LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS5004NA Emerging Programming Platforms and Technologies

Assessment Weightage & Type

30% Group Coursework

Title (Where Required):

Year and Semester: 2020-21Autumn

Group Name:			
SN	**Student Name**	**College ID**	**University ID**
1.	**Rhythm Bhandari**	**Np01cp4a190013**	**19032177**
2.	**Anurag Dahal**	**Np01cp4a190137**	**19032165**
3.	**Simon Poudyal**	**Np01cp4a190091**	**19031695**
4.	**Binisha Tandukar**	**Np01cp4a190119**	**19030859**

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Table of Contents

Table of Figures

Table of tables

1. Proposal

For this coursework, we have decided to build a "Meat products information system." As per the title, its key feature will be to add and search information regarding meat products. Learning how to use a software is cognitively demanding and often frustrating, therefore, to alleviate that problem, we will mainly be focusing on making the system clutter-free and easy to navigate. However, ease-of-navigation is just one part of the puzzle. An unreliable software, no matter how easy it is to use, is worthless. Hence, aside from making it user-friendly, we're concerned on making it reliable as well.

1.1 List of Data

• Product ID: A unique ID(number) which represents only one single product. (Integer)

• Product Name: Name of a particular product. (String)

• Type of Meat: This data represents the types of meat (Mutton, Chicken, Fish, Pork, Buffalo) which are to be selected via a combobox.

• Available Qty: The available amount of a particular meat product in the store. This data will be in Kgs and is to be selected via a combo box. (Integer)

• Discount: Lets the user know whether discount is available in that product or not. It will have two options (Yes or No) to be chosen via radio buttons and will add a 10% discount to the final price if it's checked Yes. (String)

• Price Per Kg: It represents the selling price of a meat product on a per kg basis. (Integer)

1.2 Features

• The added information will be displayed in a table.

• The UI will let the user add information regarding meat products with less to no effort.

• After the addition is successful, appropriate feedbacks will be displayed to reduce confusion.

- Adding to the feature above, it'll have the option to "search by type" as well.

1.3 Tools Used

In order to build this system, we've decided to take help of two software. Adobe XD for wireframes and NetBeans for development.

Adobe XD: Every software needs some sort of map or guideline as to how it should look and feel. Therefore, here we've taken the help of Adobe XD to create wireframes for our project. The key reason behind choosing XD is the rapidity and easiness with which it allows us to create designs and prototypes.

NetBeans: For coding our software and designing the UI, we used a free open-source IDE which has fully integrated tools. NetBeans provides the users with error detection features, auto-complete suggestion and a drag and drop mechanism to design UI in a very easy and intuitive way. This IDE also supports any Operating System that runs JVM and allows the user to take use of multiple project windows at once.

2. Individual Tasks

Rhythm Bhandari	1. Code for searching items based on price, validations and importing file data. 2. Report - Method description. 3. Report - Working mechanism of Binary Search. 4. Conclusion.
Binisha Tandukar	1. Proposal 2. Verified References. 3. GUI of the program - JTable. 4. Report -Formatting.
Simon Poudyal	1. Introduction. 2. GUI of the program. 3. Code for searching items based on type. 4. Report- Implementation of Binary Search .
Anurag Dahal	1. Testing 2. Wireframes and Illustrations. 3. GUI of the program. 4. Report - Implementation of Sorting algorithm and Working mechanism of Selection sort.

3. Introduction

This coursework chiefly concerns with building a meat products information system. Subsequently, it has been divided into three main parts: Descriptions of algorithms used, the program along with its GUI, and testing.

In the descriptions part, two algorithms have been elaborated. The first is Binary Search algorithm and the second is Selection sort algorithm. Both of them have been further divided into two parts: description of the algorithmic process and how they have been used in the program.

The actual program consists of two main functionalities: product searching and product adding. The former has two core functions, searching via price and searching via category. The latter, on the other hand, helps add the product in the information system by asking for information such as the product's name, its quantity, its type etcetera.

The third or the final part concerns with testing the program. Its purpose is to test the program for errors, which help in iteration thereby making it fully functional and usable. Hence, a total of six tests regarding potential errors have been conducted and documented in this coursework.

As has been demonstrated repeatedly by countless studies, people find attractive things easier to work with (Norman, 2004). Hence, special attention has been given to the aesthetics and usability of the program as well.

The goal of this coursework is to make students familiar with the core functionalities and methods of Java through a real-world task. By making the students explain the algorithms in their own words, the coursework also aims to establish a proper understanding of the processes involved.

Similarly, the testing portion asks students to document the testing procedures by clearly stating the actions taken, the expected results, and the final result. This helps solidify the necessary skills required for solving real world problems and documenting the solutions through the use of programming.

4. Binary Search Algorithm

4.1 Working mechanism of binary search algorithm

Binary search is an algorithm which works by repeatedly dividing in half the portion of the list that could contain the item, until you've narrowed down the possible locations to just one (Academy, 2021). The algorithm only works when the array or list in concern is sorted (Point, 2021).

To put that into perspective, here's an example.

Consider an array of length 5. It has five elements: (1, 2, 3, 4, 5) and the 1 is the number that the algorithm should search for (the search-value).

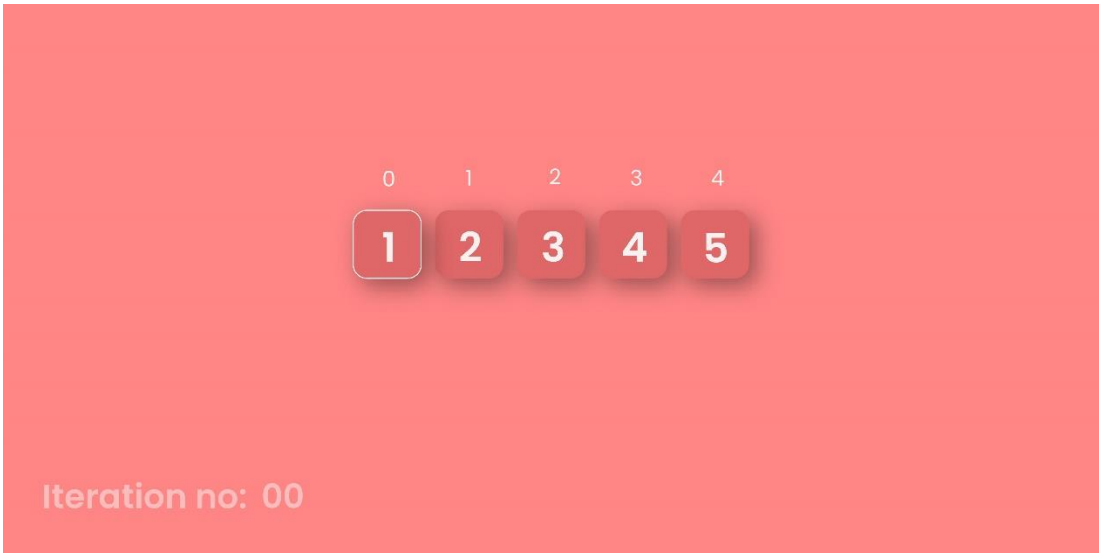

Figure 1: Binary Search 1

In the first iteration the algorithm selects index 0 as the higher-index and index 4 as the lower-index. The mid-index, then, is found by adding the highest and lowest index, and dividing them by 2. Which results in an index of 2.

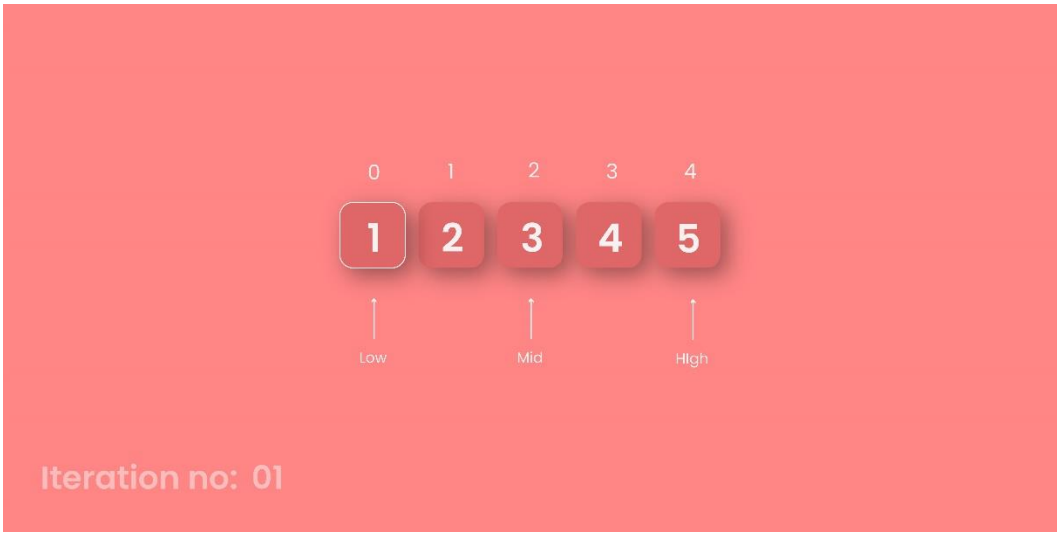

Figure 2: Binary Search 2

The algorithm, then, compares the search-value with the mid-value. If it is equal, the algorithm displays the index of the value, if it is not, the algorithm checks whether the search value is higher or lower than the mid-point.

If the search value is higher, the algorithm sets the lower-index to mid-index + 1. If it is lower, the algorithm sets the higher-index to mid-index+1. Afterwards, the program re-iterates.

In the example concerned, the algorithm compares the mid value 3 with our search value 1. As 1 is not equal to and is lower than 3, the program sets the higher-index to 1 and re-iterates.

The mid-point now becomes index 0, as (0+1)/2 is equals to 0.5, which, after rounding-off becomes 0.

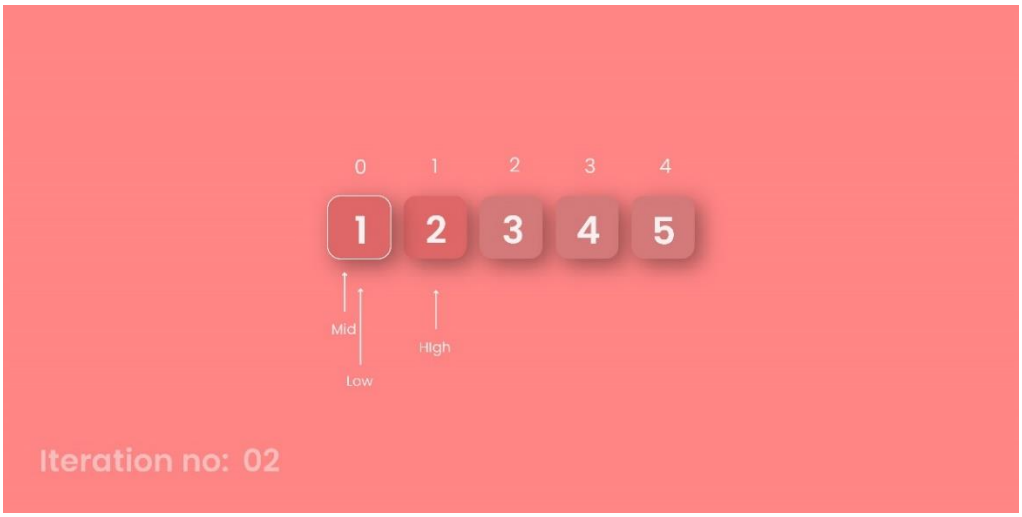

Figure 3: Binary Search 3

The program now compares the value of the mid-index (mid-value), i.e. 1 with the search-value. The comparison matches and the program displays the mid-index.

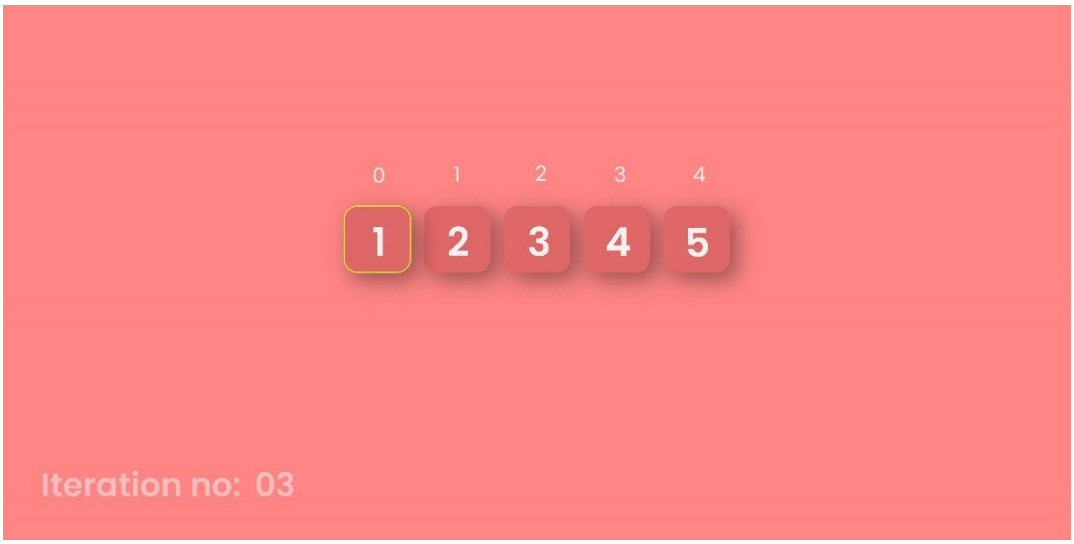

Figure 4: Binary Search 4

4.2 Implementation of Binary Search in the program.

The function binarySearch() has been used in the program to search products by price in the information system.

It takes four values: ArrayList<String> arr, int low, int high, double find. The variable “arr” is the ArrayList where the value is to be found. “Int low”, and “int high” represent the higher and lower index of the array respectively and finally, the “double find” represents the value to be found in the ArrayList.

```
public static int binarySearch(ArrayList<String> arr, int low, int high, double find){
```

Figure 5: Binary Search Implementation 1

The ArrayList arr is converted to double using the conString method and is assigned to a newly created “ArrayList<Double> prr.”

```
ArrayList<Double> prr = conString(arr);
```

Figure 6: Binary Search Implementation 2

A loop which runs when the lower index is lesser than or equal to the higher index, is set up. The mid-index is found by adding “int low” and “int high” and subsequently dividing it by 2. Afterwards, the result is stored at a variable named “int mid.”

```
while(low <= high){
    int mid = (low + high) / 2;
```

Figure 7: Binary Search Implementation 3

The value at prr(mid) is then compared with the "find". If it matches then the program returns mid, i.e. the index of our "find."

```
if (prr.get(mid) == (find)) {
    return mid;
}
```

Figure 8: Binary Search Implementation 4

If the value at prr(mid) doesn't match the "find", and is subsequently lower than the "find", the lower index or "int low" is set to "mid + 1"

```
else if (prr.get(mid) < find) {
    low = mid + 1;
```

Figure 9: Binary Search Implementation 5

If the value at prr(mid) doesn't match the find, and is subsequently higher than the "find", the higher index or "int high" is set to "mid -1"

```
}else{
    high = mid - 1;
}
```

Figure 10: Binary Search Implementation 6

Through similar successions the ArrayList keeps on getting divided into halves and the loop keeps on running until the 'find' is found. When it is found, the program returns its index.

5. Selection sort algorithm

5.1 Working mechanism of selection sort algorithm

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning (Geeksforgeeks, 2019).

Here is a list of five elements: (3, 2, 4, 1, 5), starting from an index of 1.

First, the algorithm selects the value at the lowest index (3) as minimum and compares it to other values in the list.

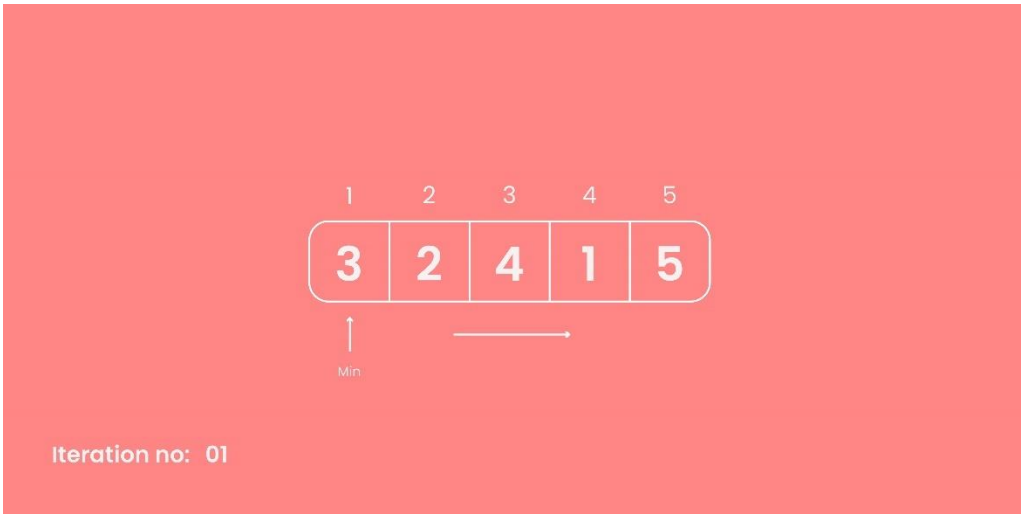

Figure 11: Selection sort 1

It then finds 1 at the fourth index and swaps it for 3. Now the algorithm selects the value at index 2 as the minimum and re-checks for values lower that that.

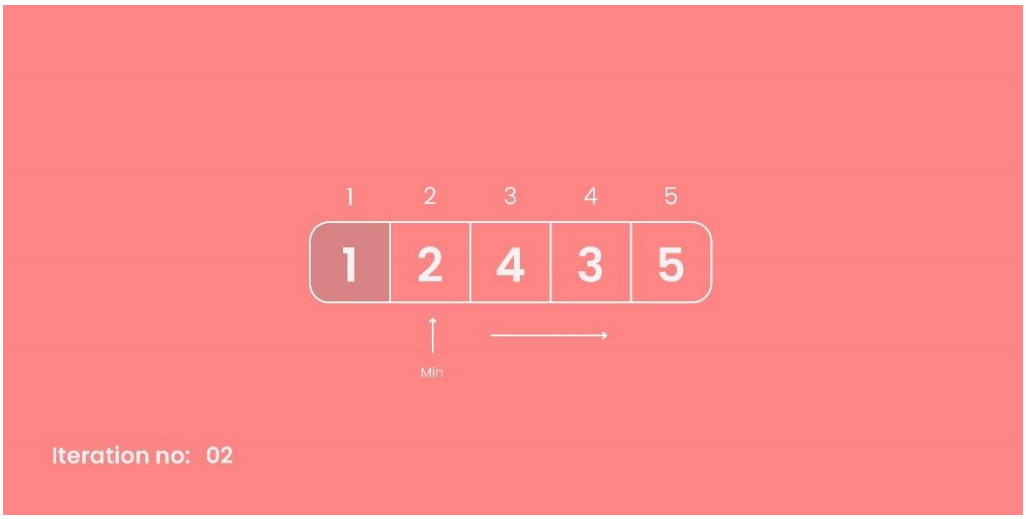

Figure 12: Selection sort 2

It finds no other value lesser than 2, hence it is left where it is. Now the algorithm selects the value at index 3 as the minimum and re-checks for values lesser than that.

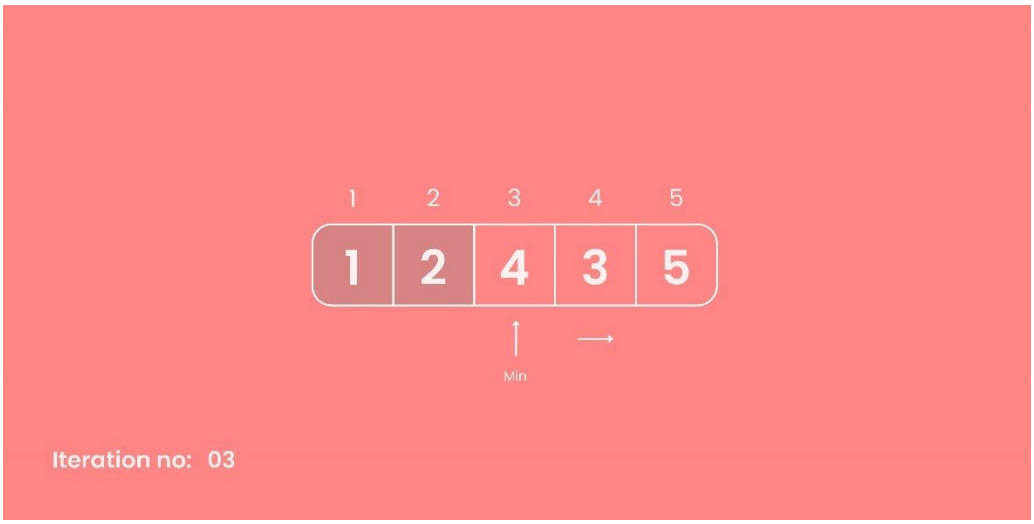

Figure 13: Selection sort 3

Subsequently, it finds 3 at index 4 and swaps places. The program, then, selects 4 as the minimum value and goes on to compare it with other values in the list.

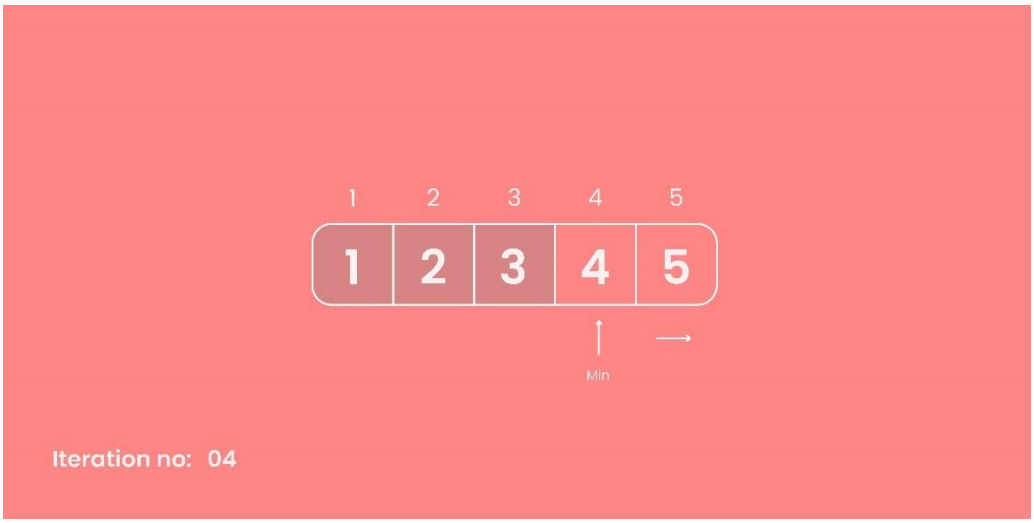

Figure 14: Selection sort 4

It finds no element lesser than 4, therefore 4 is left at index 4.

Finally, 5 is selected as the minimum value, and because there are no other values in the array to compare to, the algorithm stops.

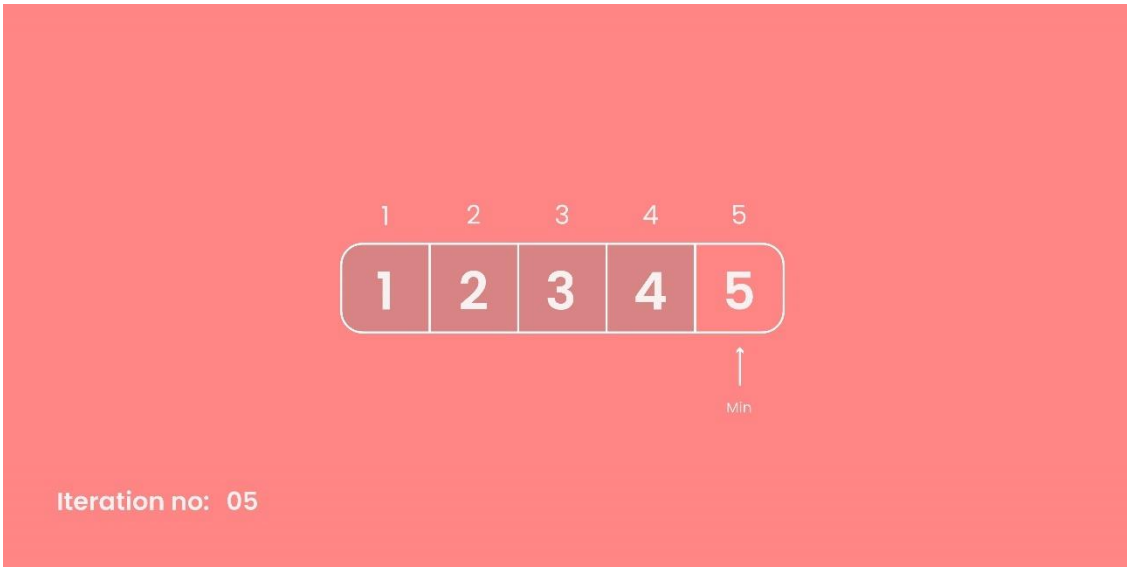

Figure 15: Selection sort 5

Here's the sorted list.

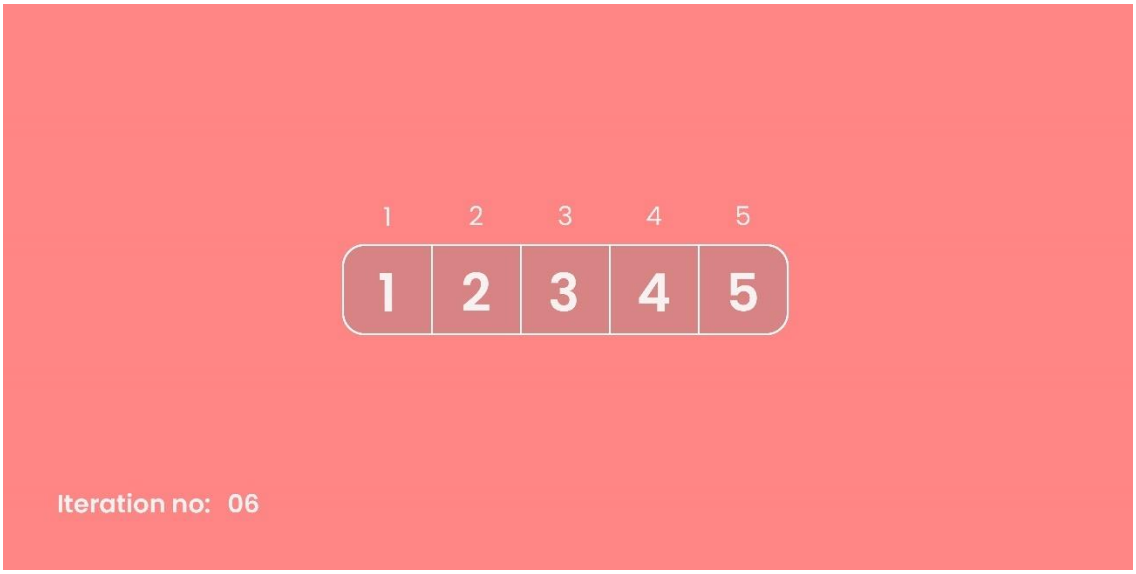

Figure 16: Selection sort 6

5.2 Implementation of selection sort in the program

The ArrayList containing price is sorted by the doSelectionSort() function in the program.

Here is how it has been implemented.

Firstly, an ArrayList of ArrayLists (aList) is passed through the function doSelectionSort().

```
public static void doSelectionSort(ArrayList<ArrayList<String>> aList) {
```

Figure 17: Selection sort implementation 1

The ArrayList at index 5 of aList is converted to Double by using the conString method and is assigned to the newly created ArrayList called newList.

```
    ArrayList<Double> newList = conString(aList.get(5));
```

Figure 18: Selection sort implementation 2

A loop, with int i = 0, which runs until i is less than the size of newList is created. Subsequently another int p is created which stores the lower index of the particular iteration.

Afterwards, another loop within the first loop is created. This loop checks whether there is any value less than the minimum value of P. If the condition is satisfied, the value of the int j is swapped with the value of int p.

```
    for (int i = 0; i < newList.size(); i++) {
        int p = i;
        for (int j = i; j < newList.size(); j++) {
            if (newList.get(j) < newList.get(p))
                p = j;
        }
```

Figure 19: Selection sort implementation 3

Now a new variable “double min” is created which stores the value of newList at index p. Afterwards, , the value at index p is assigned with the value at index i. Subsequently, the index i, or index 0 of newList is assigned with min.

```
}
double min = newList.get(p);
newList.set(p, newList.get(i));
newList.set(i, min);
```

Figure 20: Selection sort implementation 4

The program goes through iterations, until the list is sorted.

Now, after the ArrayList having price values is sorted. Other ArrayLists containing other value have to be sorted as well. For that purpose, the function sortOtherValue() has been called. It takes the same values of int p and int i, and rapidly sorts itself.

```
for(int k = 0; k < 5; k++){
    sortOtherValue(aList.get(k),p,i);
```

Figure 21: Selection sort implementation 5

```
public static void sortOtherValue(ArrayList<String> nam, int p, int i){
    String man = nam.get(p);
    nam.set(p,nam.get(i));
    nam.set(i, man);
}
```

Figure 22: Selection sort implementation 6

6. Method Description

Class MeatProductInfo

1. **private void mainAddBtnActionPerformed** – This method is notified when the JButton mainAddBtn (Add a Product) in main panel is clicked. This sets add panel to visible and hides other panels. This takes the user to the add product interface.

2. **private void mainSearchBtnActionPerformed** – This method is notified when the JButton mainSearchBtn (Search a Product) in main panel is clicked. This sets search panel to visible and hides other panels. This takes the user to the search interface.

3. **private void searchBackMouseClicked** – This method is invoked when the JLabel searchBack (< Back) in search panel is clicked. This sets main panel to visible and hides other panels. This takes the user back to the main interface.

4. **private void addButtonActionPerformed** – This action performed method is notified when the JButton addButton (Add Product) is clicked. This method retrieves the user inputs in the form and stores in respective String variables. Variables meatID, name and price get value from respective JTextField. Whereas type and weight getSelectedItem() from respective JComboBox. String discount retrieves from selected JRadioButton. If the user input is validated successfully, data is stored in tables (productTable and allTable). Also, the meatID and name are stored in two ArrayList (listID and listName) for future checking of repeating data. Else, suitable message reporting the error is displayed.

5. **private void existingBackMouseClicked** – This method is invoked when JLabel existingBack(< Back) is clicked. This sets add panel to visible and hides other panels. This takes the user back to the add product interface.

6. **private void priceSearchBtnActionPerformed** – This method is notified when the JButton priceSearchBtn(Search By Price) is clicked. Firstly, it calls clearData() method to eliminate errors of stored data. Then, addData() method is called which adds data to the

ArrayList meatList from the JTable. Entered price in the JTextField is stored in String sPrice which is further validated. If user input is correct, meatList is passed through doSelectionSort() which uses the selection sort technique to sort data in ascending order of price. ArrayList of price, lowest price index, highest price index and user entered price is passed through binarySearch() method which returns the index of user entered price. Finally, message dialog of searched price info is shown. Respective validation errors are notified in message box.

7. **private void showAllBtnActionPerformed()** – This method is notified when the JButton showAllBtn (Show all products) in search panel is clicked. This sets all Products panel to visible and hides other panels. This takes the user to the All Products interface where JTable allTable is shown.

8. **private void allBackMouseClicked() -** This method is invoked when JLabel allBack(< Back) is clicked. This sets search panel to visible and hides other panels. This takes the user back to the search product interface.

9. **Private void typeSearchBtnActionPerformed() -** This method is notified when the JButton typeSearchBtn(Search By Type) is clicked. Firstly, it calls clearData() method to eliminate errors of stored data. Then, addData() method is called which adds data to the ArrayList meatList from the JTable. String type retrieves the selected value from JComboBox typeSearchBox. According to the selected value, typeSearch(type) is called which returns an array. That array is used to display the number of products and their names in a message box. If the returned array is empty, appropriate message is displayed.

10. **private void typeSearchbtnMouseEntered()** – This sets different color to the button creating an animation when the cursor enters the button.

11. **private void jMenuImportActionPerformed()** – When the jMenuImport (‘Open’) menu item of jMenuFile(File) is clicked this method is notified. This prompts the user a Yes/No question (Do you want to choose file?) where ‘Yes’ initializes 0 in int userAnswer and ‘No’ initializes 1. Switch case is used according to userAnswer where in case 1, importFile is initialized with default file and in case 2, JFileChooser is used where the user searches and selects importFile as required. Then, the BufferedReader class is used to read importFile. With proper formatting, words are split according to ‘/’ in between and stored in an array row. Row is added in the table. Loop is used to repeat this process for all lines.

12. **private void jMenuExitActionPerformed()** – When the jMenuExit (‘Exit’) menu item of jMenuFile(File) is clicked this method is notified. This method asks for user confirmation in a Yes/No question(Do you want to close the program?). Clicking ‘Yes’ closes the program.

13. **private void jMenuClearActionPerfomed()** - When the jMenuClear (‘Clear) menu item of jMenuFile(File) is clicked this method is notified. This method initializes the number of rows in an int rowC. If the number of rows is more than 0, it clears both tables and calls clearData() and clearCheckData. This clears everything in the program.

14. **private void jMenuHelpMouseClicked()** - When the jMenuHelp(Help) is clicked this method is notified. This then opens the user help file “UserManual.pdf”.

15. private void clear() – When called this method clears all the JTextFields, selection of JComboBoxs and JRadioButton selection.

16. **public void clearData()** – When called this method clears the ArrayLists meatList, mID, mName, mType, mWeight, mDiscount, mPrice.

17. **public void addCheckData()** – When called this method adds meatID to the ArrayList listID and name to the ArrayList listName taking data from the productTable.

18. **public void clearCheckData()** – When called this method clears the ArrayLists listID and listName.

19. **public void addData()** - When called this method gets the number of rows present in productTable and initializes it in int rowCounter. rowCounter is used in loop to retrieve individual value from each box in a row and assigned to String variables. Respective string variable is added to ArrayList like String meatID to ArrayList mID. All the ArrayLists mID, mName, mType, mWeight, mDiscount, mPrice are added to ArrayList of ArrayList meatList.

20. **private String[ ] typeSearch(String type)**- When called with the parameter type, this method returns a String array. 5 empty strings are initialized and an array details of length 10. For loop to cover all rows is used. And an if loop is used to keep it in bounds. According to the value from the table, switch case is used to store it in respective meat type. For example, if the meat is of pork, name of the meat is stored in String porkDetails and pCount is increased. All the names and counts are stored in array details which is returned.

21. **public static void doSelectionSort(ArrayList<ArrayList<String>> aList)** – This method is called to sort the ArrayList of ArrayList. It sorts all the ArrayLists in aList according to one ArrayList of aList. In this case, ArrayList of price is used. The String ArrayList is passed through conString to return ArrayList<Double> newList. Selection sort is used to sort the newList. To sort other ArrayLists former and new indexes with ArrayList are passed through sortOtherValues. mPrice of meatList is cleared and replaced with the new order of prices in String.

22. **public static void sortOtherValue(ArrayList<String> nam, int p, int i)** – This method is called to sort other ArrayLists. According to the index of price and their order, values are sorted accordingly.

23. **public static int binarySearch(ArrayList<String> arr, int low, int high, double find)** – This method is called when searching for price. It returns the index of searched price.

String ArrayList is converted to ArrayList of Double passing through conString(). The binary searching algorithm is used to find index of price.

24. **public static ArrayList conString(ArrayList<String> arr)** – This ArrayList returns ArrayList of Double. New ArrayList of Double newList is declared of the same size as arr. Double.parseDouble is used to change datatype from String to Double.

7. Testing

7.1 Test to show whether the program can run in Netbeans.

Test Number	1
Testing Purpose	To show whether the program can run in Netbeans.
Action Performed	The program was compiled and run in Netbeans.
Expected Result	The program will run successfully.
Actual Result	The program ran successfully

Table 1: Test 1

Screenshots

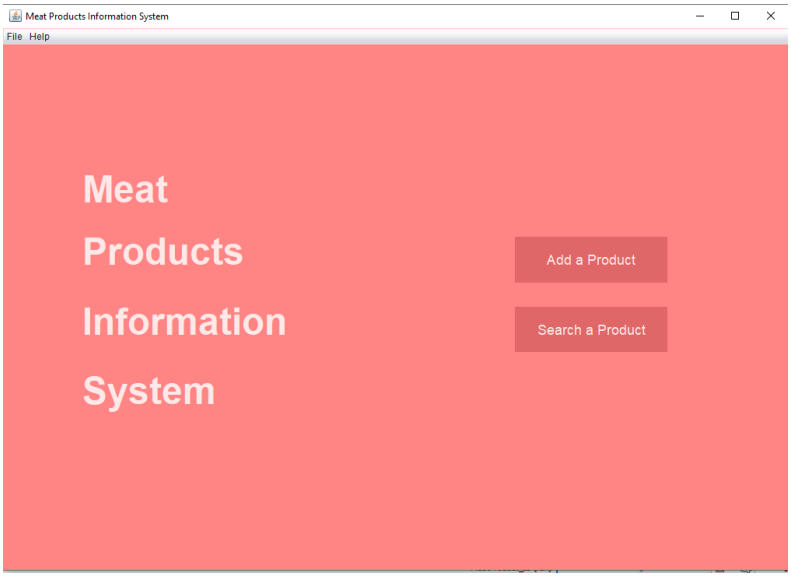

Figure 23: Running the program

7.2 Testing to show whether new product can be added or not.

Test Number	2
Testing Purpose	To show whether new product can be added or not.
Action Performed	Appropriate textboxes were filled with appropriate data and the add product button was clicked.
Expected Result	The product will be added successfully.
Actual Result	The product was added successfully.

Table 2: Testing 2

Screenshots

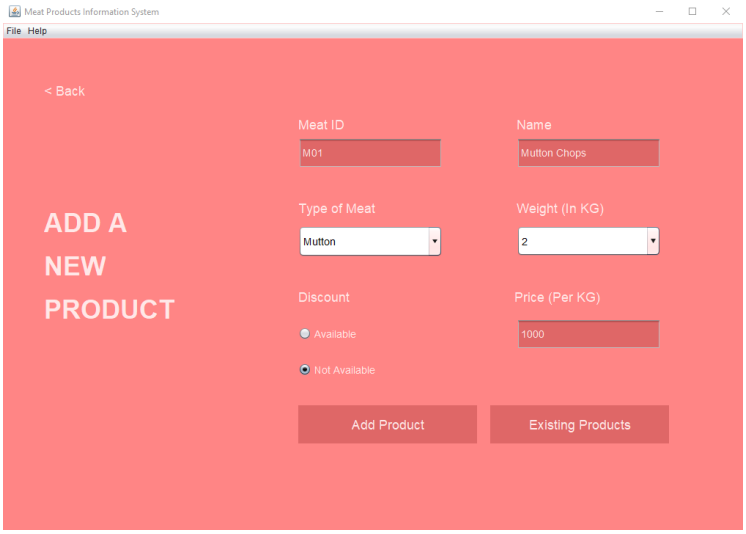

Figure 24: Adding a product.

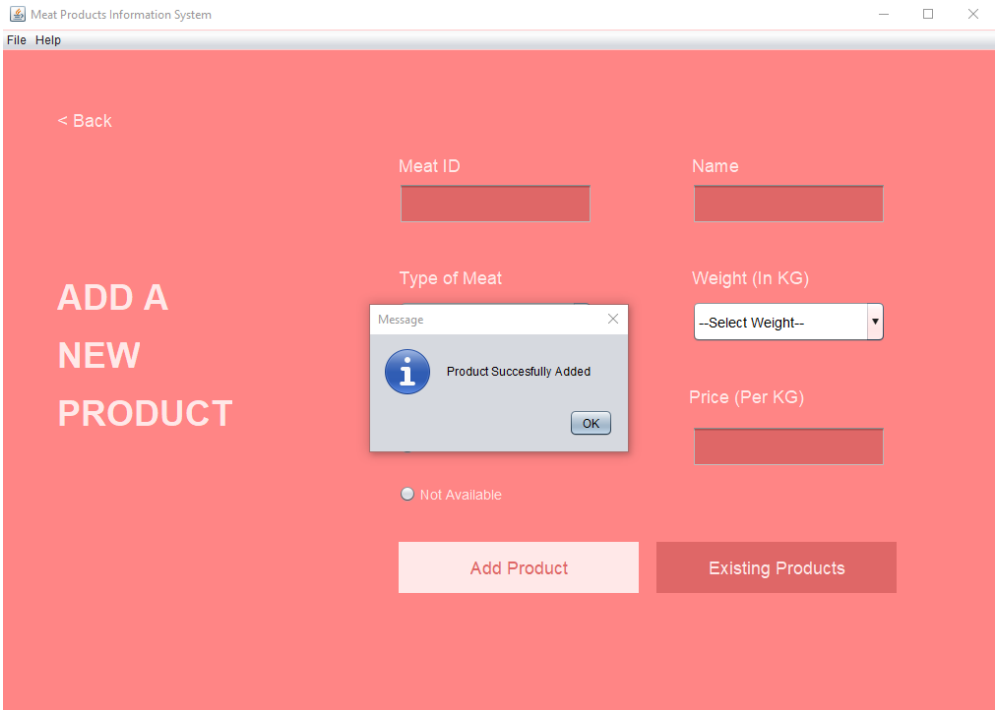

Figure 25: Product Successfully added.

7.3 Testing to show whether the products can be searched by price.

Test Number	3
Testing Purpose	To show whether the products can be searched by price.
Action Performed	A price of 1000 was put in the textbox and the search by price button was clicked.
Expected Result	The results will display successfully.
Actual Result	The message box displayed the results successfully.

Table 3: Test 3

Screenshots

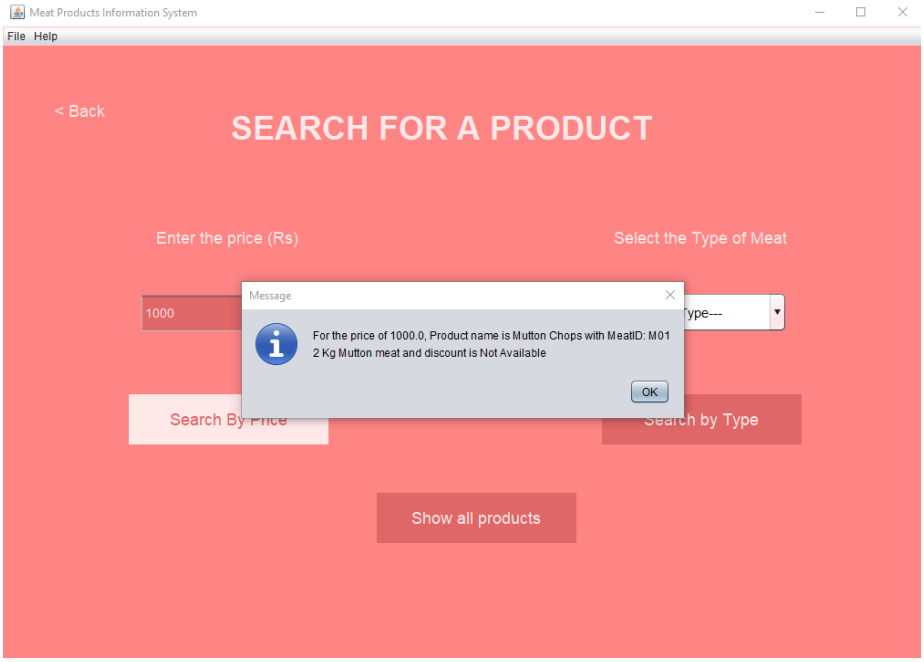

Figure 26: Searching for a product.

7.4 Testing to show whether the product can be searched by type.

Test Number	4
Testing Purpose	To show whether the product can be searched by type.
Action Performed	Type 'Mutton' was selected from the combo box and the search by type button was entered.
Expected Result	The result will be displayed successfully.
Actual Result	The result was displayed successfully by a message box.

Table 4: Test 4

Screenshot

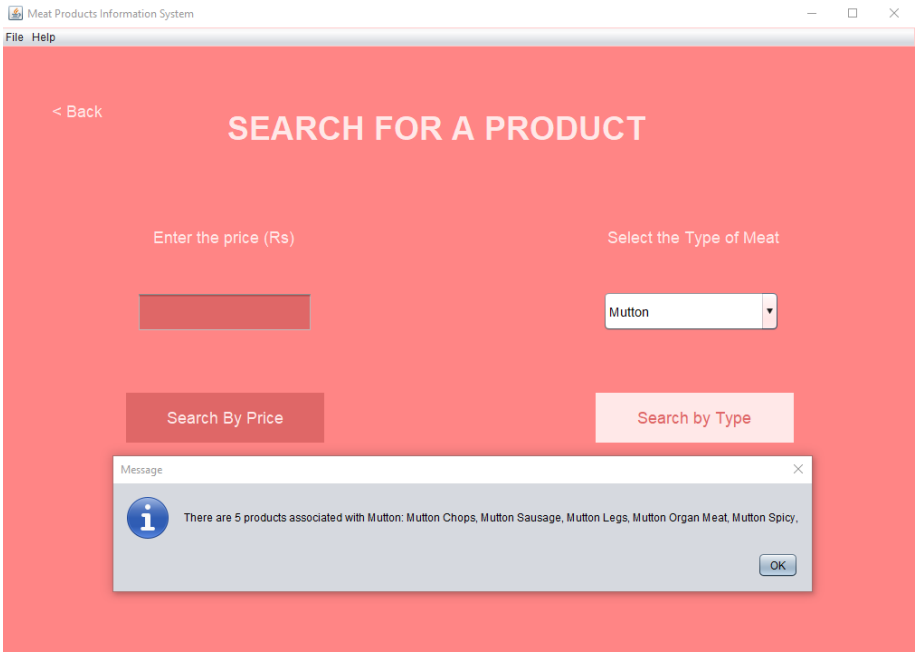

Figure 27: Search by type

7.5 Testing to show whether the program accepts unsuitable values.

Test Number	5
Testing Purpose	To show whether the program accepts unsuitable values.
Action Performed	A numeric value was entered in the Name textbox and the add product button was clicked.
Expected Result	A message box that indicates that the value is unsuitable appears.
Actual Result	The message box appeared successfully.

Table 5: Test 5

Screenshot

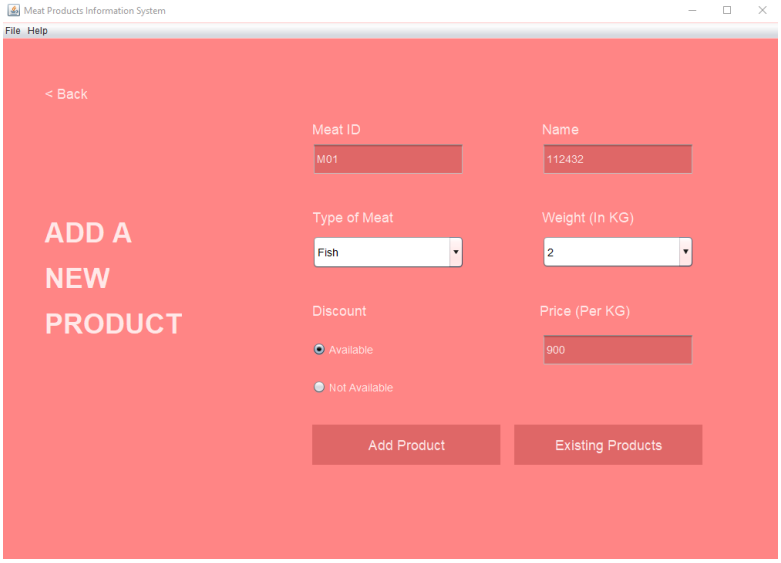

Figure 28: Adding unsuitable values

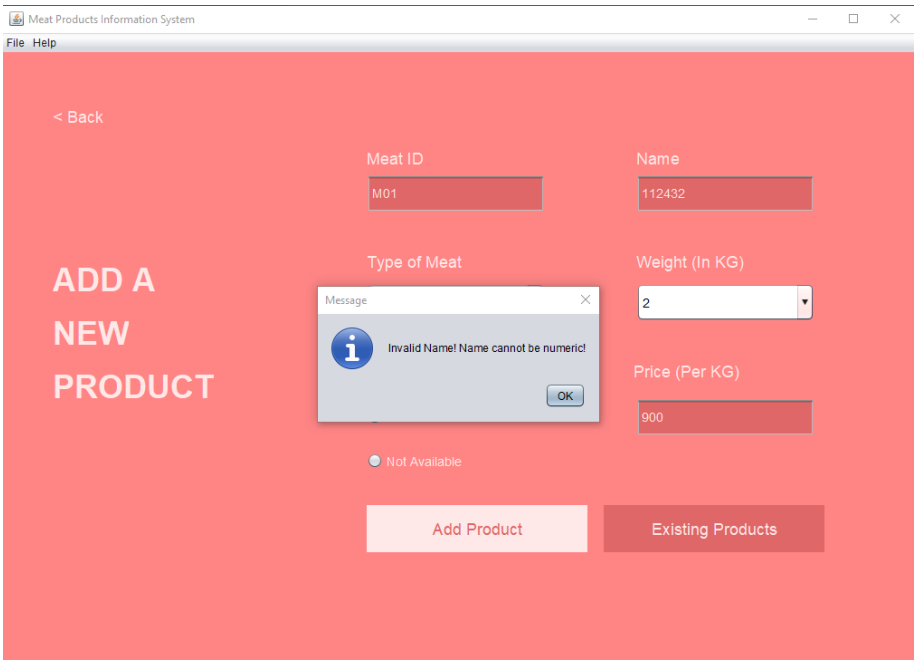

Figure 29: Unsuitable values error message

7.6 Testing to show whether the program accepts empty values

Test Number	6
Testing Purpose	To show whether the program accepts empty values.
Action Performed	The Meat ID textbox was left empty and the add product button was clicked.
Expected Result	A message box that indicates that the value is unsuitable appears.
Actual Result	The message box appeared successfully.

Table 6: Test 6

Screenshots

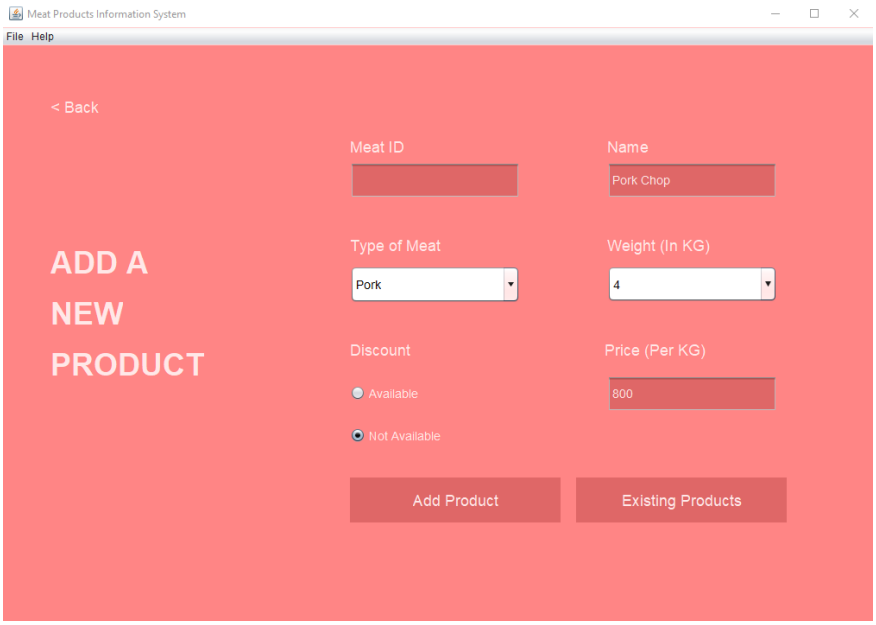

Figure 30: Leaving textboxes empty

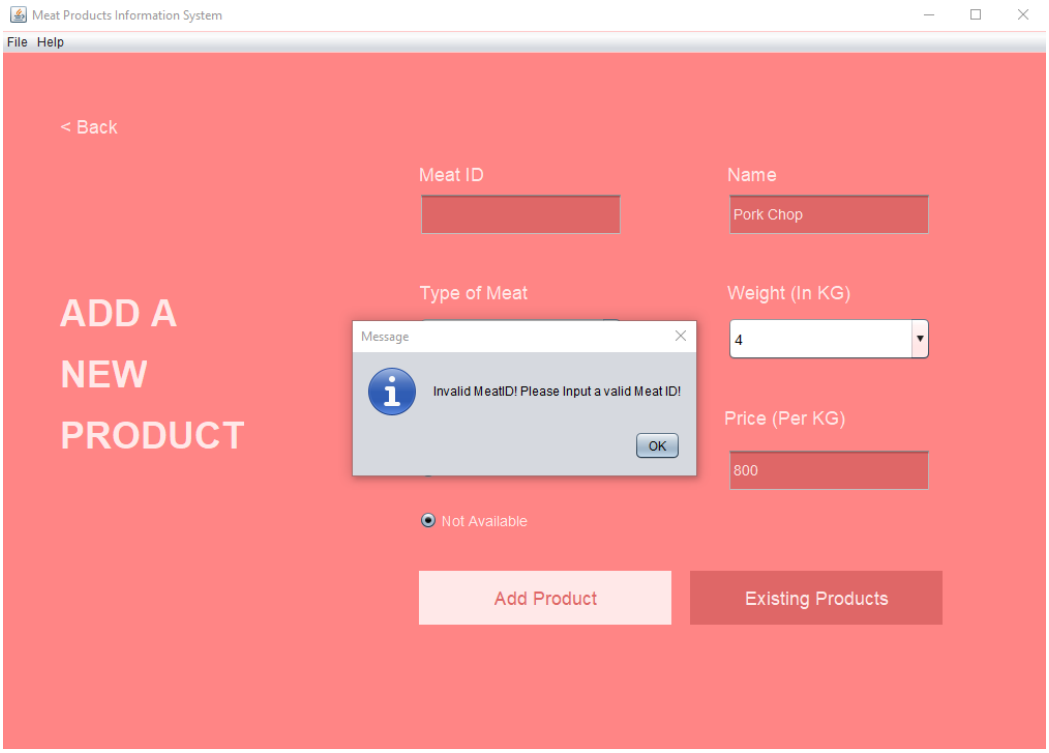

Figure 31: Leaving textboxes empty error message

7.7 Testing to show whether the program can import a file from menu.

Test Number	7
Testing Purpose	To show whether the program can import a file from menu.
Action Performed	The file menu was clicked, and the open button was clicked.
Expected Result	The file will be imported.
Actual Result	The file was imported successfully.

Table 7: Test 7

Screenshots

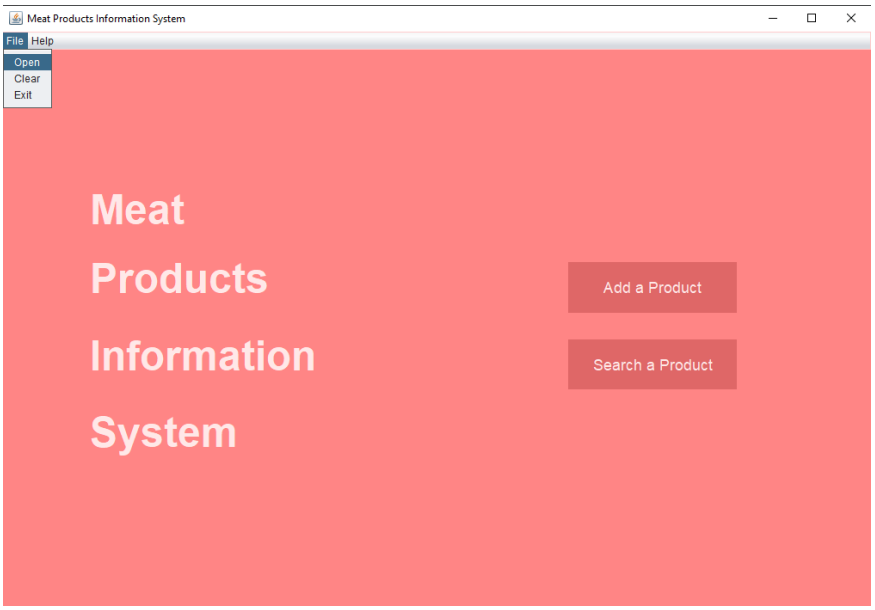

Figure 32: Importing a file from menu

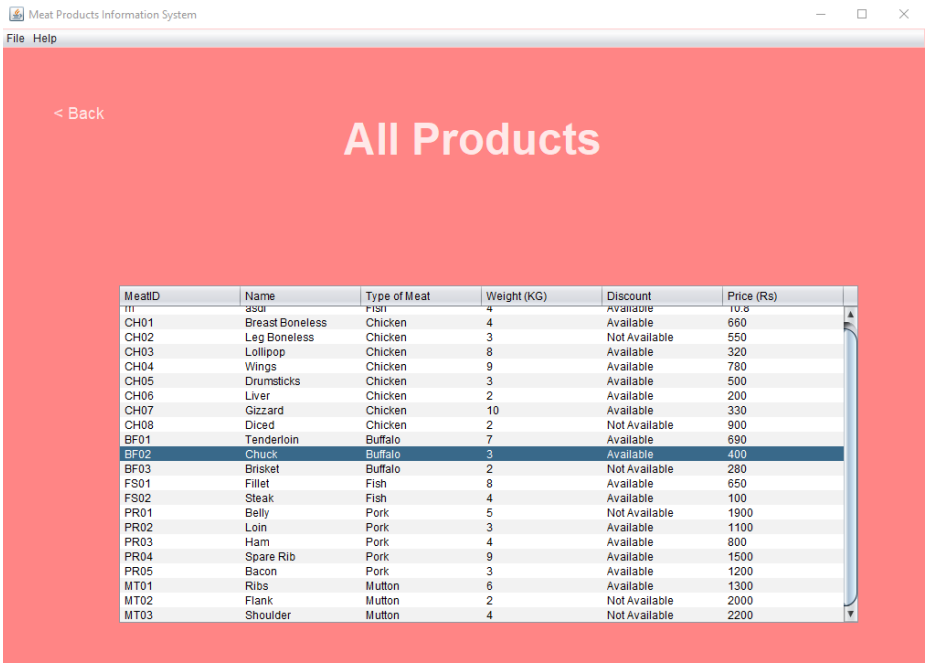

Figure 33: File importing successful

7.8 Testing to show whether the program can reduce the price by 10% if discount is available.

Test Number	8
Testing Purpose	Testing to show whether the program can reduce the price by 10% if discount is available.
Action Performed	While adding a product, its price was set to 1000 and the discount available radio button was clicked.
Expected Result	The price of the product will be reduced by 10%.
Actual Result	The price has been reduced to 900 successfully.

Table 8: Test 8

Screenshots

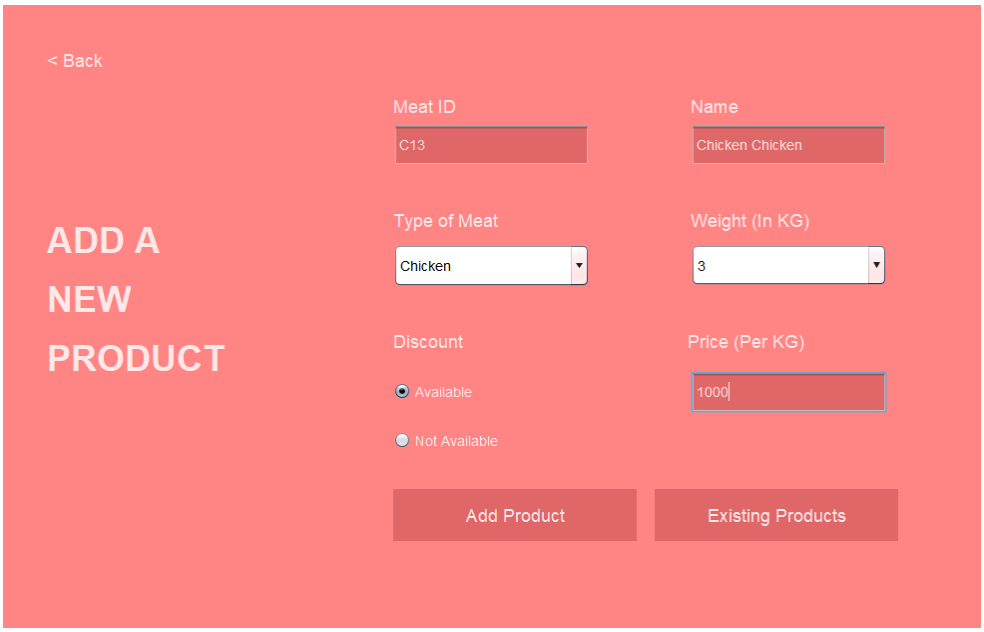

Figure 34: Testing the discount function

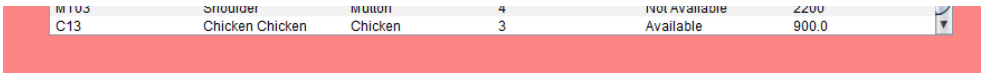

Figure 35: Discount successfully applied

8. Conclusion

We ran into countless problems ever since we started the coursework. Firstly, the group-member to whom the task of creating wireframes had been delegated, kept on procrastinating until the deadlines. His misunderstanding of the coursework instruction, afterwards led to multiple re-iterations of the wireframes until they met the requirements.

Secondly, the wireframes were a bit fancier, and hence little complicated than what Java Swing could do. This led to spending a large chunk of time on meeting the aesthetic requirements which detracted us from the main task of creating a functional and usable program. One key problem which occurred here was that of the buttons. They kept on retaining their original (default) background color, and therefore took multiple attempts to fix.

Thirdly, the initial version of our program, as we realized later, didn't have any validation method for imported data. Subsequently, we faced issues such as seeing numbers in the place of names and letters in the place of price. After a re-iteration, this bug was fixed.

Lastly, we faced multiple errors on our searching algorithm. This pushed our program completion further towards the deadline and consumed a lot of time. After watching a handful of tutorials on the internet, we found a solution and the program ran smoothly.

The Stoics had a famous saying that goes, 'the obstacle is the way.' Tantamount to this, we took every mistake as a feedback and kept on improving the program further. Every little heuristics and skills we've gathered through this coursework have made us better programmers. Whatever we have learnt here, we believe, will be crucial while solving real world problems in the future.

9. References

Academy, K., 2021. *www.khanacademy.org.* [Online]
Available at: https://www.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search
[Accessed 21 01 2021].

Geeksforgeeks, 2019. *www.geeksforgeeks.org.* [Online]
Available at: https://www.geeksforgeeks.org/selection-sort/
[Accessed 21 01 2021].

Norman, D., 2004. *Emotional Design: Why we love (or hate) everyday things.* 1st ed. New York: Basic books.

Point, T., 2021. */www.tutorialspoint.com.* [Online]
Available at:
https://www.tutorialspoint.com/data_structures_algorithms/binary_search_algorithm.htm
[Accessed 21 01 2021].

Appendix

```
/*
* 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 MeatProduct_IS;

import java.awt.Desktop;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import javax.swing.JOptionPane;
import java.util.ArrayList;
import javax.swing.JFileChooser;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author Rhythm
 */
public class MeatProductInfo extends javax.swing.JFrame {

    ArrayList<ArrayList<String>> meatList;
    ArrayList<String> mID = new ArrayList<>();
    ArrayList<String> mName = new ArrayList<>();
```

```
ArrayList<String> mType = new ArrayList<>();
ArrayList<String> mWeight = new ArrayList<>();
ArrayList<String> mDiscount = new ArrayList<>();
ArrayList<String> mPrice = new ArrayList<>();

ArrayList<String> listID;
ArrayList<String> listName;

DefaultTableModel modProductTable;
DefaultTableModel modAllTable;
/**
 * Creates new form Main
 */



/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    buttonGroup1 = new javax.swing.ButtonGroup();
    jButton2 = new javax.swing.JButton();
    mainPanel = new javax.swing.JPanel();
    mainLable1 = new javax.swing.JLabel();
```

```
mainLable2 = new javax.swing.JLabel();
mainLable3 = new javax.swing.JLabel();
mainLable4 = new javax.swing.JLabel();
mainSearchBtn = new javax.swing.JButton();
mainAddBtn = new javax.swing.JButton();
addPanel = new javax.swing.JPanel();
addBack = new javax.swing.JLabel();
IDlable = new javax.swing.JLabel();
typeLable = new javax.swing.JLabel();
nameLable = new javax.swing.JLabel();
weightLable = new javax.swing.JLabel();
discountLable = new javax.swing.JLabel();
priceLable = new javax.swing.JLabel();
addLable1 = new javax.swing.JLabel();
addLable2 = new javax.swing.JLabel();
addLable3 = new javax.swing.JLabel();
IDTF = new javax.swing.JTextField();
typeBox = new javax.swing.JComboBox<>();
radioAvailable = new javax.swing.JRadioButton();
radioNotAvailable = new javax.swing.JRadioButton();
addButton = new javax.swing.JButton();
nameTF = new javax.swing.JTextField();
weightBox = new javax.swing.JComboBox<>();
priceTF = new javax.swing.JTextField();
existingButton = new javax.swing.JButton();
searchPanel = new javax.swing.JPanel();
searchBack = new javax.swing.JLabel();
jLabel1 = new javax.swing.JLabel();
```

```
jLabel2 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
typeSearchBox = new javax.swing.JComboBox<>();
priceSearchTF = new javax.swing.JTextField();
showAllBtn = new javax.swing.JButton();
priceSearchBtn = new javax.swing.JButton();
typeSearchBtn = new javax.swing.JButton();
existingPanel = new javax.swing.JPanel();
existingLable = new javax.swing.JLabel();
jScrollPane1 = new javax.swing.JScrollPane();
productTable = new javax.swing.JTable();
existingBack = new javax.swing.JLabel();
allProductsPanel = new javax.swing.JPanel();
allLabel = new javax.swing.JLabel();
jScrollPane2 = new javax.swing.JScrollPane();
allTable = new javax.swing.JTable();
allBack = new javax.swing.JLabel();
jMenuBar1 = new javax.swing.JMenuBar();
jMenuFile = new javax.swing.JMenu();
jMenuImport = new javax.swing.JMenuItem();
jMenuClear = new javax.swing.JMenuItem();
jMenuExit = new javax.swing.JMenuItem();
jMenuHelp = new javax.swing.JMenu();

jButton2.setText("jButton2");

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
mainPanel.setBackground(new java.awt.Color(255, 133, 133));

mainLable1.setFont(new java.awt.Font("Lato Heavy", 1, 48)); // NOI18N
mainLable1.setForeground(new java.awt.Color(255, 232, 232));
mainLable1.setText("Meat");

mainLable2.setFont(new java.awt.Font("Lato Heavy", 1, 48)); // NOI18N
mainLable2.setForeground(new java.awt.Color(255, 232, 232));
mainLable2.setText("Products");

mainLable3.setFont(new java.awt.Font("Lato Heavy", 1, 48)); // NOI18N
mainLable3.setForeground(new java.awt.Color(255, 232, 232));
mainLable3.setText("Information");

mainLable4.setFont(new java.awt.Font("Lato Heavy", 1, 48)); // NOI18N
mainLable4.setForeground(new java.awt.Color(255, 232, 232));
mainLable4.setText("System");

mainSearchBtn.setBackground(new java.awt.Color(223, 103, 103));
mainSearchBtn.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
mainSearchBtn.setForeground(new java.awt.Color(255, 232, 232));
mainSearchBtn.setText("Search a Product");
mainSearchBtn.setToolTipText("");
mainSearchBtn.setActionCommand("");
mainSearchBtn.setAutoscrolls(true);
mainSearchBtn.setBorder(null);
mainSearchBtn.setContentAreaFilled(false);
mainSearchBtn.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
```

```
mainSearchBtn.setDefaultCapable(false);
mainSearchBtn.setFocusPainted(false);
mainSearchBtn.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
mainSearchBtn.setOpaque(true);
mainSearchBtn.setSelected(true);
mainSearchBtn.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        mainSearchBtnMouseEntered(evt);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        mainSearchBtnMouseExited(evt);
    }
});
mainSearchBtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mainSearchBtnActionPerformed(evt);
    }
});

mainAddBtn.setBackground(new java.awt.Color(223, 103, 103));
mainAddBtn.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
mainAddBtn.setForeground(new java.awt.Color(255, 232, 232));
mainAddBtn.setText("Add a Product");
mainAddBtn.setActionCommand("");
mainAddBtn.setBorder(null);
mainAddBtn.setContentAreaFilled(false);
mainAddBtn.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
mainAddBtn.setDefaultCapable(false);
```

```
        mainAddBtn.setFocusPainted(false);
        mainAddBtn.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        mainAddBtn.setOpaque(true);
        mainAddBtn.setSelected(true);
        mainAddBtn.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent evt) {
                mainAddBtnMouseEntered(evt);
            }
            public void mouseExited(java.awt.event.MouseEvent evt) {
                mainAddBtnMouseExited(evt);
            }
        });
        mainAddBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mainAddBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout mainPanelLayout = new javax.swing.GroupLayout(mainPanel);
        mainPanel.setLayout(mainPanelLayout);
        mainPanelLayout.setHorizontalGroup(
            mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(mainPanelLayout.createSequentialGroup()
                .addGap(102, 102, 102)

.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                    .addComponent(mainLable2)
                    .addComponent(mainLable1)
```

```
                .addComponent(mainLable4)
                .addComponent(mainLable3))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 281,
Short.MAX_VALUE)

.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG, false)
                .addComponent(mainAddBtn, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(mainSearchBtn, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
Short.MAX_VALUE))
            .addGap(156, 156, 156))
        );
        mainPanelLayout.setVerticalGroup(
            mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(mainPanelLayout.createSequentialGroup()
                .addGap(154, 154, 154)
                .addComponent(mainLable1)
                .addGap(18, 18, 18)

.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                .addComponent(mainLable2)
                .addComponent(mainAddBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 59,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                .addComponent(mainLable3)
                .addComponent(mainSearchBtn, javax.swing.GroupLayout.PREFERRED_SIZE,
58, javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
        .addGap(18, 18, 18)
        .addComponent(mainLable4)
        .addContainerGap(200, Short.MAX_VALUE))
);

addPanel.setBackground(new java.awt.Color(255, 133, 133));

addBack.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
addBack.setForeground(new java.awt.Color(255, 232, 232));
addBack.setText("< Back");
addBack.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
addBack.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        addBackMouseClicked(evt);
    }
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        addBackMouseEntered(evt);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        addBackMouseExited(evt);
    }
});

IDlable.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
IDlable.setForeground(new java.awt.Color(255, 232, 232));
IDlable.setText("Meat ID");

typeLable.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
```

```
typeLable.setForeground(new java.awt.Color(255, 232, 232));
typeLable.setText("Type of Meat");

nameLable.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
nameLable.setForeground(new java.awt.Color(255, 232, 232));
nameLable.setText("Name");

weightLable.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
weightLable.setForeground(new java.awt.Color(255, 232, 232));
weightLable.setText("Weight (In KG)");

discountLable.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
discountLable.setForeground(new java.awt.Color(255, 232, 232));
discountLable.setText("Discount");

priceLable.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
priceLable.setForeground(new java.awt.Color(255, 232, 232));
priceLable.setText("Price (Per KG)");

addLable1.setFont(new java.awt.Font("Lato Heavy", 1, 36)); // NOI18N
addLable1.setForeground(new java.awt.Color(255, 232, 232));
addLable1.setText("ADD A");

addLable2.setFont(new java.awt.Font("Lato Heavy", 1, 36)); // NOI18N
addLable2.setForeground(new java.awt.Color(255, 232, 232));
addLable2.setText("NEW");

addLable3.setFont(new java.awt.Font("Lato Heavy", 1, 36)); // NOI18N
```

```
        addLable3.setForeground(new java.awt.Color(255, 232, 232));
        addLable3.setText("PRODUCT");

        IDTF.setBackground(new java.awt.Color(223, 103, 103));
        IDTF.setFont(new java.awt.Font("Lato Heavy", 0, 14)); // NOI18N
        IDTF.setForeground(new java.awt.Color(255, 232, 232));
        IDTF.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                IDTFActionPerformed(evt);
            }
        });

        typeBox.setBackground(new java.awt.Color(223, 103, 103));
        typeBox.setFont(new java.awt.Font("Lato Heavy", 0, 14)); // NOI18N
        typeBox.setForeground(new java.awt.Color(255, 232, 232));
        typeBox.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "---Select a
Type---", "Pork", "Fish", "Mutton", "Chicken", "Buffalo" }));

        buttonGroup1.add(radioAvailable);
        radioAvailable.setFont(new java.awt.Font("Lato Heavy", 0, 14)); // NOI18N
        radioAvailable.setForeground(new java.awt.Color(255, 232, 232));
        radioAvailable.setText("Available");
        radioAvailable.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                radioAvailableActionPerformed(evt);
            }
        });

        buttonGroup1.add(radioNotAvailable);
```

```
radioNotAvailable.setFont(new java.awt.Font("Lato Heavy", 0, 14)); // NOI18N
radioNotAvailable.setForeground(new java.awt.Color(255, 232, 232));
radioNotAvailable.setText("Not Available");

addButton.setBackground(new java.awt.Color(223, 103, 103));
addButton.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
addButton.setForeground(new java.awt.Color(255, 232, 232));
addButton.setText("Add Product");
addButton.setBorder(null);
addButton.setContentAreaFilled(false);
addButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
addButton.setFocusPainted(false);
addButton.setOpaque(true);
addButton.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        addButtonMouseEntered(evt);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        addButtonMouseExited(evt);
    }
});
addButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        addButtonActionPerformed(evt);
    }
});

nameTF.setBackground(new java.awt.Color(223, 103, 103));
```

```
        nameTF.setFont(new java.awt.Font("Lato Heavy", 0, 14)); // NOI18N
        nameTF.setForeground(new java.awt.Color(255, 232, 232));

        weightBox.setBackground(new java.awt.Color(223, 103, 103));
        weightBox.setFont(new java.awt.Font("Lato Heavy", 0, 14)); // NOI18N
        weightBox.setForeground(new java.awt.Color(255, 232, 232));
        weightBox.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "--Select
Weight--", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10" }));

        priceTF.setBackground(new java.awt.Color(223, 103, 103));
        priceTF.setFont(new java.awt.Font("Lato Heavy", 0, 14)); // NOI18N
        priceTF.setForeground(new java.awt.Color(255, 232, 232));

        existingButton.setBackground(new java.awt.Color(223, 103, 103));
        existingButton.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
        existingButton.setForeground(new java.awt.Color(255, 232, 232));
        existingButton.setText("Existing Products");
        existingButton.setBorder(null);
        existingButton.setContentAreaFilled(false);
        existingButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        existingButton.setFocusPainted(false);
        existingButton.setOpaque(true);
        existingButton.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent evt) {
                existingButtonMouseEntered(evt);
            }
            public void mouseExited(java.awt.event.MouseEvent evt) {
                existingButtonMouseExited(evt);
            }
```

```
        });
        existingButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                existingButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout addPanelLayout = new javax.swing.GroupLayout(addPanel);
        addPanel.setLayout(addPanelLayout);
        addPanelLayout.setHorizontalGroup(
            addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(addPanelLayout.createSequentialGroup()
                .addGap(56, 56, 56)

.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                    .addComponent(addBack, javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(addLable1)
                    .addComponent(addLable2)
                    .addComponent(addLable3))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 165,
Short.MAX_VALUE)

.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                    .addGroup(addPanelLayout.createSequentialGroup()
                        .addComponent(addButton, javax.swing.GroupLayout.PREFERRED_SIZE, 244,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
```

```
                .addComponent(existingButton, javax.swing.GroupLayout.PREFERRED_SIZE,
244, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGap(0, 97, Short.MAX_VALUE))

            .addGroup(addPanelLayout.createSequentialGroup()


.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)

                .addGroup(addPanelLayout.createSequentialGroup()


.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G, false)

                    .addComponent(typeLable)

                    .addComponent(IDlable)

                    .addComponent(radioAvailable)

                    .addComponent(radioNotAvailable)

                    .addComponent(typeBox, 0, 197, Short.MAX_VALUE)

                    .addComponent(IDTF, javax.swing.GroupLayout.DEFAULT_SIZE, 197,
Short.MAX_VALUE))

                .addGap(101, 101, 101)


.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G, false)

                    .addComponent(nameLable)

                    .addComponent(weightLable)

                    .addComponent(weightBox, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

                    .addComponent(priceTF)

                    .addComponent(nameTF, javax.swing.GroupLayout.DEFAULT_SIZE,
197, Short.MAX_VALUE)))

                .addGroup(addPanelLayout.createSequentialGroup()

                .addComponent(discountLable)

                .addGap(225, 225, 225)
```

```
                    .addComponent(priceLable)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))
    );
    addPanelLayout.setVerticalGroup(
      addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(addPanelLayout.createSequentialGroup()
        .addGap(56, 56, 56)
        .addComponent(addBack, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)

.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
          .addComponent(nameLable)
          .addComponent(IDlable))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
          .addComponent(IDTF, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE)
          .addComponent(nameTF, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(42, 42, 42)

.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
          .addGroup(addPanelLayout.createSequentialGroup()

.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
```

```
                .addComponent(typeLable)
                .addComponent(weightLable))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(typeBox, javax.swing.GroupLayout.PREFERRED_SIZE, 43, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(weightBox, javax.swing.GroupLayout.PREFERRED_SIZE, 42, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(42, 42, 42)

.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(discountLable)
                .addComponent(priceLable))
            .addGap(18, 18, 18)

.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(radioAvailable)
                .addComponent(priceTF, javax.swing.GroupLayout.PREFERRED_SIZE, 42, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addComponent(radioNotAvailable)
            .addGap(39, 39, 39)

.addGroup(addPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(addButton, javax.swing.GroupLayout.PREFERRED_SIZE, 52, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                    .addComponent(existingButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 52,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGroup(addPanelLayout.createSequentialGroup()
                    .addGap(8, 8, 8)
                    .addComponent(addLable1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addComponent(addLable2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addComponent(addLable3)))
            .addContainerGap(116, Short.MAX_VALUE))
        );

        searchPanel.setBackground(new java.awt.Color(255, 133, 133));

        searchBack.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
        searchBack.setForeground(new java.awt.Color(255, 232, 232));
        searchBack.setText("< Back");
        searchBack.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        searchBack.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                searchBackMouseClicked(evt);
            }
            public void mouseEntered(java.awt.event.MouseEvent evt) {
                searchBackMouseEntered(evt);
            }
            public void mouseExited(java.awt.event.MouseEvent evt) {
                searchBackMouseExited(evt);
```

```
        }
    });

    jLabel1.setFont(new java.awt.Font("Lato Heavy", 1, 36)); // NOI18N
    jLabel1.setForeground(new java.awt.Color(255, 232, 232));
    jLabel1.setText("SEARCH FOR A PRODUCT");

    jLabel2.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
    jLabel2.setForeground(new java.awt.Color(255, 232, 232));
    jLabel2.setText("Enter the price (Rs)");

    jLabel3.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
    jLabel3.setForeground(new java.awt.Color(255, 232, 232));
    jLabel3.setText("Select the Type of Meat");

    typeSearchBox.setBackground(new java.awt.Color(223, 103, 103));
    typeSearchBox.setFont(new java.awt.Font("Lato Heavy", 0, 14)); // NOI18N
    typeSearchBox.setForeground(new java.awt.Color(255, 232, 232));
    typeSearchBox.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "---
Select a Type---", "Pork", "Fish", "Mutton", "Chicken", "Buffalo" }));

    priceSearchTF.setBackground(new java.awt.Color(223, 103, 103));
    priceSearchTF.setFont(new java.awt.Font("Lato Heavy", 0, 14)); // NOI18N
    priceSearchTF.setForeground(new java.awt.Color(255, 232, 232));
    priceSearchTF.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            priceSearchTFActionPerformed(evt);
        }
    });
```

```
showAllBtn.setBackground(new java.awt.Color(223, 103, 103));
showAllBtn.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
showAllBtn.setForeground(new java.awt.Color(255, 232, 232));
showAllBtn.setText("Show all products");
showAllBtn.setContentAreaFilled(false);
showAllBtn.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
showAllBtn.setOpaque(true);
showAllBtn.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        showAllBtnMouseEntered(evt);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        showAllBtnMouseExited(evt);
    }
});
showAllBtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        showAllBtnActionPerformed(evt);
    }
});

priceSearchBtn.setBackground(new java.awt.Color(223, 103, 103));
priceSearchBtn.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
priceSearchBtn.setForeground(new java.awt.Color(255, 232, 232));
priceSearchBtn.setText("Search By Price");
priceSearchBtn.setContentAreaFilled(false);
priceSearchBtn.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
```

```
priceSearchBtn.setOpaque(true);
priceSearchBtn.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        priceSearchBtnMouseEntered(evt);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        priceSearchBtnMouseExited(evt);
    }
});
priceSearchBtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        priceSearchBtnActionPerformed(evt);
    }
});

typeSearchBtn.setBackground(new java.awt.Color(223, 103, 103));
typeSearchBtn.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
typeSearchBtn.setForeground(new java.awt.Color(255, 232, 232));
typeSearchBtn.setText("Search by Type");
typeSearchBtn.setContentAreaFilled(false);
typeSearchBtn.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
typeSearchBtn.setOpaque(true);
typeSearchBtn.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseEntered(java.awt.event.MouseEvent evt) {
        typeSearchBtnMouseEntered(evt);
    }
    public void mouseExited(java.awt.event.MouseEvent evt) {
        typeSearchBtnMouseExited(evt);
```

```
        }
    });
    typeSearchBtn.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            typeSearchBtnActionPerformed(evt);
        }
    });


    javax.swing.GroupLayout searchPanelLayout = new
javax.swing.GroupLayout(searchPanel);
    searchPanel.setLayout(searchPanelLayout);
    searchPanelLayout.setHorizontalGroup(

searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(searchPanelLayout.createSequentialGroup()

.addGroup(searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                .addGroup(searchPanelLayout.createSequentialGroup()
                    .addGap(168, 168, 168)
                    .addComponent(jLabel2))
                .addGroup(searchPanelLayout.createSequentialGroup()
                    .addGap(150, 150, 150)
                    .addComponent(priceSearchTF, javax.swing.GroupLayout.PREFERRED_SIZE,
194, javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
```

```
                .addComponent(typeSearchBox, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 195,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jLabel3, javax.swing.GroupLayout.Alignment.TRAILING))

            .addGap(150, 150, 150))

        .addGroup(searchPanelLayout.createSequentialGroup()

            .addGap(56, 56, 56)

            .addComponent(searchBack, javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(131, 131, 131)

            .addComponent(jLabel1)

            .addContainerGap(298, Short.MAX_VALUE))

        .addGroup(searchPanelLayout.createSequentialGroup()

            .addGap(138, 138, 138)

            .addComponent(priceSearchBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 219,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

            .addComponent(typeSearchBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 219,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(134, 134, 134))

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
searchPanelLayout.createSequentialGroup()

            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

            .addComponent(showAllBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 219,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(381, 381, 381))

    );

    searchPanelLayout.setVerticalGroup(

searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
        .addGroup(searchPanelLayout.createSequentialGroup()
            .addGap(56, 56, 56)
.addGroup(searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 67,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(searchBack, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(75, 75, 75)
.addGroup(searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASE
LINE)
                .addComponent(jLabel2)
                .addComponent(jLabel3))
            .addGap(48, 48, 48)
.addGroup(searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASE
LINE)
                .addComponent(typeSearchBox, javax.swing.GroupLayout.PREFERRED_SIZE,
44, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(priceSearchTF, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(67, 67, 67)
.addGroup(searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASE
LINE)
                .addComponent(priceSearchBtn, javax.swing.GroupLayout.PREFERRED_SIZE,
55, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(typeSearchBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 55,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(53, 53, 53)
```

```
            .addComponent(showAllBtn, javax.swing.GroupLayout.PREFERRED_SIZE, 55,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(83, 83, 83))
        );

        existingPanel.setBackground(new java.awt.Color(255, 133, 133));
        existingPanel.setToolTipText("");

        existingLable.setFont(new java.awt.Font("Lato Heavy", 1, 48)); // NOI18N
        existingLable.setForeground(new java.awt.Color(255, 232, 232));
        existingLable.setText("Existing Products");

        productTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "MeatID", "Name", "Type of Meat", "Weight (KG)", "Discount", "Price (Rs)"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        jScrollPane1.setViewportView(productTable);
```

```
        if (productTable.getColumnModel().getColumnCount() > 0) {
            productTable.getColumnModel().getColumn(0).setResizable(false);
            productTable.getColumnModel().getColumn(1).setResizable(false);
            productTable.getColumnModel().getColumn(2).setResizable(false);
            productTable.getColumnModel().getColumn(3).setResizable(false);
            productTable.getColumnModel().getColumn(4).setResizable(false);
            productTable.getColumnModel().getColumn(5).setResizable(false);
        }

        existingBack.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
        existingBack.setForeground(new java.awt.Color(255, 232, 232));
        existingBack.setText("< Back");
        existingBack.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        existingBack.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                existingBackMouseClicked(evt);
            }
            public void mouseEntered(java.awt.event.MouseEvent evt) {
                existingBackMouseEntered(evt);
            }
            public void mouseExited(java.awt.event.MouseEvent evt) {
                existingBackMouseExited(evt);
            }
        });

        javax.swing.GroupLayout existingPanelLayout = new
javax.swing.GroupLayout(existingPanel);
        existingPanel.setLayout(existingPanelLayout);
        existingPanelLayout.setHorizontalGroup(
```

```
existingPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(existingPanelLayout.createSequentialGroup()
            .addGap(56, 56, 56)
            .addComponent(existingBack, javax.swing.GroupLayout.PREFERRED_SIZE, 63, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(existingLable)
            .addGap(281, 281, 281))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, existingPanelLayout.createSequentialGroup()
            .addContainerGap(123, Short.MAX_VALUE)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 813, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(71, 71, 71))
    );
    existingPanelLayout.setVerticalGroup(

existingPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(existingPanelLayout.createSequentialGroup()
            .addGap(56, 56, 56)

.addGroup(existingPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(existingLable, javax.swing.GroupLayout.PREFERRED_SIZE, 87, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(existingBack, javax.swing.GroupLayout.PREFERRED_SIZE, 31, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 115, Short.MAX_VALUE)
```

```
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 373,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(44, 44, 44))
        );

        allProductsPanel.setBackground(new java.awt.Color(255, 133, 133));
        allProductsPanel.setToolTipText("");

        allLabel.setFont(new java.awt.Font("Lato Heavy", 1, 48)); // NOI18N
        allLabel.setForeground(new java.awt.Color(255, 232, 232));
        allLabel.setText("All Products");

        allTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

            },
            new String [] {
                "MeatID", "Name", "Type of Meat", "Weight (KG)", "Discount", "Price (Rs)"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                false, false, false, false, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return canEdit [columnIndex];
            }
        });
        jScrollPane2.setViewportView(allTable);
```

```
        if (allTable.getColumnModel().getColumnCount() > 0) {
            allTable.getColumnModel().getColumn(0).setResizable(false);
            allTable.getColumnModel().getColumn(1).setResizable(false);
            allTable.getColumnModel().getColumn(2).setResizable(false);
            allTable.getColumnModel().getColumn(3).setResizable(false);
            allTable.getColumnModel().getColumn(4).setResizable(false);
            allTable.getColumnModel().getColumn(5).setResizable(false);
        }

        allBack.setFont(new java.awt.Font("Lato Heavy", 0, 18)); // NOI18N
        allBack.setForeground(new java.awt.Color(255, 232, 232));
        allBack.setText("< Back");
        allBack.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        allBack.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                allBackMouseClicked(evt);
            }
            public void mouseEntered(java.awt.event.MouseEvent evt) {
                allBackMouseEntered(evt);
            }
            public void mouseExited(java.awt.event.MouseEvent evt) {
                allBackMouseExited(evt);
            }
        });

        javax.swing.GroupLayout allProductsPanelLayout = new
javax.swing.GroupLayout(allProductsPanel);
        allProductsPanel.setLayout(allProductsPanelLayout);
        allProductsPanelLayout.setHorizontalGroup(
```

```
allProductsPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(allProductsPanelLayout.createSequentialGroup()
            .addGap(56, 56, 56)
            .addComponent(allBack, javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(255, 255, 255)
            .addComponent(allLabel)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
allProductsPanelLayout.createSequentialGroup()
            .addContainerGap(123, Short.MAX_VALUE)
            .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 813,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(71, 71, 71))
    );
    allProductsPanelLayout.setVerticalGroup(

allProductsPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(allProductsPanelLayout.createSequentialGroup()
            .addGap(56, 56, 56)

.addGroup(allProductsPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                .addComponent(allLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 87,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(allBack, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 115,
Short.MAX_VALUE)
```

```
            .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 373,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(44, 44, 44))
        );

        jMenuBar1.setBorder(javax.swing.BorderFactory.createMatteBorder(1, 1, 1, 1, new
java.awt.Color(255, 204, 204)));
        jMenuBar1.setOpaque(false);

        jMenuFile.setText("File");

        jMenuImport.setText("Open");
        jMenuImport.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuImportActionPerformed(evt);
            }
        });
        jMenuFile.add(jMenuImport);

        jMenuClear.setText("Clear");
        jMenuClear.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuClearActionPerformed(evt);
            }
        });
        jMenuFile.add(jMenuClear);

        jMenuExit.setText("Exit");
        jMenuExit.addActionListener(new java.awt.event.ActionListener() {
```

```
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jMenuExitActionPerformed(evt);
        }
    });
    jMenuFile.add(jMenuExit);

    jMenuBar1.add(jMenuFile);

    jMenuHelp.setText("Help");
    jMenuHelp.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jMenuHelpMouseClicked(evt);
        }
    });
    jMenuHelp.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jMenuHelpActionPerformed(evt);
        }
    });
    jMenuBar1.add(jMenuHelp);

    setJMenuBar(jMenuBar1);

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(mainPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(addPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(searchPanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(existingPanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(allProductsPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(mainPanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(addPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(searchPanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(existingPanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addComponent(allProductsPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>
    public MeatProductInfo() {
        this.listName = new ArrayList<>(2);
        this.listID = new ArrayList<>(2);
        this.meatList = new ArrayList< >();
        initComponents();
        this.setTitle("Meat Products Information System");
        setLocationRelativeTo(null);
        mainPanel.setVisible(true);
        searchPanel.setVisible(false);
        addPanel.setVisible(false);
        existingPanel.setVisible(false);
        allProductsPanel.setVisible(false);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    private void mainAddBtnActionPerformed(java.awt.event.ActionEvent evt) {
        mainPanel.setVisible(false);
        searchPanel.setVisible(false);
        addPanel.setVisible(true);
        existingPanel.setVisible(false);
        allProductsPanel.setVisible(false);
    }
```

```
    private void mainSearchBtnActionPerformed(java.awt.event.ActionEvent evt) {
        mainPanel.setVisible(false);
        searchPanel.setVisible(true);
        addPanel.setVisible(false);
        existingPanel.setVisible(false);
        allProductsPanel.setVisible(false);
    }

    private void radioAvailableActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void IDTFActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void addBackMouseClicked(java.awt.event.MouseEvent evt) {
        mainPanel.setVisible(true);
        searchPanel.setVisible(false);
        addPanel.setVisible(false);
        existingPanel.setVisible(false);
        allProductsPanel.setVisible(false);
    }

    private void searchBackMouseClicked(java.awt.event.MouseEvent evt) {
        mainPanel.setVisible(true);
        searchPanel.setVisible(false);
        addPanel.setVisible(false);
```

```
    existingPanel.setVisible(false);
    allProductsPanel.setVisible(false);
}

private void addBackMouseEntered(java.awt.event.MouseEvent evt) {
    addBack.setForeground(new java.awt.Color(223, 103, 103));
}

private void addBackMouseExited(java.awt.event.MouseEvent evt) {
    addBack.setForeground(new java.awt.Color(255, 232, 232));
}

private void searchBackMouseEntered(java.awt.event.MouseEvent evt) {
    searchBack.setForeground(new java.awt.Color(223, 103, 103));
}

private void searchBackMouseExited(java.awt.event.MouseEvent evt) {
    searchBack.setForeground(new java.awt.Color(255, 232, 232));
}

private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {

    String meatID = IDTF.getText();
    String name = nameTF.getText();
    String price =  priceTF.getText();
    String discount = "";

    if (radioAvailable.isSelected()){
```

```
        discount = "Available";
    }
    if (radioNotAvailable.isSelected()){
        discount = "Not Available";
    }

    String type = (String) typeBox.getSelectedItem();
    String weight = (String) weightBox.getSelectedItem();



    if (!(meatID.isEmpty())){
        if(!mID.contains(meatID.toLowerCase())){
            if(!(name.isEmpty())){
                if(name.matches("[a-zA-Z ]+")){
                    if(!mName.contains(name.toLowerCase())){
                        if(!(typeBox.getSelectedIndex()==0)){
                            if(!(weightBox.getSelectedIndex()==0)){
                                if (!discount.isEmpty()){
                                    try{
                                        if(!(price).isEmpty()){
                                            if(Double.parseDouble(price) > 0){


                                                modProductTable =
(DefaultTableModel)productTable.getModel();
                                                modAllTable = (DefaultTableModel)allTable.getModel();
                                                modProductTable.addRow(new
Object[]{null,null,null,null,null,null});
                                                modAllTable.addRow(new
Object[]{null,null,null,null,null,null});
```

```
String details[] = {meatID, name, type, weight, discount, price};

int nextRow =0;
int rowCounter = productTable.getRowCount();
int columnCounter = productTable.getColumnCount();
boolean empty = false;
do {
   String rowChecker = (String)
productTable.getValueAt(nextRow, 0);
   if (rowChecker !=null && rowChecker.length() != 0) {

      nextRow ++;

   }
   else{
      empty = true;
   }

} while (nextRow<rowCounter && empty == false);

for (int i = 0; i < columnCounter; i++) {
   productTable.setValueAt(details[i], nextRow, i);
   allTable.setValueAt(details[i], nextRow, i);
}

mID.add(meatID.toLowerCase());
mName.add(name.toLowerCase());
```

```
                    clear();
                    JOptionPane.showMessageDialog(addPanel, "Product
Succesfully Added");


                }
                else{
                    JOptionPane.showMessageDialog(addPanel, "Invalid Price!
Price must be positive integer");
                }
            }
            else{
                JOptionPane.showMessageDialog(addPanel, "Invalid Price! Price
field cannot be empty");
            }}catch(NumberFormatException e){
                JOptionPane.showMessageDialog(addPanel, "Invalid Price! Price
must be numeric");
            }
        }
        else{
            JOptionPane.showMessageDialog(addPanel, "Select if Discount is
Available or not");
        }


    }
    else{
        JOptionPane.showMessageDialog(addPanel, "Please select weight!");
    }
}
else{
    JOptionPane.showMessageDialog(addPanel, "Please select Type of Meat!");
```

```
                }
            }
            else{
                JOptionPane.showMessageDialog(addPanel, "Name has been repeated");
            }
        }

        else{
            JOptionPane.showMessageDialog(addPanel, "Invalid Name! Name cannot be
numeric!");
        }

    }
    else{
        JOptionPane.showMessageDialog(addPanel, "Invalid Name! Please Input a valid
Name!");
        }
    }
    else{
        JOptionPane.showMessageDialog(addPanel, "Invalid ID! Meat ID has been
repeated");
    }
}
else{
    JOptionPane.showMessageDialog(addPanel, "Invalid MeatID! Please Input a valid Meat
ID!");
}


}
```

```
private void existingBackMouseClicked(java.awt.event.MouseEvent evt) {
    mainPanel.setVisible(false);
    searchPanel.setVisible(false);
    addPanel.setVisible(true);
    existingPanel.setVisible(false);
    allProductsPanel.setVisible(false);
}

private void existingBackMouseEntered(java.awt.event.MouseEvent evt) {
    existingBack.setForeground(new java.awt.Color(223, 103, 103));
}

private void existingBackMouseExited(java.awt.event.MouseEvent evt) {
    existingBack.setForeground(new java.awt.Color(255, 232, 232));
}

private void existingButtonActionPerformed(java.awt.event.ActionEvent evt) {
    mainPanel.setVisible(false);
    searchPanel.setVisible(false);
    addPanel.setVisible(false);
    existingPanel.setVisible(true);
    allProductsPanel.setVisible(false);
}

private void mainAddBtnMouseEntered(java.awt.event.MouseEvent evt) {
    mainAddBtn.setForeground(new java.awt.Color(223, 103, 103));
    mainAddBtn.setBackground(new java.awt.Color(255, 232, 232));
```

```
    }

    private void mainAddBtnMouseExited(java.awt.event.MouseEvent evt) {
        mainAddBtn.setForeground(new java.awt.Color(255, 232, 232));
        mainAddBtn.setBackground(new java.awt.Color(223, 103, 103));
    }

    private void mainSearchBtnMouseEntered(java.awt.event.MouseEvent evt) {
        mainSearchBtn.setForeground(new java.awt.Color(223, 103, 103));
        mainSearchBtn.setBackground(new java.awt.Color(255, 232, 232));
    }

    private void mainSearchBtnMouseExited(java.awt.event.MouseEvent evt) {
        mainSearchBtn.setForeground(new java.awt.Color(255, 232, 232));
        mainSearchBtn.setBackground(new java.awt.Color(223, 103, 103));
    }

    private void addButtonMouseEntered(java.awt.event.MouseEvent evt) {
        addButton.setForeground(new java.awt.Color(223, 103, 103));
        addButton.setBackground(new java.awt.Color(255, 232, 232));
    }

    private void addButtonMouseExited(java.awt.event.MouseEvent evt) {
        addButton.setForeground(new java.awt.Color(255, 232, 232));
        addButton.setBackground(new java.awt.Color(223, 103, 103));
    }

    private void existingButtonMouseEntered(java.awt.event.MouseEvent evt) {
```

```
        existingButton.setForeground(new java.awt.Color(223, 103, 103));
        existingButton.setBackground(new java.awt.Color(255, 232, 232));
    }

    private void existingButtonMouseExited(java.awt.event.MouseEvent evt) {
        existingButton.setForeground(new java.awt.Color(255, 232, 232));
        existingButton.setBackground(new java.awt.Color(223, 103, 103));
    }

    private void priceSearchBtnActionPerformed(java.awt.event.ActionEvent evt) {
        clearData();
        addData();

        String sPrice =  priceSearchTF.getText();

        try{
            if(!(sPrice).isEmpty()){
                if(Double.parseDouble(sPrice) > 0){
                    double find = Double.parseDouble(sPrice);
                    boolean present = mPrice.contains(String.valueOf(find));
                    if (present) {
                        doSelectionSort(meatList);
                        int index = binarySearch(meatList.get(5), 0, meatList.get(5).size()-1, find);
                        JOptionPane.showMessageDialog(searchPanel, "For the price of " +
meatList.get(5).get(index) +  ", Product name is " +meatList.get(1).get(index)
                                + " with MeatID: " + meatList.get(0).get(index) + "\n" +
meatList.get(3).get(index) + " Kg "
                                + meatList.get(2).get(index) + " meat "+ "and discount is " +
meatList.get(4).get(index) );
```

```
                }else{
                    JOptionPane.showMessageDialog(addPanel, "Invalid Price! Price not found");
                }
            }
            else{
                JOptionPane.showMessageDialog(addPanel, "Invalid Price! Price cannot be
negative");
            }
        }else{
            JOptionPane.showMessageDialog(addPanel, "Invalid Price! Price cannot be empty");
        }

    }
    catch(NumberFormatException e){
        JOptionPane.showMessageDialog(addPanel, "Invalid Price! Price must be numeric");
    }



}

private void showAllBtnActionPerformed(java.awt.event.ActionEvent evt) {
    mainPanel.setVisible(false);
    searchPanel.setVisible(false);
    addPanel.setVisible(false);
    existingPanel.setVisible(false);
    allProductsPanel.setVisible(true);
}
```

```
    private void allBackMouseClicked(java.awt.event.MouseEvent evt) {
        mainPanel.setVisible(false);
        searchPanel.setVisible(true);
        addPanel.setVisible(false);
        existingPanel.setVisible(false);
        allProductsPanel.setVisible(false);
    }

    private void allBackMouseEntered(java.awt.event.MouseEvent evt) {
        allBack.setForeground(new java.awt.Color(223, 103, 103));
    }

    private void allBackMouseExited(java.awt.event.MouseEvent evt) {
        allBack.setForeground(new java.awt.Color(255, 232, 232));
    }

    private void typeSearchBtnActionPerformed(java.awt.event.ActionEvent evt) {
        clearData();
        addData();

        if(!(typeSearchBox.getSelectedIndex()==0)){
            String type = (String) typeSearchBox.getSelectedItem();

            if(type.equals("Pork")){
                if(typeSearch(type)[0].isEmpty()){
                    JOptionPane.showMessageDialog(searchPanel, "There are no Products Associated
with Pork");
                }
                else{
```

```
            JOptionPane.showMessageDialog(searchPanel, "There are " + typeSearch(type)[5]
+ " products associated with Pork:"+typeSearch(type)[0]);
        }
    }
    else if(type.equals("Mutton")){
        if(typeSearch(type)[1].isEmpty()){
            JOptionPane.showMessageDialog(searchPanel, "There are no Products Associated
with Mutton");
        }
        else{

            JOptionPane.showMessageDialog(searchPanel, "There are " + typeSearch(type)[6]
+ " products associated with Mutton:"+typeSearch(type)[1]);
        }
    }
    else if(type.equals("Chicken")){
        if(typeSearch(type)[2].isEmpty()){
            JOptionPane.showMessageDialog(searchPanel, "There are no Products Associated
with Chicken");
        }
        else{
            JOptionPane.showMessageDialog(searchPanel, "There are " + typeSearch(type)[7]
+ " products associated with Chicken:"+typeSearch(type)[2]);
        }
    }
    else if(type.equals("Buffalo")){
        if(typeSearch(type)[3].isEmpty()){
            JOptionPane.showMessageDialog(searchPanel, "There are no Products Associated
with Buffalo");
        }
```

```
            else{
                JOptionPane.showMessageDialog(searchPanel, "There are " + typeSearch(type)[8]
+ " products associated with Buffalo:"+typeSearch(type)[3]);
            }
        }
        else if(type.equals("Fish")){
            if(typeSearch(type)[4].isEmpty()){
                JOptionPane.showMessageDialog(searchPanel, "There are no Products Associated
with Fish");
            }
            else{
                JOptionPane.showMessageDialog(searchPanel, "There are " + typeSearch(type)[9]
+ " products associated with Fish:" + typeSearch(type)[4]);
            }
        }
    }
    else{
        JOptionPane.showMessageDialog(searchPanel, "Please select Type of Meat!");
    }


}

private void priceSearchBtnMouseEntered(java.awt.event.MouseEvent evt) {
    priceSearchBtn.setForeground(new java.awt.Color(223, 103, 103));
    priceSearchBtn.setBackground(new java.awt.Color(255, 232, 232));
}

private void priceSearchBtnMouseExited(java.awt.event.MouseEvent evt) {
```

```
    priceSearchBtn.setForeground(new java.awt.Color(255, 232, 232));
    priceSearchBtn.setBackground(new java.awt.Color(223, 103, 103));
}

private void typeSearchBtnMouseEntered(java.awt.event.MouseEvent evt) {
    typeSearchBtn.setForeground(new java.awt.Color(223, 103, 103));
    typeSearchBtn.setBackground(new java.awt.Color(255, 232, 232));
}

private void typeSearchBtnMouseExited(java.awt.event.MouseEvent evt) {
    typeSearchBtn.setForeground(new java.awt.Color(255, 232, 232));
    typeSearchBtn.setBackground(new java.awt.Color(223, 103, 103));
}

private void showAllBtnMouseEntered(java.awt.event.MouseEvent evt) {
    showAllBtn.setForeground(new java.awt.Color(223, 103, 103));
    showAllBtn.setBackground(new java.awt.Color(255, 232, 232));
}

private void showAllBtnMouseExited(java.awt.event.MouseEvent evt) {
    showAllBtn.setForeground(new java.awt.Color(255, 232, 232));
    showAllBtn.setBackground(new java.awt.Color(223, 103, 103));
}

private void jMenuImportActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int dialogChoice = 1;
    File importFile = new File("demo1.txt");
```

```
int userAnswer = JOptionPane.showConfirmDialog(null, "Do you want to choose file? \n "
        + "(No - imports default data) ", "User Option", JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE);

switch (userAnswer) {
   case 1: importFile = new File("ImportData.txt");
           dialogChoice = 0;
           break;
   case 0: String defaultFileDirectory = "D:\\";
           JFileChooser dialogFileChooser = new JFileChooser(defaultFileDirectory);
           dialogChoice = dialogFileChooser.showOpenDialog(null);
           importFile = dialogFileChooser.getSelectedFile();
           break;
}

if (dialogChoice == JFileChooser.APPROVE_OPTION) {
   try {
      BufferedReader br = new BufferedReader(new FileReader(importFile));
      modProductTable = (DefaultTableModel)productTable.getModel();
      modAllTable = (DefaultTableModel)allTable.getModel();
      Object[] rows = br.lines().toArray();

      for(int i =0; i < rows.length; i++){
         String[] row = rows[i].toString().split("/");
         modProductTable.addRow(row);
         modAllTable.addRow(row);
      }

      addCheckData();
```

```
            JOptionPane.showMessageDialog(mainPanel, "File data has been imported in the
Product Table.");


        } catch (FileNotFoundException ex) {
            JOptionPane.showMessageDialog(this, "System cannot find the selected file.");


        } catch (NullPointerException ex){
            modProductTable.setRowCount(0);
            modAllTable.setRowCount(0);
            clearData();
            clearCheckData();
            JOptionPane.showMessageDialog(this, "Wrong file selected. Added data cleared.");
        }


      }



  }


  private void jMenuExitActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
      int userAnswer = JOptionPane.showConfirmDialog(null, "Do you want to close the
program?", "Confirm", JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE);
      if (userAnswer == JOptionPane.YES_OPTION) {
          this.dispose();
      }
  }
```

```
    private void priceSearchTFActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jMenuClearActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        int rowC = productTable.getRowCount();

        if(rowC > 0){
            modProductTable.setRowCount(0);
            modAllTable.setRowCount(0);
            clearData();
            clearCheckData();
            JOptionPane.showMessageDialog(mainPanel, "Product table is cleared!");
        }else{
            JOptionPane.showMessageDialog(mainPanel, "Product table is empty!");
        }
    }

    private void jMenuHelpActionPerformed(java.awt.event.ActionEvent evt) {

    }

    private void jMenuHelpMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        try {
            Desktop.getDesktop().open(new java.io.File("UserManual.pdf"));
        } catch (Exception ex) {
```

```
            JOptionPane.showMessageDialog(this, "File not found");
        }
    }

    private void clear(){
        IDTF.setText("");
        nameTF.setText("");
        weightBox.setSelectedIndex(0);
        typeBox.setSelectedIndex(0);
        buttonGroup1.clearSelection();
        priceTF.setText("");
    }

    public void clearData(){
        meatList.clear();
        mID.clear();
        mName.clear();
        mType.clear();
        mWeight.clear();
        mDiscount.clear();
        mPrice.clear();
    }

    public void addCheckData(){
        int rowCounter = productTable.getRowCount();
        for(int i=0; i<rowCounter; i++){
            String meatID = (String) productTable.getModel().getValueAt(i,0);
            String name = (String) productTable.getModel().getValueAt(i,1);
```

```
            listID.add(meatID.toLowerCase());
            listName.add(name.toLowerCase());
        }
    }

    public void clearCheckData(){
        listID.clear();
        listName.clear();
    }

    public void addData(){
        int rowCounter = productTable.getRowCount();
        for(int i=0; i<rowCounter; i++){
            String meatID = (String) productTable.getModel().getValueAt(i,0);
            String name = (String) productTable.getModel().getValueAt(i,1);
            String type = (String) productTable.getModel().getValueAt(i,2);
            String weight = (String) productTable.getModel().getValueAt(i,3);
            String discount = (String) productTable.getModel().getValueAt(i,4);
            String sPrice = (String) productTable.getModel().getValueAt(i,5);
            String pp = String.valueOf(Double.parseDouble(sPrice));

            mID.add(meatID);
            mName.add(name);
            mType.add(type);
            mWeight.add(weight);
            mDiscount.add(discount);
            mPrice.add(pp);
```

```
    }
    meatList.add(mID);
    meatList.add(mName);
    meatList.add(mType);
    meatList.add(mWeight);
    meatList.add(mDiscount);
    meatList.add(mPrice);
}

private String[] typeSearch(String type){
    String porkDetails = "";
    String muttonDetails = "";
    String chickenDetails = "";
    String buffDetails = "";
    String fishDetails = "";
    String[] details = new String [10];

    int nextRow =0;
    int rowCounter = productTable.getRowCount();
    int pCount = 0; int mCount = 0; int cCount = 0; int bCount = 0; int fCount = 0;
    for (int i = 0; i < rowCounter; i++) {
        String rowChecker = (String) productTable.getValueAt(nextRow, 0);
        if (rowChecker !=null && rowChecker.length() != 0) {
            rowChecker = (String) productTable.getValueAt(nextRow, 2);
            switch (rowChecker) {
                case "Pork":
                    porkDetails = porkDetails + " " + productTable.getValueAt(nextRow,1) + ",";
```

```
                pCount++;
                nextRow++;
                break;
            case "Mutton":
                muttonDetails = muttonDetails + " " + productTable.getValueAt(nextRow,1) +
",";
                mCount++;
                nextRow++;
                break;
            case "Chicken":
                chickenDetails = chickenDetails + " " + productTable.getValueAt(nextRow,1) +
",";
                cCount++;
                nextRow++;
                break;
            case "Buffalo":
                buffDetails = buffDetails + " " + productTable.getValueAt(nextRow,1) + ",";
                bCount++;
                nextRow++;
                break;
            case "Fish":
                fishDetails = fishDetails + " " + productTable.getValueAt(nextRow,1) + ",";
                fCount++;
                nextRow++;
                break;
            default:
                break;
        }
    }
```

```
            else{
                break;
            }

        }
        details[0] = porkDetails;
        details[5] = String.valueOf(pCount);
        details[1]= muttonDetails;
        details[6] =  String.valueOf(mCount);
        details[2]= chickenDetails;
        details[7] = String.valueOf(cCount);
        details[3]= buffDetails;
        details[8] = String.valueOf(bCount);
        details[4] = fishDetails;
        details[9] = String.valueOf(fCount);
        return details;
    }

    public static void doSelectionSort(ArrayList<ArrayList<String>> aList) {

        ArrayList<Double> newList = conString(aList.get(5));

        for (int i = 0; i < newList.size(); i++) {
            int p = i;
            for (int j = i; j < newList.size(); j++) {
                if (newList.get(j) < newList.get(p))
                    p = j;
            }
```

```
            double min = newList.get(p);
            newList.set(p, newList.get(i));
            newList.set(i, min);

            for(int k = 0; k < 5; k++){
                sortOtherValue(aList.get(k),p,i);
            }

        }
        aList.get(5).clear();
        for (double myDouble : newList){
            aList.get(5).add(String.valueOf(myDouble));
        }

    }

    public static void sortOtherValue(ArrayList<String> nam, int p, int i){
        String man = nam.get(p);
        nam.set(p,nam.get(i));
        nam.set(i, man);
    }

    public static int binarySearch(ArrayList<String> arr, int low, int high, double find){

        ArrayList<Double> prr = conString(arr);
        while(low <= high){
            int mid = (low + high) / 2;
            if (prr.get(mid) == (find)) {
```

```
            return mid;
        }
        else if (prr.get(mid) < find) {
            low = mid + 1;
        }else{
            high = mid - 1;
        }
    }
    return -1;
}

public static ArrayList conString(ArrayList<String> arr){
    ArrayList<Double> newList = new ArrayList<>(arr.size());
    for (String myInt : arr){
        newList.add(Double.parseDouble(myInt));
    }
    return newList;
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
    * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
```

```
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MeatProductInfo.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MeatProductInfo.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MeatProductInfo.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MeatProductInfo.class.getName()).log(java.util.logging.Leve
l.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
```

```
            new MeatProductInfo().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JTextField IDTF;
private javax.swing.JLabel IDlable;
private javax.swing.JLabel addBack;
private javax.swing.JButton addButton;
private javax.swing.JLabel addLable1;
private javax.swing.JLabel addLable2;
private javax.swing.JLabel addLable3;
private javax.swing.JPanel addPanel;
private javax.swing.JLabel allBack;
private javax.swing.JLabel allLabel;
private javax.swing.JPanel allProductsPanel;
private javax.swing.JTable allTable;
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JLabel discountLable;
private javax.swing.JLabel existingBack;
private javax.swing.JButton existingButton;
private javax.swing.JLabel existingLable;
private javax.swing.JPanel existingPanel;
private javax.swing.JButton jButton2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
```

```
    private javax.swing.JLabel jLabel3;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuClear;
    private javax.swing.JMenuItem jMenuExit;
    private javax.swing.JMenu jMenuFile;
    private javax.swing.JMenu jMenuHelp;
    private javax.swing.JMenuItem jMenuImport;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JButton mainAddBtn;
    private javax.swing.JLabel mainLable1;
    private javax.swing.JLabel mainLable2;
    private javax.swing.JLabel mainLable3;
    private javax.swing.JLabel mainLable4;
    private javax.swing.JPanel mainPanel;
    private javax.swing.JButton mainSearchBtn;
    private javax.swing.JLabel nameLable;
    private javax.swing.JTextField nameTF;
    private javax.swing.JLabel priceLable;
    private javax.swing.JButton priceSearchBtn;
    private javax.swing.JTextField priceSearchTF;
    private javax.swing.JTextField priceTF;
    private javax.swing.JTable productTable;
    private javax.swing.JRadioButton radioAvailable;
    private javax.swing.JRadioButton radioNotAvailable;
    private javax.swing.JLabel searchBack;
    private javax.swing.JPanel searchPanel;
    private javax.swing.JButton showAllBtn;
```

```
    private javax.swing.JComboBox<String> typeBox;
    private javax.swing.JLabel typeLable;
    private javax.swing.JComboBox<String> typeSearchBox;
    private javax.swing.JButton typeSearchBtn;
    private javax.swing.JComboBox<String> weightBox;
    private javax.swing.JLabel weightLable;
    // End of variables declaration

}
```