent (R) or coursework students (C). Their answer will effect some of the menu options processing. In particular menu option 2, 5, and 6 will be effected. For example, if the client indicated they are dealing with coursework students then for option 5 only for coursework students will be their overall mark and grade be computed and outputted. The menu of options displayed after the client indicates whether they are dealing with research or coursework students will keep redisplaying until the client select option 1 which is to quit the menu.

Assumption-

- Assume client will input data of the correct data type
- Assume the youngest possible student in the program can be from the year 2000
- Assume if students do not have mark information in text (CourseWorkStudentMark.txt OR ResearchStudentMark.txt) then the student will be given the default marks of 0.
- Assume input of each student coming from student.txt, researchStudentMark.txt and courseWorkStudentMark.txt will be in one line, of the correct data type, in order and not empty
- Assume (related to option 6 in the menu) the average overall mark is the average overall mark for either coursework students OR research students. Combining the average coursework students and research student average overall mark to get the average for both would not be possible since those two average marks are fundamentally different. It would be like combining the average height of a people in a classroom with the average weight of a people in a classroom.
- Assume (related to option 6) this is for every coursework or research students held in arrayList and with or without mark information in text CourseWorkStudentMark.txt OR ResearchStudentMark.txt). The client will have already selected option 2 before using option 6.
- Assume (related to option 5 in the menu) either coursework or research students will be computed and outputted. Whether it is a coursework or research students is determined by the client prior to the menu being displayed. The client will enter “C” or “R” indicating their intention.
- Assume (related to option 5 in the menu) the client will have already selected option 2 before using option 5. Option 2 sets the marks needed for option 5 thus we assume the client will have already gone through option 2
- Assume each student will have a unique student ID

- Assume calculating overallMark rounded up if decimal >= 0.5 or rounded down if < 0.5

User Guide

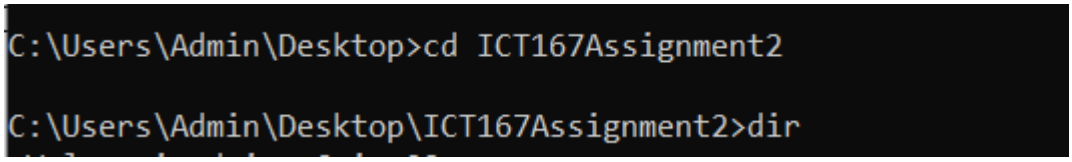
Option 1- Run with jar

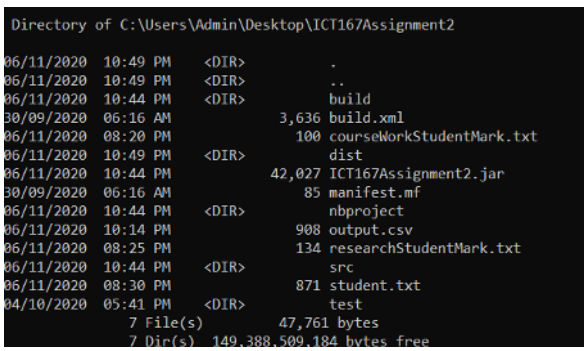
Step 1:

- Extract the ICT167Assignment2 folder to desktop

Step 2:

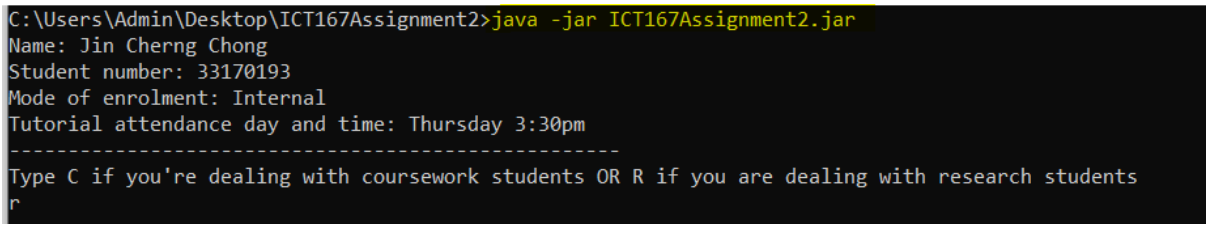
- Open up command prompt
- Go to ICT167Assignment2 directory
    - Command: Cd [ICT167Assignment2folder]





Step 3:

- Once in: ICT167Assignment2 folder → Execute the ICT167Assignment2.jar
    - Command: java -jar ICT167Assignment2.jar



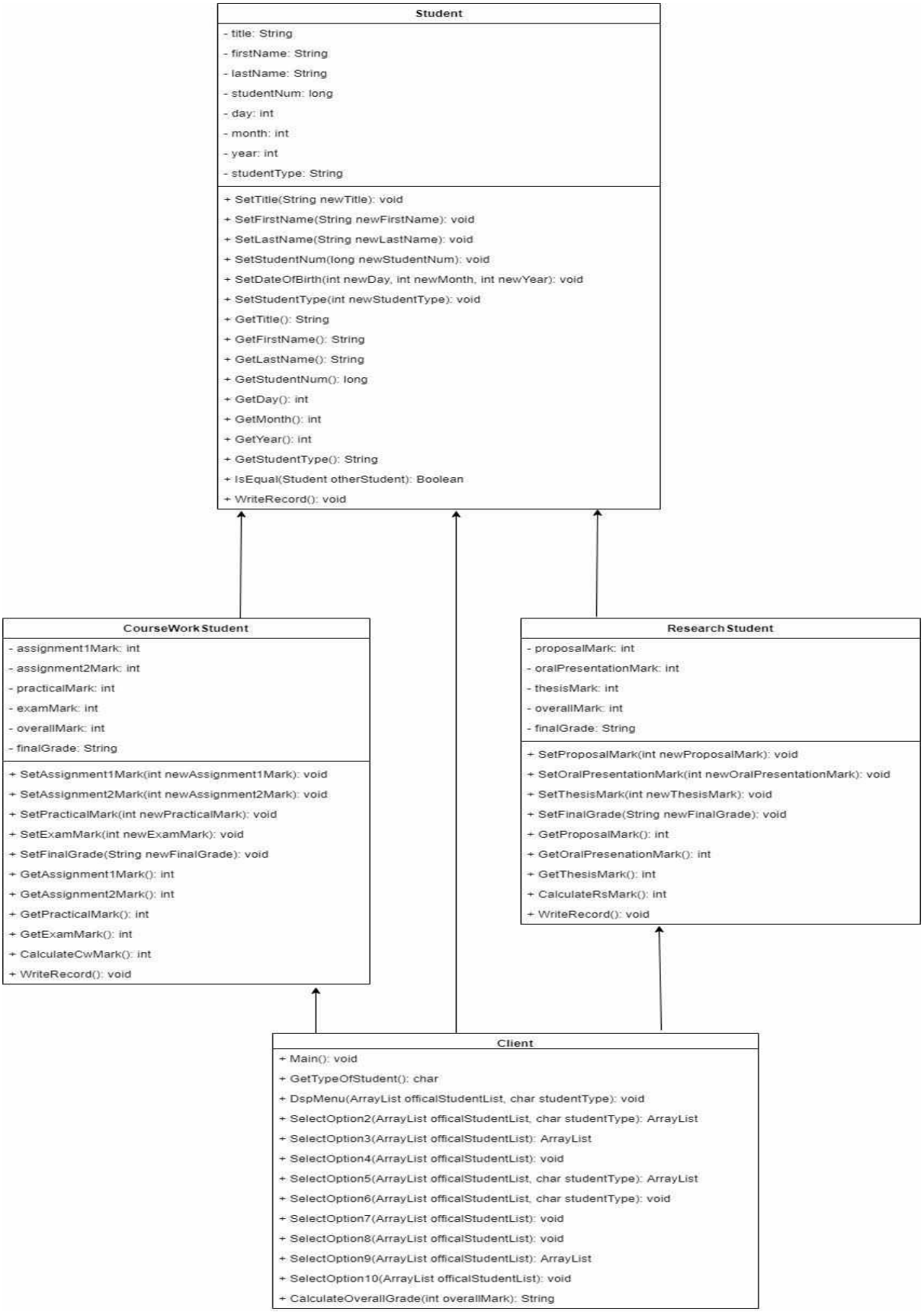
Step 4: Well done! You can now type away in the command prompt

Structure/Design/Algorithm

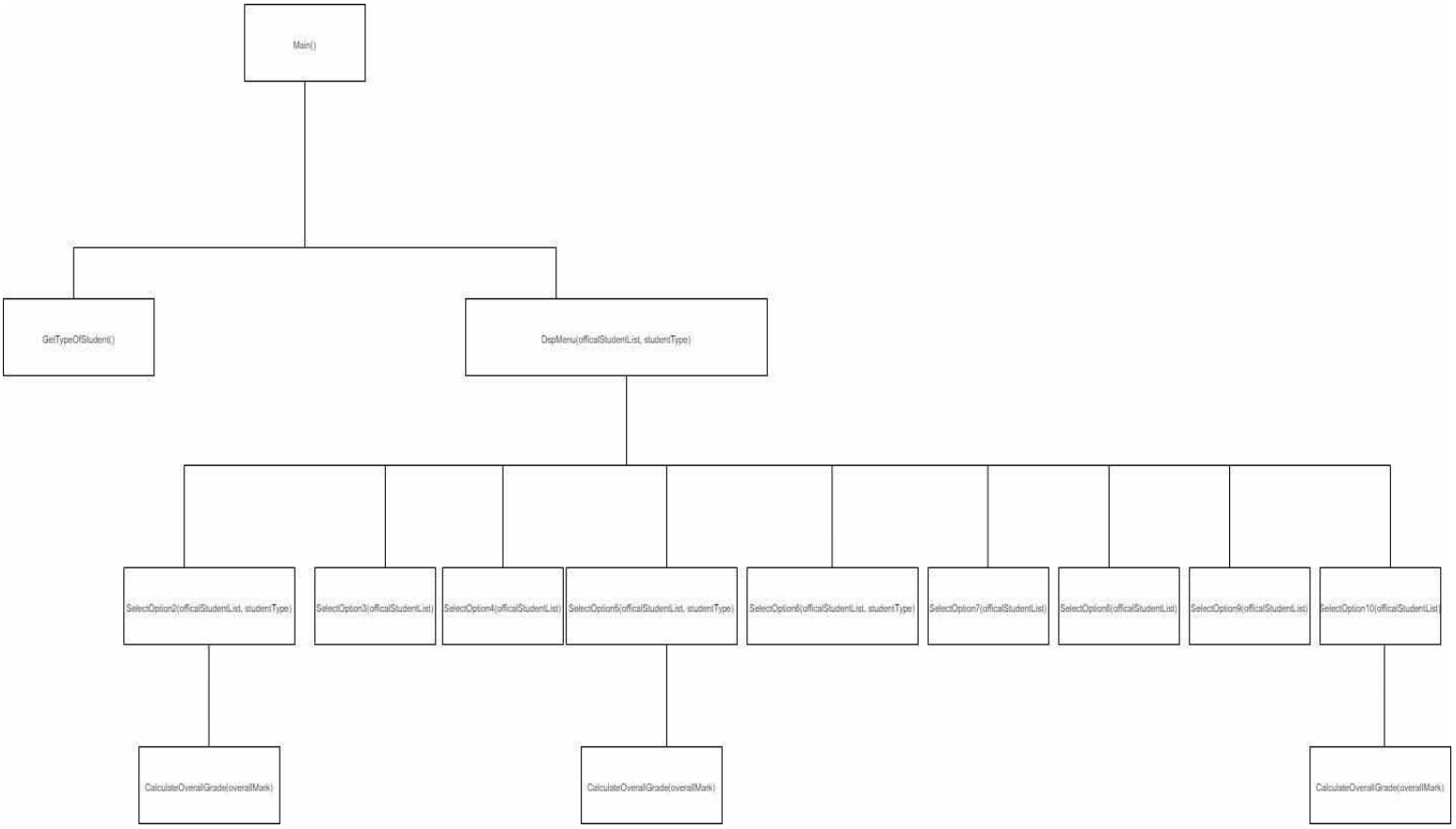
Additional method for Client class-

Methods	Justification
CalculateOverallGrade(overallMark)	This method takes the overallMark achieved by either the coursework or research students and calculates the awarded grade for student based of the overall mark. This was done to avoid code duplication since having the same method in two classes would inefficient
WriteRecord()	This method utilises polymorphism to have the same method name but output different number of attributes for different student types
Selection sort	A selection sort is used because it is one of the most efficient algorithms to sort an array of numbers in ascending order. It was selected over a bubble sort because it is more efficient in that the number of loops required to sort through the numbers to sort them is less with a selection sort

UML diagram for the client program and Student, CourseWorkStudent and ResearchStudent class


Student
- title: String
- firstName: String
- lastName: String
- studentNum: long
- day: int
- month: int
- year: int
- studentType: String
+ SetTitle(String newTitle): void
+ SetFirstName(String newFirstName): void
+ SetLastName(String newLastName): void
+ SetStudentNum(long newStudentNum): void
+ SetDateOfBirth(int newDay, int newMonth, int newYear): void
+ SetStudentType(int newStudentType): void
+ GetTitle(): String
+ GetFirstName(): String
+ GetLastName(): String
+ GetStudentNum(): long
+ GetDay(): int
+ GetMonth(): int
+ GetYear(): int
+ GetStudentType(): String
+ IsEqual(Student otherStudent): Boolean
+ WriteRecord(): void
CourseWorkStudent
- assignment1Mark: int
- assignment2Mark: int
- practicalMark: int
- examMark: int
- overallMark: int
- finalGrade: String
+ SetAssignment1Mark(int newAssignment1Mark): void
+ SetAssignment2Mark(int newAssignment2Mark): void
+ SetPracticalMark(int newPracticalMark): void
+ SetExamMark(int newExamMark): void
+ SetFinalGrade(String newFinalGrade): void
+ GetAssignment1Mark(): int
+ GetAssignment2Mark(): int
+ GetPracticalMark(): int
+ GetExamMark(): int
+ CalculateCwMark(): int
+ WriteRecord(): void
ResearchStudent
- proposalMark: int
- oralPresentationMark: int
- thesisMark: int
- overallMark: int
- finalGrade: String
+ SetProposalMark(int newProposalMark): void
+ SetOralPresentationMark(int newOralPresentationMark): void
+ SetThesisMark(int newThesisMark): void
+ SetFinalGrade(String newFinalGrade): void
+ GetProposalMark(): int
+ GetOralPresenationMark(): int
+ GetThesisMark(): int
+ CalculateRsMark(): int
+ WriteRecord(): void
Client
+ Main(): void
+ GetTypeOfStudent(): char
+ DspMenu(ArrayList officalStudentList, char studentType): void
+ SelectOption2(ArrayList officalStudentList, char studentType): ArrayList
+ SelectOption3(ArrayList officalStudentList): ArrayList
+ SelectOption4(ArrayList officalStudentList): void
+ SelectOption5(ArrayList officalStudentList, char studentType): ArrayList
+ SelectOption6(ArrayList officalStudentList, char studentType): void
+ SelectOption7(ArrayList officalStudentList): void
+ SelectOption8(ArrayList officalStudentList): void
+ SelectOption9(ArrayList officalStudentList): ArrayList
+ SelectOption10(ArrayList officalStudentList): void
+ CalculateOverallGrade(int overallMark): String

Structure chart for client program- (zoom to view more clearly)



Low level algorithm for client program-

```
Procedure void Main()

        Create studentList as new ArrayList()
        Integer StudentNo = 1;
        Character studentType = 'A'
        String typeOfStudent

        studentType = GetTypeOfStudent()



        while(hasNext(student.txt)) then

                Create student as new student()
                Boolean invalidStudentInformation = true

                title = next(student.txt)
                firstName = next(student.txt)
                surname = next(student.txt)
                studentNum = parseLong(next(student.txt))

                DOB = next(student.txt)
                String[] splitDOB = split(DOB, "/")
                day = parseInteger(splitDOB[0])
                month = parseInteger(splitDOB[1])
                year = parseInteger(splitDOB[2])

                typeOfStudent = next(student.txt)

                if(typeOfStudent == "CourseWorkStudent") then
                        Create student as new CourseWorkStudent()
                else if(typeOfStudent == "ResearchStudent") then
                        Create student as new ResearchStudent()
                else
                        Output "Error: Incorrect student type specified for student"
                EndIf
```

```
                SetTitle(Student, title)

                SetFirstName(Student, firstName)

                SetLastName(Student, lastName)

                SetStudentNum(Student, studentNum)

                SetDateOfBirth(Student, day, month, year)

                SetStudentType(typeOfStudent)


                invalidStudentInformation = (GetTitle(Student) == "None" OR
GetFirstName(Student) == "None" OR GetLastName(Student) == "None" OR GetStudentNum(Student) ==
0 OR GetDay(Student) == 0 OR GetMonth(Student) == 0 OR GetYear(Student) == 0 OR
GetStudentType(Student) == "None")


                if(invalidStudentInformation) then

                        Output "Error: Invalid information for a student: . Therefore the
program will not save the student information " + StudentNo

                else

                        Add(studentList, Student)

                EndIf


                StudentNo++




        EndWhile

        DspMenu(studentList, studentType)

EndProcedure




Procedure character GetTypeOfStudent()


        Boolean invalidStudentType = true

        Character studentType = 'A'


        do


                Output "Type C if you're dealing with coursework students OR R if dealing with
research students"

                Input studentType

                studentType = toUpperCase(studentType)


                if(studentType == C OR studentType == R) then
```

```
                        invalidStudentType = false
                else
                        Output "Invalid option!"
                EndIF

        while(invalidStudentType)

        return studentType

EndProcedure



Procedure void DspMenu(ArrayList studentList, Character studentType)

        Integer option = 0

        while(option != 1) then

                Output "Enter an option: "
                Input option

                Switch(option)
                        Case 1:
                                Output "Farewell! Exiting menu"
                        Case 2:
                                officalStudentList = SelectOption2(officalStudentList,
studentType)
                        Case 3:
                                officalStudentList = SelectOption3(officalStudentList)
                        Case 4:
                                SelectOption4(officalStudentList)
                        Case 5:
                                officalStudentList = SelectOption5(officalStudentList,
studentType)
                        Case 6:
                                SelectOption6(officalStudentList, studentType)
                        Case 7:
                                SelectOption7(officalStudentList)
                        Case 8:
```

```
                        SelectOption8(officalStudentList)
                Case 9:
                        officalStudentList = SelectOption9(officalStudentList)
                        SelectOption4(officalStudentList)
                Case 10:
                        officalStudentList = SelectOption9(officalStudentList)
                        SelectOption10(officalStudentList)
                default:
                        Output "Invalid option!"
        EndCase

    EndWhile



EndProcedure




Procedure ArrayList SelectOption2(ArrayList officalStudentList, Character studentType)

    Long studentNum = 0
    Long num



    while(hasNext(courseWorkStudentMark.txt) AND studentType = 'C') then

        num = Next(courseWorkStudentMark.txt)

        for Person To officalStudentList Do
            studentNum = GetStudentNum(person)

            if(num == studentNum) then
                //Downcast student (super class) --> courseWork student (sub
class)


                assignment1Mark = next(CourseWorkStudentMark.txt)
                SetAssignment1Mark(CourseWorkStudent, assignment1Mark)
```

```
                        assignment2Mark = next(CourseWorkStudentMark.txt)
                        SetAssignment2Mark(CourseWorkStudent, assignment2Mark)

                        practicalMark = next(CourseWorkStudentMark.txt)
                        SetPracticalMark(CourseWorkStudent, practicalMark)

                        examMark = next(CourseWorkStudentMark.txt)
                        SetExamMark(CourseWorkStudent, examMark)

                        overallMark = CalculateCwMark(CourseWorkStudent)

                        overallGrade = CalculateGrade(overallMark)
                        SetFinalGrade(CourseWorkStudent, overallGrade)

                EndIF

        EndFor

EndWhile

while(hasNext(researchStudentMark.txt) AND studentType = 'R') then

        num = Next(researchStudentMark.txt)

        for Person To officalStudentList Do
                studentNum = GetStudentNum(person)

                if(num == studentNum) then
                        //Downcast student (super class) --> research student (sub
class)

                        assignment1Mark = next(researchStudentMark.txt)
                        SetAssignment1Mark(researchStudentMark, assignment1Mark)

                        assignment2Mark = next(researchStudentMark.txt)
                        SetAssignment2Mark(researchStudentMark, assignment2Mark)

                        practicalMark = next(researchStudentMark.txt)
```

```
                                SetPracticalMark(researchStudentMark, practicalMark)

                                examMark = next(researchStudentMark.txt)
                                SetExamMark(researchStudentMark, examMark)

                                overallMark = CalculateRsMark(researchStudentMark)

                                overallGrade = CalculateGrade(overallMark)
                                SetFinalGrade(researchStudentMark, overallGrade)

                        EndIF

                EndFor

        EndWhile


        return officalStudentList

EndProcedure









Procedure ArrayList SelectOption3(ArrayList officalStudentList)

        Long clientNum = 0
        Long numOfStudent = 0
        String firstNameOfStudent
        String surnameOfStudent
        character confirmation = 'N'
        Boolean studentNumExist = false

        Output "Enter the student number identifying the student you wish to delete"
        Input clientNum
```

```
        for person To officalStudentList Do

                firstNameOfStudent = GetFirstName(person)
                surnameOfStudent = GetLastName(person)
                numOfStudent = GetStudentNum(person)

                if(clientNum == numOfStudent) then
                        Output "Are you sure you want to remove      StudentID:     (Y/N)?" +
firstNameOfStudent + surnameOfStudent + numOfStudent
                        Input confirmation
                        confirmation = toUpperCase(Character, confirmation)
                        studentNumExist = true
                EndIf

                if(confirmation == 'Y') then
                        remove(officalStudentList, person)
                        return officalStudentList
                else
                        Output "Student not removed"
                EndIf


        EndFor

        if(!studentNumExist) then
                Output "Student number entered does not exist"
        EndIf

        return officalStudentList

EndProcedure



Procedure void SelectOption4(ArrayList officalStudentList)



        for person To officalStudentList Do
```

```
                WriteRecord(officalStudentList, person)
                Output "-----------------------------------"

        EndFor


EndProcedure




Procedure ArrayList SelectOption5(ArrayList officalStudentList, character studentType)

        Integer overallMark = 0
        Long num = 0
        String overallGrade
        Long studentNum


        while(hasNext(CourseWorkStudentMark.txt) AND studentType == 'C') then

                        num = next(CourseWorkStudentMark.txt)

                        for Person To officalStudentList Do

                                studentNum = GetStudentNum(Person)

                                if(num = studentNum) then

                                        //Downcast object from arrayList (student) -->
courseWorkStudent object

                                        overallMark = CalculateCwMark(CourseWorkStudent)
                                        overallGrade = CalculateGrade(overallMark)
                                        SetFinalGrade(CourseWorkStudent, overallGrade)

                                        WriteRecord(CourseWorkStudent)
                                        Output "-----------------------------------"
```

```
                        EndIF

                EndFor

                nextLine(CourseWorkStudentMark.txt)

        EndWhile


        while(hasNext(ResearchStudentMark.txt) AND studentType == 'R') then

                num = next(CourseWorkStudentMark.txt)

                for Person To officalStudentList Do

                        studentNum = GetStudentNum(Person)

                        if(num = studentNum) then

                                //Downcast object from arrayList (student) -->
researchStudent object
                                overallMark = CalculateRsMark(ResearchStudent)
                                overallGrade = CalculateGrade(overallMark)
                                SetOverallGrade(ResearchStudent, overallGrade)

                                WriteRecord(ResearchStudent)
                                Output "-----------------------------------"
                        EndIF
                EndFor

                nextLine(ResearchStudentMark.txt)

        EndWhile

        return ArrayList

EndProcedure
```

```
Procedure void SelectOption6(ArrayList officalStudentList, character studentType)


        Integer mark = 0
        Integer totalMarkRS = 0
        Integer totalMarkCw = 0
        Integer counterCw = 0
        Integer counterRs = 0
        Boolean correctStudentType = false
        Integer averageAbove = 0
        Integer averageBelow = 0
        Strint type
        Integer averageMarkCw = 0
        Integer averageMarkRs = 0

        for Person To officalStudentList Do

                type = GetStudentType(person)

                if(type = "CourseWorkStudent") then
                        //Downcast object from arrayList (student) --> courseWorkStudent object
                        mark = CalculateCwMark(courseWorkStudent)
                        totalMarkCw += mark
                        counterCw++
                EndIF

                if(type = "ResearchStudent") then
                        //Downcast object from arrayList (student) --> ResearchStudent object
                        mark = CalculateRsMark(ResearchStudent)
                        totalMarkRs += mark
                        counterRs++
                EndIF
        EndFor

        averageMarkCw = totalMarkCw / counterCw
        averageMarkRs = totalMarkRs / counterRs


        for Integer i = 0 To (Size(officalStudentList) AND studentType = 'C') Do
```

```
        correctStudentType = false

        Student person = get(officalStudentList, i)
        type = GetStudentType(person)

        if(type = "CourseWorkStudent")  then
                //Downcast object from arrayList (student) --> courseWorkStudent object
                mark = CalculateCwMark(courseWorkStudent)
                correctStudentType = true
        EndIF

        if(mark >= averageMarkCw AND correctStudentType) then
                averageAbove++
        EndIF

        if(mark <= averageMarkCw AND correctStudentType) then
                averageBelow++
        EndIF

EndFor


for Integer i = 0 To (Size(officalStudentList) AND studentType = 'R') Do

        correctStudentType = false

        Student person = get(officalStudentList, i)
        type = GetStudentType(person)

        if(type = "CourseWorkStudent")  then
                //Downcast object from arrayList (student) --> researchStudent object
                mark = CalculateRsMark(researchStudent)
                correctStudentType = true
        EndIF

        if(mark >= averageMarkRs AND correctStudentType) then
                averageAbove++
        EndIF
```

```
                if(mark <= averageMarkRs AND correctStudentType) then
                        averageBelow++
                EndIF

        EndFor

        Output "Number of students above average- " + averageAbove
        Output "Number of students below average- " + averageBelow


EndProcedure



Procedure void SelectOption7(ArrayList officalStudentList)

        Long num = 0
        Long studentNum = 0
        Boolean studentNotFound = true

        Output "Enter a student number: "
        Input num

        for Person To officalStudentList Do

                studentNum = GetStudentNum(person)

                if(num == studentNum) then
                        WriteRecord(person)
                        studentNotFound = false
                EndIF

        EndFor

        if(studentNotFound) then
                Output "Student not found in arrayList"
        EndIF
```

```
                Output "-------------------------------------"

EndProcedure



Procedure void SelectOption8(ArrayList officalStudentList)

        String fName
        String lName
        String studentFName
        String studentLName
        Boolean studentNotFound = true

        Output "Enter first name of student"
        Input fName

        Output "Enter last name of student"
        Input lName

        for Person To officalStudentList Do

                studentFName = GetFirstName(person)
                studentLName = GetLastName(person)

                if(fName == studentFName AND lName == studentLName) then
                        WriteRecord(person)
                        studentNotFound = false
                        Output "-------------------------------------"
                EndIF

        EndFor

        if(studentNotFound) then
                Output "Student not found in arrayList"
        EndIF

EndProcedure
```

```
Procedure ArrayList SelectOption9(ArrayList officalStudentList)


        for Integer i = 0 To (i < Size(officalStudentList) - 1) Do

                Integer indexOfUnsortedSmallest = i

                for Integer j = i + 1 To (j < Size(officalStudentList)) Do

                        Create person as a new Student()
                        person = Get(officalStudentList, indexOfUnsortedSmallest)
                        Long currentSmallNum = GetStudentNum(person)

                        Create secondPerson as a new Student()
                        secondPerson = Get(officalStudentList, j)
                        Long afterNum = GetStudentNum(secondPerson)

                        if(afterNum < currentSmallNum) then
                                indexOfUnsortedSmallest = j
                        EndIF

                EndFor

                Create tempPerson as a new Student()
                tempPerson = Get(officalStudentList, indexOfUnsortedSmallest)

                Create tempPerson2 as a new Student()
                tempPerson2 = Get(officalStudentList, i)

                officalStudentList.Set(indexOfUnsortedSmallest, tempPerson2)
                officalStudentList.Set(i, tempPerson)

        EndFor

        return officalStudentList
```

```
EndProcedure



Procedure void SelectOption10(ArrayList officalStudentList)

        String OutputFilePath = "output.csv"

        Create outputStream as new PrintWriter(OutputFilePath)

        Write(outputStream, "Title" + ",")
        Write(outputStream, "Name" + ",")
        Write(outputStream, "Student Number" + ",")
        Write(outputStream, "Date of Birth" + ",")
        Write(outputStream, "StudentType" + ",")
        Write(outputStream, "OverallMark" + ",")
        Write(outputStream, "Grade" + ",")
        Write(outputStream, "Assessment1" + ",")
        Write(outputStream, "Assessment2" + ",")
        Write(outputStream, "Assessment3" + ",")
        Write(outputStream, "Assessment4" + ",")

        for Person To officalStudentList Do

                Write(outputStream, "/n")

                String title = GetTitle(person)
                String firstName = GetFirstName(person)
                String lastName = GetLastName(person)
                Long studentNum = GetStudentNum(person)
                Integer day = GetDay(person)
                Integer month = GetMonth(person)
                Integer year = GetYear(person)
                String studentType = GetStudentType(person)

                if(studentType == "courseWorkStudent") then
                        //Downcast object from arrayList (student) --> courseWorkStudent object
```

```
        Integer assignment1 = GetAssignment1Mark(courseWorkStudent)
        Integer assignment2 = GetAssignment2Mark(courseWorkStudent)
        Integer practicalMark = GetPracticalMark(courseWorkStudent)
        Integer examMark = GetExamMark(courseWorkStudent)
        Integer overallMark = CalculateCwMark(courseWorkStudent)
        String finalGrade = CalculateOverallGrade(overallMark)

        Write(outputStream, title + ",")
        Write(outputStream, firstName + " " + lastName + ",")
        Write(outputStream, studentNum + ",")
        Write(outputStream, day + "/" + month + "/" + year + ",")
        Write(outputStream, studentType + ",")
        Write(outputStream, overallMark + ",")
        Write(outputStream, finalGrade + ",")
        Write(outputStream, assignment1 + ",")
        Write(outputStream, assignment2 + ",")
        Write(outputStream, assignment3 + ",")
        Write(outputStream, assignment4 + ",")

EndIF

if(studentType == "researchStudent") then
        //Downcast object from arrayList (student) --> researchStudent object

        Integer proposalMark = GetProposalMark(researchStudent)
        Integer oralPresenationMark = GetOralPresenationMark(researchStudent)
        Integer thesisMark = GetThesisMark(researchStudent)
        Integer overallMark = CalculateRsMark(researchStudent)
        String finalGrade = CalculateOverallGrade(overallMark)

        Write(outputStream, title + ",")
        Write(outputStream, firstName + " " + lastName + ",")
        Write(outputStream, studentNum + ",")
        Write(outputStream, day + "/" + month + "/" + year + ",")
        Write(outputStream, studentType + ",")
        Write(outputStream, overallMark + ",")
        Write(outputStream, finalGrade + ",")
        Write(outputStream, proposalMark + ",")
```

```
                        Write(outputStream, oralPresenationMark + ",")
                        Write(outputStream, thesisMark + ",")

                EndIF

                Close(outputStream)
                Output "Finished writing to file"

        EndFor

EndProcedure


Procedure String CalculateOverallGrade(Integer overallMark)

        String overallGrade

        if(overallMark < 0 OR overllMark > 100) then
                Output "Overall mark not valid"
        else if(overallGrade >= 80) then
                overallGrade = HD
        else if(overallGrade >= 70) then
                overallGrade = D
        else if(overallGrade >= 60) then
                overallGrade = C
        else if(overallGrade >= 50) then
                overallGrade = P
        else if(overallGrade >= 0) then
                overallGrade = N
        EndIf

        return overallGrade

EndProcedure
```

Low level algorithm for student class-

```
private String title
private String firstName
private String lastName
private Long studentNum
private Integer day
private Integer month
private Integer year
private String studentType



Procedure Student()

        title = "None"
        firstName = "None"
        lastName = "None"
        studentNum = 0
        day = 0
        month = 0
        year = 0
        studentType = "None"

EndProcedure



Procedure Student(String initalTitle, String initialFirstName, String initialLastName, Long
initialStudentNum, Integer initialDay, Integer initialMonth, Integer initialYear, String
initialStudentType)

        title = initalTitle
        firstName = initialFirstName
        lastName = initialLastName
        studentNum = initialStudentNum
        day = initialDay
        month = initialMonth
        year = initialYear
        studentType = initialStudentType
```

```
EndProcedure


Procedure void SetTitle(String newTitle)

        if(!newTitle Is Empty AND newTitle != null) then
                title = newTitle
        else
                Output "Error: Invalid title for student"
        EndIf

EndProcedure




Procedure void SetFirstName(String newFirstName)

        if(!newFirstName Is Empty AND newFirstName != null) then
                firstName = newFirstName
        else
                Output "Error: Invalid first name for student"
        EndIf

EndProcedure


Procedure void SetLastName(String newLastName)

        if(!newLastName Is Empty AND newLastName != null) then
                lastName = newLastName
        else
                Output "Error: Invalid last name for student"
        EndIf

EndProcedure


Procedure void SetStudentNum(Long newStudentNum) //Have to deal with duplicate student ID
```

```
                studentNum = newStudentNum

EndProcedure




Procedure void SetDateOfBirth(Integer newDay, Integer newMonth, Integer newYear)


        Integer oldDay = day

        Integer oldMonth = month


        Boolean maxTwentyNineDay = (newDay >= 1 AND newDay <= 29)

        Boolean maxThirtyDay = (newDay >= 1 AND newDay <= 30)

        Boolean maxThrityOneDay = (newDay >= 1 AND newDay <=31)


        Boolean Feb = (newMonth == 2)

        Boolean thirtyDayMonth = (newMonth == 4 OR newMonth == 6 OR newMonth == 9 OR newMonth
== 11)

        Boolean thirtyOneDayMonth = (newMonth == 1 OR newMonth == 3 OR newMonth == 5 OR
newMonth == 7 OR newMonth == 8 OR newMonth == 10 OR newMonth == 12)




        if(newMonth >= 1 AND newMonth <= 12) then

                month = newMonth

        else

                Output "Error: Invalid month entered. Therefore, date of birth for student not
set"

                return

        EndIF




        if(maxTwentyNineDay AND Feb) then

                day = newDay

        else if(maxThirtyDay AND thirtyDayMonth) then

                day = newDay

        else if(maxThrityOneDay AND thirtyOneDayMonth) then

                day = newDay

        else

                Output "Error: Invalid day for student. Therefore, date of birth for student
not set"
```

```
                month = oldMonth
                return
        EndIF

        if(newYear >= 2000) then
                year = newYear
        else
                Output "Error: Invalid year entered for student. Therefore, date of birth for
student not set"
                month = oldMonth
                day = oldDay
                return
        EndIF

EndProcedure


Procedure void SetStudentType(String newStudentType)

        if(newStudentType == "CourseWorkStudent") then
                studentType = "CourseWorkStudent"
        else if(newStudentType = "ResearchStudent") then
                studentType = "ResearchStudent"
        else
                Output "Error: Invalid student type for student"
        EndIf

EndProcedure




Procedure String GetTitle

        return title

EndProcedure

Procedure String GetFirstName
```

```
        return firstName

EndProcedure

Procedure String GetLastName

        return lastName

EndProcedure

Procedure Long GetStudentNum

        return studentNum

EndProcedure

Procedure Integer GetDay

        return day

EndProcedure

Procedure Integer GetMonth

        return month

EndProcedure

Procedure Integer GetYear

        return year

EndProcedure

Procedure String GetStudentType

        return studenType

EndProcedure
```

```
Procedure Boolean IsEqual(Student otherStudent)


        Boolean sameName = (this.firstName Equals otherStudent.firstName AND this.lastName
Equals otherStudent.lastName)

        Boolean sameDOB = (this.day == otherStudent.day AND this.month == otherStudent.month
AND this.year == otherStudent.year)


        if(SameName AND sameDOB) then

                return true

        else

                return false

        EndIf


EndProcedure



Procedure void WriteRecord()


        Output "Title- " + title

        Output "Name- " + firstName + lastName

        Output "studentNum- " + studentNum

        Output "Date of Birth- "  + day + month + year

        Output "Student type- " + studenType


EndProcedure
```

Low level algorithm for CourseWorkStudent class-

```
private Integer assignment1Mark

private Integer assignment2Mark

private Integer practicalMark

private Integer examMark

private Integer overallMark

private String finalGrade



Procedure CourseWorkStudent()


        super() //     Must Invoke studentClassConstructor (base class)

        assignment1Mark = 0

        assignment2Mark = 0

        practicalMark = 0

        examMark = 0

        finalGrade = 'None'


EndProcedure



Procedure CourseWorkStudent(String initalTitle, String initialFirstName, String
initialLastName, Long initialStudentNum, Integer initialDay, Integer initialMonth, Integer
initialYear, String initialStudentType, Integer initialAssignment1Mark, Integer
initialAssignment2Mark, Integer initialPracticalMark, Integer initialExamMark, Integer
initialOverallMark, String initialFinalGrade)


        studentClassConstructor(initalTitle, initialFirstName, initialLastName,
initialStudentNum, initialDay, initialMonth, initialYear, initialStudentType) // Must Invoke
studentClassConstructor (base class)

        assignment1Mark = initialAssignment1Mark

        assignment2Mark = initialAssignment2Mark

        practicalMark = initialPracticalMark

        examMark = initialExamMark

        overallMark = initialOverallMark

        finalGrade = initialFinalGrade


EndProcedure




Procedure void SetAssignment1Mark(Integer newAssignment1Mark)
```

```
        if(newAssignment1Mark >= 0 AND newAssignment1Mark <= 100) then
                assignment1Mark = newAssignment1Mark
        else
                Output "Error: Invalid assignment 1 mark for student"
        EndIf

EndProcedure


Procedure void SetAssignment2Mark(Integer newAssignment2Mark)

        if(newAssignment2Mark >= 0 AND newAssignment2Mark <= 100) then
                assignment2Mark = newAssignment2Mark
        else
                Output "Error: Invalid assignment 2 mark for student"
        EndIf

EndProcedure


Procedure void SetPracticalMark(Integer newPracticalMark)

        if(newPracticalMark >= 0 AND newPracticalMark <= 20) then
                practicalMark = newPracticalMark
        else
                Output "Error: Invalid practical mark for student"
        EndIf

EndProcedure


Procedure void SetExamMark(Integer newExamMark)

        if(newExamMark >= 0 AND newExamMark <= 100) then
                examMark = newExamMark
        else
                Output "Error: Invalid exam mark for student"
        EndIf
```

```
EndProcedure



Procedure void SetFinalGrade(String newFinalGrade)


        if(newFinalGrade == "HD" OR newFinalGrade == "D" OR newFinalGrade == "C" OR
newFinalGrade == "P" OR newFinalGrade == "N") then

                finalGrade = newFinalGrade

        else

                Output "Error: Invalid final grade for student"

        EndIF



EndProcedure







Procedure Integer GetAssignment1Mark


        return assignment1Mark


EndProcedure


Procedure Integer GetAssignment2Mark


        return Assignment2Mark


EndProcedure


Procedure Integer GetPracticalMark


        return practicalMark


EndProcedure


Procedure Integer GetExamMark


        return examMark
```

```
EndProcedure




Procedure Integer CalculateCwMark()


        Double weightedAssignment1Mark = (double) assignment1Mark/100 * 25

        Double weightedAssignment2Mark = (double) assignment2Mark/100 * 25

        Double weightedPracticalMark = (double) practicalMark/20 * 20

        Double weightedExamMark = (double) examMark/100 * 30


        overallMark = (integer) (weightedAssignment1Mark + weightedAssignment2Mark +
weightedPracticalMark + weightedExamMark)


        Double decimalInput = (weightedAssignment1Mark + weightedAssignment2Mark +
weightedPracticalMark + weightedExamMark) - overallMark;


        if(decimalInput < 0.5) then

                return overallMark

        else

                Double decNumToRoundUp = 1 - decimalInput;


                overallMark = (integer) ((weightedAssignment1Mark + weightedAssignment2Mark +
weightedPracticalMark + weightedExamMark)   + decNumToRoundUp)


                return overallMark

        EndIF


EndProcedure




Procedure void WriteRecord()


        Output "Title- " + GetTitle()

        Output "Name- " + GetFirstName() + GetLastName()

        Output "studentNum- " + GetStudentNum()

        Output "Date of Birth- "  + GetDay() + GetMonth() + GetYear()

        Output "Student type- " + GetStudentType()
```

```
        Output "assignment 1 mark- " + assignment1Mark
        Output "assignment 2 mark- " + assignment2Mark
        Output "practical work mark- " + practicalMark
        Output "exam mark- " + examMark
        Output "overall mark- " + overallMark
        Output "final grade- " + finalGrade
EndProcedure
```

Low level algorithm for ResearchStudent class-

```
private Integer proposalMark

private Integer oralPresenationMark

private Integer thesisMark

private Integer overallMark

private String finalGrade


Procedure ResearchStudent()


        super()

        proposalMark = 0

        oralPresenationMark = 0

        thesisMark = 0

        overallMark = 0

        finalGrade = 'None'


EndProcedure



Procedure ResearchStudent(String initalTitle, String initialFirstName, String initialLastName,
Long initialStudentNum, Integer initialDay, Integer initialMonth, Integer initialYear, String
initialStudentType, Integer initialProposalMark, Integer initialOralPresenationMark, Integer
initialThesisMark, Integer initialOverallMark, String initialFinalGrade)


        super(initalTitle, initialFirstName, initialLastName, initialStudentNum, initialDay,
initialMonth, initialYear, initialStudentType)

        proposalMark = initialProposalMark

        oralPresenationMark = initialOralPresenationMark

        thesisMark = initialThesisMark

        overallMark = initialOverallMark

        finalGrade = initialFinalGrade


EndProcedure



Procedure void SetProposalMark(Integer newProposalMark)


        if(newProposalMark >= 0 AND newProposalMark <= 100) then

                proposalMark = newProposalMark

        else

                Output "Error: Invalid proposal mark for student"
```

```
        EndIf

EndProcedure


Procedure void SetOralPresenationMark(Integer newOralPresenationMark)

        if(newOralPresenationMark >= 0 AND newOralPresenationMark <= 20) then
                oralPresenationMark = newOralPresenationMark
        else
                Output "Error: Invalid oral presenation mark for student"
        EndIf

EndProcedure


Procedure void SetThesisMark(Integer newThesisMark)

        if(newThesisMark >= 0 AND newThesisMark <= 100) then
                thesisMark = newThesisMark
        else
                Output "Error: Invalid thesis mark for student"
        EndIf

EndProcedure


Procedure void SetFinalGrade(String newFinalGrade)

        if(newFinalGrade == "HD" OR newFinalGrade == "D" OR newFinalGrade == "C" OR
newFinalGrade == "P" OR newFinalGrade == "N") then
                finalGrade = newFinalGrade
        else
                Output "Error: Invalid final grade for student"
        EndIF


EndProcedure
```

```
Procedure Integer GetProposalMark

        return proposalMark

EndProcedure

Procedure Integer GetOralPresenationMark

        return oralPresenationMark

EndProcedure

Procedure Integer GetThesisMark

        return thesisMark

EndProcedure




Procedure Integer CalculateRsMark()

        Double weightedProposalMark = (Double) proposalMark/100 * 30
        Double weightedOralPresenationMark = (Double) oralPresenationMark/20 * 10
        Double weightedThesisMark = (Double) thesisMark/100 * 60



        overallMark = (integer) (weightedProposalMark + weightedOralPresenationMark +
weightedThesisMark)

        Double decimalInput = (weightedProposalMark + weightedOralPresenationMark +
weightedThesisMark) - overallMark;

        if(decimalInput < 0.5) then
                return overallMark
        else
                Double decNumToRoundUp = 1 - decimalInput;
```

```
                overallMark = (integer) ((weightedProposalMark + weightedOralPresenationMark +
weightedThesisMark)  + decNumToRoundUp)


                return overallMark

        EndIF


EndProcedure



Procedure void WriteRecord()


        Output "Title- " + GetTitle()

        Output "Name- " + GetFirstName() + GetLastName()

        Output "studentNum- " + GetStudentNum()

        Output "Date of Birth- "  + GetDay() + GetMonth() + GetYear()

        Output "Student type- " + GetStudentType()

        Output "Proposal mark- " + proposalMark

        Output "Oral presenation mark- " + oralPresenationMark

        Output "Thesis mark- " + thesisMark

        Output "Overall mark- " + overallMark

        Output "final grade- " + finalGrade


EndProcedure
```

Limitations

One shortfall of my program is the way in which the CSV file is outputted. In my output CSV file, I had to use assessment 1, assessment 2, and assessment 3 to identify the different assessments. The disadvantage of this method is that it is hard to identify which assessment is which. The advantage of this method is that the method is adaptable to many different assessments and places which is why I did it this way.

Another shortfall with my program is that it doesn't validate the student.txt input well. I had to make the assumption that the student.txt input was of the correct type. For example, the student ID is a number and not a word is an assumption I made. The reason why student.txt isn't validated well is because I used the method .next() to get input from student.txt. Therefore, due to the limitations of that method the student.txt input will always be a string. So without the assumption that student.txt will be of the correct type, a student ID as a word would be possible.

Testing

Testing has been divided into several parts. A driver program has been utilised for every part to show that every method works correctly. The following parts are tested- student (base class), courseworkStudent (sub class), researchStudent (sub class) and finally client program. The inputs come from student.txt and sometimes the client. For the client program I've tested each option

Test Table: Student class

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	Enters title of student, first name, last name, student number, date of birth and student type correctly in order on one line	Mr Jin Chong 33170193 10/2/2004 CourseWorkStudent	Title- Mr Name- Jin Chong Student Number- 33170193 Date of Birth- 10/2/2004 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- None	Success
2	Enters date of birth incorrectly with day out of range for the month	Dr Jin Bloggs 33170193 33/2/2004 reseArchStudenT	Invalid day for student. Therefore, date of birth for student not set invalid information for student. Therefore, the program will not save the student information	Success
3	Enters date of birth incorrectly with month out of range for a year	Mr Seymor Skinner 36945 20/13/2004 CourseWorkStudent	Invalid month entered. Therefore, date of birth for student not set invalid information for student. Therefore, the program will not save the student information	Success
4	Enters date of birth incorrectly with year outside the accepted range (so before year 2000)	Miss Hilder Chan 238294 4/2/1999 CourseWorkStudent	Invalid year entered for student. Therefore, date of birth for student not set invalid information for student. Therefore, the program will not save the student information	Success
5	Enter date of birth incorrectly with day, month, and year outside the valid range	Mr Roman Lewis 101818 32/13/1000 CourseWorkStudent	Invalid month entered. Therefore, date of birth for student not set	Success

			invalid information for student. Therefore, the program will not save the student information	
6	Enter research student type in different cases	Miss Kava Dickson 20202 16/11/2030 REsEarChStudent	Title- Miss Name- Kava Dickson Student Number- 20202 Date of Birth- 16/11/2030 Student type- ResearchStudent Proposal mark- 0 Oral presenation mark- 0 Thesis mark- 0 overall mark- 0 overall mark- None	Success
7	Enter coursework student type in different cases	Dr Tom Mcdonald 04011 16/11/2015 CoURsEWorKStudent	Title- Dr Name- Tom Mcdonald Student Number- 04011 Date of Birth- 16/11/2015 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- None	Success
8	Enter student type incorrectly so not courseworkstudent or researchstudent	Dr Brett Oneil 30133 32/5/2004 sess	Incorrect student type specified for student invalid information for student. Therefore, the program will not save the student information	Success
9	Enter two student objects with same names and same date of birth and name in different cases	Mr PEPpa Dom 2468 12/2/2009 ResearchStudent Dr PEppA DOm 9119 12/2/2009 CoURsEWorKStudent	Same name and DOB	Success
10	Enter two student objects without same name but same date of birth	Dr PEppA DOm 9119 12/2/2009 CoURsEWorKStudent Dr Meg Tom 2048 12/2/2009 CoURsEWorKStudent	No not equal	Success
11	Enter two student with same name but without same date of birth	Dr Meg Tom 2048 12/2/2009 CoURsEWorKStudent Dr Meg Tom 4981 14/8/2010 CoURsEWorKStudent	No not equal	Success

Result of programing testing

TestCase 1:

Title- Mr

Name- Jin Chong

Student Number- 33170193

Date of Birth- 10/2/2004

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

TestCase 2:

Error: Invalid day for student. Therefore, date of birth for student not set

Error: Invalid information for student 2. Therefore, the program will not save the student information

TestCase 3:

Error: Invalid month entered. Therefore, date of birth for student not set

Error: Invalid information for student 3. Therefore, the program will not save the student information

TestCase 4:

Error: Invalid year entered for student. Therefore, date of birth for student not set

Error: Invalid information for student 4. Therefore, the program will not save the student information

TestCase 5:

Error: Invalid month entered. Therefore, date of birth for student not set

Error: Invalid information for student 5. Therefore, the program will not save the student information

TestCase 6:

Title- Miss

Name- Kava Dickson

Student Number- 20202

Date of Birth- 16/11/2030

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- None

TestCase 7:

Title- Dr

Name- Tom Mcdonald

Student Number- 4011

Date of Birth- 16/11/2015

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

TestCase 8:

Error: Incorrect student type specified for student

Error: Invalid information for student 12. Therefore, the program will not save the student information

TestCase 9:

Same name and DOB

TestCase 10:

No not equal

TestCase 11:

No not equal

Test Table: ResearchStudent class

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	Enter maximum range number for proposalMark, oralPresenationMark and thesisMark	636334 636334 100 20 100	Title- Miss Name- Tom Rox Student Number- 636334 Date of Birth- 6/11/2030 Student type- ResearchStudent Proposal mark- 100 Oral presenation mark- 20 Thesis mark- 100 overall mark- 100 overall mark- HD	Success
2	Enter minimum range for proposalMark, oralPresenationMark and thesisMark	20202 20202 0 0 0	Title- Miss Name- Kava Dickson Student Number- 20202 Date of Birth- 16/11/2030 Student type- ResearchStudent Proposal mark- 0 Oral presenation mark- 0 Thesis mark- 0 overall mark- 0 overall mark- N	Success
3	Enter proposalMark and oralPresenationMark above acceptable range	2468 2468 101 21 10	Invalid proposal mark for student Invalid oral presenation mark for student Title- Mr Name- PEPpa Dom Student Number- 2468 Date of Birth- 12/2/2009 Student type- ResearchStudent Proposal mark- 0 Oral presenation mark- 0 Thesis mark- 10 overall mark- 6 overall mark- N	Success
4	Enter proposalMark, oralPresenationMark and thesisMark below acceptable range	59822 59822 -1 -1 -100	Invalid proposal mark for student Invalid oral presenation mark for student Invalid thesis mark for student Title- Mr Name- Raphel Nadal Student Number- 59822 Date of Birth- 20/12/2005 Student type- ResearchStudent Proposal mark- 0 Oral presenation mark- 0	Success

			Thesis mark- 0 overall mark- 0 overall mark- N	

Result of programing testing

TestCase 1:

Enter a student: 636334

Title- Miss

Name- Tom Rox

Student Number- 636334

Date of Birth- 6/11/2030

Student type- ResearchStudent

Proposal mark- 100

Oral presenation mark- 20

Thesis mark- 100

overall mark- 100

overall mark- HD

TestCase 2:

Enter a student: 20202

Title- Miss

Name- Kava Dickson

Student Number- 20202

Date of Birth- 16/11/2030

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- N

TestCase 3:

Enter a student: 2468

Error: Invalid proposal mark for student

Error: Invalid oral presenation mark for student

Title- Mr

Name- PEPpa Dom

Student Number- 2468

Date of Birth- 12/2/2009

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 15

Thesis mark- 10

overall mark- 14

overall mark- N

TestCase 4:

Enter a student: 59822

Error: Invalid proposal mark for student

Error: Invalid oral presenation mark for student

Error: Invalid thesis mark for student

Title- Mr

Name- Raphel Nadal

Student Number- 59822

Date of Birth- 20/12/2005

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- N

Test Table: CourseWorkStudent class

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	Enter maximum range number for assignment1Mark, assignment2Mark, practicalMark and examMark	12345 100 100 20 100	Title- Miss Name- Tom Rox Student Number- 12345 Date of Birth- 1/4/2020 Student type- CourseWorkStudent assignment 1 mark- 100 assignment 2 mark- 100 practical work mark- 20 exam mark- 100 overall mark- 100 overall mark- HD	Success
2	Enter maximum range number for assignment1Mark, assignment2Mark, practicalMark and examMark	1225 0 0 0 0	Title- Dr Name- Tom Mcdonald Student Number- 4011 Date of Birth- 16/11/2015 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- N	Success
3	Enter assignment1Mark, assignment2Mark, practicalMark and examMark above acceptable range	9119 101 101 21 101	Invalid assignment 1 mark for student Invalid assignment 2 mark for student Invalid practical mark for student Invalid exam mark for student Title- Dr Name- PEppA DOm Student Number- 9119 Date of Birth- 12/2/2009 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- N	Success
4	Enter assignment1Mark, assignment2Mark, practicalMark and examMark below acceptable range	4981 -1 -1 -1 -1	Invalid assignment 1 mark for student Invalid assignment 2 mark for student Invalid practical mark for student Invalid exam mark for student	Success

			Title- Dr Name- Meg Tom Student Number- 4981 Date of Birth- 14/8/2010 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- N	

Result of programing testing

TestCase 1:

Enter a student: 12345

Title- Miss

Name- Tom Rox

Student Number- 12345

Date of Birth- 1/4/2020

Student type- CourseWorkStudent

assignment 1 mark- 100

assignment 2 mark- 100

practical work mark- 20

exam mark- 100

overall mark- 100

overall mark- HD

TestCase 2:

Enter a student: 4011

Title- Dr

Name- Tom Mcdonald

Student Number- 4011

Date of Birth- 16/11/2015

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

TestCase 3:

Enter a student: 9119

Error: Invalid assignment 1 mark for student

Error: Invalid assignment 2 mark for student

Error: Invalid practical mark for student

Error: Invalid exam mark for student

Title- Dr

Name- PEppA DOm

Student Number- 9119

Date of Birth- 12/2/2009

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

TestCase 4:

Enter a student: 4981

Error: Invalid assignment 1 mark for student

Error: Invalid assignment 2 mark for student

Error: Invalid practical mark for student

Error: Invalid exam mark for student

Title- Dr

Name- Meg Tom

Student Number- 4981

Date of Birth- 14/8/2010

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

Test Table: Client program

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	Client states they are dealing with research students by entering lowercase r Client select option 1	r 1	Farewell! Exit menu	Success
2	Client states they are dealing with coursework students by entering lowercase c. Client select out of range (invalid) option	c 11	Invalid option! Enter an option:	Success

Result of programing testing

TestCase 1:

Type C if you're dealing with coursework students OR R if you are dealing with reseach students

r

Enter an option: 1

Farewell! Exit menu

TestCase 2:

Type C if you're dealing with coursework students OR R if you are dealing with reseach students

c

Enter an option: 11

Invalid option!

Enter an option:

Test Table: Client program – Option 2 (Add all marks information about a coursework or research student)

NOTE: option 5 often used to show the effects

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	Client states they are dealing with research students Add proposalMark, oralPresenationMark and thesisMark that is out of range out for a student	Mr Raphel Nadal 59822 20/12/2005 REsEarChStudent 59822 -1 -1 -100 r 2 5	Invalid proposal mark for student Invalid oral presenation mark for student Invalid thesis mark for student Title- Mr Name- Raphel Nadal Student Number- 59822 Date of Birth- 20/12/2005 Student type- ResearchStudent Proposal mark- 0 Oral presenation mark- 0 Thesis mark- 0 overall mark- 0 overall mark- N	Success
2	Client states they are dealing with coursework students Add assignment1Mark, assignment2Mark, practicalMark and examMark that is out of range for a student	Dr PEppA DOm 9119 12/2/2009 CoURsEWorKStudent 9119 101 101 21 101 C 2 5	Invalid assignment 1 mark for student Invalid assignment 2 mark for student Invalid practical mark for student Invalid exam mark for student Title- Dr Name- PEppA DOm Student Number- 9119 Date of Birth- 12/2/2009 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- N	Success
3	Client states they are dealing with research students Add proposalMark, oralPresenationMark and thesisMark for a student that doesn't exist	22222 50 10 50 R 2	student 22222 can't be found in arrayList	Success

Result of programing testing

TestCase 1:

Type C if you're dealing with coursework students OR R if you are dealing with research students

r

Enter an option: 2

Error: Invalid proposal mark for student

Error: Invalid oral presenation mark for student

Error: Invalid proposal mark for student

Error: Invalid oral presenation mark for student

Error: Invalid thesis mark for student

Enter an option: 5

Title- Mr

Name- Raphel Nadal

Student Number- 59822

Date of Birth- 20/12/2005

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- N

TestCase 2:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 2

Error: Invalid assignment 1 mark for student

Error: Invalid assignment 1 mark for student

Error: Invalid assignment 2 mark for student

Error: Invalid practical mark for student

Error: Invalid exam mark for student

Error: Invalid assignment 1 mark for student

Error: Invalid assignment 2 mark for student

Error: Invalid practical mark for student

Error: Invalid exam mark for student

Enter an option: 5

Title- Dr

Name- PEppA DOm

Student Number- 9119

Date of Birth- 12/2/2009

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

TestCase 3:

Type C if you're dealing with coursework students OR R if you are dealing with research students

r

Enter an option: 2

Exception: student 22222 can't be found in arrayList

Enter an option:

Test Table: Client program – Option 3 (Remove student from arrayList from student number)

NOTE: option 5 is often used to show the effects

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	Client states they are dealing with research students Remove student that doesn't have mark information set	R 5 3 62315 y 5	Enter the student number identifying the student you wish to delete: Are you sure you want to remove Rob Potter Student ID- 62315 (Y/N)?	Success
2	Client states they are dealing with courseworkstudents students Remove student that has mark information set	C 2 5 3 33170193 Yep 5	Enter the student number identifying the student you wish to delete: Are you sure you want to remove Jin Chong Student ID- 33170193 (Y/N)?	Success
3	Client states they are dealing with research students Remove student that doesn't exist	R 3 222222	Student number entered does not exist	Success
4	Client states they are dealing with courseworkstudents students Remove a student but don't confirm	C 3 33170193 Nope	Student not removed	Success
5	Client states they are dealing with courseworkstudents students Remove a student but confirmation input is not character Y or N	C 3 33170193 z	Student not removed	Success

Result of programing testing

TestCase 1:

Type C if you're dealing with coursework students OR R if you are dealing with research students

R

Enter an option: 5

Title- Doc

Name- Rob Potter

Student Number- 62315

Date of Birth- 4/3/2000

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- N

Enter an option: 3

Enter the student number identifying the student you wish to delete:

62315

Are you sure you want to remove Rob Potter Student ID- 62315 (Y/N)?

Y

Enter an option: 5

Title- Mr

Name- Buck tanner

Student Number- 101746

Date of Birth- 10/5/2030

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- N

TestCase 2:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 2

Enter an option: 5

Title- Mr

Name- Jin Chong

Student Number- 33170193

Date of Birth- 10/2/2004

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 60

practical work mark- 14

exam mark- 50

overall mark- 44

overall mark- N

Enter an option: 3

Enter the student number identifying the student you wish to delete:

33170193

Are you sure you want to remove Jin Chong Student ID- 33170193 (Y/N)?

Yep

Enter an option: 5

Title- Miss

Name- Tom Rox

Student Number- 12345

Date of Birth- 1/4/2020

Student type- CourseWorkStudent

assignment 1 mark- 100

assignment 2 mark- 100

practical work mark- 20

exam mark- 100

overall mark- 100

overall mark- HD

TestCase 3:

Type C if you're dealing with coursework students OR R if you are dealing with research students

r

Enter an option: 3

Enter the student number identifying the student you wish to delete:

222222

Student number entered does not exist

Enter an option:

TestCase 4:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 3

Enter the student number identifying the student you wish to delete:

33170193

Are you sure you want to remove Jin Chong Student ID- 33170193 (Y/N)?

no

Student not removed

Enter an option:

TestCase 5:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 3

Enter the student number identifying the student you wish to delete:

33170193

Are you sure you want to remove Jin Chong Student ID- 33170193 (Y/N)?

zz

Student not removed

Enter an option:

Test Table: Client program – Option 4 and Option 5

Key: … means there are more students outputted.

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	Client states they are dealing with coursework students Client selects option 5 with all valid mark information for all existing coursework students added	C 2 5	Title- Mr Name- Jin Chong Student Number- 33170193 Date of Birth- 10/2/2004 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 60 practical work mark- 14 exam mark- 50 overall mark- 44 overall mark- N … Title- Dr Name- Meg Tom Student Number- 4981 Date of Birth- 14/8/2010 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- N	Success
	Client states they are dealing with coursework students Client selects option 5 with no mark information added to coursework students	C 5	Title- Mr Name- Jin Chong Student Number- 33170193 Date of Birth- 10/2/2004 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- N … Title- Dr Name- Meg Tom Student Number- 4981 Date of Birth- 14/8/2010 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- N	Success
3	Client states they are dealing with coursework students Client selects option 4 with no mark information added to any coursework students	C 4	Title- Mr Name- Jin Chong Student Number- 33170193 Date of Birth- 10/2/2004 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- None … Title- Mr Name- Rustle Ton Student Number- 3102742 Date of Birth- 21/3/2004 Student type- ResearchStudent Proposal mark- 0 Oral presenation mark- 0 Thesis mark- 0 overall mark- 0 overall mark- None	Success
4	Client states they are dealing with research students	C 2	Title- Mr Name- Jin Chong Student Number- 33170193 Date of Birth- 10/2/2004 Student type-	Success

	Client selects option 4 with mark information added to research students	4	CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- None ... Title- Mr Name- Rustle Ton Student Number- 3102742 Date of Birth- 21/3/2004 Student type- ResearchStudent Proposal mark- 50 Oral presenation mark- 10 Thesis mark- 50 overall mark- 50 overall mark- P	

Result of programing testing

TestCase 1:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 2

Title- Mr

Name- Jin Chong

Student Number- 33170193

Date of Birth- 10/2/2004

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 60

practical work mark- 14

exam mark- 50

overall mark- 44

overall mark- N

Title- Miss

Name- Tom Rox

Student Number- 12345

Date of Birth- 1/4/2020

Student type- CourseWorkStudent

assignment 1 mark- 100

assignment 2 mark- 100

practical work mark- 20

exam mark- 100

overall mark- 100

overall mark- HD

Title- Dr

Name- Tom Mcdonald

Student Number- 4011

Date of Birth- 16/11/2015

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

Title- Dr

Name- PEppA DOm

Student Number- 9119

Date of Birth- 12/2/2009

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

Title- Dr

Name- Meg Tom

Student Number- 4981

Date of Birth- 14/8/2010

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

Enter an option:

TestCase 2:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 5

Title- Mr

Name- Jin Chong

Student Number- 33170193

Date of Birth- 10/2/2004

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

Title- Miss

Name- Tom Rox

Student Number- 12345

Date of Birth- 1/4/2020

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

Title- Dr

Name- Tom Mcdonald

Student Number- 4011

Date of Birth- 16/11/2015

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

Title- Dr

Name- PEppA DOm

Student Number- 9119

Date of Birth- 12/2/2009

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

Title- Dr

Name- Meg Tom

Student Number- 4981

Date of Birth- 14/8/2010

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- N

Enter an option:

TestCase 3:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 4

Title- Mr

Name- Jin Chong

Student Number- 33170193

Date of Birth- 10/2/2004

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Title- Miss

Name- Tom Rox

Student Number- 12345

Date of Birth- 1/4/2020

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Title- Doc

Name- Rob Potter

Student Number- 62315

Date of Birth- 4/3/2000

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- None

Title- Mr

Name- Buck tanner

Student Number- 101746

Date of Birth- 10/5/2030

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- None

Title- Miss

Name- Tom Rox

Student Number- 636334

Date of Birth- 6/11/2030

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- None

Title- Miss

Name- Kava Dickson

Student Number- 20202

Date of Birth- 16/11/2030

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- None

Title- Dr

Name- Tom Mcdonald

Student Number- 4011

Date of Birth- 16/11/2015

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Title- Mr

Name- PEPpa Dom

Student Number- 2468

Date of Birth- 12/2/2009

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- None

Title- Dr

Name- PEppA DOm

Student Number- 9119

Date of Birth- 12/2/2009

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Title- Dr

Name- Meg Tom

Student Number- 2048

Date of Birth- 12/2/2009

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Title- Dr

Name- Meg Tom

Student Number- 4981

Date of Birth- 14/8/2010

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Title- Mr

Name- Raphel Nadal

Student Number- 59822

Date of Birth- 20/12/2005

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- None

Title- Mr

Name- Rustle Ton

Student Number- 3102742

Date of Birth- 21/3/2004

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- None

Enter an option:

[More to it]

Enter a student: 3102742

Title- Mr

Name- Rustle Ton

Student Number- 3102742

Date of Birth- 21/3/2004

Student type- ResearchStudent

Proposal mark- 50

Oral presenation mark- 10

Thesis mark- 50

overall mark- 50

overall mark- P

TestCase 4:

Type C if you're dealing with coursework students OR R if you are dealing with research students

r

Enter an option: 2

Enter an option: 4

Title- Mr

Name- Jin Chong

Student Number- 33170193

Date of Birth- 10/2/2004

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

...

Title- Mr

Name- Rustle Ton

Student Number- 3102742

Date of Birth- 21/3/2004

Student type- ResearchStudent

Proposal mark- 50

Oral presenation mark- 10

Thesis mark- 50

overall mark- 50

overall mark- P

Enter an option:

Test Table: Client program – Option 6

Key: … means there are more students outputted.

NOTE: Client must select option 2 prior to using option 6

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	Client states they are dealing with coursework students Selects option 6	C 2 6	Number of students above average- 2 Number of students below average- 4	Success
2	Client states they are dealing with coursework students Selects option 6	R 2 6	Number of students above average- 4 Number of students below average- 3	Success

Result of programing testing

TestCase 1:

Type C if you're dealing with coursework students OR R if you are dealing with research students

C

Enter an option: 6

Number of students above average- 2

Number of students below average- 4

TestCase 2:

Type C if you're dealing with coursework students OR R if you are dealing with research students

R

Enter an option: 6

Number of students above average- 4

Number of students below average- 3

Test Table: Client program – Option 7 and option 8

Key: … means there are more students outputted.

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	Client enter existing number for courseWorkstudent	C 7 33170193	Title- Mr Name- Jin Chong Student Number- 33170193 Date of Birth- 10/2/2004 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- None	Success
2	Client enter existing number for ResearchStudent	C 7 62315	62315 Title- Doc Name- Rob Potter Student Number- 62315 Date of Birth- 4/3/2000 Student type- ResearchStudent Proposal mark- 0 Oral presenation mark- 0 Thesis mark- 0 overall mark- 0 overall mark- None	Success
3	Client enters student number not found	R 7 11	Student not found	Success
4	Client enters existing given name and surname, case insensitive	R 8 JiN CHoNG	Title- Mr Name- Jin Chong Student Number- 33170193 Date of Birth- 10/2/2004 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- None	Success
5	Multiple students having the same name	C 8 meg ToM	Title- Dr Name- Meg Tom Student Number- 2048 Date of Birth- 12/2/2009 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- None	Success

			Title- Dr Name- Meg Tom Student Number- 4981 Date of Birth- 14/8/2010 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- None	
6	Client enters student name not found	R 8 Taylor Nick	Student not found in arrayList	Success

Result of programing testing

TestCase 1:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 7

Enter a student number:

33170193

Title- Mr

Name- Jin Chong

Student Number- 33170193

Date of Birth- 10/2/2004

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Enter an option:

TestCase 2:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 7

Enter a student number:

62315

Title- Doc

Name- Rob Potter

Student Number- 62315

Date of Birth- 4/3/2000

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- None

Enter an option:

TestCase 3:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 7

Enter a student number:

11

Student not found

Enter an option:

TestCase 4:

Type C if you're dealing with coursework students OR R if you are dealing with research students

r

Enter an option: 8

Enter first name of student:

JiN

Enter last name of student

CHoNG

Title- Mr

Name- Jin Chong

Student Number- 33170193

Date of Birth- 10/2/2004

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Enter an option:

TestCase 5:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 8

Enter first name of student:

meg

Enter last name of student

ToM

Title- Dr

Name- Meg Tom

Student Number- 2048

Date of Birth- 12/2/2009

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Title- Dr

Name- Meg Tom

Student Number- 4981

Date of Birth- 14/8/2010

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Enter an option:

TestCase 6:

Type C if you're dealing with coursework students OR R if you are dealing with research students

R

Enter an option: 8

Enter first name of student:

Taylor

Enter last name of student

Nick

Student not found in arrayList

Enter an option:

Test Table: Client program – Option 9 and 10

Key: … means there are more students outputted.

Test #	Test description	Inputs	Expected outputs	Success/Failure
1	Sort arrayList into ascending order of student ID and output	c 9	Title- Dr Name- Meg Tom Student Number- 2048 Date of Birth- 12/2/2009 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0 practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- None Title- Mr Name- PEPpa Dom Student Number- 2468 Date of Birth- 12/2/2009 Student type- ResearchStudent Proposal mark- 0 Oral presenation mark- 0 Thesis mark- 0 overall mark- 0 overall mark- None … Title- Mr Name- Rustle Ton Student Number- 3102742 Date of Birth- 21/3/2004 Student type- ResearchStudent Proposal mark- 0 Oral presenation mark- 0 Thesis mark- 0 overall mark- 0 overall mark- None Title- Mr Name- Jin Chong Student Number- 33170193 Date of Birth- 10/2/2004 Student type- CourseWorkStudent assignment 1 mark- 0 assignment 2 mark- 0	Success

			practical work mark- 0 exam mark- 0 overall mark- 0 overall mark- None	
2	Output sorted arrayList. Client forget to sort or select option 9. Therefore, program will also do option 9	R 10	Finished writing to file	Success

Result of programing testing

TestCase 1:

Type C if you're dealing with coursework students OR R if you are dealing with research students

c

Enter an option: 9

Title- Dr

Name- Meg Tom

Student Number- 2048

Date of Birth- 12/2/2009

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Title- Mr

Name- PEPpa Dom

Student Number- 2468

Date of Birth- 12/2/2009

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- None

...

Title- Mr

Name- Rustle Ton

Student Number- 3102742

Date of Birth- 21/3/2004

Student type- ResearchStudent

Proposal mark- 0

Oral presenation mark- 0

Thesis mark- 0

overall mark- 0

overall mark- None

Title- Mr

Name- Jin Chong

Student Number- 33170193

Date of Birth- 10/2/2004

Student type- CourseWorkStudent

assignment 1 mark- 0

assignment 2 mark- 0

practical work mark- 0

exam mark- 0

overall mark- 0

overall mark- None

Enter an option:

TestCase 2:

Type C if you're dealing with coursework students OR R if you are dealing with research students

R

Enter an option: 2

...

Enter an option: 10

Finished writing to file

Enter an option:

Source program listing

Java source code for client program (Client.java)-

```
/*
 * Student Marks and Information Tracker
 * By: Jin Cherng Chong
 * 23/10/2020
 * Files: Client.java, Student.java (base class), CourseWorkStudent.java(subclass name), ResearchStudent.java (subclass name),
 * CourseWorkStudentMark.txt, researchStudentMark.txt and student.txt
 * This program keeps track of different types students at a university and their marks they obtain in a year for a generic unit
 *
*/

package ict167assignment2;

/**
 *
 * @author 33170193
 */
import java.util.*;
import java.io.*;

public class Client {

   public static void main(String[] args) {
       StudentInfo();
       int studentNo = 1;
       ArrayList<Student> studentList = new ArrayList<Student>(); //Create array list of students

       Scanner inputStream = null;
       char studentType = 'A';
       String typeOfStudent;

       studentType = GetTypeOfStudent();

       try {
           inputStream = new Scanner(new File("student.txt")); //Opens the student.txt which contains list of students
       } catch (FileNotFoundException e) {
```

```
            System.out.println("Error opening file");

            System.exit(0); //Exit program when student.txt can't be opened

        }


        while (inputStream.hasNext()) {


            Student student = null;  //NOTE: Declared new student object outside if-statement so it can be accessed. Both object courseWorkStudent and ResearchStudent ARE students

            Boolean invalidStudentInformation = true;


            String title = inputStream.next(); //Get the title, firstname, lastname and student number from student.txt

            String firstName = inputStream.next();

            String lastName = inputStream.next();

            long studentNum = Long.parseLong(inputStream.next());


            String DOB = inputStream.next(); //Get date of birth from student.txt

            String[] splitDOB = DOB.split("/");

            int day = Integer.parseInt(splitDOB[0]);

            int month = Integer.parseInt(splitDOB[1]);

            int year = Integer.parseInt(splitDOB[2]);


            typeOfStudent = inputStream.next(); //Get student type from student.txt


            if (typeOfStudent.equalsIgnoreCase("CourseWorkStudent")) {

                student = new CourseWorkStudent(); //NOTE: Why can I assign class: courseWorkStudent() to class: student? Because courseWorkStudent extends student thus courseWorkStudent Is a Student

            } else if (typeOfStudent.equalsIgnoreCase("ResearchStudent")) {

                student = new ResearchStudent(); //NOTE: Why can't I declare inside if-statement i.e ResearchStudent student = new ResearchStudent();? Because can't be accessed outside if-statement

            } else {

                System.out.printf("Error: Incorrect student type specified for student \n");

            }


            if (student != null) { //Validate whether CourseWorkStudent OR ResearchStudent has been created. A null means neither has been created

                student.SetTitle(title);

                student.SetFirstName(firstName);

                student.SetLastName(lastName);

                student.SetStudentNum(studentNum);

                student.SetDateOfBirth(day, month, year);

                student.SetStudentType(typeOfStudent);


                invalidStudentInformation = (student.GetTitle().equals("None") || student.GetFirstName().equals("None") || student.GetLastName().equals("None") || student.GetStudentNum() == 0 || student.GetDay() == 0 || student.GetMonth() == 0 || student.GetYear() == 0 || student.GetStudentType().equals("None"));
```

```
        }

        if (invalidStudentInformation) {
            System.out.printf("Error: Invalid information for student %d. Therefore, the program will not save the student information \n", studentNo);
            System.out.println();
        } else {
            studentList.add(student); //Add the student to array list
        }

        studentNo++; //Keep track new students in student.txt

    }

    inputStream.close();

    // studentList.get(1).WriteRecord();
    DspMenu(studentList, studentType);

    ////////////////////////////////////////////////////////////////////////////Class Student checking??
}

public static char GetTypeOfStudent() { //Method that gets the studentType that client wants to work with

    Boolean invalidStudentType = true;
    char studentType = 'A';
    Scanner keyboard = new Scanner(System.in);

    do {

        System.out.println("Type C if you're dealing with coursework students OR R if you are dealing with research students");
        studentType = keyboard.nextLine().charAt(0); //Gets the character input
        studentType = Character.toUpperCase(studentType); //Always converts the character input to uppercase

        if (studentType == 'C' || studentType == 'R') { //Checks whether character input is one of the valid student types
            invalidStudentType = false;
        } else {
            System.out.println("Invalid option!");
        }

    } while (invalidStudentType); //Keeps looping if the student type entered is not valid
```

```
    System.out.println("-----------------------------------------------------");

    return studentType;
}

public static void DspMenu(ArrayList<Student> studentList, char studentType) { //Method that displays a menu to the client

    int option = 0;
    ArrayList<Student> officalStudentList = studentList;
    Scanner keyboard = new Scanner(System.in);

    while (option != 1) { //Stops displaying the menu when the option entered is 1

        System.out.print("Enter an option: ");
        option = keyboard.nextInt();
        System.out.println();

        switch (option) {
            case 1:
                System.out.println("Farewell! Exit menu");
                break;
            case 2:
                officalStudentList = SelectOption2(officalStudentList, studentType);
                //officalStudentList.get(1).WriteRecord();
                break;
            case 3:
                officalStudentList = SelectOption3(officalStudentList);
                //officalStudentList.get(1).WriteRecord();
                break;
            case 4:
                SelectOption4(officalStudentList);
                break;
            case 5:
                officalStudentList = SelectOption5(officalStudentList, studentType);
                //officalStudentList.get(1).WriteRecord();
                break;
            case 6:
                SelectOption6(officalStudentList, studentType);
                break;
            case 7:
                SelectOption7(officalStudentList);
```

```
                break;
            case 8:
                SelectOption8(officalStudentList);
                break;
            case 9:
                officalStudentList = SelectOption9(officalStudentList);
                SelectOption4(officalStudentList); //Output sorted array
                break;
            case 10:
                officalStudentList = SelectOption9(officalStudentList); //Sort array just in case it hasn't been sorted yet
                SelectOption10(officalStudentList);
                break;
            default:
                System.out.println("Invalid option!");
        }
    }

}

    public static ArrayList SelectOption2(ArrayList<Student> officalStudentList, char studentType) { //Method that adds all the marks for all the students of a particular student type

    long studentNum = 0;
    long num = 0;
    int overallMark = 0;
    String overallGrade;
    Boolean studentNotInStudentList = true;

    Scanner inputStreamCw = null;
    Scanner inputStreamRs = null;

    try { //Open coursework mark txt file
        inputStreamCw = new Scanner(new File("courseWorkStudentMark.txt")); //Opens the student.txt which contains list of students
    } catch (FileNotFoundException e) {
        System.out.println("Error opening file");
        System.exit(0); //Exit program when courseWorkStudentMark.txt can't be opened
    }

    try {  //Open researchstudent mark txt file
        inputStreamRs = new Scanner(new File("researchStudentMark.txt")); //Opens the student.txt which contains list of students
    } catch (FileNotFoundException e) {
```

```
            System.out.println("Error opening file");
            System.exit(0); //Exit program when researchStudentMark.txt can't be opened
        }

        while (inputStreamCw.hasNext() && studentType == 'C') { //Loop through coursework mark file

            studentNotInStudentList = true;
            num = Long.parseLong(inputStreamCw.next());

            try {

                for (Student person : officalStudentList) { //Loop through all the students in the student arrayList
                    studentNum = person.GetStudentNum();

                    if (num == studentNum) {

                        CourseWorkStudent CourseWorkStudent = (CourseWorkStudent) person; //NOTE: Downcasting: Casting super class (Student) --> Sub class
(courseWorkStudent)

                        int assignment1Mark = Integer.parseInt(inputStreamCw.next());
                        CourseWorkStudent.SetAssignment1Mark(assignment1Mark);

                        int assignment2Mark = Integer.parseInt(inputStreamCw.next());
                        CourseWorkStudent.SetAssignment2Mark(assignment2Mark);

                        int practicalMark = Integer.parseInt(inputStreamCw.next());
                        CourseWorkStudent.SetPracticalMark(practicalMark);

                        int examMark = Integer.parseInt(inputStreamCw.next());
                        CourseWorkStudent.SetExamMark(examMark);

                        overallMark = CourseWorkStudent.CalculateCwMark();

                        overallGrade = CalculateOverallGrade(overallMark);
                        CourseWorkStudent.SetFinalGrade(overallGrade);

                        studentNotInStudentList = false;
                    }
                }

                if (studentNotInStudentList) {
```

```
                throw new Exception();

            }


        } catch (Exception e) {

            System.out.printf("Exception: student %d can't be found in arrayList \n", num);

        }


        inputStreamCw.nextLine(); //NOTE: Why did I include nextLine when I'm checking hasNext right away? Because hasNext doesn't go to the new line. It goes to the next word

    }


    while (inputStreamRs.hasNext() && studentType == 'R') { //Loop through research student mark file


        studentNotInStudentList = true;

        num = Long.parseLong(inputStreamRs.next());


        try {


            for (Student person : officalStudentList) { //Loop through all the object in arrayList (officalStudentList)

                studentNum = person.GetStudentNum();


                if (num == studentNum) {

                    ResearchStudent ResearchStudent = (ResearchStudent) person; //NOTE: Downcasting: Casting super class (Student) --> Sub class (research student)


                    int proposalMark = Integer.parseInt(inputStreamRs.next());

                    ResearchStudent.SetProposalMark(proposalMark);


                    int oralPresenationMark = Integer.parseInt(inputStreamRs.next());

                    ResearchStudent.SetOralPresenationMark(oralPresenationMark);


                    int thesisMark = Integer.parseInt(inputStreamRs.next());

                    ResearchStudent.SetThesisMark(thesisMark);


                    overallMark = ResearchStudent.CalculateRsMark();


                    overallGrade = CalculateOverallGrade(overallMark);

                    ResearchStudent.SetFinalGrade(overallGrade);


                    studentNotInStudentList = false;

                }

            }
```

```
            if (studentNotInStudentList) {
                throw new Exception();
            }

        } catch (Exception e) {
            System.out.printf("Exception: student %d can't be found in arrayList \n", num);
        }

        inputStreamRs.nextLine(); //NOTE: Why did I include nextLine when I'm checking hasNext right away? Because hasNext doesn't go to the new line. It goes to the next word
    }

    inputStreamCw.close();
    inputStreamRs.close();
    return officalStudentList;

}

public static ArrayList SelectOption3(ArrayList<Student> officalStudentList) { //Method removes student from arrayList of students

    long clientNum = 0;
    long numOfStudent = 0;
    String firstNameOfStudent;
    String surnameOfStudent;
    char confirmation = 'N';
    Boolean studentNumExist = false;

    Scanner keyboard = new Scanner(System.in);
    Scanner keyboard2 = new Scanner(System.in);

    System.out.println("Enter the student number identifying the student you wish to delete: ");
    clientNum = keyboard.nextLong(); //Get the student number client want's to remove

    for (Student person : officalStudentList) { //Loop through all the students in the student arrayList

        firstNameOfStudent = person.GetFirstName();
        surnameOfStudent = person.GetLastName();
        numOfStudent = person.GetStudentNum();

        if (clientNum == numOfStudent) { //Check whether client entered student number matches current person actual student number
```

```
            System.out.printf("Are you sure you want to remove %s %s Student ID- %d (Y/N)? \n", firstNameOfStudent, surnameOfStudent, numOfStudent);
            confirmation = keyboard2.next().charAt(0); //This gets character from string
            confirmation = Character.toUpperCase(confirmation); //Converts all characters to uppercase for consistency
            studentNumExist = true;
        }

        if (confirmation == 'Y') { //Check whether client wants to actually delete
            officalStudentList.remove(person); //Remove person from arrayList (officalStudentList). Notice person is in for loop
            return officalStudentList;
        } else {
            System.out.println("Student not removed");
            return officalStudentList;
        }

    }

    if (!studentNumExist) {
        System.out.println("Student number entered does not exist");
    }
    return officalStudentList;
}

public static void SelectOption4(ArrayList<Student> officalStudentList) { ///Method that displays details of all students in the arrayList

    for (Student person : officalStudentList) { //Loops through all the students in the student list

        System.out.println("-------------------------------------------------------");
        person.WriteRecord(); //NOTE: WHY didn't I downcast? Because Polymorphism + overrriding. refer to lecture

    }

}

public static ArrayList SelectOption5(ArrayList<Student> officalStudentList, char studentType) { //Method that compute and output the overall mark + grade for either coursework or research students

    int overallMark = 0;
    Scanner inputStreamCw = null;
    Scanner inputStreamRs = null;
    String overallGrade;
    long num = 0;
```

```
        long studentNum = 0;

        try {
            inputStreamCw = new Scanner(new File("courseWorkStudentMark.txt")); //Opens the student.txt which contains list of students
        } catch (FileNotFoundException e) {
            System.out.println("Error opening file");
            System.exit(0); //Exit program when courseWorkStudentMark.txt can't be opened
        }

        try {
            inputStreamRs = new Scanner(new File("researchStudentMark.txt")); //Opens the student.txt which contains list of students
        } catch (FileNotFoundException e) {
            System.out.println("Error opening file");
            System.exit(0); //Exit program when researchStudentMark.txt can't be opened
        }

        while (inputStreamCw.hasNext() && studentType == 'C') { //Goes through all the courseWorkStudentMark.txt file

            num = Long.parseLong(inputStreamCw.next());

            for (Student person : officalStudentList) { //Loop through all the student's object in the arrayList

                studentNum = person.GetStudentNum();

                if (num == studentNum) { //When student number = current person number then set

                    CourseWorkStudent CourseWorkStudent = (CourseWorkStudent) person; //NOTE: Downcasting: Casting super class (Student) --> Sub class

                    overallMark = CourseWorkStudent.CalculateCwMark(); //Calculate overall coursework mark
                    overallGrade = CalculateOverallGrade(overallMark); //Calculate overall grade
                    CourseWorkStudent.SetFinalGrade(overallGrade); //store grade

                    CourseWorkStudent.WriteRecord();
                    System.out.println("-----------------------------------------------------");
                }
            }

            inputStreamCw.nextLine(); //NOTE: Why did I include nextLine when I'm checking hasNext right away? Because hasNext doesn't go to the new line. It goes to the next word
        }
```

```
        while (inputStreamRs.hasNext() && studentType == 'R') { //Goes through all the researchStudent.txt file

            num = Long.parseLong(inputStreamRs.next());

            for (Student person : officalStudentList) { //Loop through all the student's object in the arrayList

                studentNum = person.GetStudentNum();

                if (num == studentNum) { //When student number = current person number then set

                    ResearchStudent ResearchStudent = (ResearchStudent) person; //NOTE: Downcasting: Casting super class (Student) --> Sub class

                    overallMark = ResearchStudent.CalculateRsMark(); //Calculate overall coursework mark
                    overallGrade = CalculateOverallGrade(overallMark); //Calculate overall grade
                    ResearchStudent.SetFinalGrade(overallGrade); //store grade

                    ResearchStudent.WriteRecord();
                    System.out.println("-------------------------------------------------");
                }
            }

            inputStreamRs.nextLine(); //NOTE: Why did I include nextLine when I'm checking hasNext right away? Because hasNext doesn't go to the new line. It goes to the next word
        }

        inputStreamCw.close();
        inputStreamRs.close();

        return officalStudentList;

    }

    public static void SelectOption6(ArrayList<Student> officalStudentList, char studentType) { //Method  calculates No. of either courseWork Students or ResearchStudents above or below the courseWork students average Or researchStudents average
                                                                                                //Displays the results to client
        int mark = 0;
        int totalMarkCw = 0;
        int totalMarkRs = 0;
        int counterCw = 0;
        int counterRs = 0;
        Boolean correctStudentType = false;
        int averageAbove = 0;
```

```
        int averageBelow = 0;
        String type;
        int averageMarkCw = 0;
        int averageMarkRs = 0;

        for (Student person : officalStudentList) { //Loop that gathers total marks and number of course work and research students

            type = person.GetStudentType();

            if (type.equalsIgnoreCase("courseWorkStudent")) { //Check if object is a courseWork student
                CourseWorkStudent courseWorkStudent = (CourseWorkStudent) person; //NOTE: Downcasting: Casting super class (Student) --> Sub class CourseWorkStudent
                mark = courseWorkStudent.CalculateCwMark();
                totalMarkCw += mark; //Get total Marks of courseWork students
                counterCw++; //Keep track of number of courseWork students
            }

            if (type.equalsIgnoreCase("researchStudent")) { //Check if object is a research student
                ResearchStudent researchStudent = (ResearchStudent) person; //NOTE: Downcasting: Casting super class (Student) --> Sub class ResearchStudent
                mark = researchStudent.CalculateRsMark();
                totalMarkRs += mark; //Get total Marks of research students
                counterRs++; //Keep track of number of research students
            }
        }

        averageMarkCw = totalMarkCw / counterCw; //Calculates average mark of courseWork
        averageMarkRs = totalMarkRs / counterRs; //Calculates average mark of researchStudents
                                                 //NOTE: WHY not use for-each? Because can't combine to conditions
        for (int i = 0; i < officalStudentList.size() && studentType == 'C'; i++) { //Loops through if studentType selected at the start was C

            correctStudentType = false;

            Student person = officalStudentList.get(i); //NOTE: why not just downcast? Because person isn't getting retrieved through for each
            type = person.GetStudentType(); //This gets the student in arrayList

            if (type.equalsIgnoreCase("CourseWorkStudent")) {
                CourseWorkStudent courseWorkStudent = (CourseWorkStudent) person; //NOTE: Downcasting: Casting super class (Student) --> Sub class CourseWorkStudent
                mark = courseWorkStudent.CalculateCwMark();
                correctStudentType = true;
            }
```

```
            if (mark >= averageMarkCw && correctStudentType) {
                averageAbove++;
            }
            if (mark <= averageMarkCw && correctStudentType) {
                averageBelow++;
            }
        }

        for (int i = 0; i < officalStudentList.size() && studentType == 'R'; i++) { //Loops through if studentType selected at the start was R
                                                    //NOTE: WHY not use for-each? Because can't combine to conditions
            correctStudentType = false;

            Student person = officalStudentList.get(i); //NOTE: why not just downcast? Because person isn't getting retrieved through for each
            type = person.GetStudentType();

            if (type.equalsIgnoreCase("ResearchStudent")) {
                ResearchStudent researchStudent = (ResearchStudent) person; //NOTE: Downcasting: Casting super class (Student) --> Sub class ResearchStudent
                mark = researchStudent.CalculateRsMark();
                correctStudentType = true;
            }

            if (mark >= averageMarkRs && correctStudentType) {
                averageAbove++;
            }
            if (mark <= averageMarkRs && correctStudentType) {
                averageBelow++;
            }
        }

        System.out.printf("Number of students above average- %d \n", averageAbove);
        System.out.printf("Number of students below average- %d \n", averageBelow);
    }

    public static void SelectOption7(ArrayList<Student> officalStudentList) { //Method that gets the client to enter a ID and display corresponding details
about the student

        long num = 0;
        long studentNum = 0;
        Boolean studentNotFound = true;

        Scanner keyboard = new Scanner(System.in);
```

```
        System.out.println("Enter a student number: ");
        num = keyboard.nextLong(); //Client enters student number

        for (Student person : officalStudentList) {
            studentNum = person.GetStudentNum(); //Get the person object number

            if (num == studentNum) { //compare the user num with the person object number
                person.WriteRecord(); //NOTE: WHY didn't I downcast? Because Polymorphism + overrriding. refer to lecture
                studentNotFound = false;
            }
        }

        if (studentNotFound) {
            System.out.println("Student not found");
        }

        System.out.println("-----------------------------------------------------");

    }

    public static void SelectOption8(ArrayList<Student> officalStudentList) { //Method that gets the client to enter first name and last name and display corresponding details about the student(s)

        String fName;
        String lName;
        String studentFName;
        String studentLName;
        Boolean studentNotFound = true;

        Scanner keyboard = new Scanner(System.in);
        Scanner keyboard2 = new Scanner(System.in);

        System.out.println("Enter first name of student: ");
        fName = keyboard.nextLine();

        System.out.println("Enter last name of student");
        lName = keyboard2.nextLine();

        for (Student person : officalStudentList) { //Loop through all student objects in arrayList (officalStudentList)
```

```
            studentFName = person.GetFirstName();

            studentLName = person.GetLastName();


            if (fName.equalsIgnoreCase(studentFName) && lName.equalsIgnoreCase(studentLName)) { //compare user entered first name and last name with object first and last name

                person.WriteRecord();  //NOTE: WHY didn't I downcast? Because Polymorphism + overrriding. refer to lecture

                studentNotFound = false;

                System.out.println("-----------------------------------------------------");

            }

        }


        if (studentNotFound) { //Output error message if student not found

            System.out.println("Student not found in arrayList");

        }

    }


    public static ArrayList SelectOption9(ArrayList<Student> officalStudentList) { //Method that sorts arrayList by student number. Uses Selection sort


        for (int i = 0; i < officalStudentList.size() - 1; i++) { //Outer loop goes through all elements and stops at last element.


            int indexOfUnsortedSmallest = i;


            for (int j = i + 1; j < officalStudentList.size(); j++) { //J starts at next element after i


                Student person = new Student();

                person = officalStudentList.get(indexOfUnsortedSmallest); //Note: why can't you store into integer? Because you must get object then access attributes

                long currentSmallNum = person.GetStudentNum();


                Student secondPerson = new Student();

                secondPerson = officalStudentList.get(j);

                long afterNum = secondPerson.GetStudentNum();


                if (afterNum < currentSmallNum) { //Stores current smallest element in a variable and goes through all elements are keeps comparing

                    indexOfUnsortedSmallest = j;

                }

            }


            Student tempPerson = new Student();

            tempPerson = officalStudentList.get(indexOfUnsortedSmallest); //Get object storing UnsortedSmallestNumber


            Student tempPerson2 = new Student();
```

```
        tempPerson2 = officalStudentList.get(i); //Get the object storing the current unsortedNumber location we are dealing with (first in line)

        officalStudentList.set(indexOfUnsortedSmallest, tempPerson2);    //This object goes in the spot where the object with UnsortedSmallestNumber used to be

        officalStudentList.set(i, tempPerson); //Once you store the object with the UnsortedSmallestNumber value into an object. It get's placed at the back of sorted list

      }

      return officalStudentList;

   }

   public static void SelectOption10(ArrayList<Student> officalStudentList) { //Method that writes sorted ArrayList to CSV

      String OutputFilePath = "output.csv";

      try {

         PrintWriter outputStream = new PrintWriter(OutputFilePath);

         outputStream.write("Title" + ","); //These statements write the headings into csv
         outputStream.write("Name" + ",");
         outputStream.write("Student Number" + ",");
         outputStream.write("Date of Birth" + ",");
         outputStream.write("StudentType" + "-" + ",");
         outputStream.write("OverallMark" + ",");
         outputStream.write("Grade" + ",");
         outputStream.write("Assessment1" + ",");
         outputStream.write("Assessment2" + ",");
         outputStream.write("Assessment3" + ",");
         outputStream.write("Assessment4" + ",");

         for (Student person : officalStudentList) { //Loops through all students in arrayList

            outputStream.write("\n");

            String title = person.GetTitle(); //Get the student information which is relevant for both courseWork and research Students
            String firstName = person.GetFirstName();
            String lastName = person.GetLastName();
            Long studentNum = person.GetStudentNum();
            int day = person.GetDay();
            int month = person.GetMonth();
            int year = person.GetYear();
```

```
            String studentType = person.GetStudentType();


            if (studentType.equalsIgnoreCase("courseWorkStudent")) { //Check whether we are dealing with courseWorkStudents

                CourseWorkStudent courseWorkStudent = (CourseWorkStudent) person; //NOTE: Downcasting: Casting super class (Student) --> Sub class CourseWorkStudent

                int assignment1 = courseWorkStudent.GetAssignment1Mark(); //Get information specific to CourseWorkStudent only

                int assignment2 = courseWorkStudent.GetAssignment2Mark();

                int practicalMark = courseWorkStudent.GetPracticalMark();

                int examMark = courseWorkStudent.GetExamMark();

                int overallMark = courseWorkStudent.CalculateCwMark();

                String finalGrade = CalculateOverallGrade(overallMark);


                outputStream.write(title + ","); //Writes all the information about courseWorkStudent to CSV

                outputStream.write(firstName + " " + lastName + ",");

                outputStream.write(studentNum + ",");

                outputStream.write(day + "/" + month + "/" + year + ",");

                outputStream.write(studentType + ",");

                outputStream.write(overallMark + ",");

                outputStream.write(finalGrade + ",");

                outputStream.write(assignment1 + ",");

                outputStream.write(assignment2 + ",");

                outputStream.write(practicalMark + ",");

                outputStream.write(examMark + ",");


            }


            if (studentType.equalsIgnoreCase("ResearchStudent")) { //Check whether we are dealing with researchStudents

                ResearchStudent researchStudent = (ResearchStudent) person; //NOTE: Downcasting: Casting super class (Student) --> Sub class researchStudent

                int proposalMark = researchStudent.GetProposalMark(); //Get information specific to researchStudents only

                int oralPresenationMark = researchStudent.GetOralPresenationMark();

                int thesisMark = researchStudent.GetThesisMark();

                int overallMark = researchStudent.CalculateRsMark();

                String finalGrade = CalculateOverallGrade(overallMark);


                outputStream.write(title + ","); //Writes all the information about ResearchStudent to CSV

                outputStream.write(firstName + " " + lastName + ",");

                outputStream.write(studentNum + ",");

                outputStream.write(day + "/" + month + "/" + year + ",");

                outputStream.write(studentType + ",");

                outputStream.write(overallMark + ",");

                outputStream.write(finalGrade + ",");
```

```
                outputStream.write(proposalMark + ",");
                outputStream.write(oralPresenationMark + ",");
                outputStream.write(thesisMark + ",");
            }
        }

        outputStream.close();
        System.out.println("Finished writing to file");
    } catch (IOException e) {
        System.out.println("Can't output to file");
    }

}

public static String CalculateOverallGrade(int overallMark) { //Method that calculates the grade given the overall mark received

    String overallGrade = null;

    if (overallMark < 0 || overallMark > 100) {
        System.out.println("Overall mark not valid");
    } else if (overallMark >= 80) {
        overallGrade = "HD";
    } else if (overallMark >= 70) {
        overallGrade = "D";
    } else if (overallMark >= 60) {
        overallGrade = "C";
    } else if (overallMark >= 50) {
        overallGrade = "P";
    } else if (overallMark >= 0) {
        overallGrade = "N";
    }

    return overallGrade;
}

 public static void StudentInfo() {

    System.out.println("Name: Jin Cherng Chong ");
    System.out.println("Student number: 33170193 ");
    System.out.println("Mode of enrolment: Internal ");
    System.out.println("Tutorial attendance day and time: Thursday 3:30pm");
```

```
        System.out.println("-------------------------------------------------");
    }

}
```

Java source code for student class(Student.java)-

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ict167assignment2;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class Student {

    private String title;
    private String firstName;
    private String lastName;
    private long studentNum;
    private int day;
    private int month;
    private int year;
    private String studentType;

    public Student() {

        title = "None";
        firstName = "None";
        lastName = "None";
        studentNum = 0;
        day = 0;
        month = 0;
        year = 0;
        studentType = "None";
```

```
    }

    public Student(String initialTitle, String initialFirstName, String initialLastName, long initialStudentNum, int initialDay, int initialMonth, int initialYear, String initialStudentType) {

        title = initialTitle;
        firstName = initialFirstName;
        lastName = initialLastName;
        studentNum = initialStudentNum;
        day = initialDay;
        month = initialMonth;
        year = initialYear;
        studentType = initialStudentType;
    }

    /**
     * Pre-condition: title is a string that is neither non null or empty
     * Post-condition: assigns the newTitle parameter string to the current
     * title instance variable or displays an error message
     *
     */
    public void SetTitle(String newTitle) {

        if (!newTitle.isEmpty() && newTitle != null) { //Validates whether the title set by the user is empty or null. An error will be outputed when either of those are true
            title = newTitle;
        } else {
            System.out.println("Error: Invalid title for student"); //Output error message when string in the parameter is empty or null
        }

    }

    /**
     * Pre-condition: firstName is a string that is neither non null or empty
     * Post-condition: assigns the newFirstName parameter string to the current
     * firstName instance variable or displays an error message
     *
     */
    public void SetFirstName(String newFirstName) {

        if (!newFirstName.isEmpty() && newFirstName != null) { //Validates whether the firstName set by the user is empty or null. An error will be outputed when either of those are true
```

```
        firstName = newFirstName;
    } else {
        System.out.println("Error: Invalid first name for student"); //Output error message when string in the parameter is empty or null
    }

}

/**
 * Pre-condition: lastName is a string that is neither non null or empty
 * Post-condition: assigns the newLastName parameter string to the current
 * lastName instance variable or displays an error message
 *
 *
 */
public void SetLastName(String newLastName) {

    if (!newLastName.isEmpty() && newLastName != null) { //Validates whether the lastName set by the user is empty or null. An error will be outputed when either of those are true
        lastName = newLastName;
    } else {
        System.out.println("Error: Invalid last name for student"); //Output error message when string in the parameter is empty or null
    }

}

/**
 * Pre-condition: newStudentNum is a long Post-condition: assigns the
 * newStudentNum parameter long to the current studentNum instance variable
 *
 *
 *
 */
public void SetStudentNum(long newStudentNum) { //Have to deal with duplicate student ID?

    studentNum = newStudentNum;

}

/**
 * Pre-condition: newDay is an integer, newMonth is an integer, and newYear
 * is an integer Post-condition: assigns the newDay Integer parameter to the
```

```
    * current day instance variable, assigns the newMonth Integer parameter to
    * the current month instance variable, and assigns the newYear Integer
    * parameter to the current year instance variable.
    *
    *
    *
    */
   public void SetDateOfBirth(int newDay, int newMonth, int newYear) {

      int oldDay = day;
      int oldMonth = month;

      Boolean maxTwentyNineDay = (newDay >= 1 && newDay <= 29); //Declares the minimum and maximum days for the months. Three variables needed as there are three different maximum days
      Boolean maxThrityDay = (newDay >= 1 && newDay <= 30);
      Boolean maxThrityOneDay = (newDay >= 1 && newDay <= 31);

      Boolean Feb = (newMonth == 2); //Identifies the month entered
      Boolean thrityDayMonth = (newMonth == 4 || newMonth == 6 || newMonth == 9 || newMonth == 11);
      Boolean thrityOneDayMonth = (newMonth == 1 || newMonth == 3 || newMonth == 5 || newMonth == 7 || newMonth == 8 || newMonth == 10 || newMonth == 12);

      if (newMonth >= 1 && newMonth <= 12) { //Check month entered is an actual month.
         month = newMonth;                  //NOTE: Why isn't day checked first? Because month determines day range
      } else {
         System.out.println("Error: Invalid month entered. Therefore, date of birth for student not set");
         return;
      }

      if (maxTwentyNineDay && Feb) { //Check day entered is compatible with month entered
         day = newDay;
      } else if (maxThrityDay && thrityDayMonth) {
         day = newDay;
      } else if (maxThrityOneDay && thrityOneDayMonth) {
         day = newDay;
      } else {
         System.out.println("Error: Invalid day for student. Therefore, date of birth for student not set");
         month = oldMonth;
         return; //returns early because it's pointless to furthur and we don't want to deal with a situation where year may be valid but month and day are not. Because then we have many different combinations
      }
```

```
        if (newYear >= 2000) { //Check year entered is a valid year. The program assumes youngest student is born in year 2000. No max year added since this program will function years into the future
            year = newYear;
        } else {
            System.out.println("Error: Invalid year entered for student. Therefore, date of birth for student not set");
            month = oldMonth;
            day = oldDay;
            return;
        }

    }


    /**
     * Pre-condition: studentType is a string that is either a researchStudent or courseWorkStudent
     * Post-condition: assigns the newStudentType parameter string to the current
     * studentType instance variable or displays an error message
     *
     *
     */

    public void SetStudentType(String newStudentType) {

        if (newStudentType.equalsIgnoreCase("CourseWorkStudent")) {
            studentType = "CourseWorkStudent";
        } else if (newStudentType.equalsIgnoreCase("ResearchStudent")) {
            studentType = "ResearchStudent";
        } else {
            System.out.println("Error: Invalid student type for student");
        }

    }

    /**
     * Post-condition: returns the instance variable title as a string
     */
    public String GetTitle() {
        return title;
    }

    /**
```

```
 * Post-condition: returns the instance variable firstName as a string
 */
public String GetFirstName() {
    return firstName;
}

/**
 * Post-condition: returns the instance variable lastName as a string
 */
public String GetLastName() {
    return lastName;
}

/**
 * Post-condition: returns the instance variable studentNum as a long
 */
public long GetStudentNum() {
    return studentNum;
}

/**
 * Post-condition: returns the instance variable day as an integer
 */
public int GetDay() {
    return day;
}

/**
 * Post-condition: returns the instance variable month as an integer
 */
public int GetMonth() {
    return month;
}

/**
 * Post-condition: returns the instance variable year as an integer
 */
public int GetYear() {
    return year;
}
```

```
    /**
     * Post-condition: returns the instance variable studentType as a string
     */
    public String GetStudentType() {
        return studentType;
    }


    /**
     * Pre-condition: otherStudent is a student object Post-condition: Returns a
     * boolean value indicating whether the two objects have the same name and
     * date of birth
     *
     *
     *
     */
    public boolean IsEqual(Student otherStudent) {

        Boolean sameName = (this.firstName.equalsIgnoreCase(otherStudent.firstName) && this.lastName.equalsIgnoreCase(otherStudent.lastName)); //Verify whether two object have same name
        Boolean sameDOB = (this.day == otherStudent.day && this.month == otherStudent.month && this.year == otherStudent.year); //Verify whether two objects have same date of birth

        if (sameName && sameDOB) { //Check whether two student objects have same name and date of birth. Return true when they do have same name and date of birth
            return true;
        } else {
            return false;
        }

    }

    /**
     * Pre-condition: object must be instantiated Post-condition: displays the
     * initial (unset) instance variables of the object
     *
     *
     */
    public void WriteRecord() {

        System.out.printf("Title- %s \n", title);
        System.out.printf("Name- %s %s \n", firstName, lastName);
```

```
        System.out.printf("Student Number- %d \n", studentNum);
        System.out.printf("Date of Birth- %d/%d/%d \n", day, month, year);
        System.out.printf("Student type- %s \n", studentType);

    }

    public static void main(String[] args) { //driver method for test purposes only
        //driver method for test purposes only

        int studentNo = 1;
        ArrayList<Student> studentList = new ArrayList<Student>(); //Create array list of students

        Scanner inputStream = null;
        char studentType = 'A';
        String typeOfStudent;

        try {
            inputStream = new Scanner(new File("student.txt")); //Opens the student.txt which contains list of students
        } catch (FileNotFoundException e) {
            System.out.println("Error opening file");
            System.exit(0); //Exit program when student.txt can't be opened
        }

        while (inputStream.hasNext()) {

            Student student = null;  //NOTE: Declared new student object outside if-statement so it can be accessed. Both object courseWorkStudent and
ResearchStudent ARE students
            Boolean invalidStudentInformation = true;

            String title = inputStream.next(); //Get the title, firstname, lastname and student number from student.txt
            String firstName = inputStream.next();
            String lastName = inputStream.next();
            long studentNum = Long.parseLong(inputStream.next());

            String DOB = inputStream.next(); //Get date of birth from student.txt
```

```
            String[] splitDOB = DOB.split("/");

            int day = Integer.parseInt(splitDOB[0]);

            int month = Integer.parseInt(splitDOB[1]);

            int year = Integer.parseInt(splitDOB[2]);



            typeOfStudent = inputStream.next(); //Get student type from student.txt



            if (typeOfStudent.equalsIgnoreCase("CourseWorkStudent")) {

               student = new CourseWorkStudent(); //NOTE: Why can I assign class: courseWorkStudent() to class: student? Because courseWorkStudent extends student thus courseWorkStudent Is a Student

            } else if (typeOfStudent.equalsIgnoreCase("ResearchStudent")) {

               student = new ResearchStudent(); //NOTE: Why can't I declare inside if-statement i.e ResearchStudent student = new ResearchStudent();? Because can't be accessed outside if-statement

            } else {

               System.out.printf("Error: Incorrect student type specified for student \n");

            }



            if (student != null) { //Validate whether CourseWorkStudent OR ResearchStudent has been created. A null means neither has been created

               student.SetTitle(title);

               student.SetFirstName(firstName);

               student.SetLastName(lastName);

               student.SetStudentNum(studentNum);

               student.SetDateOfBirth(day, month, year);

               student.SetStudentType(typeOfStudent);



               invalidStudentInformation = (student.GetTitle().equals("None") || student.GetFirstName().equals("None") || student.GetLastName().equals("None") || student.GetStudentNum() == 0 || student.GetDay() == 0 || student.GetMonth() == 0 || student.GetYear() == 0 || student.GetStudentType().equals("None"));

            }



            if (invalidStudentInformation) {

               System.out.printf("Error: Invalid information for student %d. Therefore, the program will not save the student information \n", studentNo);

               System.out.println();

            } else {

               studentList.add(student); //Add the student to array list

            }



            studentNo++; //Keep track new students in student.txt



         }



         inputStream.close();
```

```
 //To add: Output all the students in array list + Way to test isEqual method + a way to test SetStudentType!!!
//studentList.get(0).WriteRecord();

  for (Student person : studentList) { //Loops through all the students in the student list

     System.out.println("-------------------------------------------------");
     person.WriteRecord(); //NOTE: WHY didn't I downcast? Because Polymorphism + overrriding. refer to lecture

}



      Student student1 = new Student(); //Testing isEqual
      Student student2 = new Student();
      Student student3 = new Student();
      Student student4 = new Student();
      student1 = studentList.get(7);
      student2 = studentList.get(8);
      student3 = studentList.get(9);
      student4 = studentList.get(10);

      Boolean equal = student1.IsEqual(student2);
      Boolean onlySameDOB = student2.IsEqual(student3);
      Boolean onlySameName = student3.IsEqual(student4);

      if(equal) {
          System.out.println("Same name and DOB");
      } else {
          System.out.println("No not equal");
      }

      if(onlySameDOB) {
          System.out.println("Same name and DOB");
      } else {
          System.out.println("No not equal");
      }

      if(onlySameName) {
          System.out.println("Same name and DOB");
      } else {
```

```
                System.out.println("No not equal");
            }



    }


    //Driver method for test purposes only
}
```

Java source code for CourseWorkStudent class(CourseWorkStudent.java)-

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ict167assignment2;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

/**
 *
 * @author Admin
 */
public class CourseWorkStudent extends Student {

    private int assignment1Mark;
    private int assignment2Mark;
    private int practicalMark;
    private int examMark;
    private int overallMark;
    private String finalGrade;

    public CourseWorkStudent() {

        super();
        assignment1Mark = 0;
        assignment2Mark = 0;
        practicalMark = 0;
        examMark = 0;
        overallMark = 0;
        finalGrade = "None";

    }
```

```
    public CourseWorkStudent(String initialTitle, String initialFirstName, String initialLastName, long initialStudentNum, int initialDay, int initialMonth, int initialYear, String initialStudentType, int initialAssignment1Mark, int initialAssignment2Mark, int initialPracticalMark, int initialExamMark, int initialOverallMark, String initialFinalGrade) {

        super(initialTitle, initialFirstName, initialLastName, initialStudentNum, initialDay, initialMonth, initialYear, initialStudentType);
        assignment1Mark = initialAssignment1Mark;
        assignment2Mark = initialAssignment2Mark;
        practicalMark = initialPracticalMark;
        examMark = initialExamMark;
        overallMark = initialOverallMark;
        finalGrade = initialFinalGrade;

    }

    /**
     * Pre-condition: newAssignment1Mark is an integer Post-condition: assigns
     * the newAssignment1Mark parameter integer to the current assignment1Mark
     * instance variable
     *
     *
     */
    public void SetAssignment1Mark(int newAssignment1Mark) {

        if (newAssignment1Mark >= 0 && newAssignment1Mark <= 100) { //Check assignment 1 mark entered is a valid mark
            assignment1Mark = newAssignment1Mark;
        } else {
            System.out.println("Error: Invalid assignment 1 mark for student");
        }

    }

    /**
     * Pre-condition: newAssignment2Mark is an integer Post-condition: assigns
     * the newAssignment2Mark parameter integer to the current assignment2Mark
     * instance variable
     *
     *
     */
    public void SetAssignment2Mark(int newAssignment2Mark) {

        if (newAssignment2Mark >= 0 && newAssignment2Mark <= 100) { //Check assignment 2 mark entered is a valid mark
            assignment2Mark = newAssignment2Mark;
```

```
    } else {
        System.out.println("Error: Invalid assignment 2 mark for student");
    }

}

/**
 * Pre-condition: newPracticalMark is an integer Post-condition: assigns the
 * newPracticalMark parameter integer to the current practicalMark instance
 * variable
 *
 *
 */
public void SetPracticalMark(int newPracticalMark) {

    if (newPracticalMark >= 0 && newPracticalMark <= 20) { //Check partical mark entered is a valid mark
        practicalMark = newPracticalMark;
    } else {
        System.out.println("Error: Invalid practical mark for student");
    }

}

/**
 * Pre-condition: newExamMark is an integer Post-condition: assigns the
 * newExamMark parameter integer to the current examMark instance variable
 *
 *
 */
public void SetExamMark(int newExamMark) {

    if (newExamMark >= 0 && newExamMark <= 100) { //Check exam mark entered is a valid mark
        examMark = newExamMark;
    } else {
        System.out.println("Error: Invalid exam mark for student");
    }

}

 /**
```

```
    * Pre-condition: newFinalGrade is a String Post-condition: assigns the

    * newFinalGrade parameter String to the current finalGrade instance variable

    *

    *

    */


   public void SetFinalGrade(String newFinalGrade) {




        if (newFinalGrade == "HD" || newFinalGrade == "D" || newFinalGrade == "P" || newFinalGrade == "C" || newFinalGrade == "N") { //Check final grade entered
is a valid grade

            finalGrade = newFinalGrade;

        } else {

            System.out.println("Error: Invalid final grade for student");

        }


   }









   /**

    * Post-condition: returns the instance variable title as a integer

    */

   public int GetAssignment1Mark() {

       return assignment1Mark;

   }


   /**

    * Post-condition: returns the instance variable firstName as a integer

    */

   public int GetAssignment2Mark() {

       return assignment2Mark;

   }


   /**

    * Post-condition: returns the instance variable lastName as a integer

    */
```

```
    public int GetPracticalMark() {

        return practicalMark;

    }


    /**

     * Post-condition: returns the instance variable lastName as a integer

     */

    public int GetExamMark() {

        return examMark;

    }


    /**

     * Pre-condition: otherStudent is a student object Post-condition: Returns a

     * boolean value indicating whether the two objects have the same name and

     * date of birth

     *

     */

    public int CalculateCwMark() {


        double weightedAssignment1Mark = (double) assignment1Mark / 100 * 25; //Must convert Integer to double. Double allows for dealing with decimal

        double weightedAssignment2Mark = (double) assignment2Mark / 100 * 25;

        double weightedPracticalMark = (double) practicalMark / 20 * 20;

        double weightedExamMark = (double) examMark / 100 * 30;




       overallMark = (int) (weightedAssignment1Mark + weightedAssignment2Mark + weightedPracticalMark + weightedExamMark); //Add all the weighted
assesments to get the overallMark

       double decimalInput = (weightedAssignment1Mark + weightedAssignment2Mark + weightedPracticalMark + weightedExamMark) - overallMark; //Get the
decimal to round


        if (decimalInput < 0.5) { //Check if decimal needs to be rounded down

            return overallMark;

        } else { //Check if decimal needs to be rounded up

            double decNumToRoundUp = 1 - decimalInput; //Determine the decimal needed to be added to make rounded the number up


            overallMark = (int) ((weightedAssignment1Mark + weightedAssignment2Mark + weightedPracticalMark + weightedExamMark)  + decNumToRoundUp);


            return overallMark;

        }


    }
```

```
/**
 * Pre-condition: object must be instantiated Post-condition: displays the
 * initial (unset) instance variables of the object
 *
 *
 */
public void WriteRecord() {

    System.out.printf("Title- %s \n", GetTitle());
    System.out.printf("Name- %s %s \n", GetFirstName(), GetLastName());
    System.out.printf("Student Number- %d \n", GetStudentNum());
    System.out.printf("Date of Birth- %d/%d/%d \n", GetDay(), GetMonth(), GetYear());
    System.out.printf("Student type- %s \n", GetStudentType());
    System.out.printf("assignment 1 mark- %d \n", assignment1Mark);
    System.out.printf("assignment 2 mark- %d \n", assignment2Mark);
    System.out.printf("practical work mark- %d \n", practicalMark);
    System.out.printf("exam mark- %d \n", examMark);
    System.out.printf("overall mark- %d \n", overallMark);
    System.out.printf("overall mark- %s \n", finalGrade);
}

public static void main(String[] args) { //driver method for test purposes only
    //driver method for test purposes only

    int studentNo = 1;
    ArrayList<Student> studentList = new ArrayList<Student>(); //Create array list of students

    Scanner inputStream = null;
    char studentType = 'C';
    String typeOfStudent;
    Scanner inputStreamCw = null;
    Scanner keyboard = new Scanner(System.in);
    Boolean studentNotInStudentList = true;
    int counter = 0;
    long num = 1;
    long studentNum = 1;
    long markStudentNum = 1;
    int studentLocation = 0;

    try {
        inputStream = new Scanner(new File("student.txt")); //Opens the student.txt which contains list of students
```

```
    } catch (FileNotFoundException e) {
        System.out.println("Error opening file");
        System.exit(0); //Exit program when student.txt can't be opened
    }

    while (inputStream.hasNext()) {

        Student student = null;  //NOTE: Declared new student object outside if-statement so it can be accessed. Both object courseWorkStudent and ResearchStudent ARE students
        Boolean invalidStudentInformation = true;

        String title = inputStream.next(); //Get the title, firstname, lastname and student number from student.txt
        String firstName = inputStream.next();
        String lastName = inputStream.next();
        studentNum = Long.parseLong(inputStream.next());

        String DOB = inputStream.next(); //Get date of birth from student.txt
        String[] splitDOB = DOB.split("/");
        int day = Integer.parseInt(splitDOB[0]);
        int month = Integer.parseInt(splitDOB[1]);
        int year = Integer.parseInt(splitDOB[2]);

        typeOfStudent = inputStream.next(); //Get student type from student.txt

        if (typeOfStudent.equalsIgnoreCase("CourseWorkStudent")) {
            student = new CourseWorkStudent(); //NOTE: Why can I assign class: courseWorkStudent() to class: student? Because courseWorkStudent extends student thus courseWorkStudent Is a Student
        } else if (typeOfStudent.equalsIgnoreCase("ResearchStudent")) {
            student = new ResearchStudent(); //NOTE: Why can't I declare inside if-statement i.e ResearchStudent student = new ResearchStudent();? Because can't be accessed outside if-statement
        } else {
            System.out.printf("Error: Incorrect student type specified for student \n");
        }

        if (student != null) { //Validate whether CourseWorkStudent OR ResearchStudent has been created. A null means neither has been created
            student.SetTitle(title);
            student.SetFirstName(firstName);
            student.SetLastName(lastName);
            student.SetStudentNum(studentNum);
            student.SetDateOfBirth(day, month, year);
            student.SetStudentType(typeOfStudent);
```

```
            invalidStudentInformation = (student.GetTitle().equals("None") || student.GetFirstName().equals("None") || student.GetLastName().equals("None") ||
student.GetStudentNum() == 0 || student.GetDay() == 0 || student.GetMonth() == 0 || student.GetYear() == 0|| student.GetStudentType().equals("None"));

        }


        if (invalidStudentInformation) {

            System.out.printf("Error: Invalid information for student %d. Therefore, the program will not save the student information \n", studentNo);

            System.out.println();

        } else {

            studentList.add(student); //Add the student to array list

        }


        studentNo++; //Keep track new students in student.txt


    }


    inputStream.close();


    System.out.print("Enter a student: ");

    num = keyboard.nextLong();


    try {

        inputStreamCw = new Scanner(new File("courseWorkStudentMark.txt")); //Opens the student.txt which contains list of students

    } catch (FileNotFoundException e) {

        System.out.println("Error opening file");

        System.exit(0); //Exit program when student.txt can't be opened

    }


    try {


        for (Student person : studentList) {


            studentNum = person.GetStudentNum();

            //System.out.println(studentNum);


            if (num == studentNum) {

                studentNotInStudentList = false;

                studentLocation = counter;

                // System.out.println("found!!!!");


            }
```

```
            counter++;

        }


        if (studentNotInStudentList) {

            throw new Exception();

        }


        while (inputStreamCw.hasNext() && studentType == 'C') {


            markStudentNum = Long.parseLong(inputStreamCw.next());


            if (num == markStudentNum) {


                Student temp = studentList.get(studentLocation);

                CourseWorkStudent CourseWorkStudent = (CourseWorkStudent) temp; //NOTE: Why don't I need to set i.e
officalStudentList.set(studentLocation, CourseWorkStudent);? Because list holds pointers to objects thus when you change something in temp you change it in
orginal object from list


                int assignment1Mark = Integer.parseInt(inputStreamCw.next());

                CourseWorkStudent.SetAssignment1Mark(assignment1Mark);


                int assignment2Mark = Integer.parseInt(inputStreamCw.next());

                CourseWorkStudent.SetAssignment2Mark(assignment2Mark);


                int practicalMark = Integer.parseInt(inputStreamCw.next());

                CourseWorkStudent.SetPracticalMark(practicalMark);


                int examMark = Integer.parseInt(inputStreamCw.next());

                CourseWorkStudent.SetExamMark(examMark);


                int overallMark= CourseWorkStudent.CalculateCwMark();

                String finalGrade = CalculateOverallGrade(overallMark);

                CourseWorkStudent.SetFinalGrade(finalGrade);

            }


            inputStreamCw.nextLine(); //NOTE: Why did I include nextLine when I'm checking hasNext right away? Because hasNext doesn't go to the new line.
It goes to the next word

        }


    } catch (Exception e) {

        System.out.println("Exception: student can't be found in arrayList");

    }
```

```
        //To add: Output all the students in array list + Way to test isEqual method!!!


         for (Student person : studentList) { //Loops through all the students in the student list


            System.out.println("-----------------------------------------------------");

            person.WriteRecord(); //NOTE: WHY didn't I downcast? Because Polymorphism + overrriding. refer to lecture


        }


    }


    public static String CalculateOverallGrade(int overallMark) { //driver method for test purposes only

        //driver method for test purposes only


        String overallGrade = null;


        if (overallMark < 0 || overallMark > 100) {

            System.out.println("Overall mark not valid");

        } else if (overallMark >= 80) {

            overallGrade = "HD";

        } else if (overallMark >= 70) {

            overallGrade = "D";

        } else if (overallMark >= 60) {

            overallGrade = "C";

        } else if (overallMark >= 50) {

            overallGrade = "P";

        } else if (overallMark >= 0) {

            overallGrade = "N";

        }


        return overallGrade;

    }


}
```

Java source code for ResearchStudent class(ResearchStudent.java)-

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ict167assignment2;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;

public class ResearchStudent extends Student {

    private int proposalMark;
    private int oralPresenationMark;
    private int thesisMark;
    private int overallMark;
    private String finalGrade;

    public ResearchStudent() {

        super();
        proposalMark = 0;
        oralPresenationMark = 0;
        thesisMark = 0;
        overallMark = 0;
        finalGrade = "None";
    }

    public ResearchStudent(String initialTitle, String initialFirstName, String initialLastName, long initialStudentNum, int initialDay, int initialMonth, int initialYear, String initialStudentType, int initialProposalMark, int initialOralPresenationMark, int initialThesisMark, int initialOverallMark, String initialFinalGrade) {

        super(initialTitle, initialFirstName, initialLastName, initialStudentNum, initialDay, initialMonth, initialYear, initialStudentType);
        proposalMark = initialProposalMark;
        oralPresenationMark = initialOralPresenationMark;
        thesisMark = initialThesisMark;
```

```
    overallMark = initialOverallMark;
    finalGrade = initialFinalGrade;
}

/**
 * Pre-condition: newProposalMark is an integer Post-condition: assigns the
 * newProposalMark parameter integer to the current proposalMark instance
 * variable
 *
 *
 */
public void SetProposalMark(int newProposalMark) {

    if (newProposalMark >= 0 && newProposalMark <= 100) { //Check proposal mark entered is a valid mark
        proposalMark = newProposalMark;
    } else {
        System.out.println("Error: Invalid proposal mark for student");
    }

}

/**
 * Pre-condition: newOralPresenationMark is an integer Post-condition:
 * assigns the newOralPresenationMark parameter integer to the current
 * oralPresenationMark instance variable
 *
 *
 */
public void SetOralPresenationMark(int newOralPresenationMark) {

    if (newOralPresenationMark >= 0 && newOralPresenationMark <= 20) { //Check oral presenation mark entered is a valid mark
        oralPresenationMark = newOralPresenationMark;
    } else {
        System.out.println("Error: Invalid oral presenation mark for student");
    }

}

/**
 * Pre-condition: newThesisMark is an integer Post-condition: assigns the
 * thesisMark parameter integer to the current thesisMark instance variable
```

```
     *
     *
     */
    public void SetThesisMark(int newThesisMark) {

        if (newThesisMark >= 0 && newThesisMark <= 100) { //Check thesis mark entered is a valid mark
            thesisMark = newThesisMark;
        } else {
            System.out.println("Error: Invalid thesis mark for student");
        }

    }

    /**
     * Pre-condition: newFinalGrade is a String Post-condition: assigns the
     * newFinalGrade parameter String to the current finalGrade instance
     * variable
     *
     *
     */
    public void SetFinalGrade(String newFinalGrade) {

        if (newFinalGrade == "HD" || newFinalGrade == "D" || newFinalGrade == "P" || newFinalGrade == "C" || newFinalGrade == "N") { //Check final grade entered
is a valid grade
            finalGrade = newFinalGrade;
        } else {
            System.out.println("Error: Invalid final grade for student");
        }

    }

    /**
     * Post-condition: returns the instance variable title as a integer
     */
    public int GetProposalMark() {
        return proposalMark;
    }

    /**
     * Post-condition: returns the instance variable firstName as a integer
     */
```

```
    public int GetOralPresenationMark() {
        return oralPresenationMark;
    }

    /**
     * Post-condition: returns the instance variable lastName as a integer
     */
    public int GetThesisMark() {
        return thesisMark;
    }

    public int CalculateRsMark() {

        double weightedProposalMark = (double) proposalMark / 100 * 30;
        double weightedOralPresenationMark = (double) oralPresenationMark / 20 * 10;
        double weightedThesisMark = (double) thesisMark / 100 * 60;

        overallMark = (int) (weightedProposalMark + weightedOralPresenationMark + weightedThesisMark); //Add all the weighted assesments to get the
overallMark

        double decimalInput = (weightedProposalMark + weightedOralPresenationMark + weightedThesisMark) - overallMark; //Get the decimal to round

        if (decimalInput < 0.5) { //Check if decimal needs to be rounded down
            return overallMark;
        } else { //Check if decimal needs to be rounded up
            double decNumToRoundUp = 1 - decimalInput; //Determine the decimal needed to be added to make rounded the number up

            overallMark = (int) ((weightedProposalMark + weightedOralPresenationMark + weightedThesisMark)  + decNumToRoundUp);

            return overallMark;
        }
    }

    /**
     * Pre-condition: object must be instantiated Post-condition: displays the
     * initial (unset) instance variables of the object
     *
     *
     */
    public void WriteRecord() {
```

```
        System.out.printf("Title- %s \n", GetTitle());
        System.out.printf("Name- %s %s \n", GetFirstName(), GetLastName());
        System.out.printf("Student Number- %d \n", GetStudentNum());
        System.out.printf("Date of Birth- %d/%d/%d \n", GetDay(), GetMonth(), GetYear());
        System.out.printf("Student type- %s \n", GetStudentType());
        System.out.printf("Proposal mark- %d \n", proposalMark);
        System.out.printf("Oral presenation mark- %d \n", oralPresenationMark);
        System.out.printf("Thesis mark- %d \n", thesisMark);
        System.out.printf("overall mark- %d \n", overallMark);
        System.out.printf("overall mark- %s \n", finalGrade);
    }

    public static void main(String[] args) { //driver method for test purposes only
        //driver method for test purposes only

        int studentNo = 1;
        ArrayList<Student> studentList = new ArrayList<Student>(); //Create array list of students

        Scanner inputStream = null;
        char studentType = 'R';
        String typeOfStudent;
        Scanner inputStreamRs = null;
        Scanner keyboard = new Scanner(System.in);
        Boolean studentNotInStudentList = true;
        int counter = 0;
        long num = 1;
        long studentNum = 1;
        long markStudentNum = 1;
        int studentLocation = 0;

        try {
            inputStream = new Scanner(new File("student.txt")); //Opens the student.txt which contains list of students
        } catch (FileNotFoundException e) {
            System.out.println("Error opening file");
            System.exit(0); //Exit program when student.txt can't be opened
        }

        while (inputStream.hasNext()) {

            Student student = null;  //NOTE: Declared new student object outside if-statement so it can be accessed. Both object courseWorkStudent and
ResearchStudent ARE students
```

```
        Boolean invalidStudentInformation = true;

        String title = inputStream.next(); //Get the title, firstname, lastname and student number from student.txt
        String firstName = inputStream.next();
        String lastName = inputStream.next();
        studentNum = Long.parseLong(inputStream.next());

        String DOB = inputStream.next(); //Get date of birth from student.txt
        String[] splitDOB = DOB.split("/");
        int day = Integer.parseInt(splitDOB[0]);
        int month = Integer.parseInt(splitDOB[1]);
        int year = Integer.parseInt(splitDOB[2]);

        typeOfStudent = inputStream.next(); //Get student type from student.txt

        if (typeOfStudent.equalsIgnoreCase("CourseWorkStudent")) {
            student = new CourseWorkStudent(); //NOTE: Why can I assign class: courseWorkStudent() to class: student? Because courseWorkStudent extends student thus courseWorkStudent Is a Student
        } else if (typeOfStudent.equalsIgnoreCase("ResearchStudent")) {
            student = new ResearchStudent(); //NOTE: Why can't I declare inside if-statement i.e ResearchStudent student = new ResearchStudent();? Because can't be accessed outside if-statement
        } else {
            System.out.printf("Error: Incorrect student type specified for student \n");
        }

        if (student != null) { //Validate whether CourseWorkStudent OR ResearchStudent has been created. A null means neither has been created
            student.SetTitle(title);
            student.SetFirstName(firstName);
            student.SetLastName(lastName);
            student.SetStudentNum(studentNum);
            student.SetDateOfBirth(day, month, year);
            student.SetStudentType(typeOfStudent);

            invalidStudentInformation = (student.GetTitle().equals("None") || student.GetFirstName().equals("None") || student.GetLastName().equals("None") || student.GetStudentNum() == 0 || student.GetDay() == 0 || student.GetMonth() == 0 || student.GetYear() == 0 || student.GetStudentType().equals("None"));
        }

        if (invalidStudentInformation) {
            System.out.printf("Error: Invalid information for student %d. Therefore, the program will not save the student information \n", studentNo);
            System.out.println();
        } else {
            studentList.add(student); //Add the student to array list
```

```
    }

    studentNo++; //Keep track new students in student.txt

}

inputStream.close();

System.out.print("Enter a student: ");
num = keyboard.nextLong();

try {
    inputStreamRs = new Scanner(new File("researchStudentMark.txt")); //Opens the student.txt which contains list of students
} catch (FileNotFoundException e) {
    System.out.println("Error opening file");
    System.exit(0); //Exit program when student.txt can't be opened
}

try {

    for (Student person : studentList) {

        studentNum = person.GetStudentNum();
        //System.out.println(studentNum);

        if (num == studentNum) {
            studentNotInStudentList = false;
            studentLocation = counter;
            // System.out.println("found!!!!");

        }

        counter++;
    }

    if (studentNotInStudentList) {
        throw new Exception();
    }

    while (inputStreamRs.hasNext() && studentType == 'R') {
```

```
            markStudentNum = Long.parseLong(inputStreamRs.next());


            if (num == markStudentNum) {


                Student temp = studentList.get(studentLocation);

                ResearchStudent ResearchStudent = (ResearchStudent) temp; //NOTE: Why don't I need to set i.e officalStudentList.set(studentLocation, CourseWorkStudent);? Because list holds pointers to objects thus when you change something in temp you change it in orginal object from list


                int proposalMark = Integer.parseInt(inputStreamRs.next());

                ResearchStudent.SetProposalMark(proposalMark);


                int oralPresenationMark = Integer.parseInt(inputStreamRs.next());

                ResearchStudent.SetOralPresenationMark(oralPresenationMark);


                int thesisMark = Integer.parseInt(inputStreamRs.next());

                ResearchStudent.SetThesisMark(thesisMark);


                int overallMark = ResearchStudent.CalculateRsMark();

                String finalGrade = CalculateOverallGrade(overallMark);

                ResearchStudent.SetFinalGrade(finalGrade);


            }


            inputStreamRs.nextLine(); //NOTE: Why did I include nextL
        }


    } catch (Exception e) {

        System.out.println("Exception: student can't be found in arrayList");

    }


    //To add: Output all the students in array list + Way to test isEqual method!!!


     for (Student person : studentList) { //Loops through all the students in the student list


        System.out.println("-------------------------------------------------");

        person.WriteRecord(); //NOTE: WHY didn't I downcast? Because Polymorphism + overrriding. refer to lecture


    }
```

```
    }

    public static String CalculateOverallGrade(int overallMark) { //driver method for test purposes only
        //driver method for test purposes only

        String overallGrade = null;

        if (overallMark < 0 || overallMark > 100) {
            System.out.println("Overall mark not valid");
        } else if (overallMark >= 80) {
            overallGrade = "HD";
        } else if (overallMark >= 70) {
            overallGrade = "D";
        } else if (overallMark >= 60) {
            overallGrade = "C";
        } else if (overallMark >= 50) {
            overallGrade = "P";
        } else if (overallMark >= 0) {
            overallGrade = "N";
        }

        return overallGrade;
    }

}
```