the UNIVERSITY of GREENWICH

COMP1752 – Object-Oriented Programming

[000938568]
University of Greenwich
3/31/2018

Table of Contents

1 INTRODUCTION

Please note that this report should **only** include the extra work completed for the coursework and **not** the work carried out for the lab sessions as required in the weekly tasks.

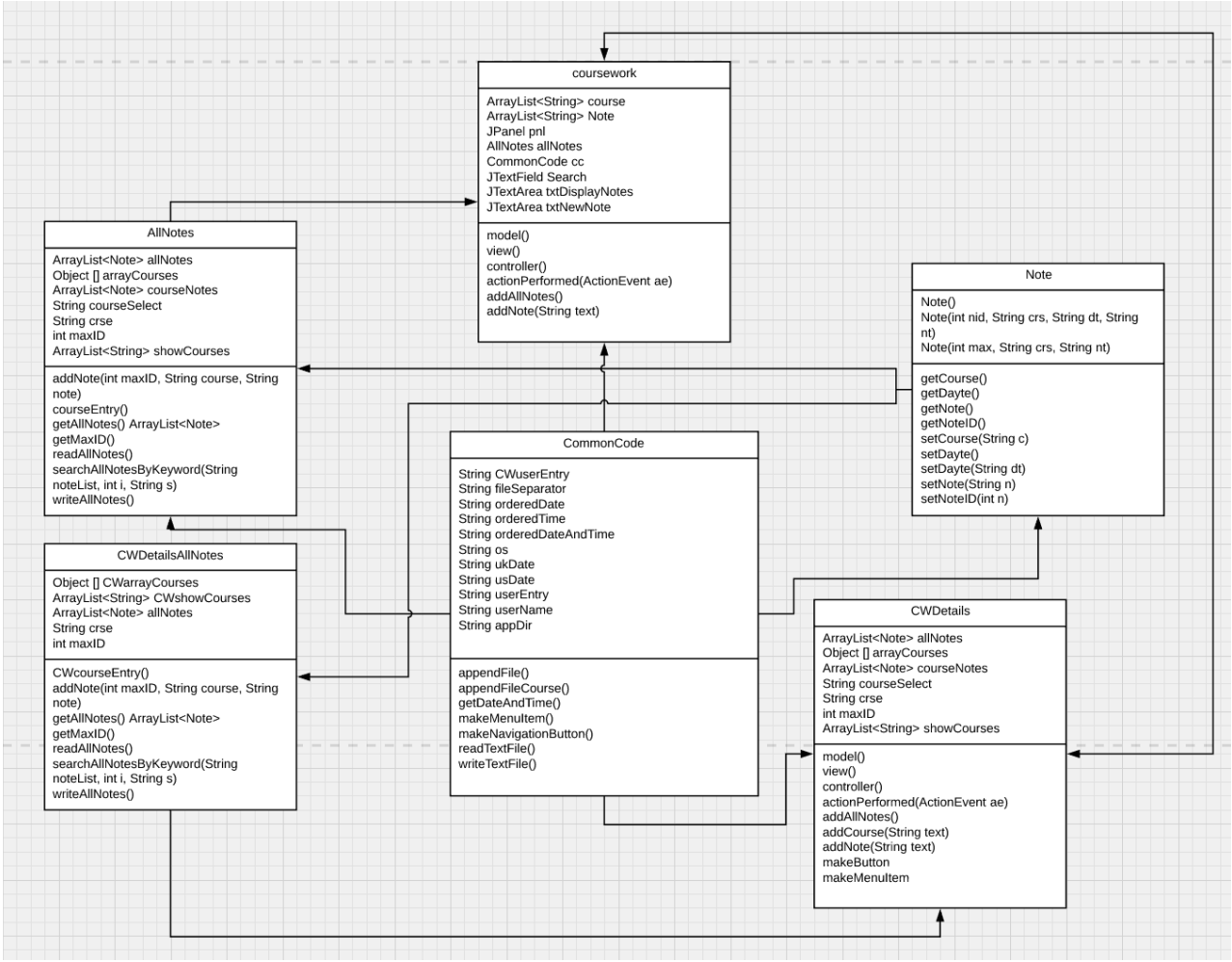
The Program that I have developed follows and delivers the key requirements provided in the Coursework Specification. These requirements include adding a course to the list of courses in a text file, also making relevant files for the new course, which can be accessed at a later date in the program so they can be edited by the user to assist them with writing notes during lectures.

The Program has been developed with being user-friendly in mind, this means I have attempted to ensure that the user cannot click or enter anything on the running program that will cause exceptions to occur or the program crashing. To ensure that it has been user-friendly I have added in statements to ensure that empty or null values cannot be entered into input fields, as these are likely to crash the program.
I have also solved the common issue of circular instantiation within the program, as switching between the two windows (coursework and the requirements page) would constantly create new instances of themselves, this is not a good idea for many reasons, so I have managed to fix this issue so that the user will not encounter this problem.

2 CLASS DIAGRAM

Include a UML class diagram showing the all the classes that you have implemented and the relationship between them. Your narrative for section 2.2 should also describe the design decisions you made and the OOP techniques used (200-400 words).

2.1 THE CLASS DIAGRAM



2.2 A Brief Explanation of the Diagram

The above diagram shows the relationships between the various classes within the coursework project, the diagram shows each classes respective members, including public variables and all of its functions, including public, private or protected functions.

CommonCode is inherited in many classes across this project; we can see using the diagram above that CommonCode is inherited to **four** different classes, these classes we can identify as

CWDetails

Coursework

Note

AllNotes

This means that these four classes above can also use the variables and functions, which are found in the CommonCode class. We can identify using the diagram above that the arrows in these cases only point in one direction; these are **outwards** from CommonCode into these classes, showing the direction of inheritance. This means that the four classes above cannot share their variables the other way with CommonCode, this only works with variables that are provided by CommonCode.

We can also identify that there is one line that extends into both directions, this line can be found on the right hand side of the diagram and connects the classes' coursework and CWDetails. These classes are able to access each other's data thanks to inheritance.
Both the classes' coursework and CWDetails to not output variables or functions to any other classes apart from each other, we can identify that both of these classes inherit variables and functions from CommonCode and their respective AllNotes classes (example being coursework inherits AllNotes whilst CWDetails inherits CWDetailsAllNotes.

Abstraction has been used in this Project as I have six classes, two of which have been created by myself.
Inheritance has been used extensively in this project as discussed above, with CommonCode being inherited across four classes in the project.
Encapsulation can be seen in the diagram above, especially in the Note method.
The note method contains various functions such as getCourse() and getDayte(), these functions are forms of Encapsulation that return values to themselves, so that they can be accessed by other classes or functions. Calling "getDayte()" for example will return the date to the function that's calling it.
Polymorphism can also be seen throughout the project, especially in coursework and CWDetails, as both of these classes feature override methods in conjunction with the actionPerformed function in both classes.

3 THE CODE FOR THE EXTRA CLASSES CREATED

Add the code you have written for each of the classes implemented here. Copy and paste relevant code - actual code please, not screen shots! Make it easy for the tutor to read. Add explanation if necessary – though your in-code comments should be clear enough.

3.1 CLASS ALLNOTES

```
     // Function to create a frame upon program start. Will ask the user to
     select
    // a course, once selected it will then display within txtDisplayNotes.


    public void courseEntry() {


        JFrame courseEntryFrame = new JFrame("Input Dialog");

        String courseSelection = (String)
JOptionPane.showInputDialog(courseEntryFrame,

                "Please select a course",

                "Course Entry",

                JOptionPane.QUESTION_MESSAGE,

                null,

                arrayCourses,

                arrayCourses[0]);


        userEntry = courseSelection;

}
```

The code function above is called within the AllNotes Constructor before the 'readAllNotes' function. The Function 'courseEntry' will display a JOptionPane asking the user to select a course from a list of courses that it displays. It will then add this selection to the String 'courseSelection'. courseSelection is then copied into the String userEntry, which I have added in CommonCode.

3.2 CLASS COMMONCODE

```
// Function will append the new course to the CourseList and create the
relevant files afterwards.


    public void appendFile() throws IOException {

        String addCourse = JOptionPane.showInputDialog(null, "Please enter
the code you wish to add to the course list:");

        JOptionPane.showMessageDialog(null, "The following course code will
be added to the course list when you next open: " + addCourse);


        //This will append the new course to CourseList.txt

        Writer output;

        output = new BufferedWriter(new FileWriter("CourseList.txt", true));

        output.append("\r" + addCourse);

        output.close();




        //This writes the new file, addCourse is the prefix to .txt

        Writer outpoot;

        outpoot = new BufferedWriter(new FileWriter(addCourse + ".txt",
true));

        outpoot.append("1     COMP1752    2017-02-23 09:54:22     "+"Course
Page:" + " "+ "'"+addCourse+"'"+ "\r");

        outpoot.close();


        //This writes the new requirements file, addCourse is the prefix to
CW.txt

        Writer output1;

        output1 = new BufferedWriter(new FileWriter(addCourse + "CW.txt",
true));

        output1.append("1     COMP0000    2017-01-01 00:00:00
      "+"Requirements Page:" + " "+ "'"+addCourse+"'"+ "\r");

        output1.close();

    }
```

The code above will append the entered String from the JOptionPane into the CourseList file using the Writer function so that it can be selected by the user at a later date.
In addition to this, the function will continue to create 2 more instances of 'Writer', these will;

Create a new file, using the entered String as a prefix to ".txt", entering in "hello" for example will then create a file called "hello.txt". I have also coded the Writer function to append a String to this new file ;

```
"1    COMP0000    2017-01-01 00:00:00 "+"Requirements Page:" + " "+
"'"+addCourse+"'"+ "\r");
```

This String inputs text into the file essentially saying the file name and what type of page it is (note page / requirements page). It also features a maxID, course code and an empty date, this is so that it can be read and written from txtDisplayNotes and txtWriteNotes respectively without any NumberExceptions appearing.

3.3 Class AllNotes

```
// The ArrayList showCourses reads from "CourseList.txt" and converts
    // itself into an Array "arrayCourses", this is because the
    // JOptionPane cannot accept ArrayLists, only Arrays.

    public ArrayList<String> showCourses  = readTextFile("CourseList.txt");
// Reading from "CourseList.txt"
    Object[] arrayCourses = showCourses.toArray();
```

In this project I have decided to not use the ComboBox and to only use a JOptionPane to display the CourseList selection to the user.

3.4 Class AllNotes

```
readNotes = readTextFile(appDir + fileSeparator + userEntry + ".txt");
```

This is a small excerpt of code found in AllNotes.readAllNotes.
I have changed the file path to include the variable userEntry as a prefix to the String ".txt" , userEntry is an empty String at the beginning of the program, this is then added too when the user selects a course to load from the CourseList, the selected course is added to the variable userEntry.
userEntry is then used to find the relevant file for the selected course by using the above code.

4 WHITE BOX TESTING

Design white box testing of your system (using a test table) and provide evidence of both the functionality of your program and the testing results.

4.1 TESTING FOR CLASS ALLNOTES

Test Case	Expected Output	Actual Output
Successfully calls functions from the Constructor	AllNotes will call the defined functions from the Constructor	AllNotes successfully calls the functions found in the Constructor, these includes; "courseEntry();" and "readAllNotes();"
Course Selected = COMP1152	COMP1152 will load into txtDisplayNotes	COMP1152 loads into txtDisplayNotes correctly.
Course Selected = CourseList	StackOverflow Error = CourseList cannot be loaded in two instances	StackOverflow Error generated. String CourseList now removed from the file CourseList, this error is now not possible.
Course Added = "CourseList"	The String "CourseList" will be added into the file CourseList, selecting this however on next load will crash the program.	The String was added successfully into CourseList, selecting it does crash the program however.
writeAllNotes() Function	The program will be able to run this function by being able to correctly write to a note, which will then be added to the correct corresponding file/	The program is able to use the function writeAllNotes, and correctly writes to a text file.
readAllNotes() Function	The program will be able to run this function by reading all the notes in a text file and displaying it in the txtDisplayNotes Text Area.	The program is able to use the function readAllNotes and correct reads text from a file, and outputs it to txtDisplayNotes Text Area.

4.2 TESTING FOR CLASS COURSEWORK

Test Case	Expected Output	Actual Output
Successfully calls functions from the Constructor	Coursework will call the defined functions from the Constructor	Coursework successfully calls the functions; Model(); View(); Controller(); From the constructor upon load.
Exit Button Clicked	Program will close, using the System.Exit(0); command.	Program closes successfully
Add Note Button Clicked Functionality: Notes Entered	Program will add the notes entered by the user in the txtWriteNotes textarea, and add them into the txtDisplayNotes area. These new notes will also be written to the corresponding file.	Notes added by the user successfully loads into the txtDisplayNotes Text Area and loads it into the corresponding Text File.
Add Note Button Clicked Functionality: No notes entered	Program will display a dialog box advising the user that empty notes or null values cannot be entered into the file.	Program successfully displays a dialog box advising the user that empty notes cannot be entered. It also clears the txtNewNote Text Area so the non-valid String(s) are removed and the user can attempt to enter a valid note.
Add Note Button Clicked Functionality: Only spaces entered by user	Program will display a dialog box advising the user that empty notes or null values cannot be entered into the file.	Program successfully displays a dialog box advising the user that empty notes cannot be entered. It also clears the txtNewNote Text Area so the non-valid String(s) are removed and the user can attempt to enter a valid note.
Add Note Button Clicked Functionality: Multiple Lines trying to be entered	Program will allow multiple lines to be entered	Program allows multiple lines to be entered into the txtDisplayNotes screen. However, when attempting to load this program after the program closes, you will encounter an error and the program will crash.

Open Requirements Window Button Clicked	Program will open an instance of the class CWDetails using the code; CWDetails cw = new CWDetails();	Program successfully opens the requirements page window.
Add a New Course Button Clicked	Program will display a dialog box asking the user to enter in a new course	Program successfully displays a dialog box asking the user to enter in a new course.
New Course Button Functionality: Enter String	Program will append the String entered by the user into the CourseList text file. It will also generate a text file for the String and a requirements page.	Program successfully appends the String entered by the user into the text file CourseList. It generates two text files, one is the notes file, the second file generated is the requirements page.
New Course Button Functionality: No String entered	Program will accept the value as “null” and create a file called “null.txt” and “nullCW.txt”, it will also append the String “null” to the CourseList file.	Program accepts the value of “null” from the user and creates a file called “null.txt” and “nullCW.txt”, it also appends the String “null” to CourseList file. This is not ideal, so I will add code to not accept null values.
New Course Button Functionality: Spaces entered only	Program will accept the value of just spaces entered. It will create a file called “ .txt” (there is a space in front of “.txt”. It will also append the spaces entered into the CourseList file.	Program accepts the value of spaces entered and creates the two files with a space as a prefix to “.txt” It also appends the entered spaces into the CourseList file. This is not ideal, I will add code to not accept null or empty values.
Close Button Clicked (found in menu bar)	Program will close, using the System.exit(0); command.	Program closes successfully.
Search Bar Functionality: Allows text to be entered	Search Bar will allow text to be entered	Search Bar allows text to be entered in Successfully
Search Bar Functionality: Searches for entered String once search button has been	Search Bar will search for the text entered into the Search Bar by the user, it will then	Search Bar searches successfully for the String entered in by the User, it then

clicked	display the results found in the txtDisplayNotes Text Area.	displays the found Strings in the txtDisplayNotes Area.
Search Bar Functionality: Empty Value in Field && search button clicked	The program will have no response to searching for an empty String.	The program successfully ignores the users request to search for an empty String and no functions are loaded.
Search Bar Functionality: Spaces entered only	The program accepts the spaces as valid values and will search for spaces.	The program will search for all lines of String, which contain spaces and display this in the txtDisplayNotes area.
Reset Search Bar Button Clicked Functionality: String in search bar	The program will clear the contents of the search bar field and reset the txtDisplayNotes Text Area to show the original file without any search parameters placed on it.	The program successfully clears the contents of the search bar field. It also resets the search parameters placed on the text in the txtDisplayNotes text area, so the original file is now being displayed, like it was before any searches took place.
Reset Search Bar Button Clicked Functionality: null value in search bar	No value has been entered into the search bar, this means nothing will be searched for within the text and nothing will change on screen.	No value has been entered in by the user, this means that nothing has changed on the screen and the search bar functionality has not been performed – this is intended.
New Button Clicked (found in Menu Bar)	Program will add the notes entered by the user in the txtWriteNotes textarea, and add them into the txtDisplayNotes area. These new notes will also be written to the corresponding file.	Notes added by the user successfully loads into the txtDisplayNotes Text Area and loads it into the corresponding Text File.

4.3 TESTING FOR CLASS CWDETAILS

Test Case	Expected Output	Actual Output
Successfully calls functions from the Constructor	CWDetails will call the defined functions from the Constructor	CWDetails successfully calls the functions; Model(); View(); Controller();

		From the constructor upon load.
Course Selection Functionality: Does the correct requirements page open for the corresponding course	Once a course has been selected, the program will open the corresponding requirements page	Once a course has been selected from the JOptionPane, it opens the corresponding requirements page file
Course Selection Functionality: Selected Course does not have the corresponding requirements page file.	CWDetails txtDisplayNotes Text Area will not display anything to the user and will report to the console displaying a "File Not Found" output.	Nothing is displayed in the text area, and it will show in the console that the File has not been found. A good idea would be to notify the user that the File does not exist when they're in the JOptionPane.
New Note Button Clicked (found in Menu Bar)	Program will add the notes entered by the user in the txtWriteNotes textarea, and add them into the txtDisplayNotes area. These new notes will also be written to the corresponding requirements file.	Notes added by the user successfully loads into the txtDisplayNotes Text Area and loads it into the corresponding requirements Text File.
Close Button Clicked (found in Menu Bar)	This button has not been coded and will not perform any function.	Clicking the button performs no function on screen, in the code, or in the console after running. I will now remove this button and its code as it is obsolete.
Return to Notes Window	Once clicked, it will instantiate a new version of the coursework class, this button is used so that the user can return back to the main coursework class window.	Clicking the button loads up the coursework class window successfully. It does however seem to open up many instances of itself you click between the two buttons on both the coursework window and the CWDetails window. This problem is called Circular Instantiation and I will add code to fix this problem which I will append to the end of this testing table.
Create Note Button Clicked Functionality (found in Menu Bar): Notes entered	Program will add the notes entered by the user in the txtWriteNotes textarea, and add them into the	Notes added by the user successfully loads into the txtDisplayNotes Text Area and loads it into the

	txtDisplayNotes area. These new notes will also be written to the corresponding file.	corresponding Text File.
Create Note Button Clicked Functionality (found in Menu Bar): No notes entered	Program will display a dialog box advising the user that empty notes or null values cannot be entered into the file.	Program successfully displays a dialog box advising the user that empty notes cannot be entered. It also clears the txtNewNote Text Area so the non-valid String(s) are removed and the user can attempt to enter a valid note.
Close Button Clicked (found on the Button Bar)	This button has not been coded and will not perform any function.	Clicking the button performs no function on screen, in the code, or in the console after running. I will now remove this button and its code as it is obsolete.
Exit Button Clicked (found on the Button Bar)	Program will close, using the System.Exit(0); command.	Program closes successfully
Exit Button Clicked (found on the Menu Bar)	Program will close, using the System.Exit(0); command.	Program closes successfully
Add Note Button Clicked Functionality: Notes Entered	Program will add the notes entered by the user in the txtWriteNotes textarea, and add them into the txtDisplayNotes area. These new notes will also be written to the corresponding Requirements file.	Notes added by the user successfully loads into the txtDisplayNotes Text Area and loads it into the corresponding Requirements Text File.
Add Note Button Clicked Functionality: No notes entered	Program will add the notes entered by the user in the txtWriteNotes textarea, and add them into the txtDisplayNotes area. These new notes will also be written to the corresponding file.	Program successfully displays a dialog box advising the user that empty notes cannot be entered. It also clears the txtNewNote Text Area so the non-valid String(s) are removed and the user can attempt to enter a valid note.
Add Note Button Clicked Functionality: Only spaces entered by user	Program will display a dialog box advising the user that empty notes or null values cannot be entered into the file.	Program successfully displays a dialog box advising the user that empty notes cannot be entered. It also clears the txtNewNote Text Area so the non-valid String(s) are removed and the user can attempt to enter a

		valid note.

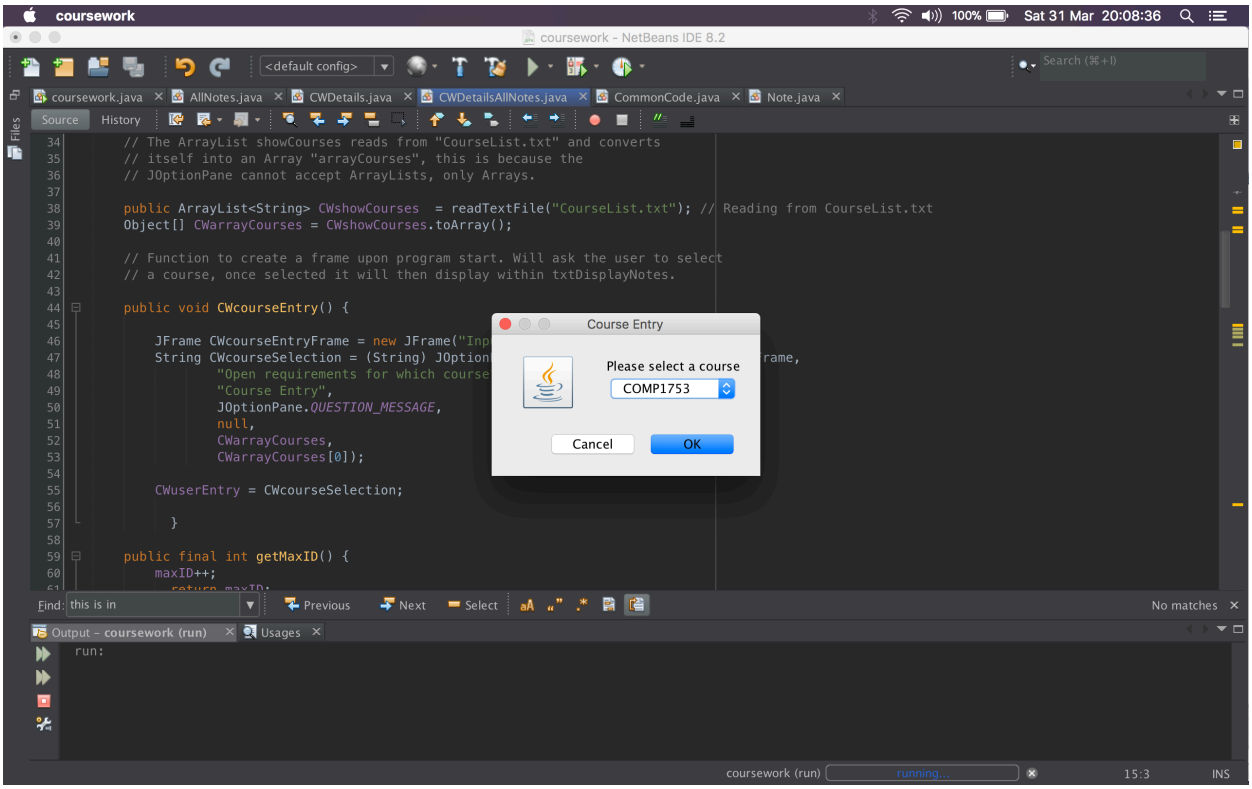
4.4 TESTING FOR CLASS CWDETAILSALLNOTES

Test Case	Expected Output	Actual Output
Successfully calls functions from the Constructor	CWDetails will call the defined functions from the Constructor	CWDetails successfully calls the functions; CWcourseEntry(); readAllNotes(); From the constructor upon load.
Course Selected = COMP1152	COMP1152 will load into CWDetails's txtDisplayNotes	COMP1152 loads into CWDetail's txtDisplayNotes correctly.
Course Selected = null	The program will accept this entry and will provide the variable "userEntry' with the String of "null"	The program accepts "null" as an acceptable value and proceeds to search for a file called null.txt This is not ideal and this functionality will be removed in the future.
Course Selected = User closes JOptionPane	The program will accept this entry and will provide the variable "userEntry' with the String of "null"	The program accepts "null" as an acceptable value and proceeds to search for a file called null.txt This is not ideal and this functionality will be removed in the future.
writeAllNotes() Function	The program will be able to run this function by being able to correctly write to a note, which will then be added to the correct corresponding file/	The program is able to use the function writeAllNotes, and correctly writes to a text file.
readAllNotes() Function	The program will be able to run this function by reading all the notes in a text file and displaying it in the txtDisplayNotes Text Area.	The program is able to use the function readAllNotes and correct reads text from a file, and outputs it to txtDisplayNotes Text Area.

5 Screen Shots of the Program Working

Provide screen shots that demonstrate the features that you have implemented. Give a brief description for each (up to 100 words) to explain which features are being demonstrated. Make sure that the screen shots make clear what you have implemented and achieved.

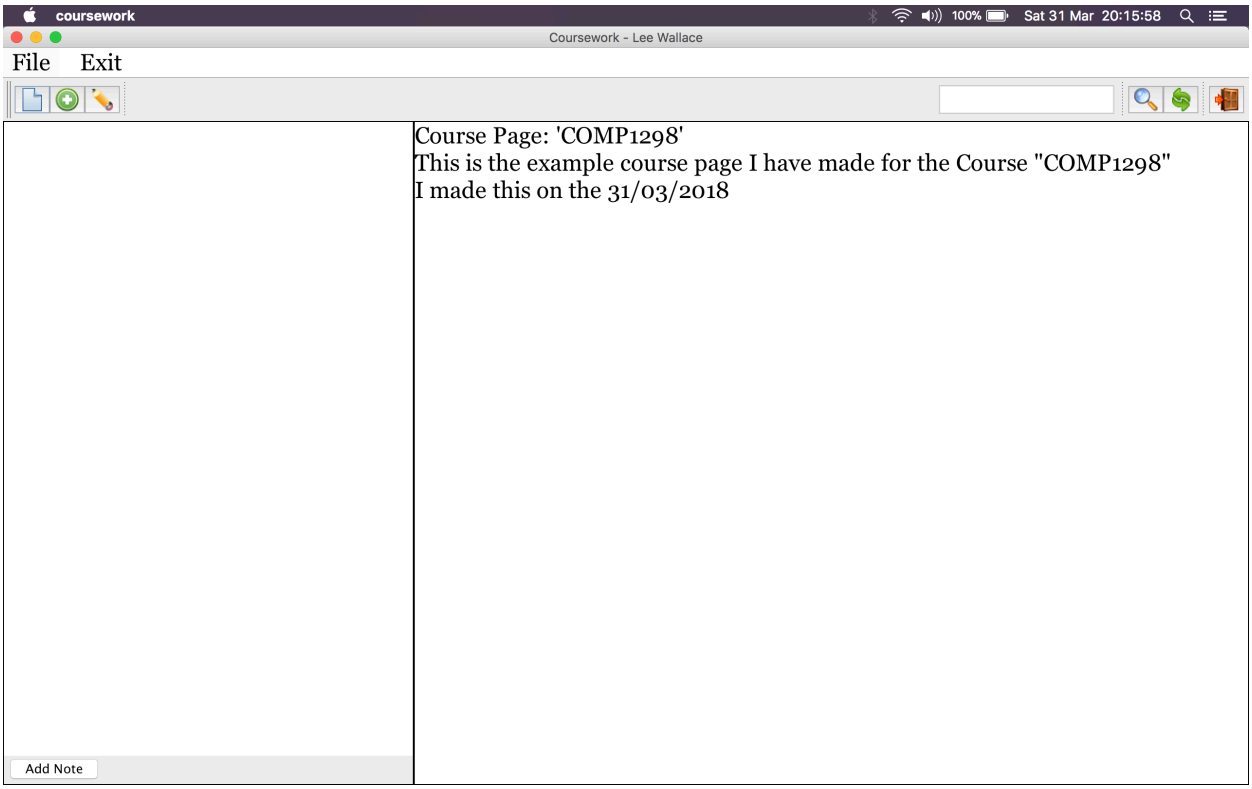
5.1 Screen 1 – Starting the Program



5.2 Screen 1 – A brief description

Once the program has been started, it will display a JOptionPane asking the user to select a course, this list of courses is being read from the CourseList.txt file, once the course has been selected, it will find the correct file (in the above instance, it will search for the file "COMP1753.txt", this file will then be displayed in the main coursework page in the txtDisplayNotes Text Area.

5.3 Screen 2 – Course Selected

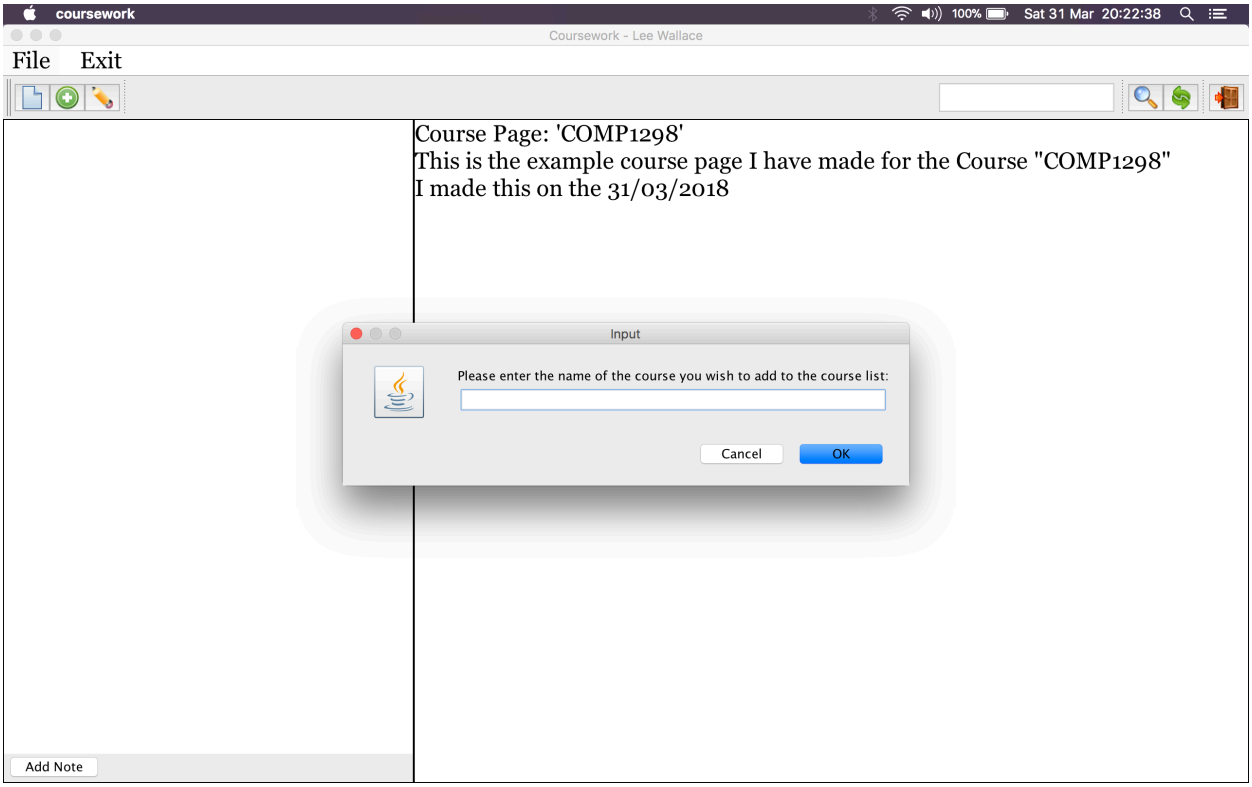


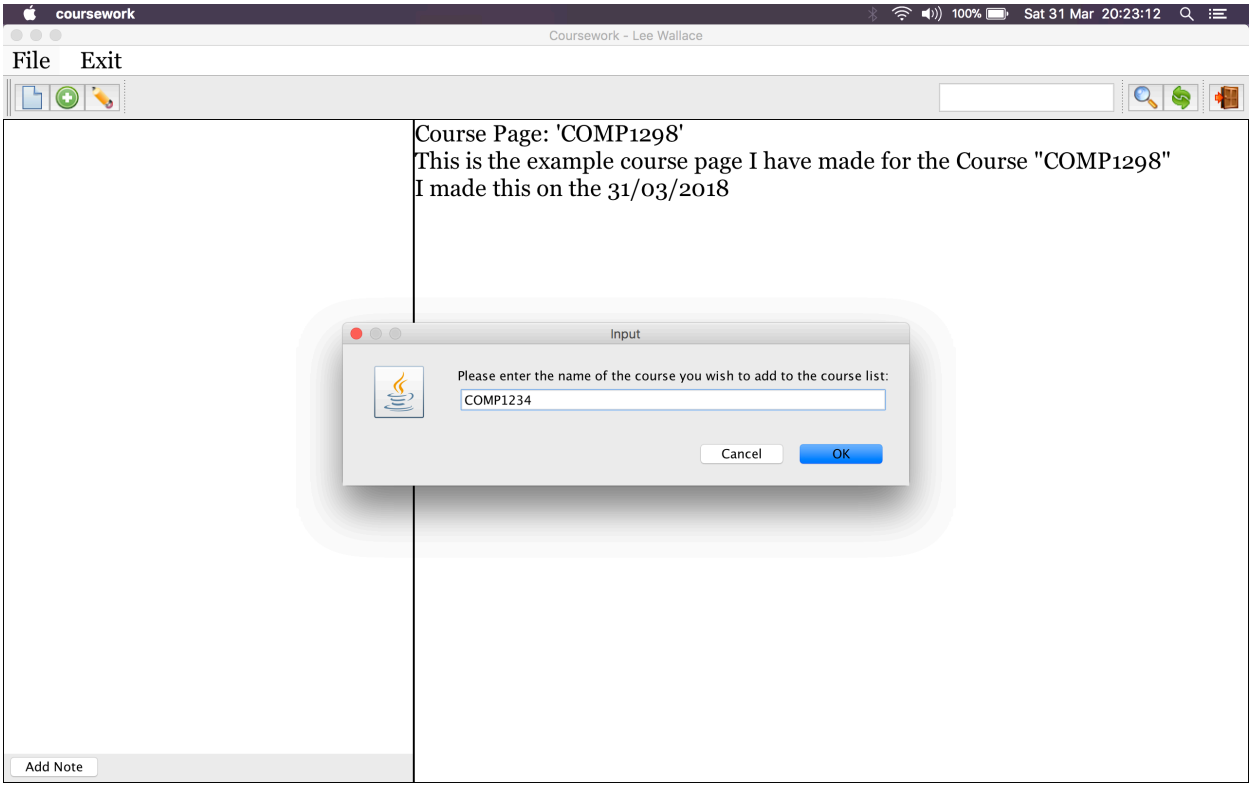
5.4 Screen 2 – A brief description

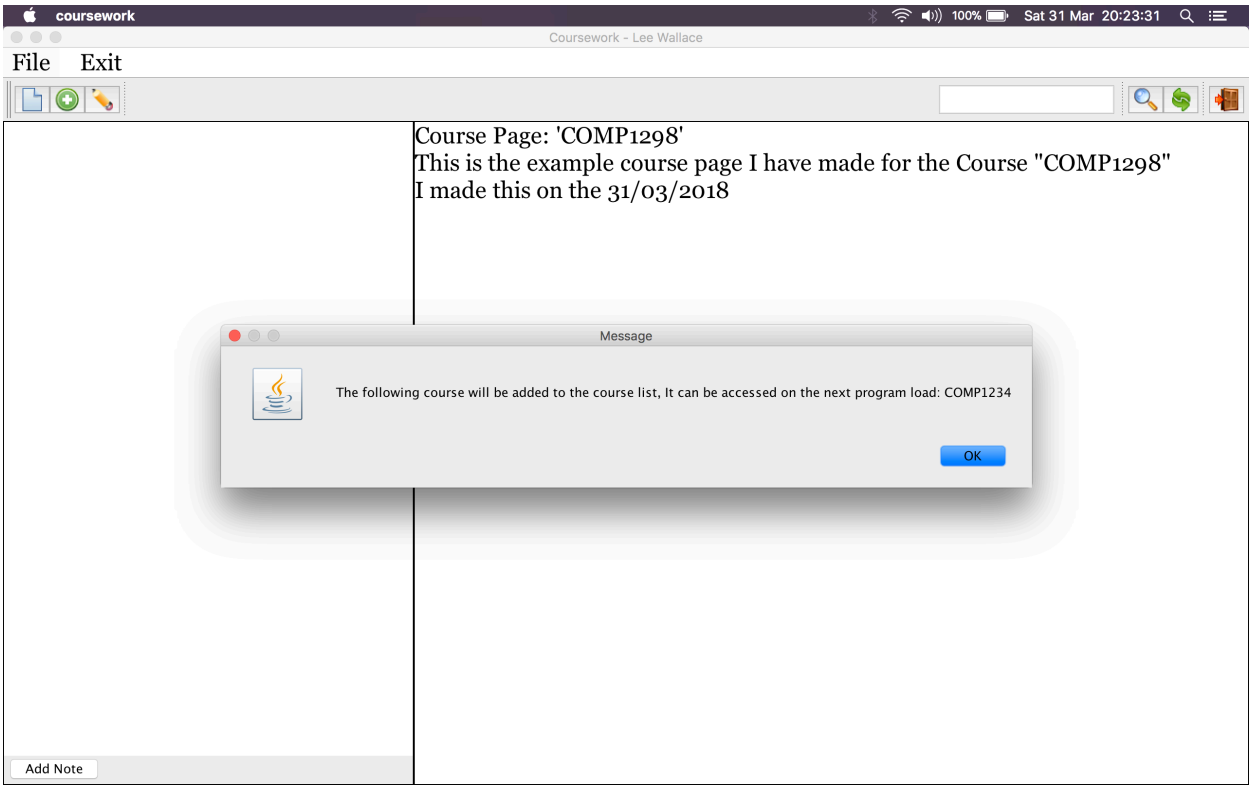
Once the course has been selected it will then display in the txtDisplayNotes Text Area, as seen in the screenshot above. I have added two new notes to this file to show that it is capable of having notes written and read to it.

The screenshot above shows the main buttons on the left hand side of the Menu Bar, these include the button to switch to the Requirements window, Adding a new note to the file and Creating a New Course.
The right hand side of the Menu Bar contains the search bar, with the search button, search bar reset button and the exit from program button.

5.5 Screen 3 – Adding a New Course







5.6 Screen 3 – A brief description

The above sequence of screenshots shows the process of adding in a new course to the CourseList.txt file.
The first screenshot shows the program producing a button asking the user what they wish to name the new course they are about to add into the course list.
The second screenshot shows that I have added in the String “COMP1234” as the new course name.
After I enter in this new course name, the program will produce a message dialog to notify us that it has been successfully added into the program and can be accessed upon the next program load.

Once the process is complete the program will append the entered String into the CourseList.txt file and create two new files, one with the name of the String appending to “.txt”, and another one with the name of the String appending to “CW.txt”.

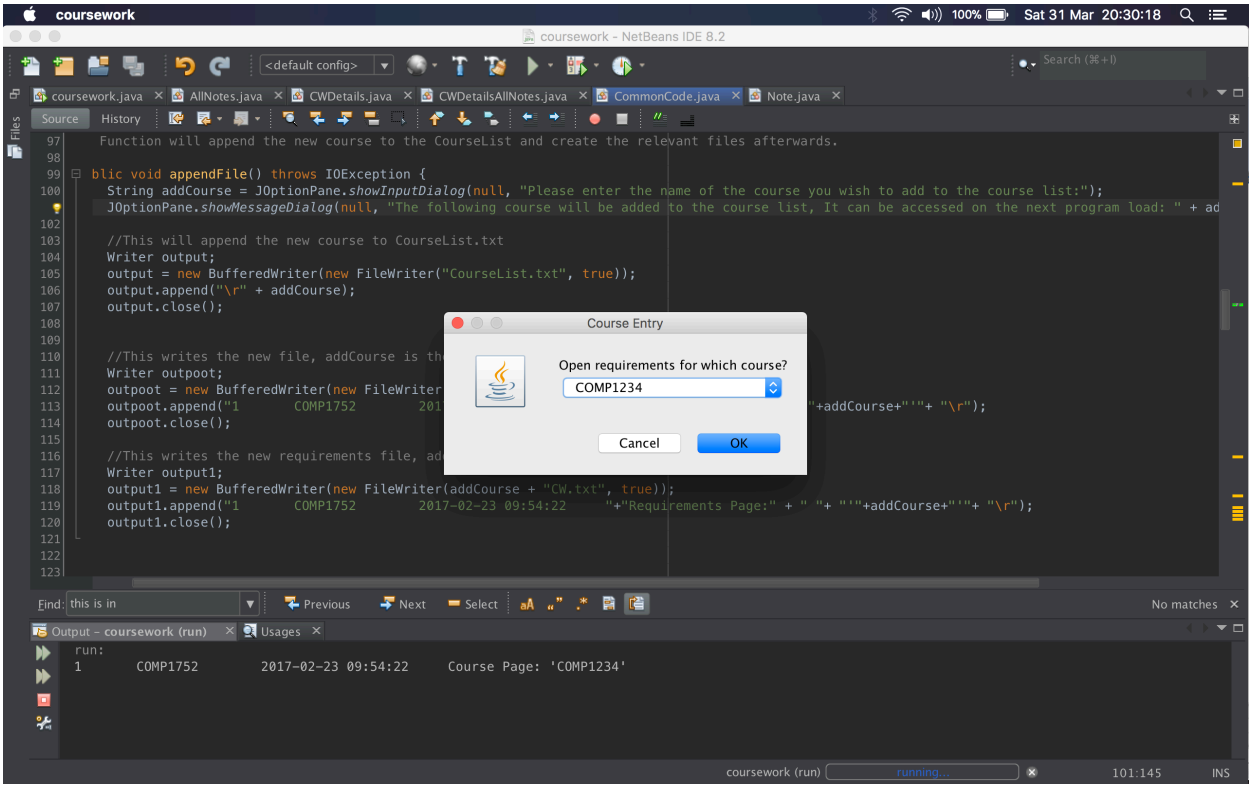
In this case for example, entering “COMP1234” into this function means COMP1234 is appended to the CourseList.txt file. In addition to this, a new file called “COMP1234.txt” is created, and “COMP1234CW.txt” is also created, this second file is used for the requirements page.

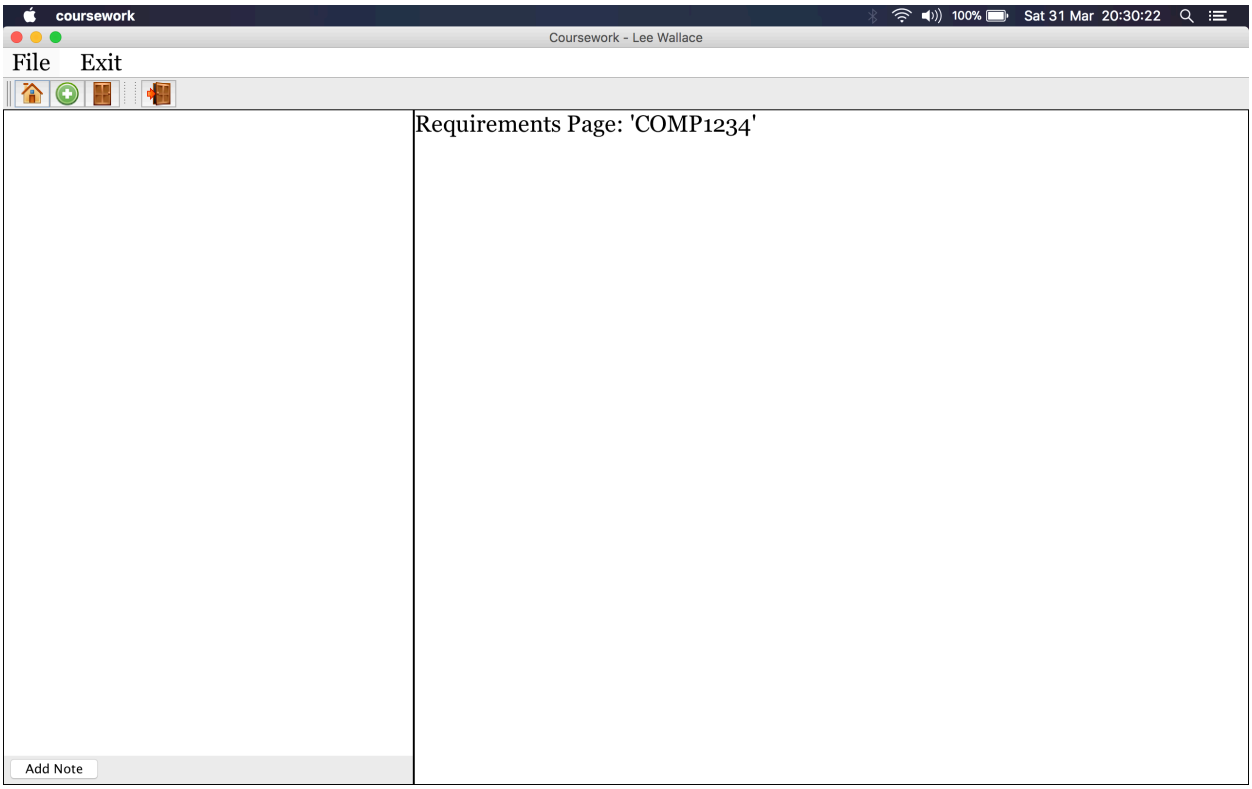
Upon loading the program after adding a course and selecting the new course, the selected course will display in the txtDisplayNotes area with the String “Course Page: COMP1234”.

The COMP1234 in the above line is whatever the file name is, if I had a course called “MATH9876” for example, the program will append the String
“Course Page: MATH9876”.

When these files have been created I have also written code to append a String to the newly created files.

5.7 Screen 4 – CWDetails Page / Requirements Page





5.8 Screen 4 – A Brief Description

The two screenshots above show what will display on screen once the user has selected the option to change to the CWDetails / Requirements Page Screen.

It will display an JOptionPane asking the user to select a course from the courselist, the selection from the courselist will open up the requirements page for the course, as once a course is created it creates two files, the notes file and the requirements file.

We saw in the previous set of screenshots that the program will automatically append a String to the new file when it has been created; this is also the case with the Requirements page.
The new file we created above called "COMP1234" has had its requirements page appended with the String

Requirements Page: "COMP1234"

This is so the user can easily differentiate between the two files, and to ensure that the user knows what screen they are currently on, whether this is the notes page or requirements page.

Adding text to both of the new files also allows them to be read by the program and to be written too correctly, if they did not have this String appended to them, the program would not know how to handle an empty .txt file once it has been loaded.

6 THE EVALUATION

Give a summary of the program and discuss what you would do if you had more time to work on the program. Answer the following questions for the reflection and write at least 400 words overall.

6.1 WHAT WENT WELL?

From using the code provided and my own general knowledge of Java, I believe I was able to construct a program that meets the requirements provided in the Coursework Specification.
I was able to make new classes such as "CWDetailsAllNotes" which allowed me to control how the requirements page handled opening notes, this includes ensuring that the files being opened by each window was the correct one. I also managed to provide a fix for the problem of circular instantiation by closing the current window when switching to a different window, such as switching from the notes page to the requirements page.

I feel like I was capable of ensuring that the logic of the program was kept suitable, ensuring that functions are given the proper return values (such as public String compared to public void).
I also ensured that classes were inheriting the correct parent class, this includes CWDetailsAllNotes inheriting CommonCode.
CWDetailsAllNotes inheriting CommonCode was very useful in this instance as the public String "userEntry" this contains what the user selected and is found in CommonCode, this means that it can be used across all the classes that inherit it so the user selection can be read across all of these child classes.

I also believe I kept the code as tidy as possible throughout the project, this includes indented code and correct naming conventions of classes, procedures, functions and variables throughout the program.

6.2 WHAT WENT LESS WELL?

Throughout coding the project there were a few major issues that I come across, some of these issues include circular instantiation, not being able to properly code the combobox as I'd like too and variables not being read correctly.

I managed to fix the issue of circular instantiation towards the end of the deadline for this project, the code I have used essentially closes the current window when you wish to switch to a different JFrame in a separate class, such as switching from the coursework class to CWDetails class (requirements window).

The combobox was also a big issue I had encountered whilst trying to write the program, I found it difficult to get the combobox toString() selection to the AllNotes class from the main coursework class.

I tried many different methods including recursion and utilizing a parent variable from CommonCode. None of these methods I was able to get functioning however; so after some research I that using a JOptionPane containing a selection box would be a good way to start the program and display the current selection in the CourseList.txt file. Whilst I do not believe that my method of displaying the course selection to the user is the most intuitive, I do feel that it still provides the user with a streamlined process of selecting a course and adding a course to the CourseList file.

6.3 What was learned?

I have learned that writing a program on this scale is much more difficult that it first may seem as there are many logical problems that have to be solved and thought about before they are being implemented. I also believe that actually writing the code for the program is the easiest thing to do compared to mapping out the logic of the program and how it all functions together.

This is also my first interaction with the Java Programming Language and I feel that the lack of knowledge in Java's syntax and technicalities withheld me from creating a program to a much higher standard to what I wanted compared to what I currently have as my final code.
To remedy this I feel like studying and trying new projects with Java in my spare time will significantly help me with future Object-Orientated Programming Projects and Programming in general.

I have also learnt that I should ask for more assistance whilst in the sessions for Java Programming, as I feel like I have left it until too late to take initiative and ask more in-depth questions regarding Object-Orientated Programming.

6.4 How would a similar task be completed differently?

A Similar task could be completed differently by allocating more time for myself to create the program. I believe that if I had more time I would be able to implement more features into the program to ensure that the user enjoys a more relaxed experience with the program, rather than worrying about crashing the program with an incorrect character entry or null value.

I would also attempt to significantly improve my Java Programming skills by taking out books from the Library or utilizing tools on the Internet, as I believe that my lack of knowledge of Java Constructs such as Recursion and Super Classes prevented me from making the program to the best of my ability, which is something I do wish to improve on in the future.

6.5 How could the course be improved?

I believe the course could be improved in a few key aspects, these aspects include having the coursework and marking criteria properly explained to them from the beginning.
I feel that once the course started I was unable to grasp what we were actually being marked on for the module.

I also strongly feel that being asked to code our coursework before our ‘boilerplate’ code was given to us was a very odd decision from my point of view. This is because the code being provided to us was constantly changing every week to ensure that it was as efficient as possible, I feel that this constantly changing code was hindering me from making any solid progress towards my coursework at the start as I was concerned whether the code would significantly change once we were given it weekly, thus breaking the program and rendering the code I had entered for my Coursework useless.

Although I do believe that once all of the code was handed to us in the end, I was able to properly focus on the Coursework Criteria and start programming in the requirements.
I feel that the first few weeks of the coursework being released was a waste of time as we did not have the code to ‘complete’ the program that was given to us. I could of used this extra time to implement more features into the program and make it better as a whole, including adding delete and amend buttons.

7 SELF-GRADING

Please assess yourself objectively for each section shown below and then enter the total mark you expect to get.

Viva 1 (3)

Not attended or no work demonstrated – 0
Work demonstrated was not up to the standard expected – 1
Work demonstrated was up to the standard expected – 2
Work demonstrated exceeded the standard expected – 3

Viva 2 (3)

Not attended or no work demonstrated – 0
Work demonstrated was not up to the standard expected – 1
Work demonstrated was up to the standard expected – 2
Work demonstrated exceeded the standard expected – 3

Viva 3 (4)

Not attended or no work demonstrated – 0
Work demonstrated was not up to the standard expected – 1
Work demonstrated was up to the standard expected – 2
Work demonstrated exceeded the standard expected – 3 or 4

For this section I think I got : 3 out of 10

Professional programming style (10)

Coding (up to 5)

Indenting has not been used – 0
Indenting has been used occasionally – 1
Indenting has been used, but not regularly – 2
Indenting has been used regularly – 3 to 4
All code has been indented correctly – 5

Naming of variables, procedure and classes (up to 5)

Professional naming has not been used – 0
Professional naming has been used occasionally – 1
Professional naming has been used, but not regularly – 2
Professional naming has been used regularly – 3 to 4
All items have been named professionally correctly – 5

For this section I think I got : 9 out of 10

Use of OOP techniques (50)

Abstraction (20)

No extra classes or objects have been created – 0
Classes and objects have been created superficially – 1 to 7
Classes and objects have been created and used correctly – 8 to 13
New and useful classes and objects have been created – 14 to 17
The use of classes and objects exceeds the specification – 18 to 20

Encapsulation (10)

No encapsulation has been used – 0
Class variables have been encapsulated superficially – 1 to 3
Class variables have been encapsulated correctly – 4 to 6
The use of encapsulation exceeds the specification – 7 to 10

Inheritance (10)

No inheritance has been used – 0
Classes have been inherited superficially – 1 to 3
Classes have been inherited correctly – 4 to 6
The use of inheritance exceeds the specification – 7 to 10

Polymorphism (10)

No polymorphism has been used – 0
A procedure has been polymorphised – 1 to 3
A procedure has been polymorphised and used appropriately – 4 to 6
The use of polymorphism exceeds the specification – 4 to 10

For this section I think I got : 45 out of 50

Testing (10)

Testing has not been demonstrated in the documentation – 0
Little white box testing has been documented – 1 to 3
White box testing has been documented for all the coursework – 4 to 6
White box testing has been documented for the whole program – 7

For this section I think I got : 7 out of 10

Evaluation (10)

No evaluation was shown in the documentation – 0
The evaluation shows a lack of thought – 1 to 3
The evaluation shows thought – 4 or 5
The evaluation shows clearly demonstrates increased awareness – 6 or 7

For this section I think I got : 7 out of 10

Documentation (10)

The documentation cannot be understood on first reading – 0
The documentation is readable, but a section(s) are missing – 1 to 3
The documentation is complete – 4 to 6
The documentation is complete and of a high standard – 7

For this section I think I got : 7 out of 10

I think my overall mark would be : 60+ out of 100