Name	Yuvraj Chandrakant Haryan
ID	D3581054
Module	Object Oriented Programming
Module Code	CIS4037-N
Course	MSc. Computing

Assessment	Task 8: Development Report

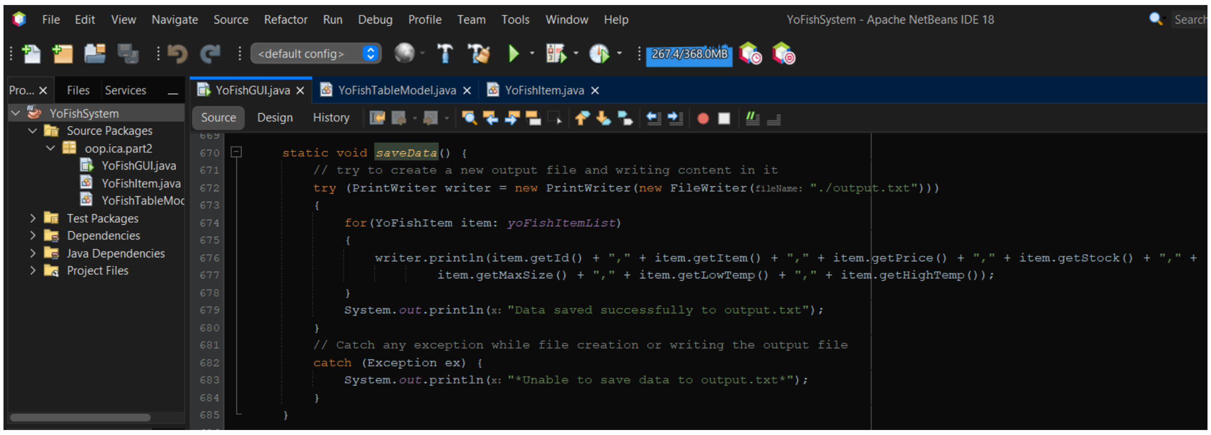
Title	ICA Part 2 – Task 8

Table of Contents

Section 1: Discussion of GUI Application

Java Swing API was found as completely new to me. Working with the libraries was a bit of challenge at first. Studied a ML system based on java swing GUI (Saudagar *et al.*, 2023) . However, as the project progressed, I was able to grab a good hold on the GUI. The design mode offered by it helped a lot in development, dragging and dropping elements saved a lot of time and proved to be a low code solution (Calçada and Bernardino, 2022) .

 It was discovered that there was some reusable code in Task 3's console application. The code for reading from an input file in task 3, worked in the GUI application with minimum number of alterations. To be precise just the line of code where data read from the file was used to create an 'ArtPrint' object had to be changed to create and 'YoFishItem' object instead. The writing to output file code had no significant changes. Apart from that, also the code used for buying and adding of stock in Task 3 was used as a reference in the application. It was specifically referred for noting how the user selected item would be fetched from the array list and how its fields i.e. stock field would be changed.

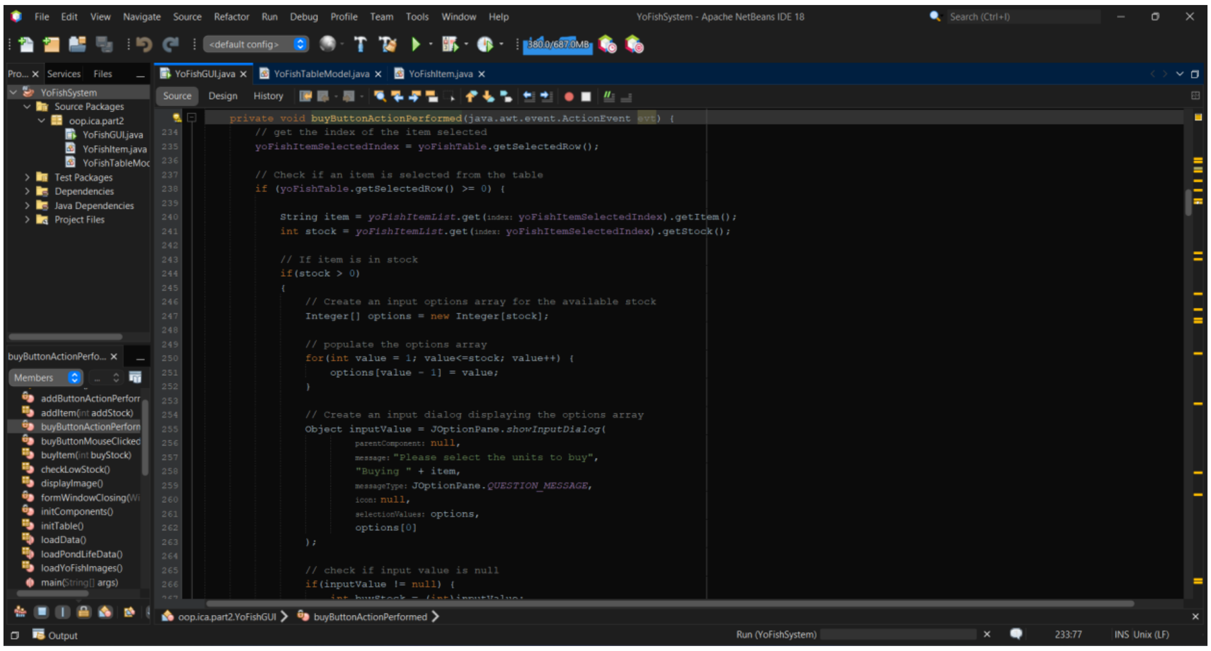


```java
    static void saveData() {
        // try to create a new output file and writing content in it
        try (PrintWriter writer = new PrintWriter(new FileWriter(fileName: "./output.txt")))
        {
            for(YoFishItem item: yoFishItemList)
            {
                writer.println(item.getId() + "," + item.getItem() + "," + item.getPrice() + "," + item.getStock() + "," +
                        item.getMaxSize() + "," + item.getLowTemp() + "," + item.getHighTemp());
            }
            System.out.println(x: "Data saved successfully to output.txt");
        }
        // Catch any exception while file creation or writing the output file
        catch (Exception ex) {
            System.out.println(x: "*Unable to save data to output.txt*");
        }
    }
```

Fig 1: Saving data to an output file (Reusable code from Task 3)

Section 2: Discussion of Working Application

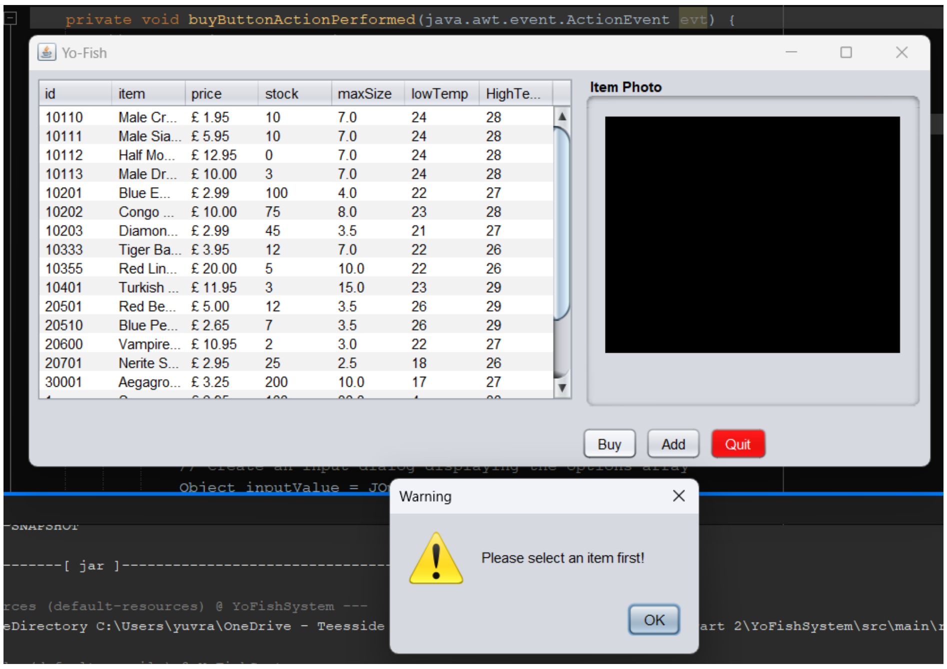
Buy operation is selected to discuss its functionality in this section. Three cases have been considered while developing the buy option, these would be discussed below:



```java
private void buyButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // get the index of the item selected
    yoFishItemSelectedIndex = yoFishTable.getSelectedRow();

    // Check if an item is selected from the table
    if (yoFishTable.getSelectedRow() >= 0) {

        String item = yoFishItemList.get(index: yoFishItemSelectedIndex).getItem();
        int stock = yoFishItemList.get(index: yoFishItemSelectedIndex).getStock();

        // If item is in stock
        if(stock > 0)
        {
            // Create an input options array for the available stock
            Integer[] options = new Integer[stock];

            // populate the options array
            for(int value = 1; value<=stock; value++) {
                options[value - 1] = value;
            }

            // Create an input dialog displaying the options array
            Object inputValue = JOptionPane.showInputDialog(
                    parentComponent: null,
                    message: "Please select the units to buy",
                    "Buying " + item,
                    messageType: JOptionPane.QUESTION_MESSAGE,
                    icon: null,
                    selectionValues: options,
                    options[0]
            );

            // check if input value is null
            if(inputValue != null) {
```

Fig 2: Buy button action code

1) **IF user has selected an item before clicking 'Buy':** If the user has selected an item from the JTable and then clicked buy button, then the 'getSelectedRow()' method on the JTable returns a number greater than 0. If it returns a 0, then a warning dialog has been shown to the user advising him to select an Item before making a purchase.



Yo-Fish

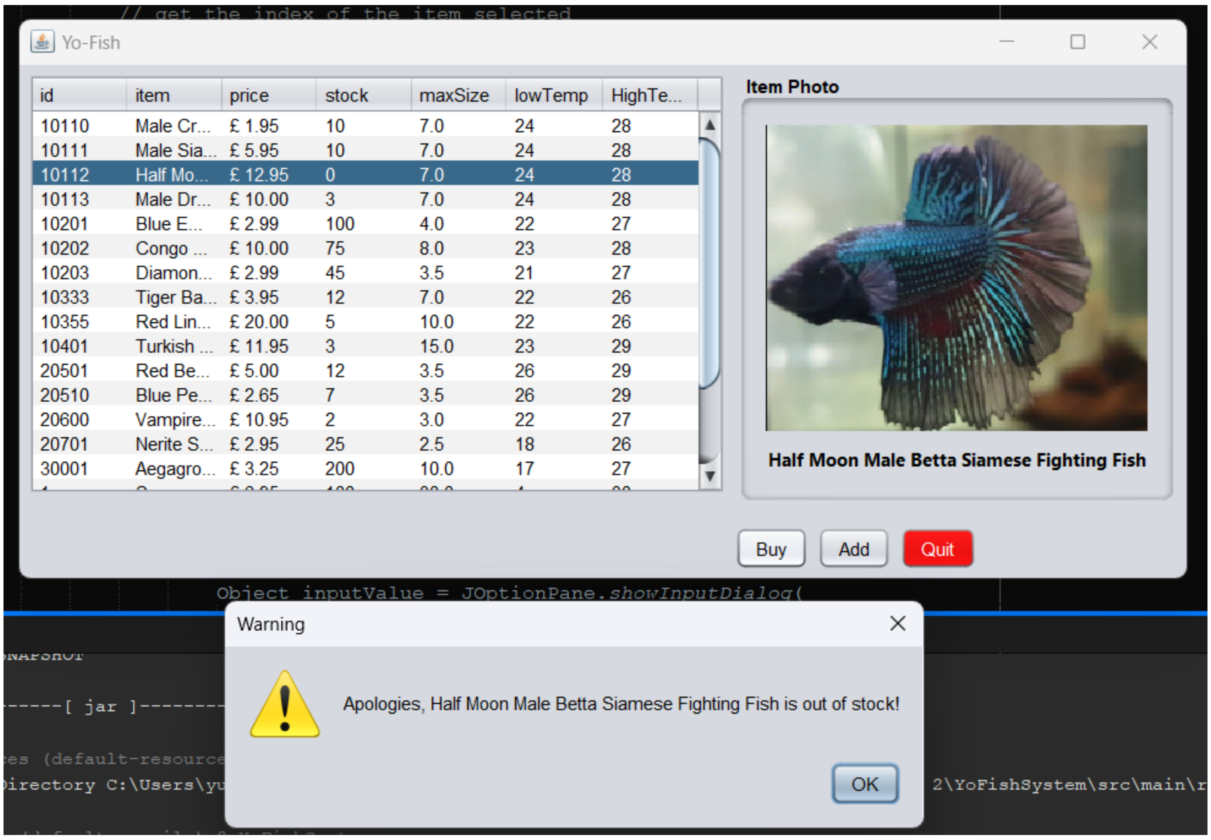
id	item	price	stock	maxSize	lowTemp	HighTe...
10110	Male Cr...	£ 1.95	10	7.0	24	28
10111	Male Sia...	£ 5.95	10	7.0	24	28
10112	Half Mo...	£ 12.95	0	7.0	24	28
10113	Male Dr...	£ 10.00	3	7.0	24	28
10201	Blue E...	£ 2.99	100	4.0	22	27
10202	Congo ...	£ 10.00	75	8.0	23	28
10203	Diamon...	£ 2.99	45	3.5	21	27
10333	Tiger Ba...	£ 3.95	12	7.0	22	26
10355	Red Lin...	£ 20.00	5	10.0	22	26
10401	Turkish ...	£ 11.95	3	15.0	23	29
20501	Red Be...	£ 5.00	12	3.5	26	29
20510	Blue Pe...	£ 2.65	7	3.5	26	29
20600	Vampire...	£ 10.95	2	3.0	22	27
20701	Nerite S...	£ 2.95	25	2.5	18	26
30001	Aegagro...	£ 3.25	200	10.0	17	27

Warning

Please select an item first!

Fig 3: Item not selected before buy (Warning)

2) **IF stock of that item is less than or equal to zero:** If the stock field of the selected 'YoFishItem' object has a value less than or equal to 0 i.e. no items are remaining to buy, then a dialog box is made to pop-up warning the user that the item is out of stock.



```
// get the index of the item selected
Object inputValue = JOptionPane.showInputDialog(
NAPSHOT
-----[ jar ]--------
es (default-resource
irectory C:\Users\yu
2\YoFishSystem\src\main\r
```

Yo-Fish

id	item	price	stock	maxSize	lowTemp	HighTe...
10110	Male Cr...	£ 1.95	10	7.0	24	28
10111	Male Sia...	£ 5.95	10	7.0	24	28
10112	Half Mo...	£ 12.95	0	7.0	24	28
10113	Male Dr...	£ 10.00	3	7.0	24	28
10201	Blue E...	£ 2.99	100	4.0	22	27
10202	Congo ...	£ 10.00	75	8.0	23	28
10203	Diamon...	£ 2.99	45	3.5	21	27
10333	Tiger Ba...	£ 3.95	12	7.0	22	26
10355	Red Lin...	£ 20.00	5	10.0	22	26
10401	Turkish ...	£ 11.95	3	15.0	23	29
20501	Red Be...	£ 5.00	12	3.5	26	29
20510	Blue Pe...	£ 2.65	7	3.5	26	29
20600	Vampire...	£ 10.95	2	3.0	22	27
20701	Nerite S...	£ 2.95	25	2.5	18	26
30001	Aegagro...	£ 3.25	200	10.0	17	27

Item Photo

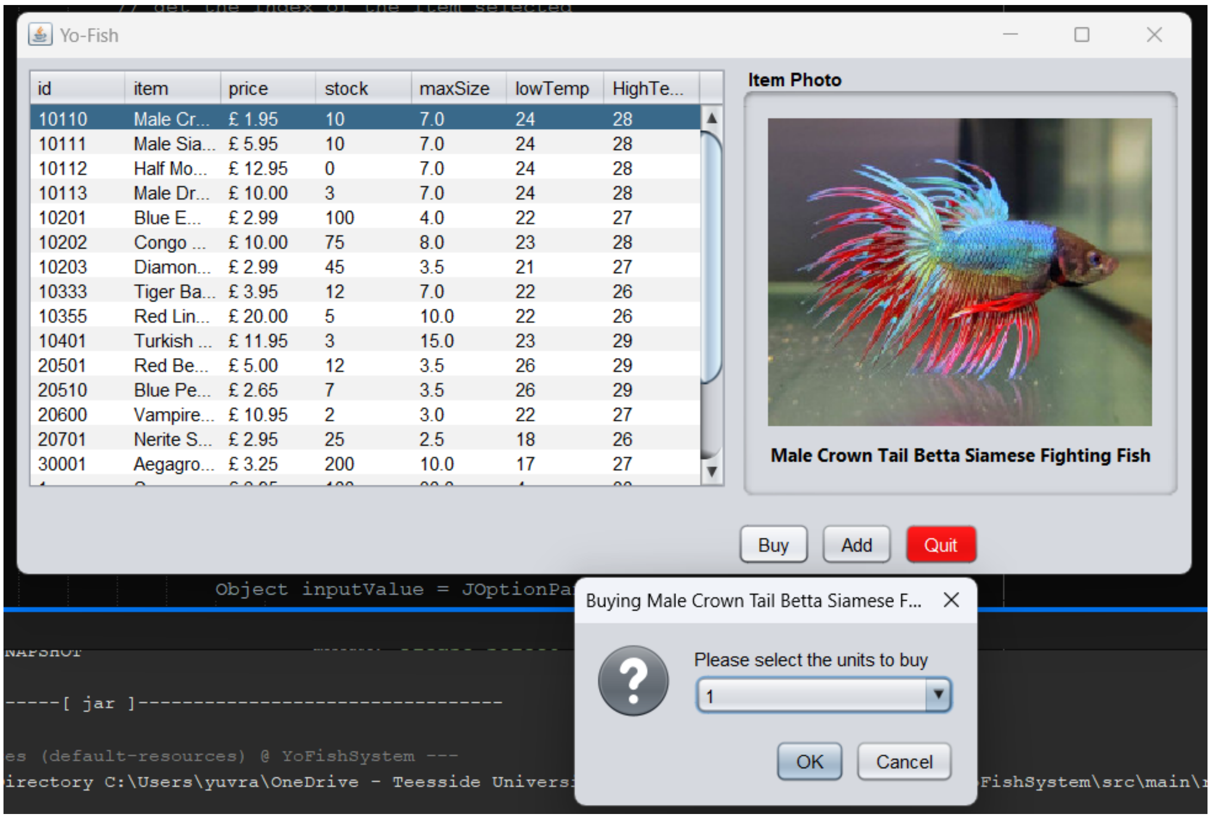
Half Moon Male Betta Siamese Fighting Fish

Warning

Apologies, Half Moon Male Betta Siamese Fighting Fish is out of stock!

Fig 4: Item out of stock (Warning)

3) **IF stock is greater than zero:** If the stock field of the selected 'YoFishItem' object has a value greater than 0 i.e. its in stock, then an input dialog box is shown to the user, asking him to select the quantity of items he wishes to buy.
A dropdown input has been chosen over a traditional input box because it eliminates the hassle of placing user input validations by restricting the user to choose amongst a list of valid inputs. In this case a dropdown of integers ranging from 1 to the maximum number of items in stock for the selected YoFish product. This takes away the risk of user entering an invalid input e.g. a senseless integer, alphabets or special characters.
At the end of a successful purchase a 'Confirmation of Sale' message box has been displayed to the user showcasing name of the fish bought, its price in pounds, number of units purchased and the stock remaining after the purchase.



id	item	price	stock	maxSize	lowTemp	HighTe...
10110	Male Cr...	£ 1.95	10	7.0	24	28
10111	Male Sia...	£ 5.95	10	7.0	24	28
10112	Half Mo...	£ 12.95	0	7.0	24	28
10113	Male Dr...	£ 10.00	3	7.0	24	28
10201	Blue E...	£ 2.99	100	4.0	22	27
10202	Congo ...	£ 10.00	75	8.0	23	28
10203	Diamon...	£ 2.99	45	3.5	21	27
10333	Tiger Ba...	£ 3.95	12	7.0	22	26
10355	Red Lin...	£ 20.00	5	10.0	22	26
10401	Turkish ...	£ 11.95	3	15.0	23	29
20501	Red Be...	£ 5.00	12	3.5	26	29
20510	Blue Pe...	£ 2.65	7	3.5	26	29
20600	Vampire...	£ 10.95	2	3.0	22	27
20701	Nerite S...	£ 2.95	25	2.5	18	26
30001	Aegagro...	£ 3.25	200	10.0	17	27

Male Crown Tail Betta Siamese Fighting Fish

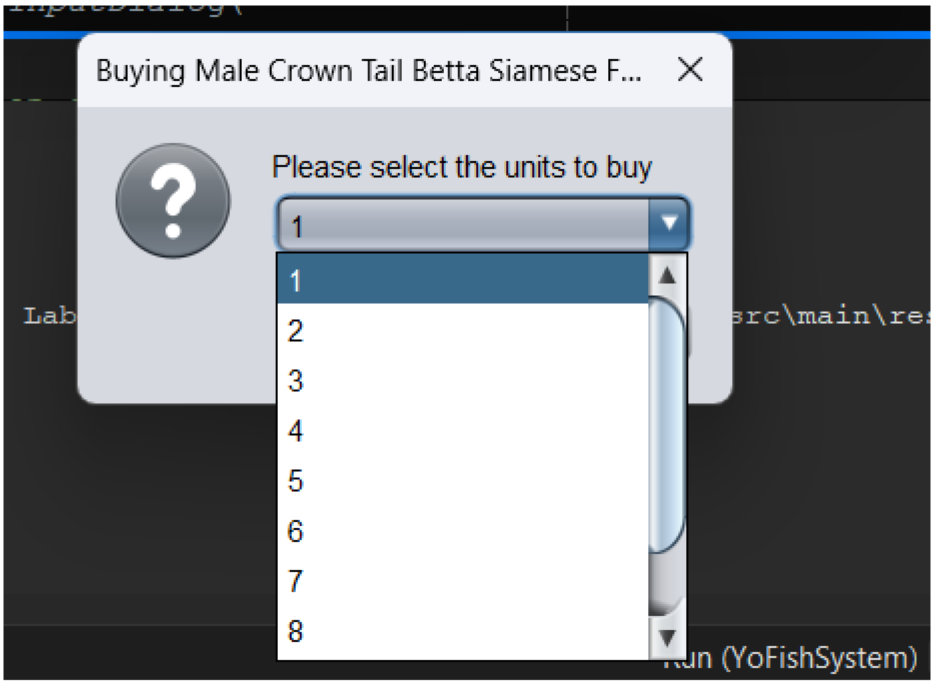
Buying Male Crown Tail Betta Siamese F...

Please select the units to buy

1

```
Object inputValue = JOptionPa
-----[ jar ]---------------------------------
es (default-resources) @ YoFishSystem ---
irectory C:\Users\yuvra\OneDrive - Teesside Univers          FishSystem\src\main\
```

Fig 5: Input dialog for buy operation



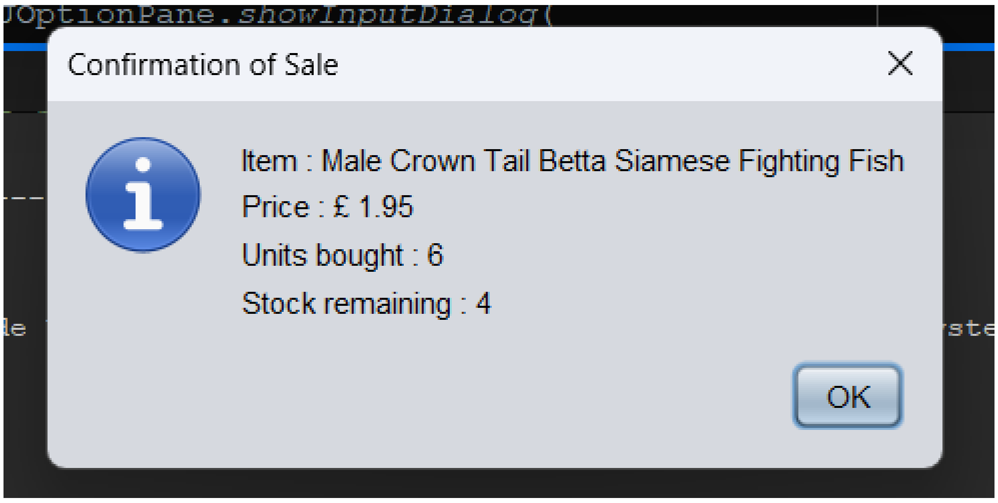
Buying Male Crown Tail Betta Siamese F...

Please select the units to buy

1

1
2
3
4
5
6
7
8

Fig 6: Dropdown input



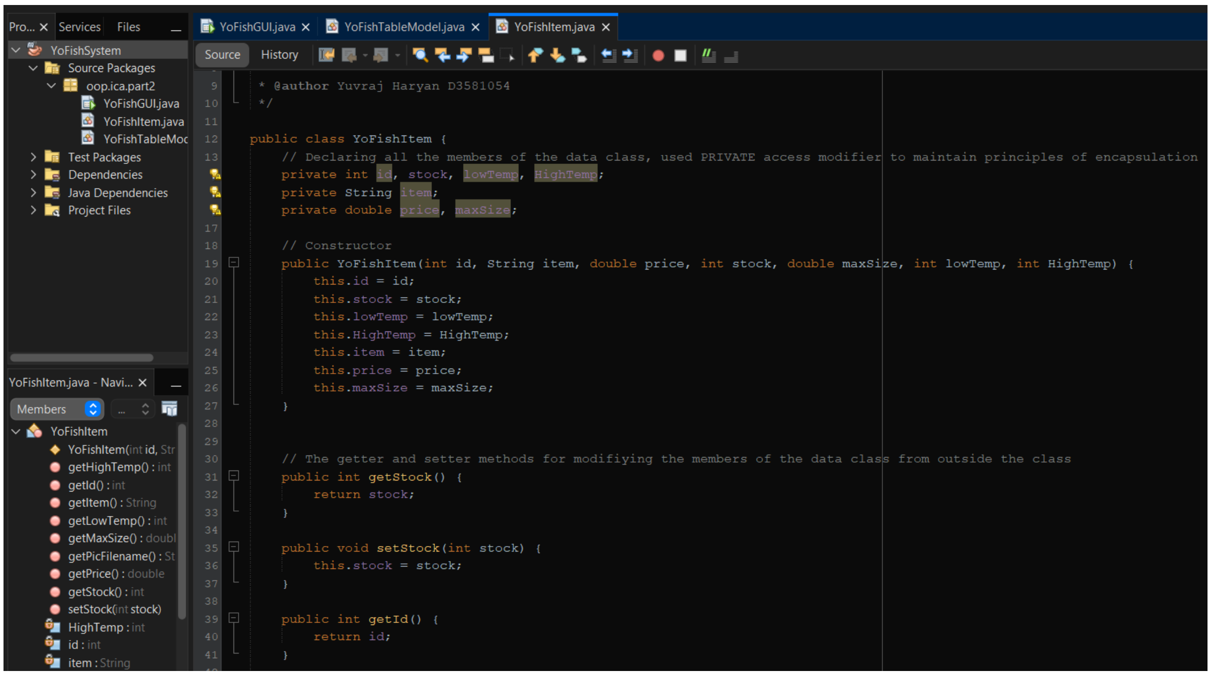
Confirmation of Sale

Item : Male Crown Tail Betta Siamese Fighting Fish
Price : £ 1.95
Units bought : 6
Stock remaining : 4

Fig 7: Confirmation of Sale dialog

Section 3: Discussion of Merged Application

Design principles of both the applications were to be studied to be able to merge them (Hu, 2023) .When the data classes of 'YoFishSystem' and 'Pondlife' were compared it was observed that both the classes have members which are similar. To be precise only one field wasn't common in the both the classes, namely the 'notes' field in the Pondlife data class which holds the description of the product being sold. And as the pondlife items were to be translated as YoFish items this field was chosen to be eliminated and rest of the fields for Pondlife products could be read and stored in the corresponding fields of the YoFish item while instantiating an 'YoFishItem' object.



```java
 * @author Yuvraj Haryan D3581054
 */

public class YoFishItem {
    // Declaring all the members of the data class, used PRIVATE access modifier to maintain principles of encapsulation
    private int id, stock, lowTemp, HighTemp;
    private String item;
    private double price, maxSize;

    // Constructor
    public YoFishItem(int id, String item, double price, int stock, double maxSize, int lowTemp, int HighTemp) {
        this.id = id;
        this.stock = stock;
        this.lowTemp = lowTemp;
        this.HighTemp = HighTemp;
        this.item = item;
        this.price = price;
        this.maxSize = maxSize;
    }


    // The getter and setter methods for modifiying the members of the data class from outside the class
    public int getStock() {
        return stock;
    }

    public void setStock(int stock) {
        this.stock = stock;
    }

    public int getId() {
        return id;
    }
```

```
YoFishItem
    YoFishItem(int id, Str
    getHighTemp() : int
    getId() : int
    getItem() : String
    getLowTemp() : int
    getMaxSize() : doubl
    getPicFilename() : St
    getPrice() : double
    getStock() : int
    setStock(int stock)
    HighTemp : int
    id : int
    item : String
```

Fig 8: Yo-Fish item data class

The code used for reading data from the ‘yo-fish.txt’ and ‘pondlife.txt’ input files is almost the same. Minor alterations were required where the application reads the comma separated values form both the input files as the sequence of values in the input file should match the order of parameters of the constructor for the ‘YoFishItem’ class. This was achieved simply by reading the values in different orders for different input files and adding them in the same sequence to the constructor parameters. ‘loadData()’ and ‘loadPondLifeData()’ were the 2 functions created for loading data from the Yo-Fish and Pondlife input files respectively.

Section 4: Discussion of Improvement

Improvement: Not offering the user an option to add new items for sale to the application has made the application rigid and is limiting its used case. User should be provided with a functionality to add new products to the YoFishSystem. This would make the application more useful and fit for the purpose.

Changes required to the application: A new button needs to be added to the main GUI menu for user to add a new item to the product list ‘Add new item’. An input form in the form of dialog needs to be introduced which would receive data regarding the item e.g. its name, price, stock units, etc. This would be then displayed in the main table on the GUI once the item is successfully added.

Code implementation: Event listeners should be set up for the new ‘Add new item’ button. Code should be written for popping-up a new dialog containing the input form for the item data. The data entered by the user in the input fields needs to be validated at code level to ensure that acceptable data types have been provided. Using valid data a ‘YoFishItem’ object should be instantiated which will be appended to the main Array List and also to the data object of the ‘YoFishTableModel’ (AbstractTableModel) so that it will be reflected in the JTable.

Testing the improvements: When developing a GUI application testing the event listeners by generating events for the event queue is crucial (Newmarch, 1999). Mainly the input form for receiving the data regarding the new item needs to be tested. Test cases are required to be written focusing on checking the input validations. Application should be inputted with invalid data and edge cases to verify the error handling mechanism. Buying and restocking operations should also be performed once on the newly added items.

References

Calçada, A. and Bernardino, J. (2022) 'Experimental Evaluation of Low Code development, Java Swing and JavaScript programming', *Proceedings of the 26th International Database Engineered Applications Symposium,* , pp. 103-112. Available at: https://doi.org/10.1145/3548785.3548792

Hu, C. (2023) *An Introduction to Software Design : Concepts, Principles, Methodologies, and Techniques.* 1st edn.

Newmarch, J.D. (1999) 'Testing Java Swing-based applications', *Proceedings Technology of Object-Oriented Languages and Systems (Cat. no.PR00393),* , pp. 156-165. Available at: https://doi.org/10.1109/TOOLS.1999.796479

Saudagar, S. *et al.* (2023) 'ML-based Java UI for Residence Predictor', *2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT),* , pp. 838-843. Available at: https://doi.org/10.1109/IDCIoT56793.2023.10053480