IN-COURSE ASSESSMENT (ICA) SPECIFICATION

Module Title	Module Leader	Jackie Barker
Object Oriented Programming (Semester 2)	**Module Code**	CIS4037-N
	Submission Final Date	Wednesday, 10th May 2023
ICA Element2: Console Application	**Submission Method**	
	Online (Blackboard)	☑
	Middlesbrough Tower	☐

Online Submission Notes

- Please carefully follow the instructions given in this Assignment Specification.
- Extensions or MITS should be requested using the Extenuating Circumstances (EC) form available at https://www.tees.ac.uk/sections/stud/handbook/extenuating_circumstances.cfm
- Instructions for submitting completed EC forms:
  - Short extensions requests are emailed to either Module Leader or Course Leader
  - Long Extension requests are submitted to Course Leader or scedt-assessments@tees.ac.uk
  - MITS requests are submitted to SLSMitigatingCircumstances@tees.ac.uk
- If Extenuating Circumstances (extension or MITS) is granted, a fully completed and signed Extenuating Circumstances form must be emailed to scedt-assessments@tees.ac.uk or submitted to the School Reception.

Module Assessment Notes

The module assessment is composed of two elements:

- Element 1 is a Portfolio of Team programming tasks and Individual written a task, contributing 30% to the module mark

- Element 2 is a Portfolio of Individual programming tasks and a written task, contributing 70% to the module mark

To pass the module students need to attain a minimum overall mark of 50%. If a student has attained 50% overall, but failed one of the elements, they will still pass if the student submitted a genuine attempt for the failed element.

FULL DETAILS OF THE ASSIGNMENT ARE ATTACHED

INCLUDING MARKING & GRADING CRITERIA

Contents

1. Introduction

The assessment for the module is comprised of two elements, this specification is for Element 2.

1.1 ICA Element 2 Overview

The second element is worth 70% of the module's mark. **Each student will individually** develop a small Java application to simulate buying and selling of sports item and write a short technical and self-reflective report.

The emphasis is on demonstrating:

- The use of the Java programming language.
- The use of Java APIs.
- Good use of object-oriented programming techniques.
- Basic GUI creation using the Swing API.
- Selecting and implementing an appropriate design pattern to improve the overall architecture of the solution

1.2 ICA Element 2 Tasks

The programming tasks for ICA Element 2 continue from the programming tasks in ICA Element1 but will now be GUI based. The application should be developed using Apache NetBeans and tested to correctly run in either IT0.11 or IT0.13

Table 1: ICA Element 2 tasks

ICA Task	Description	ICA Weighting
5	GUI Application	25 %
6	Enhanced Application	15 %
7	Merged Application	15 %
8	Development Report	15 %

The work for the above tasks should be submitted to Blackboard before 4.00pm on Wednesday, 10th May 2023.

2. Task 5 [25%] – GUI Application

2.1 Task 5 Overview

Each student should develop a GUI based version of the prototype application from ICA Element 1 Task3. When stated the application will look like that depicted in Figure 1 below.

Figure 1: GUI Application after being started

The operation of the application is as follows

1. When the application is started, data is loaded from an input file into a generic Array List

   - If the ArrayList is empty, the Application warns the user and closes

   - If the ArrayList is populated, the Application displays loaded data to a Table and displays the GUI.

   - Note that Pounds and Pence are combined into a price, which is composed of a £ symbol, a space, pounds value, decimal point, and pence value. The combined price value should be fixed to two decimal places

2. To Buy one unit of an item, the user selected the table row for the item and clicks the Buy button

   - The application checks if there is stock of the item and if so decrements the stock quantity

   - Confirms the sale or no-sale via a pop-up dialog

   - If appropriate, updates both the table and Array List

3. To Add to stock, one unit of an item, the user selected the table row for the item and clicks the Add button

   - The application increments the stock quantity

   - Confirms the adding of stock via a pop-up dialog

   - Updates both the table and Array List

4. To exit the application the user clicks the Quit Button

   - The application will save contents of ArrayList to an Output file

   - Close the application

2.2 Task 5 Naming and GUI Requirements

Students should ensure they use the following names in setting up their NetBeans Project.

Table 2: Required NetBeans project naming

What	Name
NetBeans Project	ASCSystem
Group ID	oop
Package	oop.ica.e2
JFrame Class File	SportsShopGUI.java
Data Class File	ASCStockItem.java
Abstract Table Model File	ASCTableModel.java

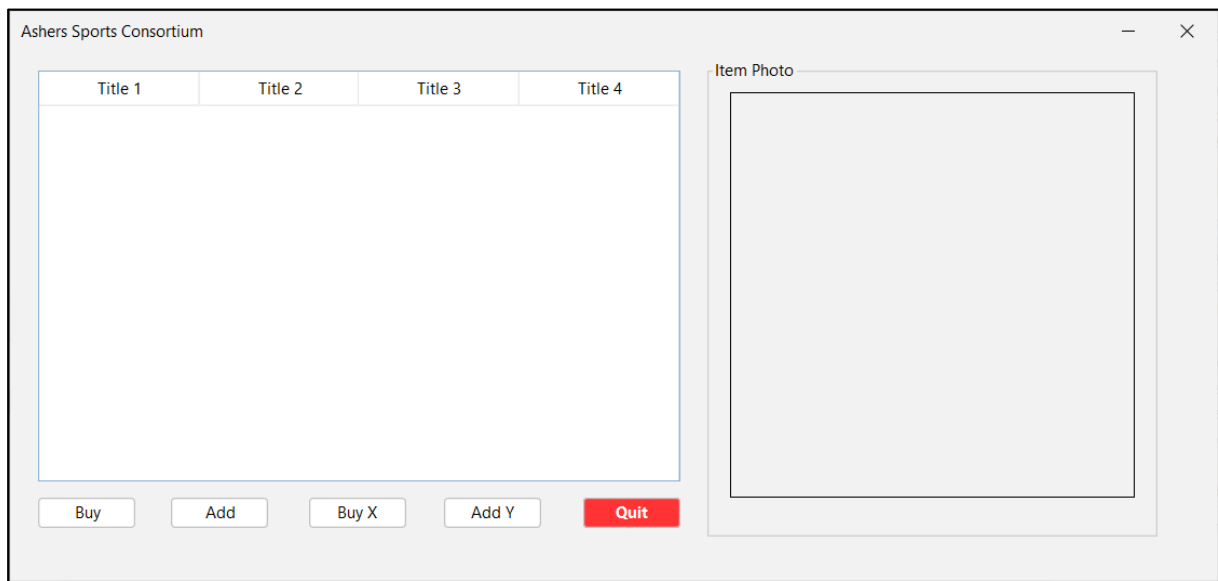The GUI design should look like that depicted in Figure2 below.

Figure 2: GUI design

The GUI components are composed of a `JFrame` , a `JPanel` and Java Swing controls, as specified in Tables 3 and 4 below.

Table 3: JFrame Components

Control	Name
JTable	ascStockItem
JButton	buyButton
JButton	addButton
JButton	buyXButton
JButton	addYButton
JButton	quitButton
JPanel	photoPanel

Table 4: JPanel Components

Control	Name
JLabel	`photoLabel`
JLabel	`itemLabel`

Although students will create *buyXButton* and *addYButton*, *photoLabel* and *itemLabel* in Task 5, they will not be coded until Task 6.

The Photo Label should be set large enough to display an inage which can be 300 pixels wide and or 300 pixels. High. The Item Label should be located below photo label and set as wide as the photo label.

2.3 Data Files

The input file is available through Blackboard and the output file is to be generated b the application:

- Input filename: **AsherSportsConsortium3.csv**
- Output filename: **asc_output.txt**

2.4 Information Messages

User confirmation messages should be displayed using `JOptionPane`. For example, if the user successfully buys an Item, then a confirmation like that depicted in Figure 3 should be displayed.

Confirmation of Sale ×

Item: 7oceans Goggles
Price £ £ 24.00
Units bought: 1
Stock remaining: 7

OK
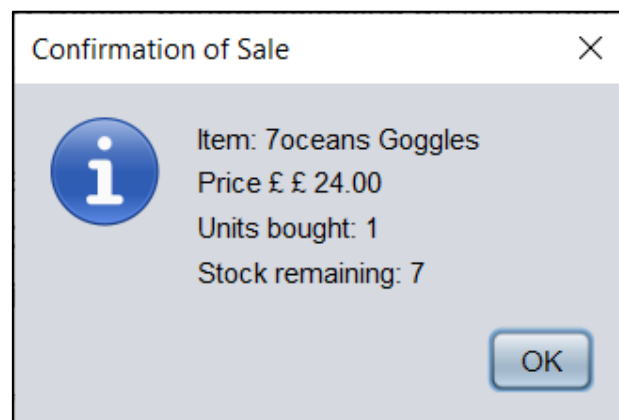
Figure 3: Confirmation Pop-up Dialog

Validation error messages should also be displayed using `JOptionPane`. For example, if the user Clicks the Buy Button without selecting an item, then a warning like that depicted in Figure 4 should be displayed.
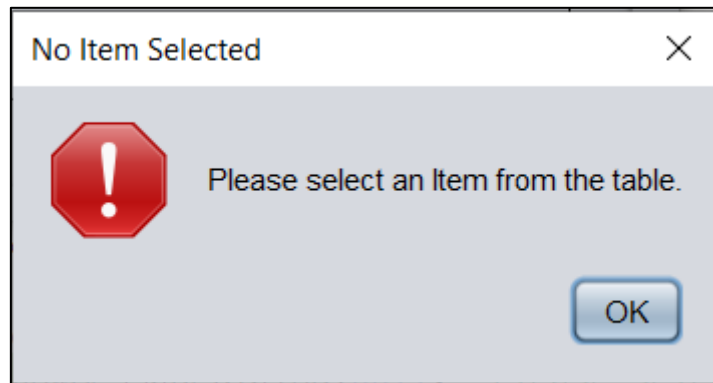

Figure 4: Validation Error Pop-up Dialog

Details of runtime errors and exceptions should be displayed to the NetBeans console using the System Error print steam, i.e., `System.err.println()`.

3. Task 6 [15%] – Enhanced Application

In this task, the GUI Application will be enhanced with the following features. The same NetBeans Project will be used and saved without any changes to file, package, or class names.

3.1 Item Photograph

When an item is selected from the table the photograph for that item should be displayed in the `photoLabel` control. In addition, the title of the item is displayed in the `itemLabel` control.

The photographs for all the stock items are provided in a zip file, **asc_pics.zip**, which is available on Blackboard. To set up the image files to be used by Java code:

1. Download the zip file

2. Extract contents of zip file

3. Within the `ASCSystem` NetBeans Project folder manually create a sub folder named **photos** and

4. Copy all the photograph files to the sub folder

For each item the corresponding photograph file consist of the Item code with a jpg file extension. For example, the item with code `FTB4444444` should have an image file named **FTB4444444.jpg**.

The `ASCStockItem` data class should be modified to include a method named `getPhotoFilename()`, which return the filename for the item.

Students should write and invoke methods in the JFrame class:

- To preload the image files into Buffered Image ArrayList named `photoList`
  - If an image file cannot be loaded add a null image to the array list.
- Loads the Buffered Image into `photoLabel` when an item is selected in the Table
  - The `itemLabel` should then display the title of the displayed item

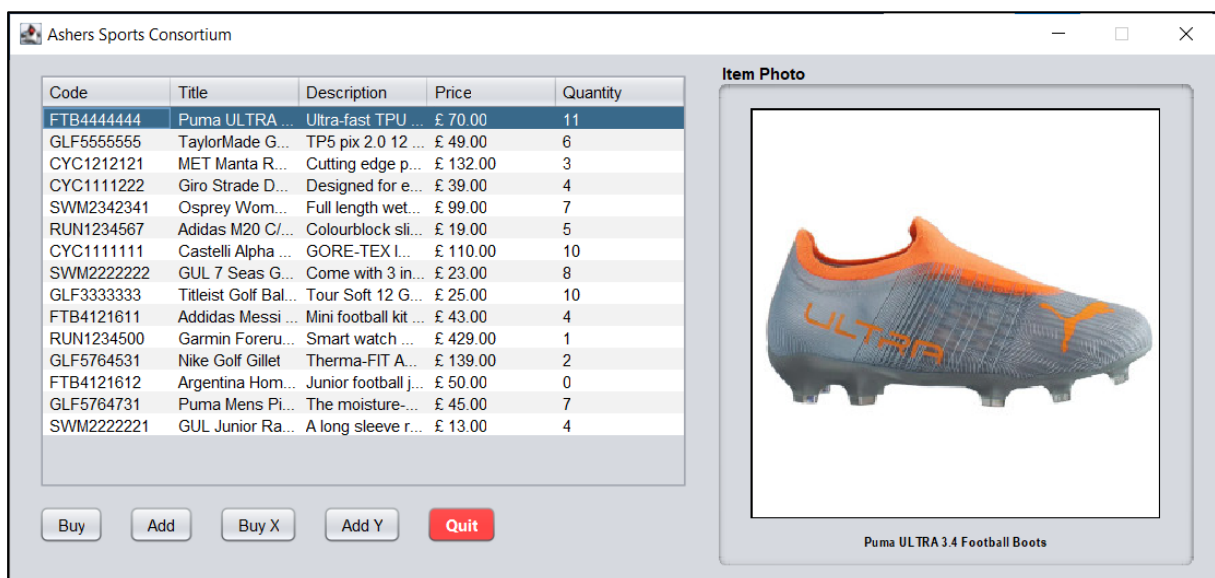Figure below displays the correct operation of displaying an item image when the first table row was selected.

Figure 5: Photo of Item displayed

3.2 Low Stock Warning

When an item is selected from the table the stock quantity is checked. If it is less than five, a pop-up dialog is displayed to warn that the item has low stock.
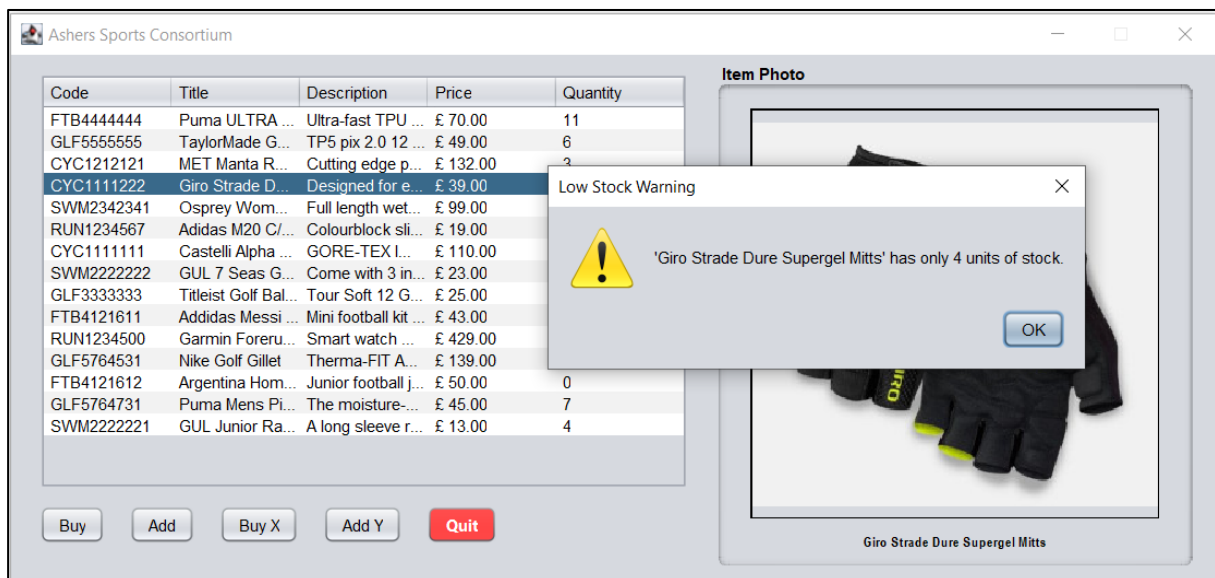
Figure 6: Low stock warning displayed

Write a method in the JFrame class which implements the above feature.

3.3 Buy X Button

When an Item is selected, and the user clicks the buyX Button:

1. The application will check if user has selected an item from the table, if not the user will be warned

2. Next the application will check is there is more than zero stock, if not the user will be warned

3. If there is stock, a pop-up dialog will request a quantity to be bought
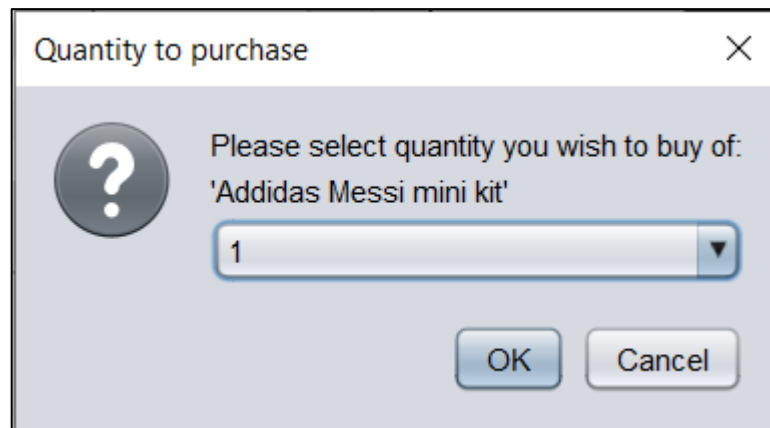
Figure 7: Input Dialog

4. If the user clicks cancel or closes the dialog, the operation should be quietly cancelled

5. The application will validate the input quantity as a positive nonzero integer, for which there is enough stock to purchase

   • Students can implement validation via an option list that displays quantities from 1 to how much quantity is in stock
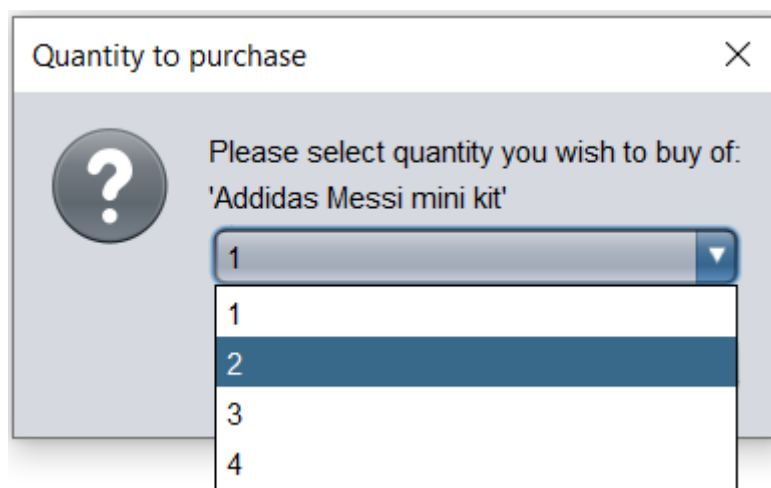

Figure 8: Input Dialog with Option List

   • Alternatively, students can use an input dialog that displays an standard text field. Providing additional code to validate the input

6. If there is sufficient stock:

   • The sale is made, table & array list are updated, and the user is informed.
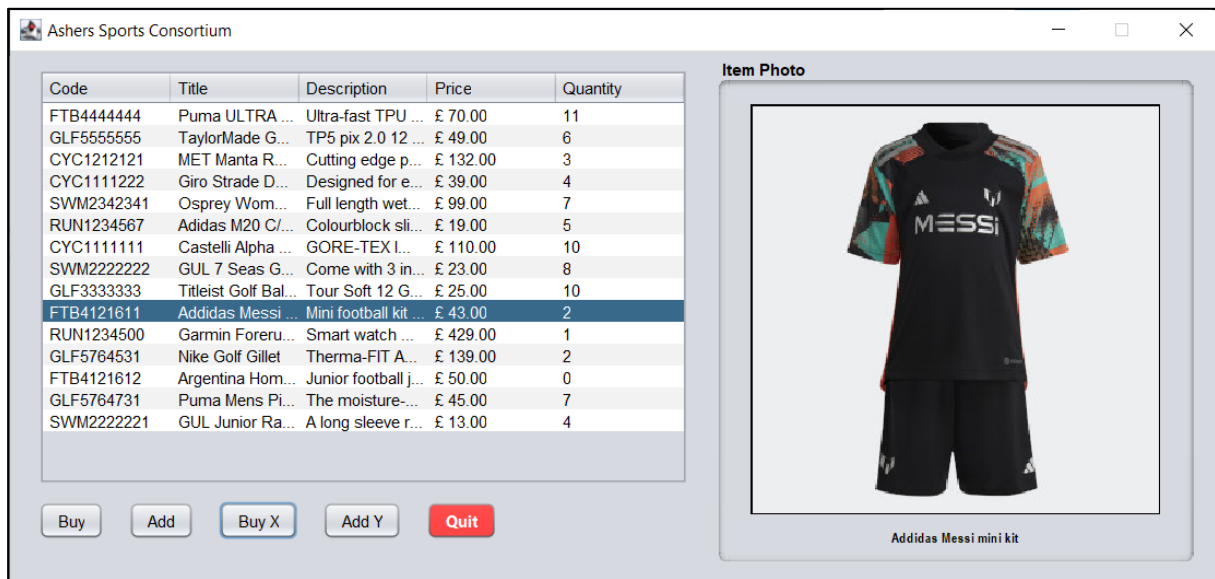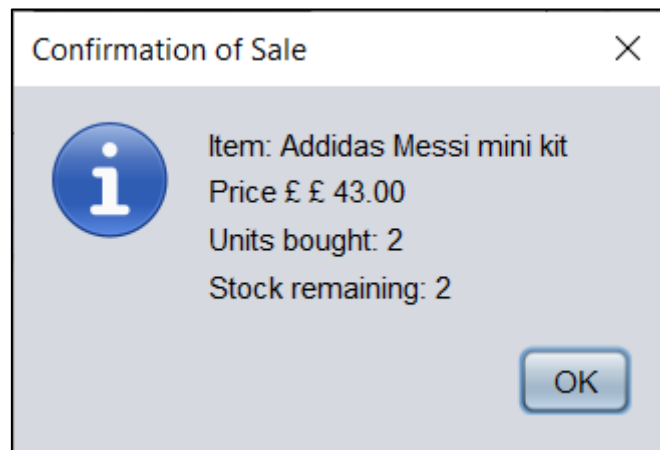
Figure 9: Table Updated

Figure 10: Confirmation of sale

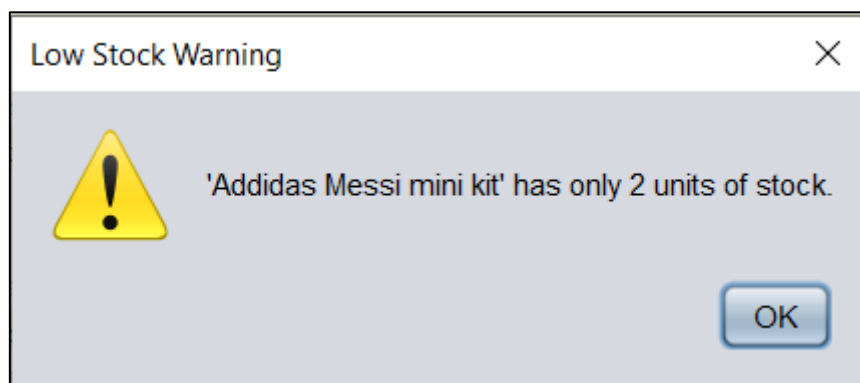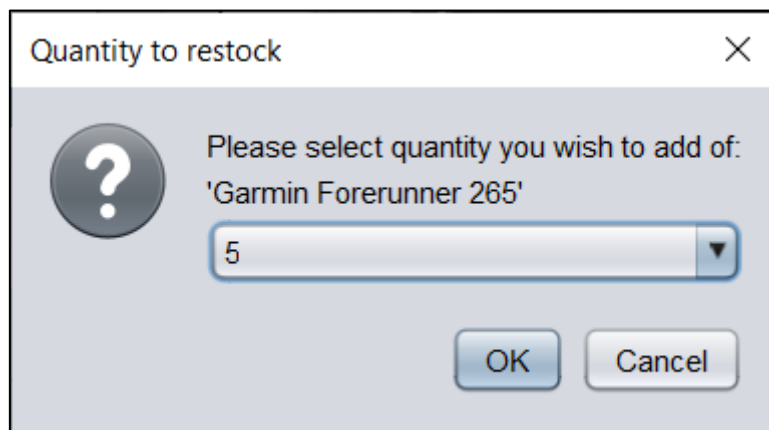- A low stock check is made, and the user warned if appropriate

Figure 11: low stock warning

3.4 Add Y Button

When an Item is selected, and the user clicks the addY Button:

1. The application will check if user has selected an item from the table, if not the user will be warned

2. Next the application will display a pop-up dialog which will request a quantity to be added to stock

Figure 12: Add Stock Input Dialog

3. If the user clicks cancel or closes the dialog, the operation should be quietly cancelled

4. The application will validate the input quantity as a positive nonzero integer, in the range five to ten.
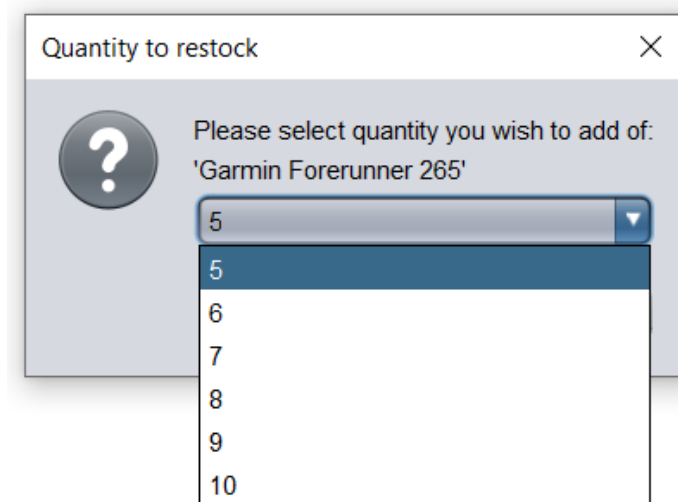   - Students can implement validation via an option list that displays integers from five to ten

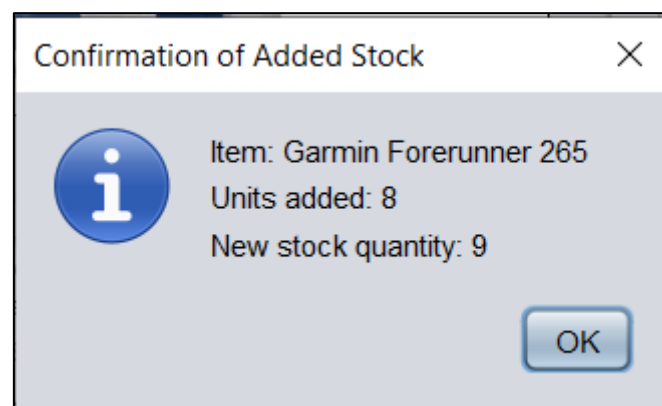**Figure 13: Input Dialog with Option List**

- Alternatively, students can use an input dialog that displays an standard text field. Providing additional code to validate the input

5. If the input is valid:

- The application will add the input to the stock for the item.

- The table & array list are updated
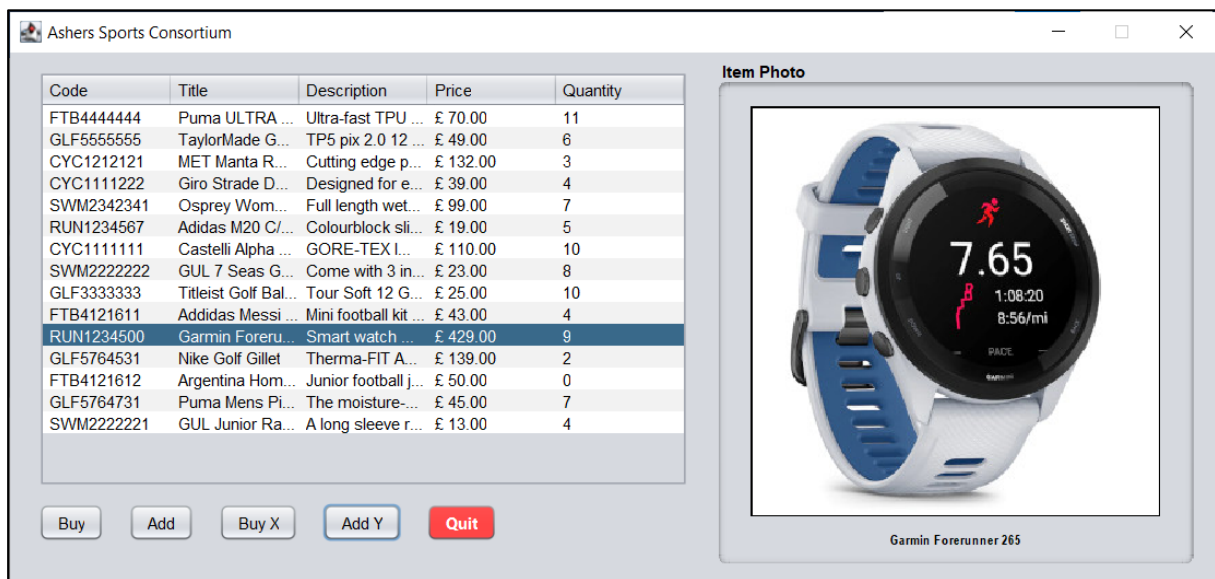
Figure 14: Updated Table

- The user is informed

Figure 15: Confirmation of Added Stock

4.   Task 7 [15%] – Merged Application

In this task, the Enhanced Application will be merged with data from another system. The same NetBeans Project will be used and saved without any changes to file, package, or class names.

4.1 Data Merger

Asher's Sports Consortium buys Teesside Skates. The head office wants a single application for buying and stock-control. Unfortunately, it is too expensive to replace "Teesside Skates" system at the moment. It has been decided that all "Teesside Skates" product items are to be "translated" as needed to "ASCStockItem" format and the two shops are to use the same application, i.e., "ASCSystem".

4.2 Teesside Skates System

The Application for Teesside Skates is provided on Blackboard. This application was at Prtotype stage, when the buyout took place.
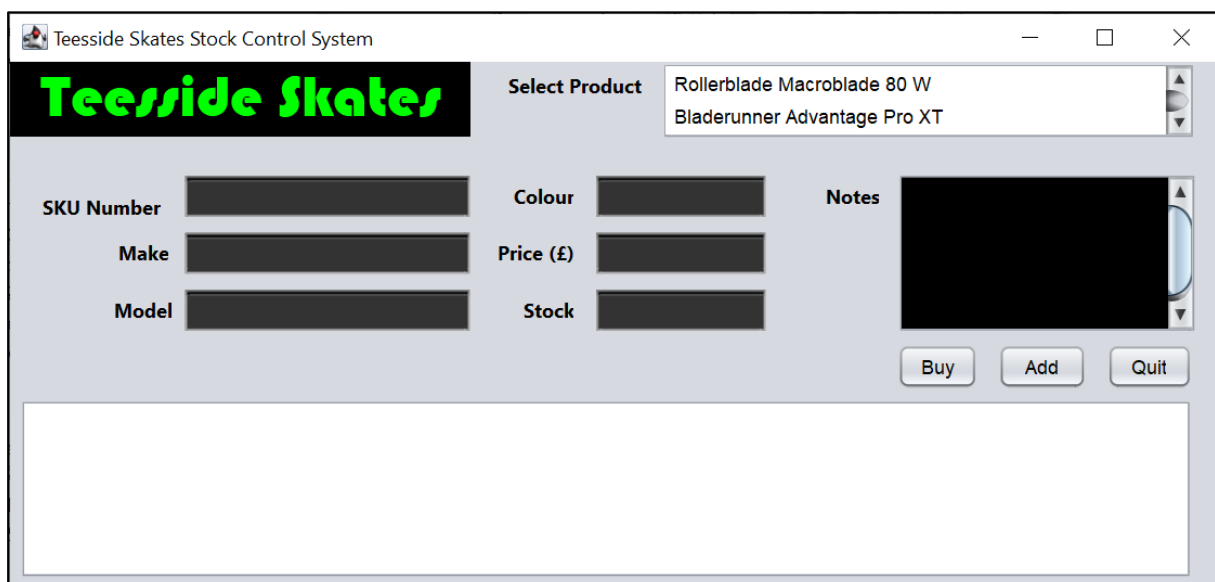

Figure 16: Teesside Skates system when started

Teesside Skates system uses a List Control instead of a table, displays information to a combination of text fields and text areas.

Figure 17: Display of data when an element in List control has been selected

The system only allows buying or stocking of a single unit for an item.
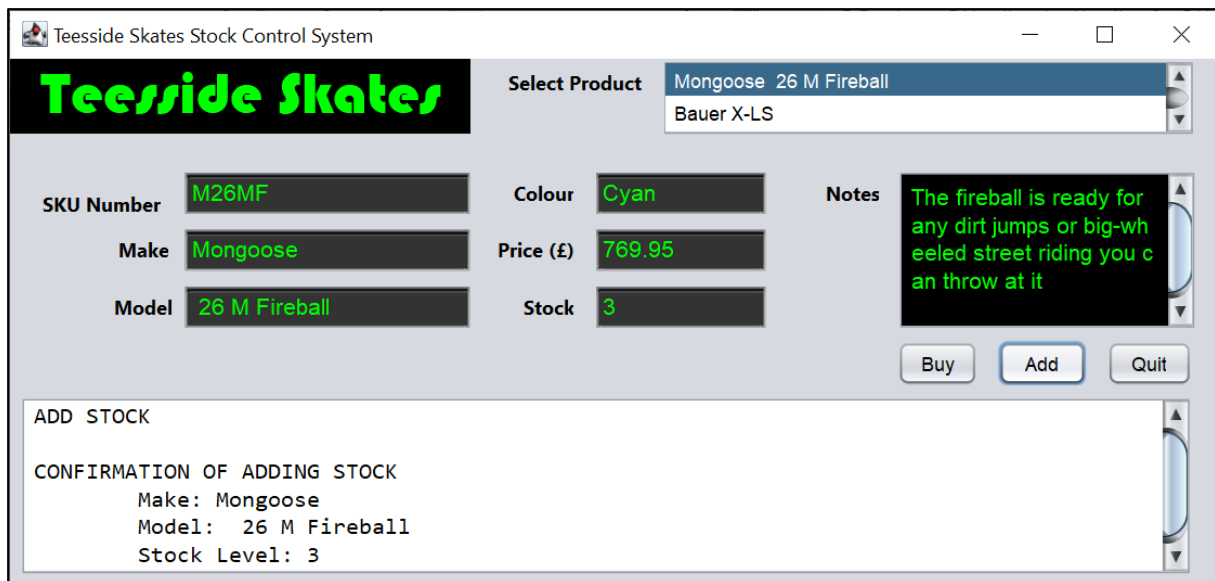

Figure 18: Successful Buy Operation

Figure 19: Successful Add Stock operation

The quit operation save data to the output file and closes the application. Any validation messages during any operation are displayed to the main text area. Figure 20 depicts outcome of clicking Buy button without selecting a product.
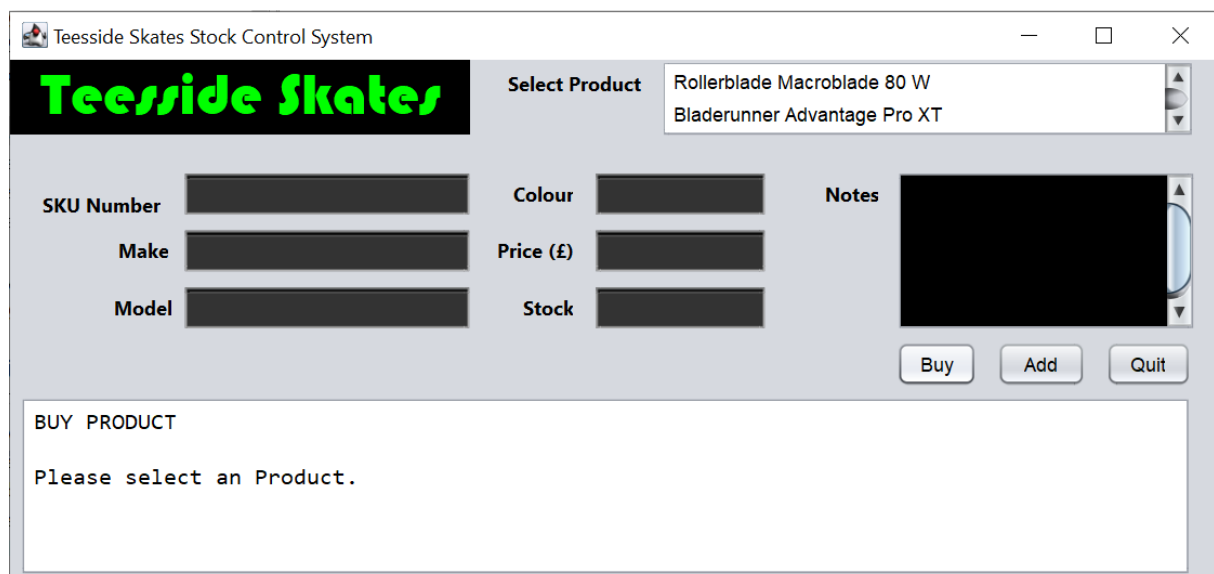

Figure 20: Validation Example

4.3 Teesside Skates Data
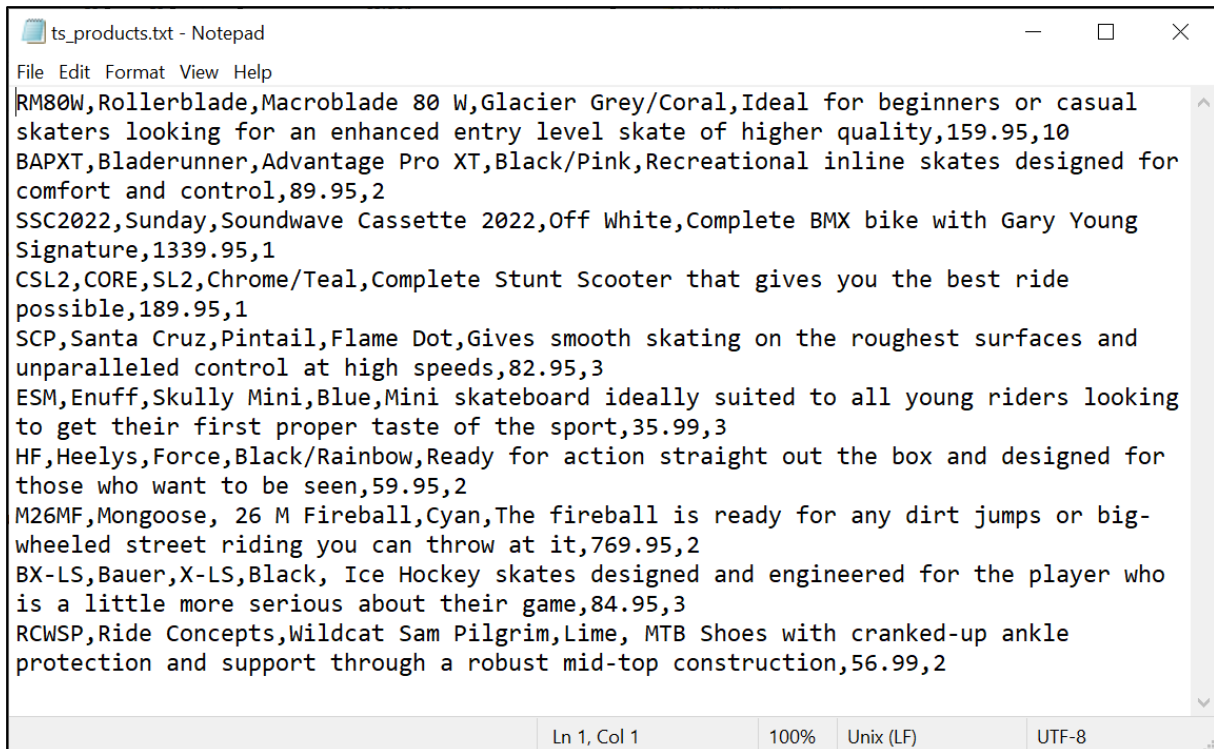
The Data class for Teesside Skates is named TSProduct and Table 5 below shows the class diagram for it.

Table 5 Data Class Diagram for Teesside Skates System

```
TSProduct
─────────────────────────────────────────────────────────────
 - skuNumber: String
 - make: String
 - model: String
 - colour: String
 - notes: String
 - price: double
 - stock: int
─────────────────────────────────────────────────────────────
 + TSProduct(String, String, String, String, String, double, int)

 + getSkuNumber(): String

 + getMake(): String

 + getModel(): String

 + getColour(): String

 + getNotes(): String

 + getPrice(): double

 + getStock(): int

 + setPrice(double): void

 + setStock(int): void

 + increaseStock(): void

 + decreaseStock(): void
```

The Data class is classed as complete and should not be modified.

A sample data file, ts_products.txt, is provided for testing purposes whose contents are depicted by Figure 21 below.

Figure 21: ts_products.txt

Images for the items in the input file are contained in the zip file `ts_pics.zip`. Both the Teesside Skates System NetBeans Project, the input file and zip file are available on Blackboard.

4.4 Merged Application

The enhanced Application needs to be modified in order to load and process the sample data and photographs from Teesside Skates.

1. Copy the file `ts_products.txt` to same location as `AsherSportsConsortium3.csv` within the ASCSystem NetBeans Project folder

2. Extract the contents of the zip file `ts_pics.zip` to the photos sub folder.

3. Use a suitable design pattern to make "TS" products behave as if they were "ASC" stock items.

4. Modify SportsShopGUI.java to load data from both input files into the single Array List.

Note that no changes should be made to the contents of the input files or original data class files. No changes are required to the Teesside Skates System.

Figure 22 below depict the merged application after being started. Note that the table shows Teesside Skates Products as if they were ASC stock Items.
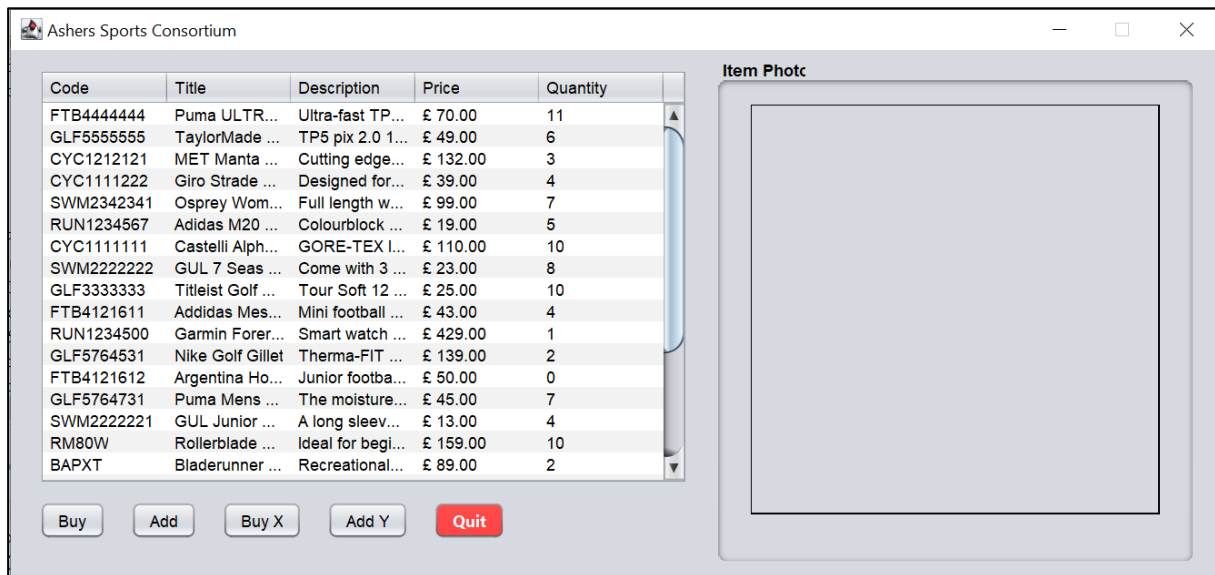
Figure 22: Merged Application after being started

Figure 23 demonstrates that when an TS Product is selected the application works just like he enhanced application, in that the photograph is displayed and low stock warning triggered if quantity is less than five.
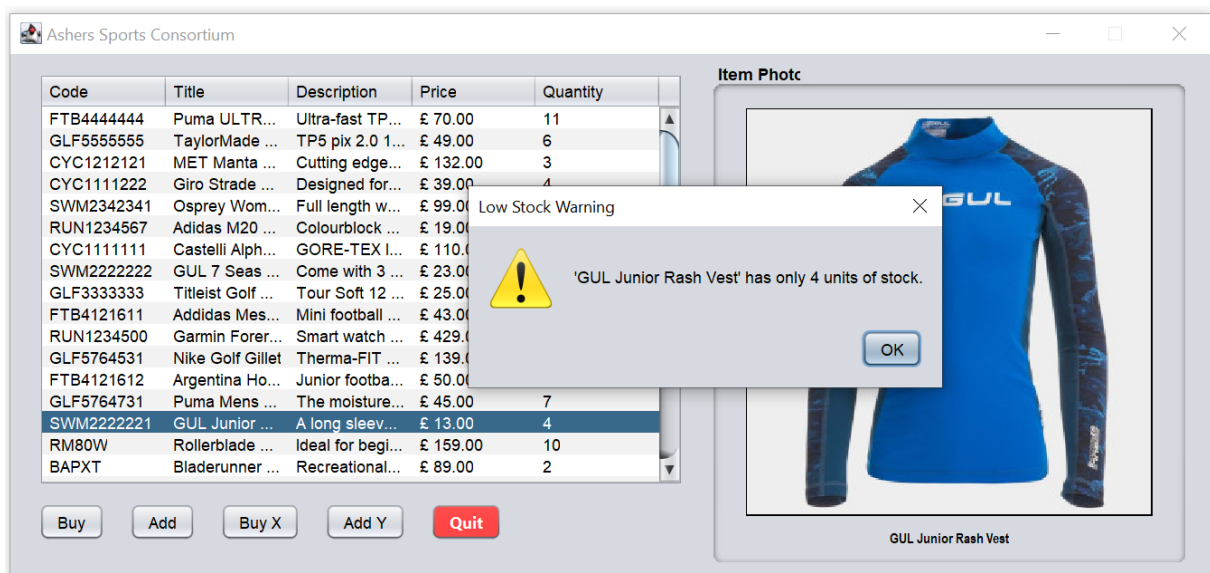
Figure 23: Merged Application when a TS Product is selected

5. Task 8 [15%] - Development Report

5.1 Report Structure

Students will need to submit a 1500 words development report. The report should contain the following:

Title Page

Table of Contents

1. Discussion of GUI Application
2. Discussion of Enhanced Application
3. Discussion of Merged Application
4. Discussion of Improvement
5. References

Students should ensure that screenshots of application running, and appropriate code snippets are provided to support discussion.

5.2 Report Section 1: Discussion of GUI Application

Students should discuss the main issues they faced when developing the GUI Application. In particular the reuse of console-based code from Task3 and what changes they made to that code. If the application does not work correctly, focus the discussion what does work and what does not work.

5.3 Report Section 2: Discussion of Enhanced Application

Students should select two of the enhancements and discuss the work they undertook to implement the selected enhancements two of the enhancements. If the enhancements do not work correctly, focus the discussion what does work and what does not work.

5.4 Report Section 3: Discussion of Merged Application

Identify the relevant design pattern. Then discuss what changes were required to the ASCSystem to allow it to use data from both input files. If the merged application did not work correctly, focus the discussion what does work and what does not work.

5.5 Report Section 4: Discussion of Improvement

In this section students should discuss one improvement they could make to the Merged Application. For the Improvements students should:

- Clearly explain what the improvement is and why it would benefit the application
- What changes would be required to the application
- What would be key code in implementing the improvement
- How could the improved application be tested

Potential examples of improvements that students can discuss include:

- Implementing a second or alternative design pattern
- Making a login feature for a user
- Allowing user to add a new item for sale
- Allowing user to remove an item for sale
- Using a database instead of input and output files

5.6 Word Limit

The word limit is 1500 words (strict). The Title page, contents listing, footnotes, code listings, tabular data, headings, captions, citations (within round brackets), references section and any appendices are excluded from the word count.

It is likely that most students will find the word limit restrictive, as such student should be very selective on their content. Avoid listing a whole program file and focus on what code was relevant to the section

5.7 Additional Information

- Reference Style: Cite Them Right

- File formats: DOCX, DOC or PDF

- File Name:
  - Format: *surname-userID*-OOP-Task8
  - Example: Rashid-u0018369-OOP-Task8.pdf

- Submit to Blackboard by 4.00pm on Wednesday 10 May 2023 (week13)

- Worth: Contributes 15% towards module mark

Appendix A.   Module Learning Outcomes

The following tables provide the learning outcomes for 'Computer Technologies and Operating Systems' module.

Personal and Transferable Skills	
1.	Produce appropriate software documentation to communicate the programming concepts and techniques underpinning their solutions.
2.	Demonstrate a sound understanding of the Java API and the ability to make good use of the information provided therein when building solutions
3.	Develop team leadership skills by taking the team lead role to identify and develop solutions to practical problems.

Research, Knowledge and Cognitive Skills	
4.	Analyse complex and/or incomplete problem specifications, justify the design and technical methodologies used, and recognise and argue for alternative approaches.
5.	Demonstrate a systematic and critical understanding of Object Oriented concepts, event handling and the development of Graphical User Interfaces.
6.	Select appropriate programming techniques and abstract Object-Oriented concepts, and critically evaluate their effectiveness in a given scenario.

Professional Skills	
7.	Design and implement efficient Java solutions to unfamiliar problem specifications and critically evaluate the processes and facilities used in an autonomous manner.

Appendix B.   Marking Criteria

Tasks 5, 6 and 7 (55%)				
Programming Practice	**Distinction**	**Merit**	**Pass**	**Fail**
	The code is written to a very high and consistent standard, elegantly specified. The code is easy to read and understand by someone other than the original author. Comments are provided, clear descriptions of algorithmic implementations that are not immediately obvious by the code itself. The code is clearly demonstrating that it is very easy to extend and maintain.	The code is written to a good standard, although not always consistent. The code is easy to read and understand by someone other than the original author. Comments are provided, although it maybe inconsistent in clarity or usefulness. The code appears to be generally easy to extend or maintain with little effort.	It performs reasonably, although not consistently. The code is written to a reasonable standard, although the author may not have always considered that others would need to read and understand their code in the future. Some of the requirements have been met. Comments are present, although they may be inconsistent in their usefulness or appropriateness. It is not immediately clear, or it looks like quite a lot of effort would be required to extend and/or maintain the current code-base	Code is poorly written, lacking in in structure. There is little or no consideration that other programmers may need to be able to read and understand the code in the future. Few, if any, useful or appropriate comments are provided. The current code-base is messy and would require considerable effort to extend and/or maintain - probably a complete rewrite would be required. .

Task 8 (15%)				
Development Report	**Distinction**	**Merit**	**Pass**	**Fail**
	Full (but concise) explanation of the problem domain and the objectives of each task; clear insight into main challenges. Very good discussion of development work for the three applications. The content is clearly selective and relevant, in order to satisfy the word limit. Code snippets concisely discussed and appropriate issues raised. Appropriate improvement discussed in terms of what benefit it would bring. Relevant methods identified for changes. Appropriate code suggested to implement the change. Choice of figures and tables compliments the discussion and are all appropriately numbered and captioned.	*Technical authorship presents a clear explanation of the problem; discussion/critique shows a good appreciation of main issues & objectives.* Good discussion of development work for the three applications. The content is relevant, Code Blocks are discussed, and issues potentially raised. Appropriate improvement identified. Some discussion of benefits and changes Relevant classes identified for changes. Appropriate pseudo code suggested to implement the change. Good range of figures and tables to support the discussion with some attempt to number and caption.	*Technical authorship presents a clear explanation of the problem; discussion/critique shows a good appreciation of main issues & objectives.* Adequate discussion of development work for two or three of the applications. The content is more general than specific. Classes are discussed, and issues potentially identified. An improvement is identified, for which a basic discussion of benefits and or changes is provided. Suggestions to implement the change, lack detail. Limited range of figures and tables provided, lacking numbering and or captions.	*Technical authorship is below the standard expected of a Masters degree.* Unsatisfactory discussion of development work for any or all of the applications. The content is irrelevant or too general. Limited identification or required changes, with either no or inappropriate issues raised. Basic improvement identified, but not suitably discussed as to its benefits and or how it could be implemented If any figures and tables are provided, they do not support the discussion and may be provided in lieu of discussion, for example screenshot of code with no discussion.