LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS5004NI Emerging Programming Platforms & Technologies

Assessment Weightage & Type

30% Group Coursework

Year and Semester

2018-19 Autumn / 2018-19 Spring

Group Name:			
SN	**Student Name**	**College ID**	**University ID**
1.	**Aditya Jung Karki**	NP01CP4A170096	17031128
2.	**Aayusha Shrestha**	NP01CP4A170099	17031143
3.	**Binayak Dev Joshi**	NP01CP4A170085	17031105
4.	**Kushal Bhattarai**	NP01CP4A170221	17031137

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.

Acknowledgment

We would like to express our deepest appreciation for all the individuals who have assisted us in completing our group work. The engagement in this group work has given us much pleasure because we have learned and gained a lot from it. We would like to especially like to thank our tutor Mr. Prithivi Maharjan for providing us with some valuable guidelines for the project throughout our numerous consultations.

We would also like to thank all our peers for provided direct/indirect guidance in writing and developing our project. The comment and suggestions that they have provided us have helped us on many fronts of our group work.

We would also like to expand our deepest gratitude to all those who have taken a keen interest in our group work and taken the time to provide us with their useful feedback and constructive criticism.

Finally, the team members itself have made a big contribution to the project by helping each other and building up the needed inspiration and enthusiasm for completing this project.

Abstract

The Information is a system that has an interconnected system of processing raw data and then organizing and managing the data for the user to view. Our coursework inquires us to create an information system that deals with the menu system where the file is to be imported into the application table and then organized in a meaningful manner. The data also needs to be sorted and searched for a result. For the sorting and the searching of the data, we have used the selection sort and the binary search algorithm. We have decided to create an information system for a pet store that handles all the information of the pet and has all the features of adding, deleting, updating, viewing, searching, sorting and importing a file in the application.

Table of Contents

Table of Figures

Table of Tables

1. Individual Tasks

The coursework assigned for this semester was related to designing a GUI a company of our choice which could add data. Our system is based on a pet store allowing the user to add details of the pets. Our group is formed of 4 members and each of us were assigned a specific task. The description of the group members and task are given below.

1.1 Kushal Bhattarai (ID: 17031137)

- **Managed the group as the Group Leader**
  In the group meeting, with the approval of all the member, he was appointed as the leader. He managed the team very well with good communication skills and divided the work for every individual to complete.

- **Searching and sorting part of the code**
  Kushal was responsible for searching and sorting part of the coding process. Kushal has used selection sort for the sorting part and binary search algorithm for the search button. He also explained the method that he used in coding the search and sort algorithms.

- **The coding part of validation and data structure**
  The coding part was equally divided among all the group members. He has done the validation part of the program. A dialog box appears with the appropriate message when any data is input, data is viewed and error message when detected.

- **Description of Binary Search and Selection Sort**
  The sorting and the searching of the data were programmed by Kushal using the binary search and the selection sort algorithm. Then he also explained the working of the algorithm in the report

1.2 Binayak Dev Joshi (ID: 17031105)

- **Designing the GUI**

  Design the GUI and coding rest of the program was done by Binayak. New and creative ideas were discussed, and the many prototypes were designed. Firstly, the wireframes for all the frame was created. He then finalized the design and created the GUI of the project with the mutual agreement between the group members.

- **Coding of adding and updating data**

  Binayak completed the adding and updating data. He managed the coding part of the action listener of add and update button. He also did few of the validation for the program. Then he described all the methods that he used while programming the features.

- **Testing**

  Binayak was responsible for testing the robustness of the application. He ensured the search and sort features worked that they should. He also checked all the validation in the form when the user adds or updates any data. All the testing was then documented.

- **Development**

  Binayak was responsible for documenting the development of the whole project. The procedures followed, the concepts that were utilized and the terminologies were explained and described by him.

1.3 Aditya Jung Karki (ID: 17031128)

- **Data collection**

  Aditya was responsible for collecting data of all the pets and storing them in a .csv file. He researched about all the pets we wanted to store and collected the pictures. He also stored them manually in the excel file. He ensured the quality of our data didn't hold back our system.

- **Coding of the help and login button**

  He was responsible for the code of the help button. Help button displays information of all the buttons and their shortcuts. He also coded for the login button. The login button only allows the admin to login. Then the method used by Aditya

- **Formatting the document**

  The format of the report was structured properly by Aditya. He also wrote the introduction, conclusion and the analysis of the project.

- **Appendix**

  The appendix of the report was made by Aditya. The appendix includes the screenshot of all the code of the application. He took the screenshot of all the codes from the project and then showcased it in the appendix section.

1.4 Aayusha Shrestha (ID: 17031143)

- **Testing**

  Aayusha was responsible for testing the other features of the application. She tested the add, delete, updates and the viewing feature of the application. She made sure the data were manipulated as they should be. Then all the testing were documented in the report.

- **Coding part for delete and view button**

  She wrote code for the delete button to delete any data from the tables. She was also responsible for the view button. The view button displays all the information of the selected pet including the photo.

- **Development**

  Aayusha also collaborated with Binayak in documenting the development of the whole project. She was responsible for showcasing the wireframes, the GUI in the report. All the procedures that were followed were clearly explained in the development.

2. Introduction

The coursework inquires us to develop a java-swing based application that consists of a component for collecting, storing, sorting and searching the data. We have decided to create a Pet Store application for simpler access to the information about types of pets based on the category, prices, and age. In this application, we aim to provide the user the freedom to view details, add, delete, modify and search information of the pets. Our main goal for this application is to be user-friendly, appealing and practical in use. GUI for this application will be clean cut and easily understandable for an overall good experience. This application will allow pet stores and pet lovers to log in to the system, sort the information and view the details of their favorite pets. Studying the industry today, the customers are looking for easy to use the application without any complications that get the job done. As developers, we aim to match the exact needs of the customer with this application. In order to build this application and make it functional, our group members will sit meetings, understand the requirement and plan the process of development with a strive to make an efficient, flexible and easy-to-use application.

This application was developed using NetBeans which is an integrated development environment for developing an application in Java, PHP, and C++. The NetBeans offers various powerful swing components that allow the developer to make beautiful GUI. Various components of swing such as the menu bar, table, frames, buttons etc. were used to make the application. The application was developed in a modular fashion implementing some of the techniques of object-oriented programming in Java.

The application development is solely based on extracting and organizing information which binds various programming techniques among which the binary search and the selection sort algorithm are the leading approaches in the application.

All the tasks on hand were distributed among the individual in the group. The group work is comprised of coding, reporting, developing the GUI and testing the application. All these tasks were tackled by the group strategically to ensure each of the members of the group can use their traits to their maximum.

3. The body of the report

3.1 Selection sort

The algorithm finds the smallest element in the array and swaps it with the first position, then the second smallest element is found and swap with the 2nd position element. Until the array is sorted, the loop runs continuously. The algorithm is called Selection sort because it selects the smallest element to its proper place.

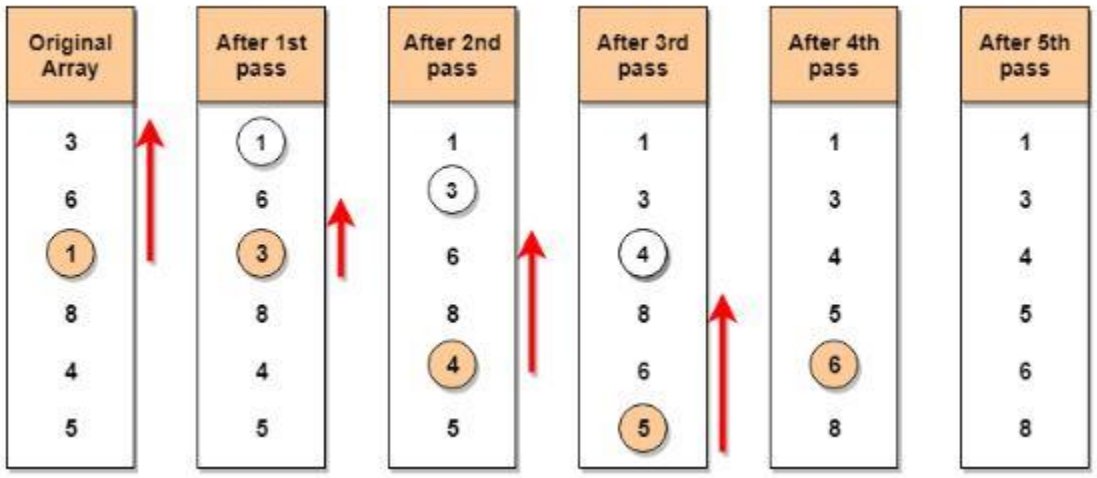


Figure 1: selection sort

Selection Sort requires two nested `for` loops to complete itself, one `for` loop is in the function sortArray (int [] a, ArrayList array), and inside the first loop we are making a call to another function `minimumPosition (int [] a, int value)`, which has the second (inner) `for` loop as well as the swap(int[] arrr, int i, int j) is called to swap the elements of array. Collections.swap (array, minPos, i) is used to swap the elements of the ArrayList present.

In the program, the button named searchPrice searches the price. The method gets the row count from the table, the price was stored in an array named 'price'. The imported data from a file that was stored in an ArrayList was copied to a local copyArray ArrayList. The method SelectionSort.sortArray (int [] price, ArrayList array) is called giving parameters price array and copyArray to sort the array. In sortArray the selection sorting is done by finding minimum price

present in the price array and swapped to the first position. Similarly, in the ArrayList also the swapping is done according to the price array so that the data of the sorted ArrayList can be displayed in the table after proper sorting. After the data is sorted and returned the table is cleared and the sorted ArrayList's data is displayed in the table present in the application.

3.2 Binary Search

The time complexity of Binary Search is 0(log n) n.

Pseudocode for binary search:

The input taken is an array named a price, let the number n of elements in the array and search value, the number being searched for. The output is the searched data.

```
Step 1    Let min = 0 and max = n-1.
Step 2    Guess as the average of max and min, rounded down
               (so that it is an integer).
Step 3    If price [guess] equals searchvalue, then stop. You
               found it! Return guess.
Step 4    If the guess was too low, that is, price [guess] <
               searchvalue, then set min = guess + 1.
Step 5    Otherwise, the guess was too high. Set max = guess -1.
Step 6    Go back to step 2.
```

In the program, the button named searchPrice searches the price. The method gets the row count from the table, the price was stored in the array named a price. The imported data from a file that was stored in an ArrayList was copied to a local copyArray ArrayList. The method SelectionSort.sortArray (int [] price, ArrayList array) is called giving parameters price array and copyArray to sort the array. After sorting, the low and high value to search and key are stored in a variable and the searchList (int [] price, int low, int high, int key) method is called with parameters. In the searchList (int price [], int low, int high, int key) method, mid value is found and checked if the mid value is equal to the key if the key is greater than the value present in price[mid] , the searchList method is again called giving low value as mid+1 and if the key is less than the value present in price[mid], the searchList method is again called giving high value as mid-1. If the key is equal to the price [mid] value then the key mid value is returned. And if the searched price is not found -1 is returned. If the return value is -1 then the dialog box

appears that searched data is found. And if the index of the price array is returned then from the sorted copy array the index values are stored in a variable and displayed in the dialog box with the price of the data which is equal to the searched value.

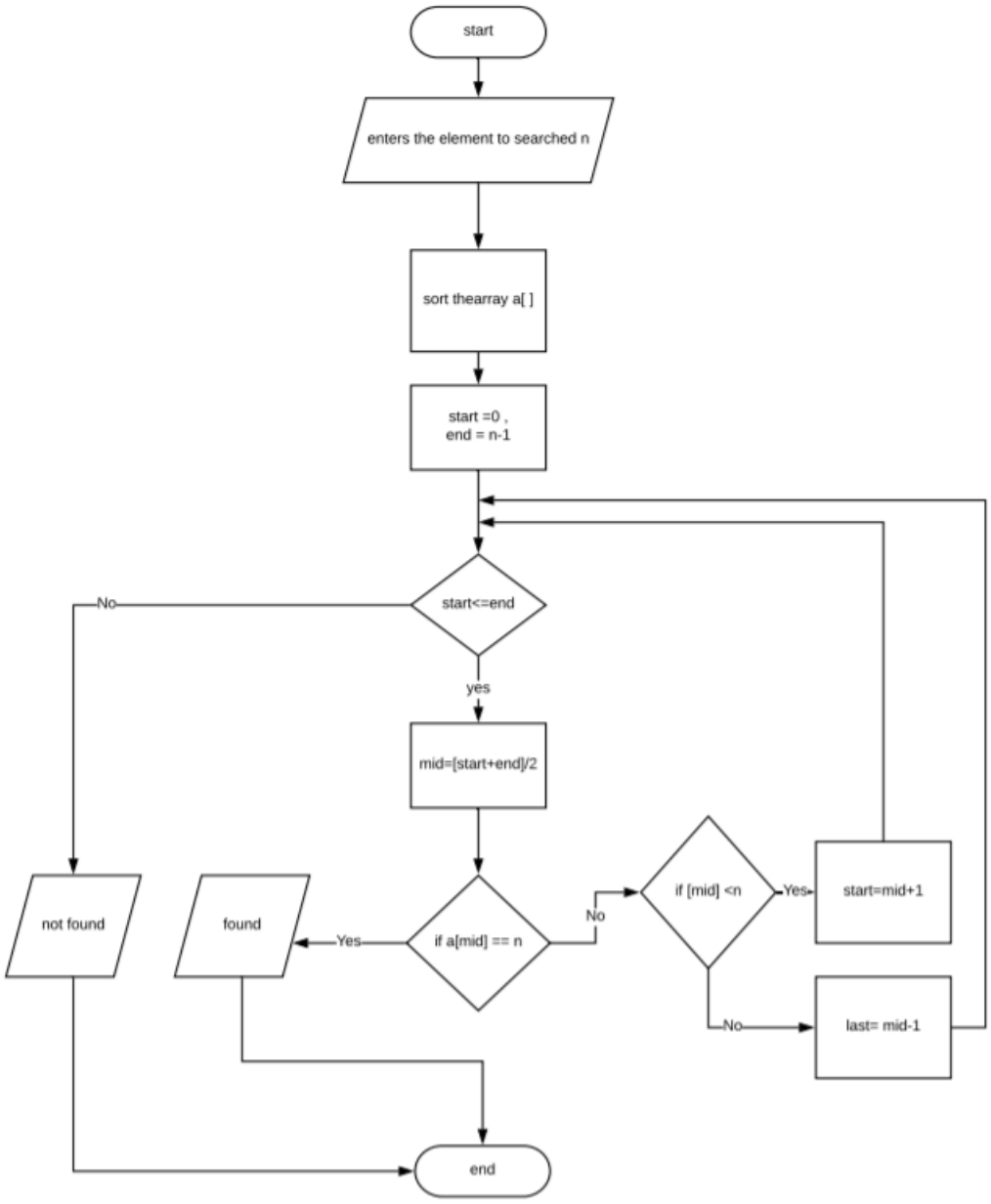


Figure 2: Flowchart of binary search

Algorithm	Best case	Expected	Worst case
Selection sort	$O(N^2)$	$O(N^2)$	$O(N^2)$
Binary search	O(1)	O(log N)	O(log N)

Table 1: Complexity of the Binary Search Algorithm

3.2 List of data

Here is the list of data in our application contains:

- Pet Number: Unique Id for each of the pet available in the store. (TextField: Integer)
- Pet Name: The name of the breed of the dog. (Text Field: String)
- Category: Each pet in the store falls in different category like dog, cat, fish, horse etc. (JComboBox: String)
- Age: This allows the user to know the current age of the pet in months. (TextField: Integer)
- Lifespan: This data provides information about the average life-span of the species. (TextField: String)
- Size: This gives the size of each pet. (CheckBox: String)
- Nature: This describes the nature of each pet whether its' loving, aggressive. (Radio button: String)
- Quantity: This data informs about the remaining pets available in the inventory. (TextField: Integer)
- Price: This data provides with the price of each pet in the inventory. (TextField: Integer)
- Photo: Each pet will have a photo. (File Chooser: String)

3.4 Method Description

- MenuInfo

Public	Void	**refresh()** This method was called to refresh the table. The method fetches the data from a two-dimensional array having all the values all sets in the table.
private	void	**availabilityActionPerformed(java.awt.event.ActionEvent evt)** This method was called to search the categories and calls update table method. On finding the search result it also displays a dialog box for all the details.
public	boolean Void	**categorySelection(String checkItem, int row, int quantity, int columnCount, int rowCount, String[][] available) {** This method was called when the search categories button was clicked and return Boolean data. It filters the data for the selected category and then returns true if the data for the category has been found.
private	Void	**windowMousePressed(java.awt.event.ActionEvent evt)** This method was called to take a coordinate from the working screen.
private	Void	**windowMouseDragged(java.awt.event.ActionEvent evt)** This method calls the windowMousePressed() method and sets the frame to the coordinates updated on dragging the frame.
private	Void	**Menu_exitActionPerformed(java.awt.event.ActionEvent evt)** This method closes the application.
Private	Void	**searchPriceActionPerformed(java.awt.event.ActionEvent evt)** This method the price in the array, copies the main data from the

		table, calls sortArray() method to sort, calls search list method and displays dialog box according to the found price or not found price.
public	int	**Searchlist(int price[ ],intlow,int high,int key)** This method used for binary searching
private	Void	**searchKeyTyped(java.awt.event.KeyEvent evt)** This method doesn't let the user to type characters other than digit.
private	Void	**sortPrice1ActionPerforned(java.awt.event.KeyEvent evt)** This method stores price and data in an array and arraylist and calls sortArray() method to sort and updates the table..
private	Void	**updateTable (String[ ] [ ] array)** This method updates the table.
private	Void	**Import_csvActionPerformed(java.awt.event.ActionEvent evt)** This method chooses a .csv file from file chooser sets path. Adds to the array, adds to the table and updates the category.
private	Void	**addCategory()** This method adds a category to the combo box selecting a unique category.
private	Void	**addtoTable()** This method adds data from Array List to jtable.
private	Void	**addtoArray()** This method used BufferReader and stores in array and also calls .csv to ArrayList method.
private	Void	**CSVtoArrayList(String CSV)** This method takes CSV file and converts .csv file from object to string arraylist.
private	Void	**addActionPerformed(java.awt.event.ActionEvent evt)**

		This method calls the get objMethod from AddData frame.
public	Void	**addData()** This method adds new data to the model.
Public static	Void	**Main()** The main method is similar for all. This method is responsible for the loading the GUI components.
private	Void	**exitMouseExited(java.awt.event.MouseEvent evt)** This method performs the hover effect of the exit button.
private	Void	**exitMouseMoved(java.awt.event.MouseEvent evt)** This method performs the hover effect for button.
private	Void	**adminActionPerformed(java.awt.event.ActionEvent evt)** This is the login method for admin.
private	Void	**delActionPerformed(java.awt.event.MouseEvent evt)** This method deletes data using petID.
private	Void	**view_detailsActionPerformed(java.awt.event.ActionEvent evt))** This method takes index and petID and calls details,viewdetails() method.
public	Void	**checkCategory(String category_)** This method is called after adding or updating the data. If the user has added a new category it stores that unique category in an array.
Public	Int	**getRowCount()** Returns the row count of the table.
Private	Void	**Log_outActionPerformed(java.awt.event.ActionEvent evt)** This method logs out the user from the staff privileges.
Public	Void	**User_guideActionPerformeed(java.awt.event.ActionEvent evt)** This method calls the object of the class AppHelp and makes it visible.

Private	Void	**minimizeActionPerformed(java.awt.eventActionEvent evt)** This method minimizes the application.
Private	Void	**exitActionperformed(java.awt.event.ActionEvent evt)** This method closes the application.
Private	Void	**refreshActionPerformed(java.awt.event.ActionEvent evet)** This method calls the refresh() method.

- SelectionSort

Public Static	Void	**Swap(int[ ]a, int i, int j)** This method swaps the integer array called a.
Public Static	Void	**sortArray(int [ ] a, ArrayList array)** This method calls the minimum position and calls swap from selectionsort.java file and also calls collections.swap() from the library.
Public Static	Int	**MinimumPosition(int [ ] a, int from)** This method returns minimum position by checking from to index.

- AddData

Public Static	Void	**getObj()** This method makes a new Object for the class if the reference variable is null initially and does not allow the frame to make new objects again until it is closed.
Public	Void	**setIdDefault()** This method checks for the largest pet id in the two dimensional ArrayList and then sets the pet id in the add form incrementing I by one.

Private	String	**getRadioSelection()** This method checks the radio selection from the add form and then returns the value.
Private	String	**getCheckSelection()** This method checks the selection in the checkbox group and then returns the value.
Private	boolean	**checkForInt(String id, String price, String quantity)** This method checks sure any integer value is a whole number and non-negative.
Public	Void	**clear()** This method clears all the selection in the add form.
Private	Void	**addActionPerformed(java.awt.event.ActionEvent evt)** This method adds all the values entered by the user in the add form into the two-dimensional ArrayList and table.
Private	void	**addImageActionPerformed(java.awt.event.ActionEvent evt)** This method opens file chooser to choose image file for pet.
Private	void	**ageKeyTyped(java.awt.event.KeyEvent evt)** This method is used to input only digit characters in the textfield.
Private	void	**quantityKeyTyped(java.awt.event.KeyEvent evt)** This method is used to input only digit characters in the textfield.
Private	void	**nameKeyTyped(java.awt.event.KeyEvent evt)** This method is used to exclude only digit characters in the textfield.
Private	void	**priceKeyTyped(java.awt.event.KeyEvent evt)** This method is used to input only digit characters in the textfield.

- Details

Public	Void	**viewDetails(int id, ArrayList<ArrayList<String>> array)** This method checks for all the values from the corresponding selected row of the table in the two-dimensional array. It then sets all the values in the frame for the user to see. This method also scales the image to fit the ratio of the JLabel.

- Update

Public	Constructor	**Update (int rowIndex, String id, String name, String category, String age, String lifespan, String size, String nature, String quantity, String price, String image)** This method takes in the values and then initializes those values.
Private	boolean	**checkNature(String nature)** This method checks the value of the global variable and then sets the selection in the update form.
Private	Void	**checkSize(String size)** This checks the value from the constructor and then checks the corresponding check box.
Private	Void	**updateActionPerformed (java.awt.event.ActionEvent evt)** This method gets the value from the text fields, checkbox and radio button validates the input and then replaces the value in the two-dimensional array and the table if any changes have been made.
Private	void	**imageChooseActionPerformed(java.awt.event.ActionEvent evt)** This method opens file chooser to choose image file for pet.
Private	String	**radioSelection()** This method checks the radio selection from the add form and then returns the value.

Private	String	**checkSelection()** This method checks the selection in the checkbox group and then returns the value.
Public	void	**setValue()** this method sets the value to in the frame getting values from the constructor.

3.4 Development

The development of the project is based on making an information system that interrelates different components of collecting data, processing the data and then organizing them. “Information systems are combinations of hardware, software, and telecommunications networks that people build and use to collect, create, and distribute useful data, typically in organizational settings.” (Bourgeois & Bourgeois, 2014) The development process also required us to carefully analyze the requirement. For the development of the application, it was necessary to reduce the components of the application to understand the basic working. Before the development process was initiated, the pre-development process was taken into account They are as follows:

1. The prototype of the application was conceptualized, and the members of the group were accounted for bringing new ideas for the table.

2. After the theme of the application (Pet Store) was agreed upon the whole group held meeting for brainstorming. The GUI, the features, the types of data for the application was discussed and the proposal was drafted.

3. that the data required for the application was collected and made sure the data were of utmost quality.

4. Then various prototype for the GUI was made by making wireframes for the application. After that, the development and the coding for the application were started.

Some of the steps mentioned above may be analogous to the development stages of software engineering. The group tried to develop the application in a systemic way so some of the techniques used are software engineering were used to some degree. Since the application we are developing is of a small scale it is not necessary to go through all the development stages thoroughly.

3.3.1 Wireframe prototyping of the GUI

- **Dashboard Wireframe**

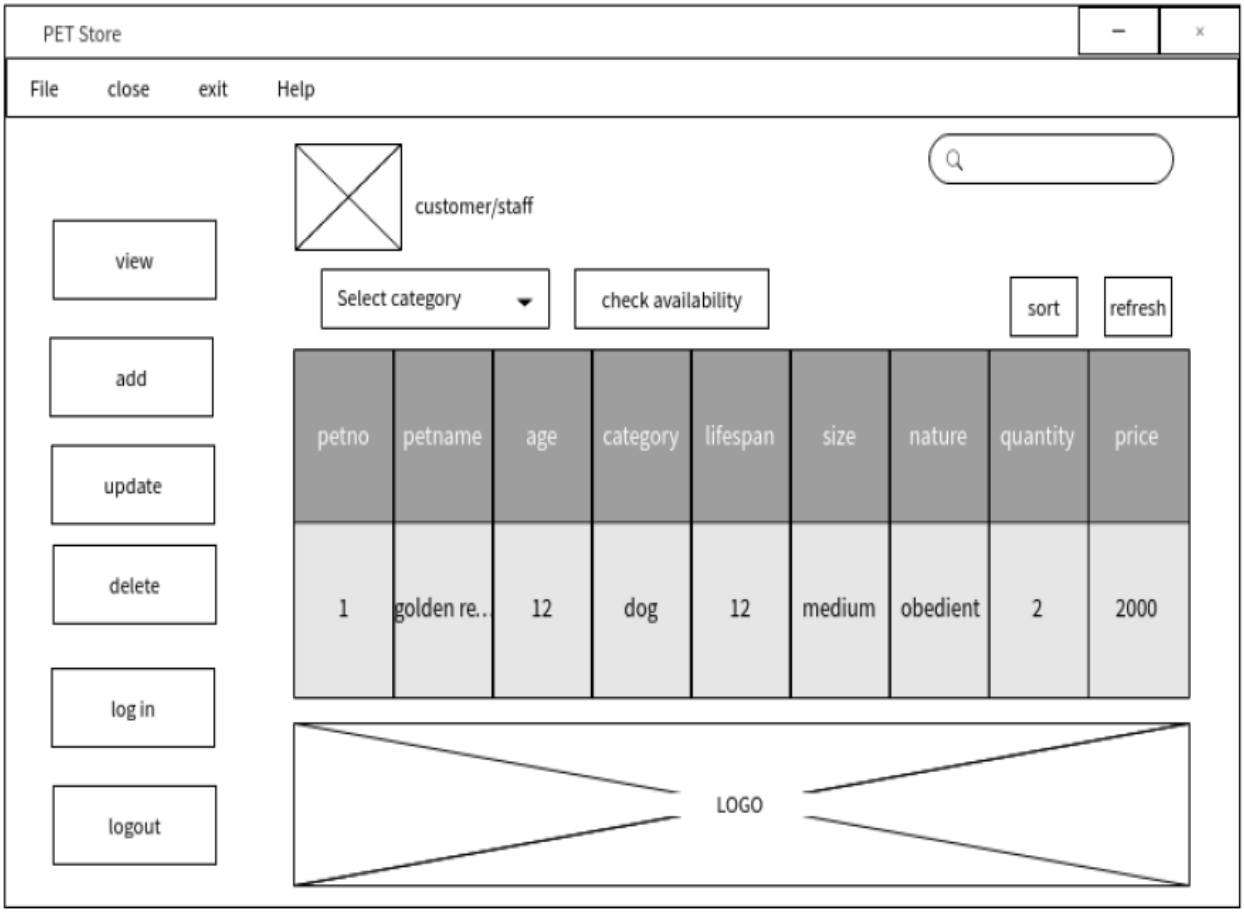


Figure 3: Dashboard wireframe

- **Add form Wireframe**

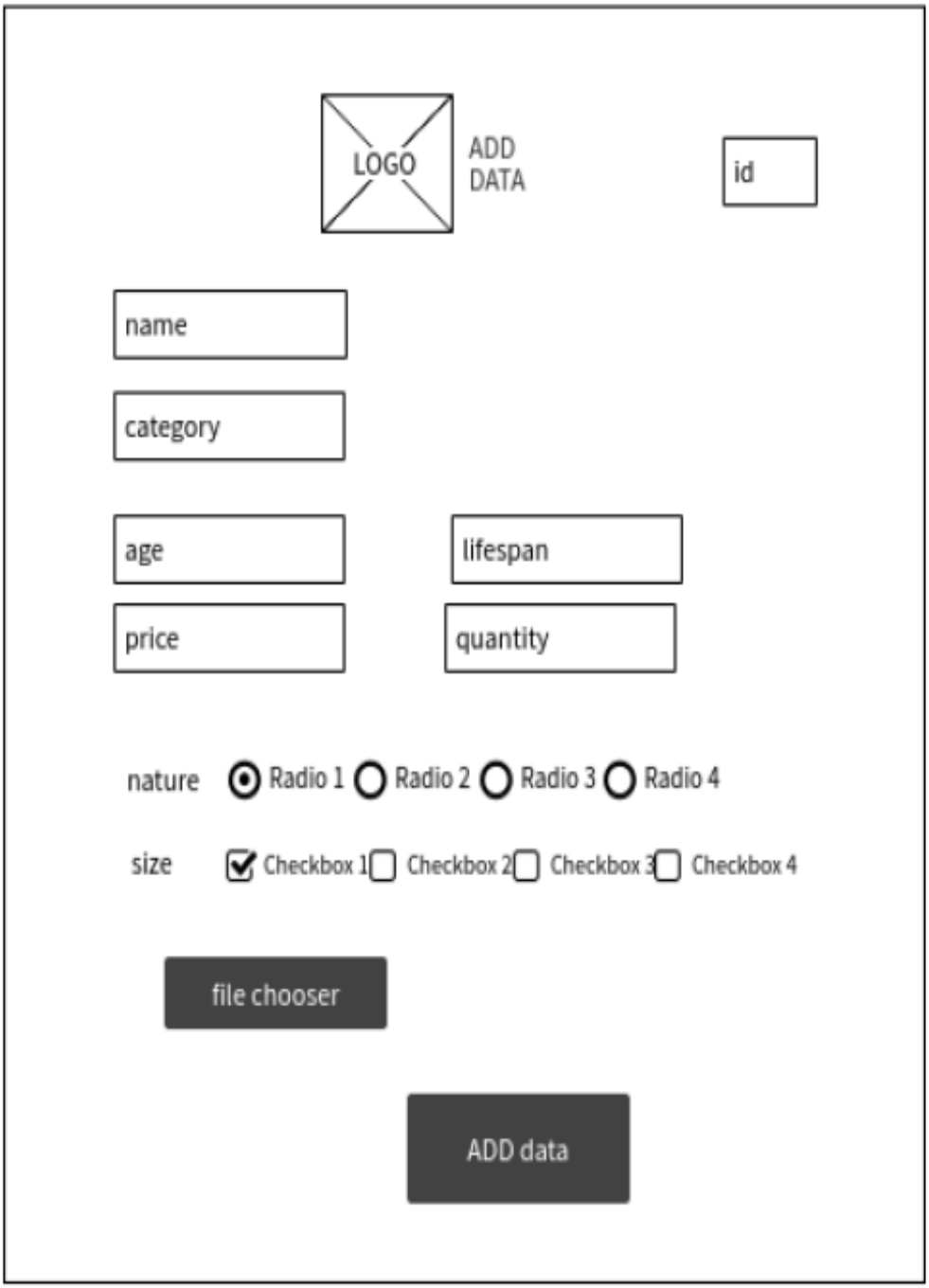


Figure 4: Add Form Wireframe

- **Update Form Wireframe**

Figure 5: Update Form Wireframe

- **View Wireframe**

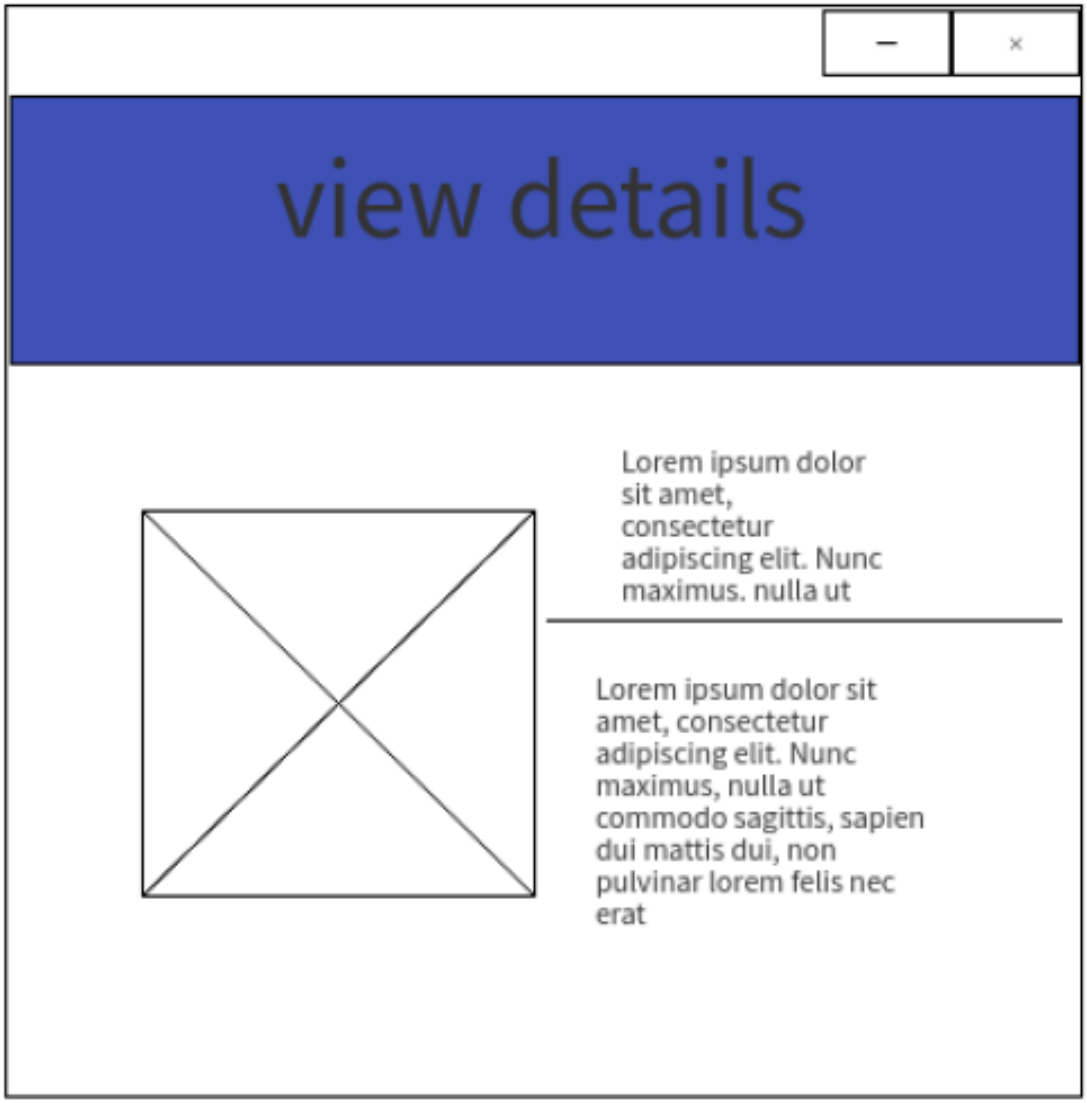


Figure 6: View wireframe

3.3.2 Developing the GUI

After developing the prototype of how the application will look the GUI components were made with the help of the swing components available in the NetBeans IDE. The NetBeans allows the user to drag, drop and position the GUI components from the palette. The palette offers numerous amounts of swing components. The GUI of our application uses the frame, JButtons, jLabels, ComboBoxes, Radio Buttons, Check Boxes, Text Fields that has been used creatively to give the application a modern and appealing look. Also, the java provides many swing libraries which helps to change the way certain components behave in the application.

- **Dashboard**

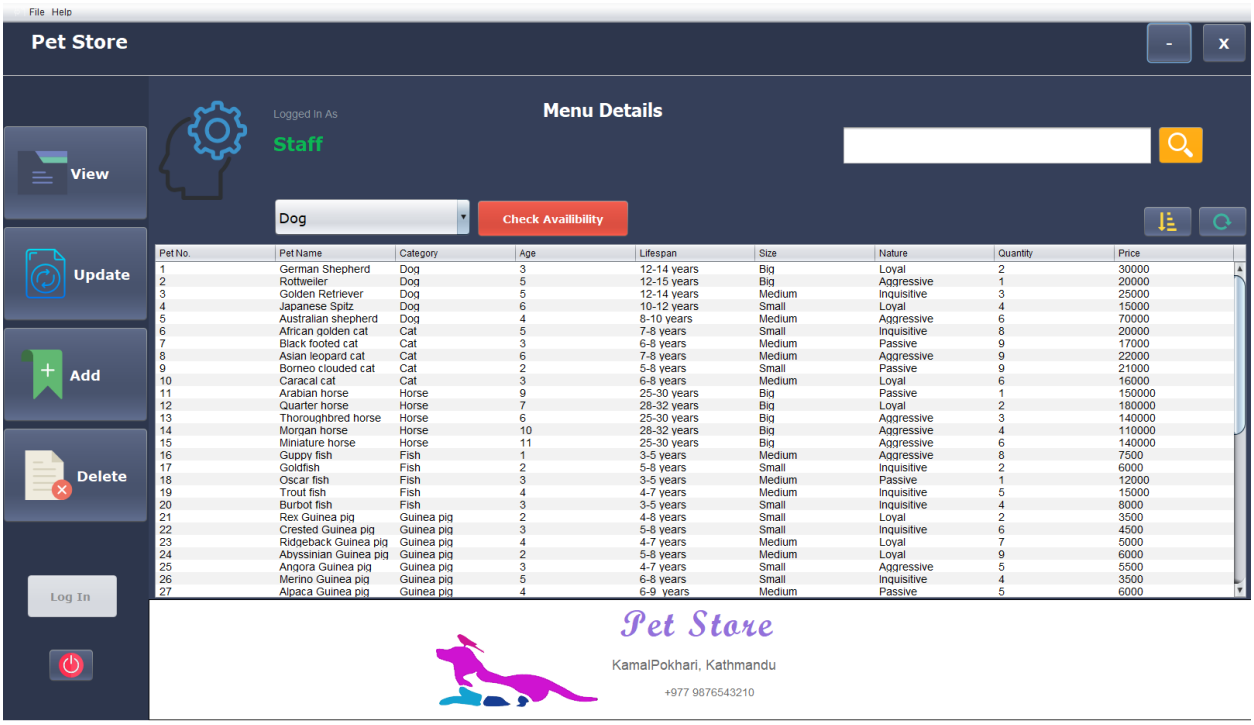


Figure 7: Main dashboard of the application

- **Add Form**

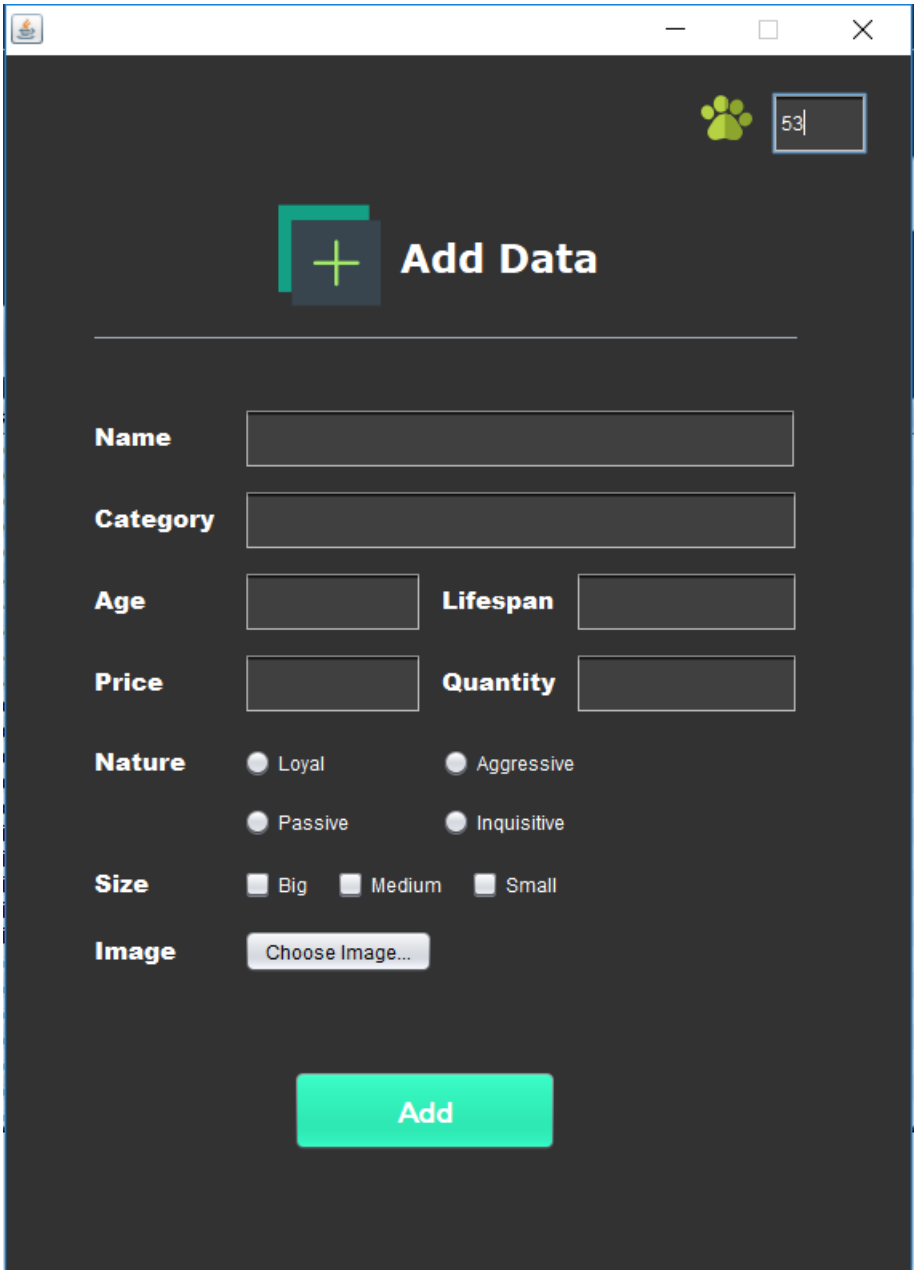


Figure 8: Add form for adding data

- **Update Form**

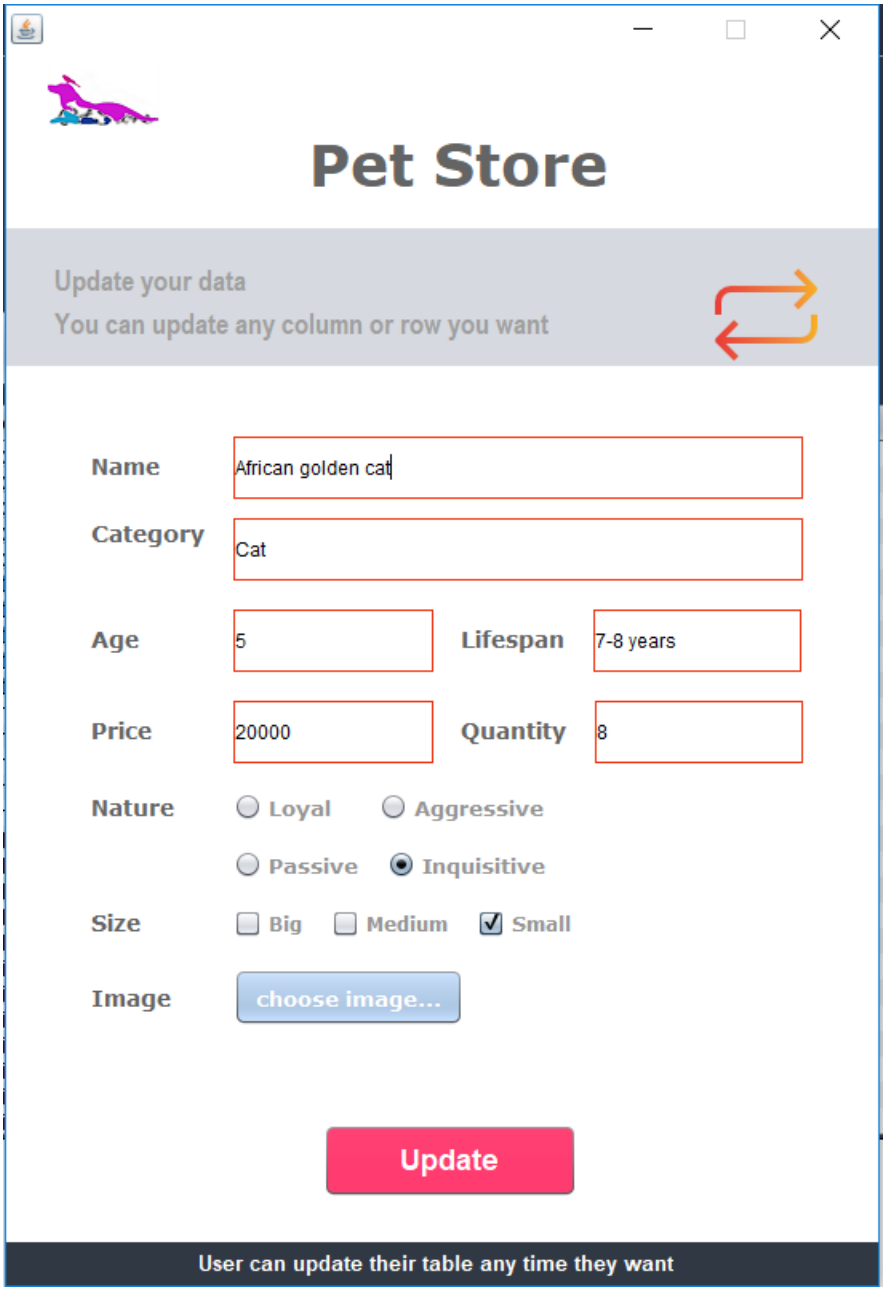


Figure 9: Update form for updating the data

- **View Frame**

Figure 10: Frame for viewing the details of a pet

3.3.3 Programming the application

Our application has multiple frames for which we need to communicate our data and information between two different frames. For this, we used the object-oriented programming techniques which also makes our application modular and easy to develop. For the object-oriented programming, we mainly adopted the encapsulation method to restrict the explicit use of methods in different classes and hiding the information. Object-oriented has a number of advantages that are as follows:

1. Object-oriented programming makes the program modular which makes the features a class easy to use and extend.

2. The object-oriented programming makes use separate java classes which makes it easier to distributes the tasks to multiple people when working in a group.

3. Debugging the application is also easy since the application is modular the errors are easy to find and easy to maintain.

4. The overall development process also becomes much faster and easier since multiple people can work on different modules.

- **Some uses of Object-oriented programming**

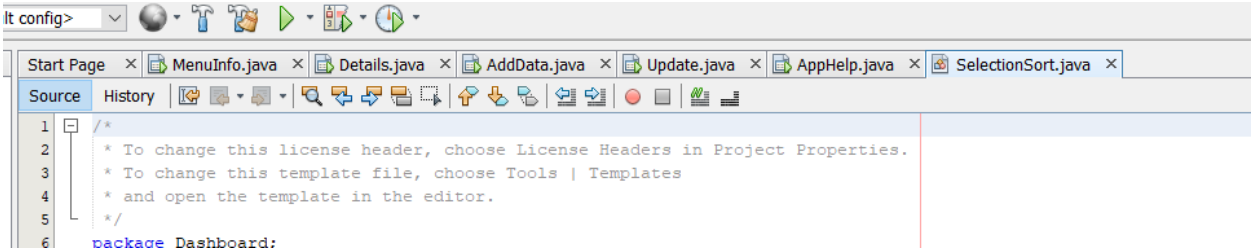


Figure 11: Using separate classes for different functions

```
     * @return
     */
    public static AddData getObj() {
        if (obj1 == null) {
            obj1 = new AddData();
        }
        return obj1;
    }
```

Figure 12: Initializing the object of a class

```
try {

    AddData.getObj().setVisible(true);
    AddData.getObj().setIdDefault();

    refresh();
    model.setRowCount(0);
    addToTable();
```

Figure 13: Calling non-static method from other classes using objects

3.3.4 User Guide

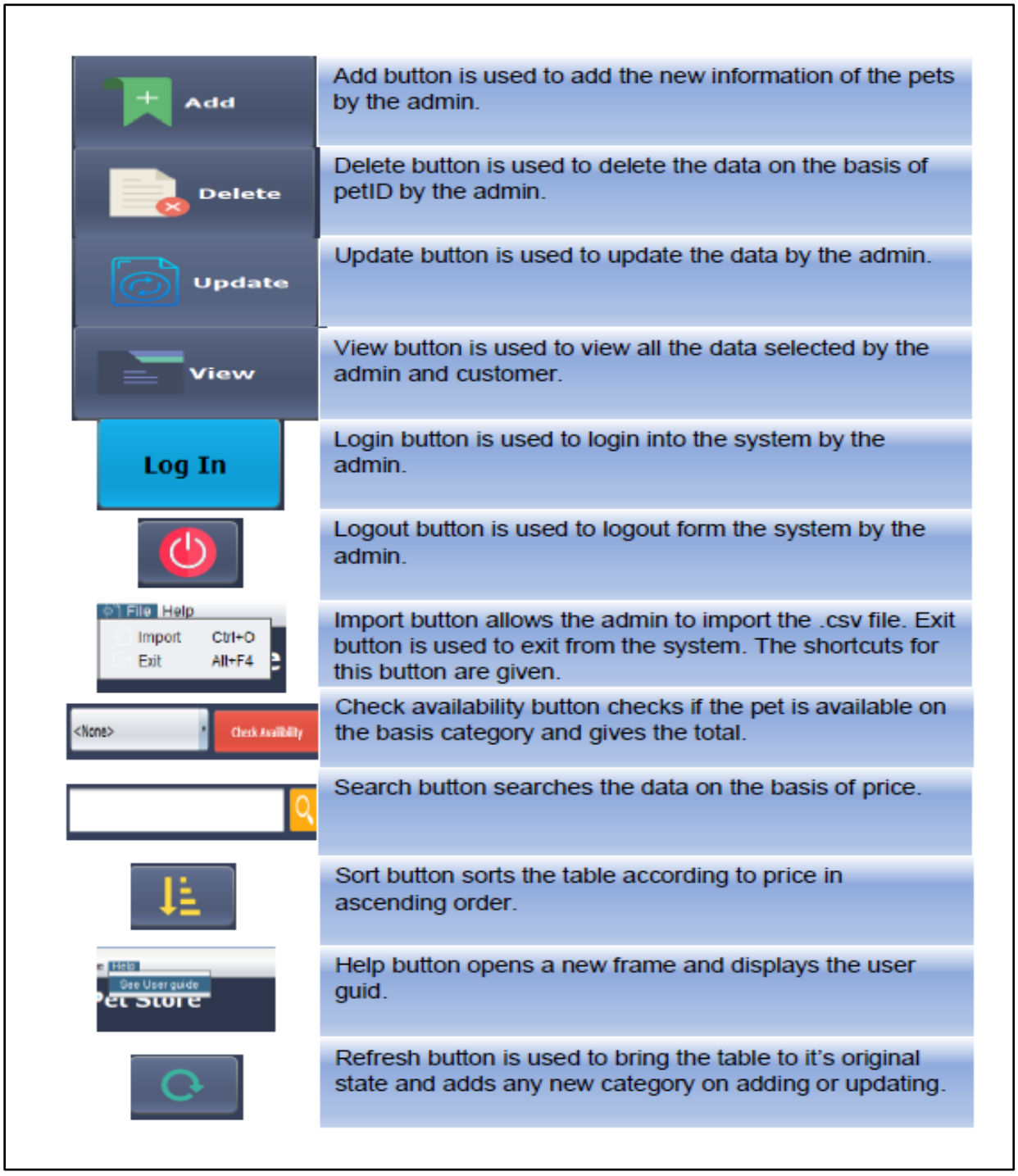


Figure 14: User guide

3.5 List of features

Our application contains the following features:

- The user can sort the information based on price in ascending order.
- The user can also manage and record their information in a CSV format and import it into the application using a file explorer.
- One of the fundamental features of this application will allow the user to search for information using a binary search algorithm.
- This application allows the staff to add, delete and update the information for flexibility and maintainability of the data.
- The application will also have a simple privilege system that only allows the staff to make crucial changes like adding, deleting and modifying data in the application which maintains the integrity of the information.
- With outstanding and modern GUI, the user will be able to navigate in the application with much ease.
- The GUI also introduces a splash screen.
- The application also has error handling that prevents the user from providing invalid inputs by displaying a message prompt.
- The application also has specific shortcut keys for importing and exiting.
- The application will also help to give the available stock of a category like dogs, cats, horses etc.
- On hovering over different components inside the application the user will be able to learn about the different functionalities using the tool tip.
- The application also provides with a help menu that provides information about the general working of the application.
- The user will also be able to view all data or specified data with filtering.

4. Testing

After the development of the application, it is customary to test the application for any errors or bugs. One of many ways of testing an application is **Black Box Testing.** This testing is involved with testing the components considering the inputs, outputs and general functionalities as per the required specification. This testing makes sure the application fits well with the requirements as well as the implied ones. Several tests were done on the application to ensure the fundamentals of the application is working properly. (Limiye, 2009)

Test 1

Objective	The run the project in the Netbeans
Action	The project was compiled, built and ran with Netbeans.
Expected Result	The project should compile and run.
Actual Result	The project ran successfully,
Test Remark	Success.

Table 2: Testing if the projects run in NetBeans

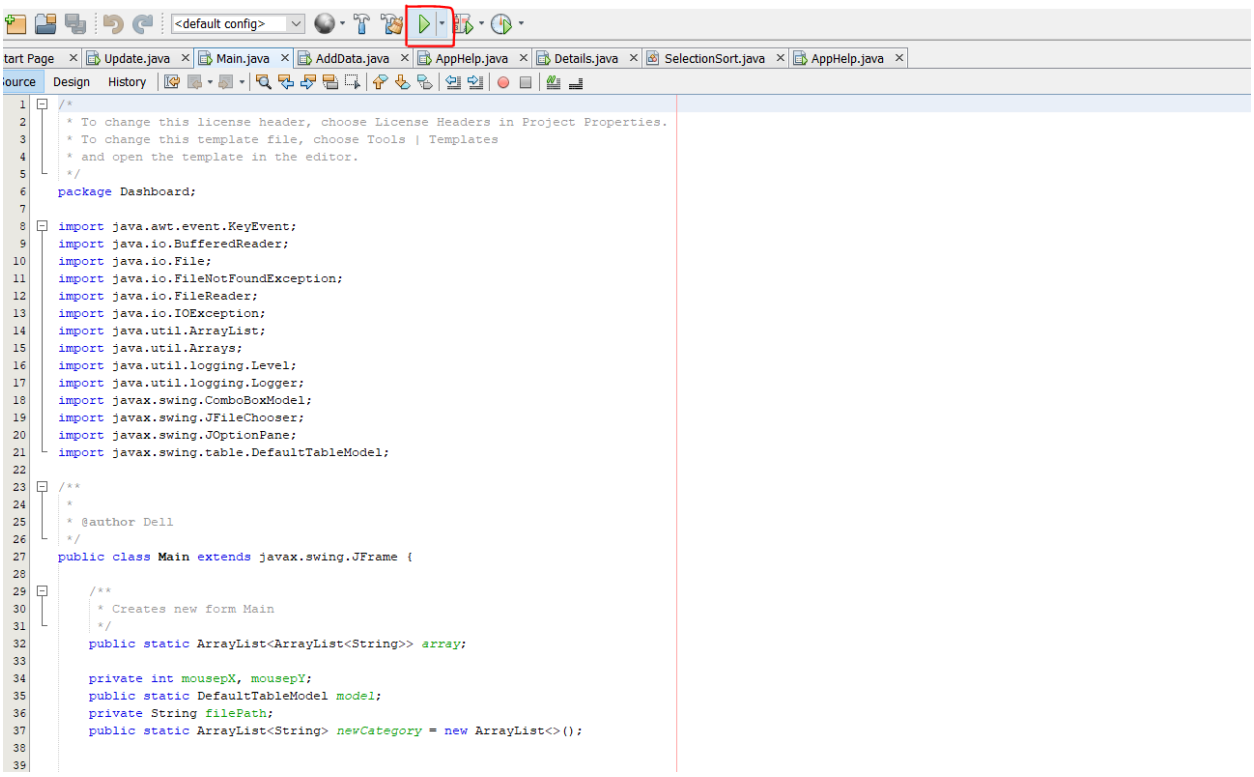


Figure 15: Running the project

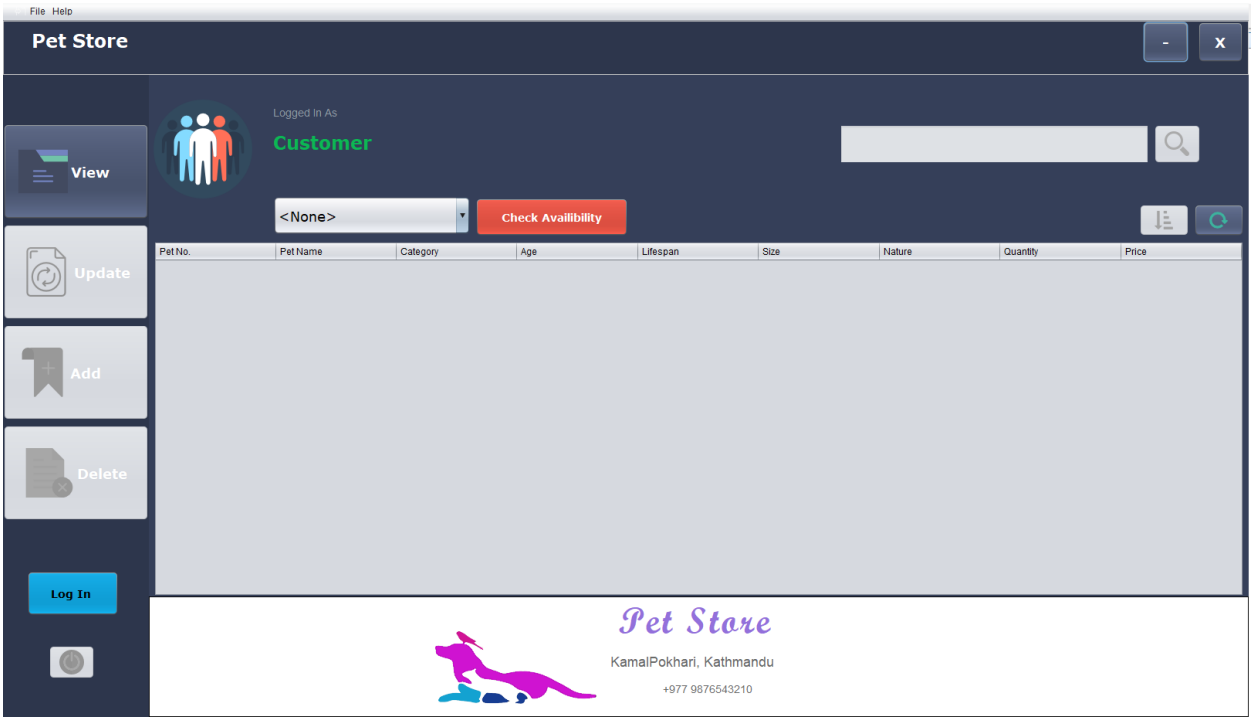


Figure 16: The application running from NetBeans

Test 2

Objective	To import a CSV file into the table.
Action	The file was selected with the help of file selector and then tried to import it into the table.
Expected Result	The file should be imported into the table with all the data in the correct order.
Actual Result	The CSV file was successfully imported into the table.
Test Remark	Success.

Table 3: Testing the import feature of the application

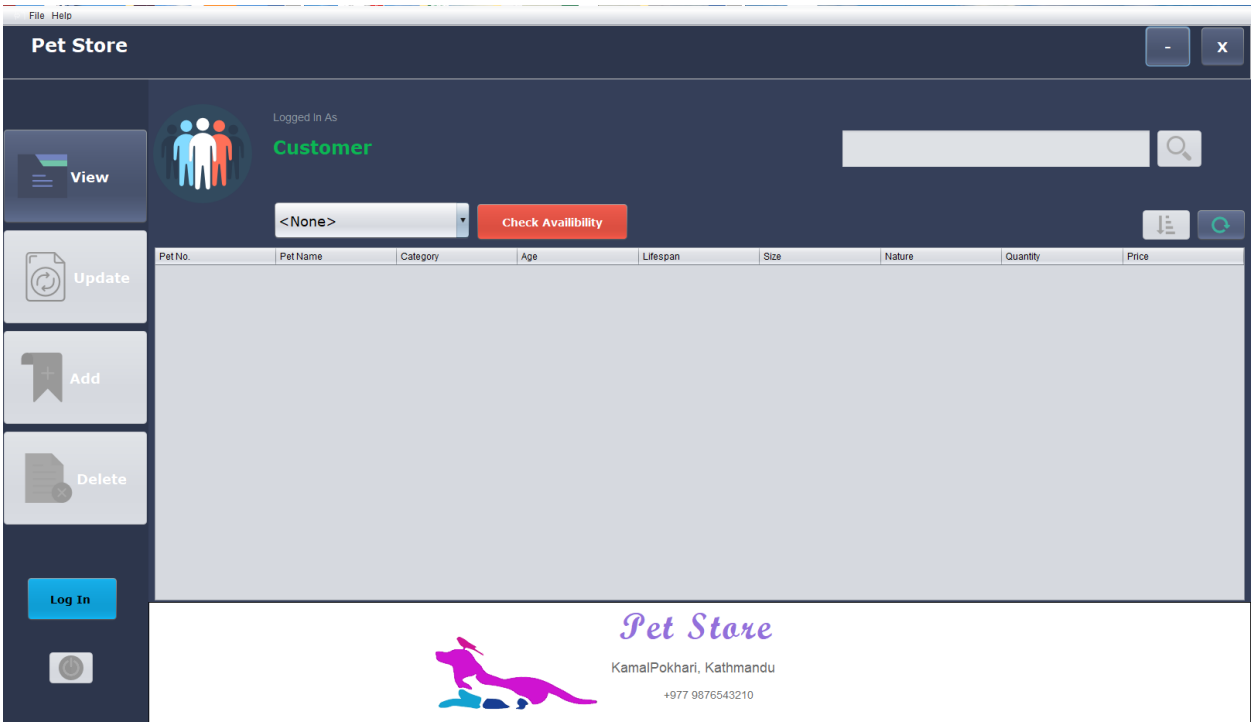


Figure 17: Empty table before importing

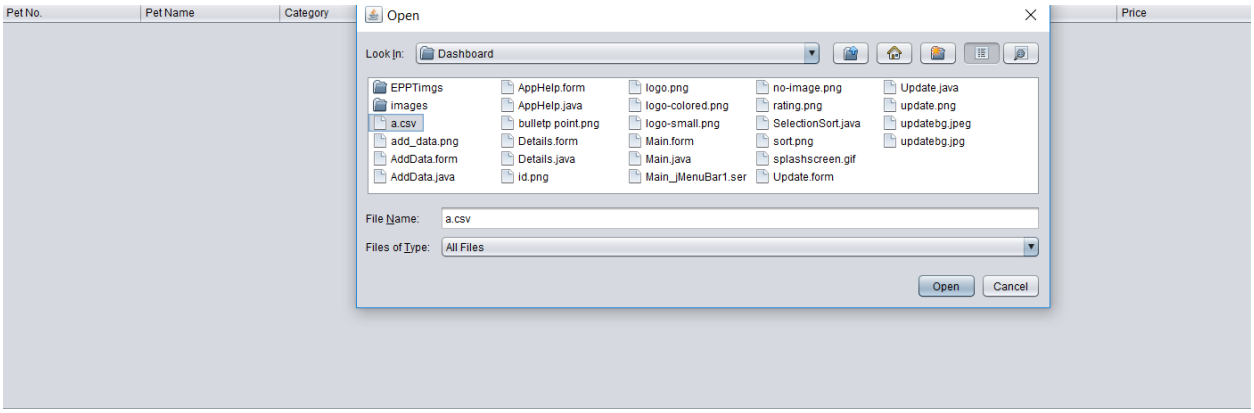


Figure 18: Selecting a csv file to import into the table

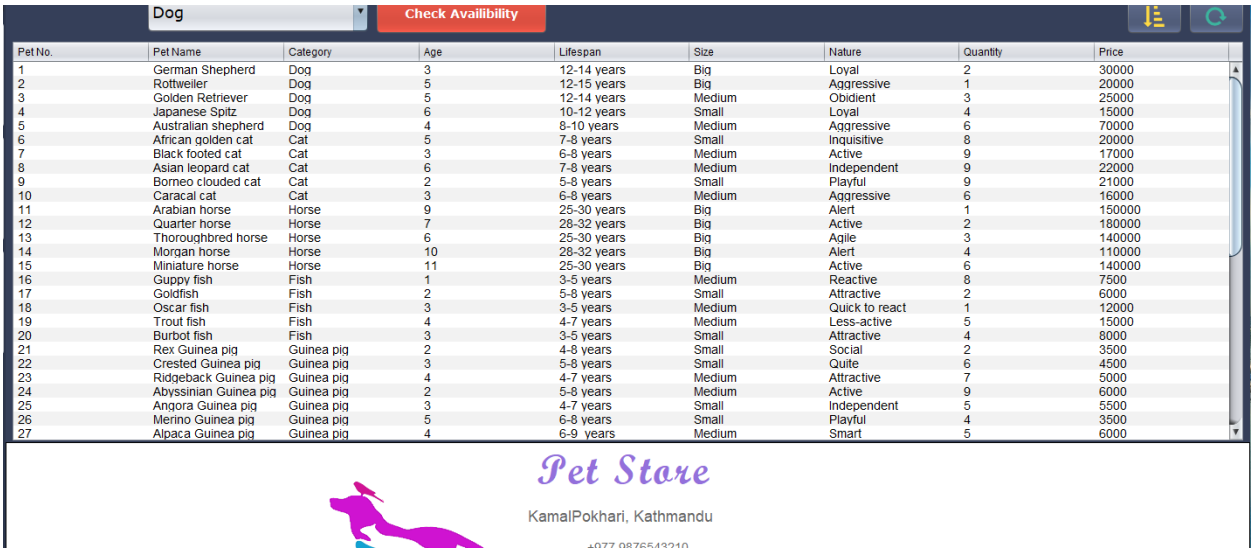


Figure 19: Data is imported into the table

Test 3

Objective	To add a new row into the table.
Action	All the fields in the add form were inputted for a new pet and then the add button was clicked.
Expected Result	All the data should be added to the table with an appropriate message in a dialog box.
Actual Result	Data was successfully added to the table.
Test Remark	Success.

Table 4: Testing the add feature of the application

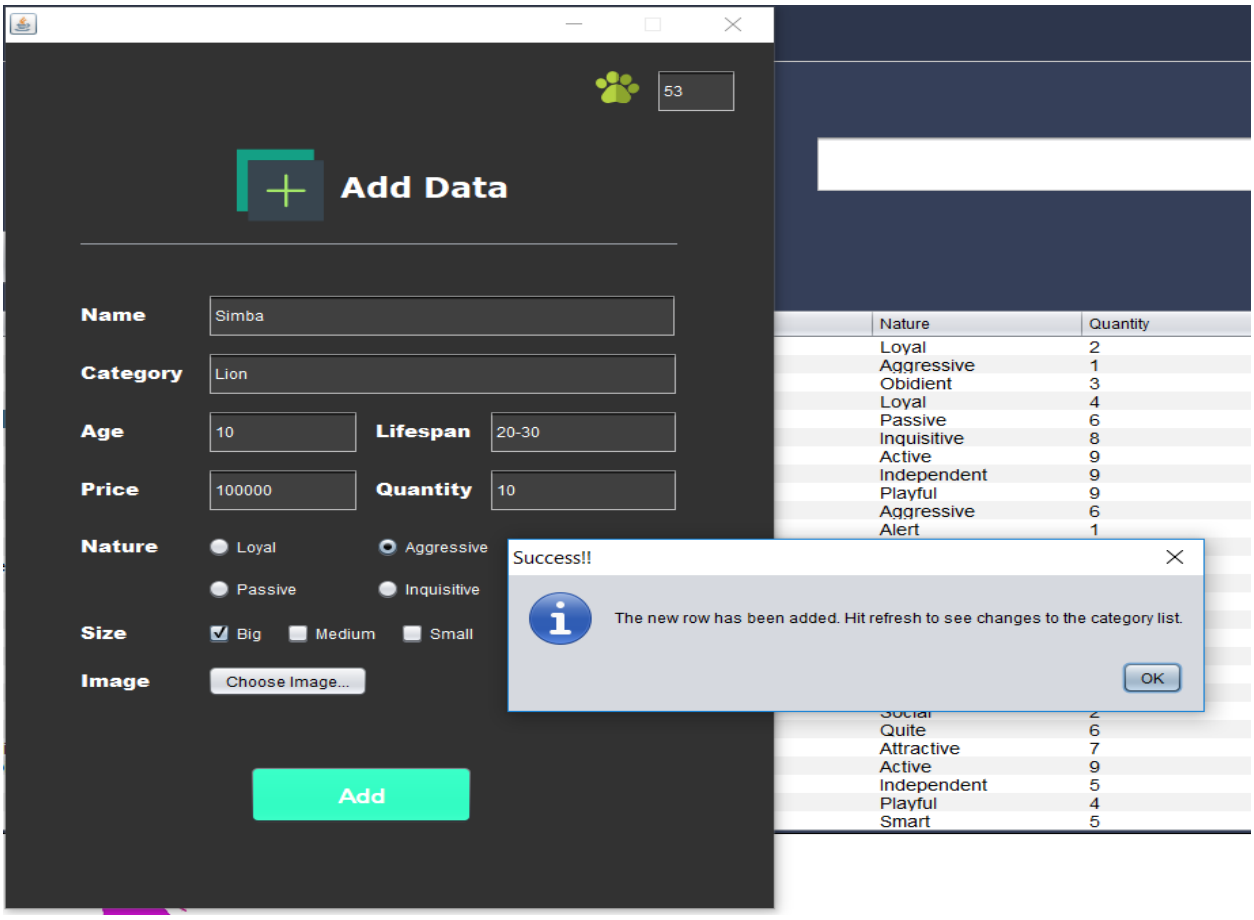


Figure 20: Adding new row into the table

Figure 21: New row added to the table

Test 4

Objective	To update a row in the table.
Action	The row to be updated was selected then the update button was clicked. Then a new frame appears that sets all the values in the text fields, checkbox or the radio button. Then certain values were changed to see the changes.
Expected Result	The values that have been changed should be changed in the table as well.
Actual Result	The row was successfully updated.
Test Remark	Success.

Table 5: Testing the update feature of the application

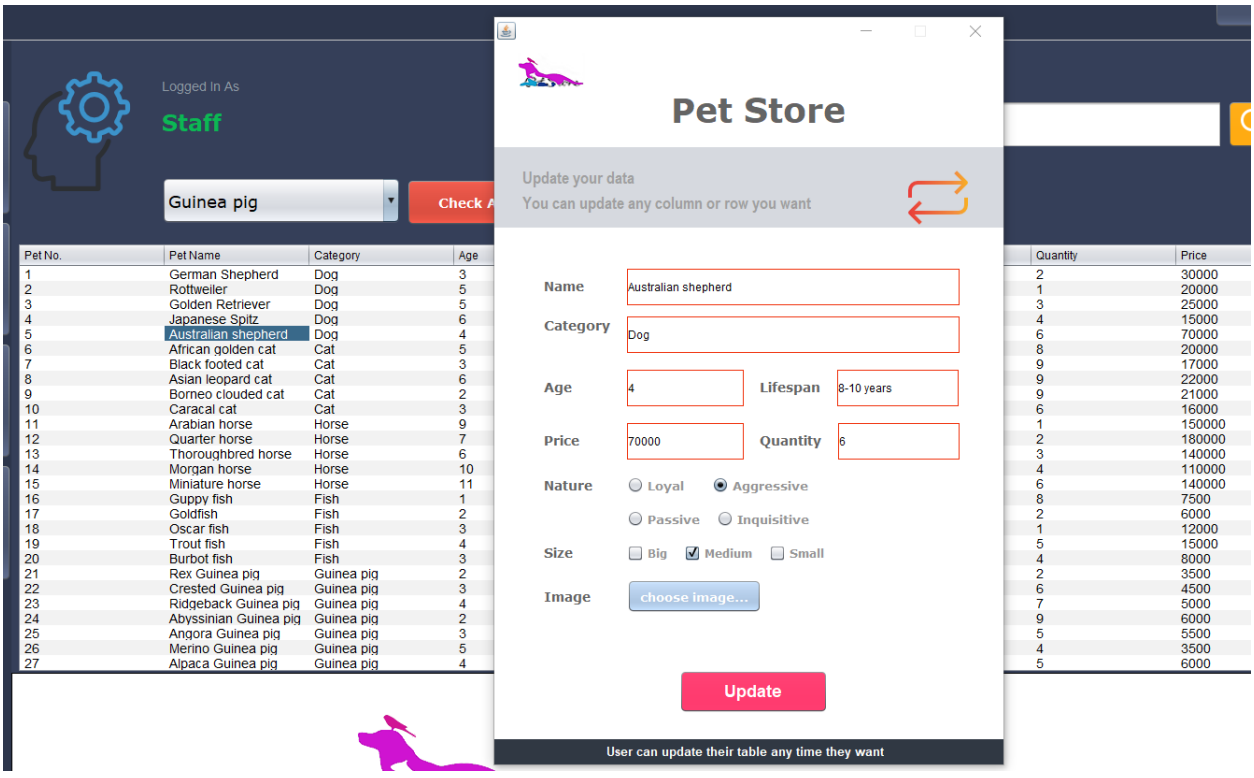


Figure 22: Selecting a row to update

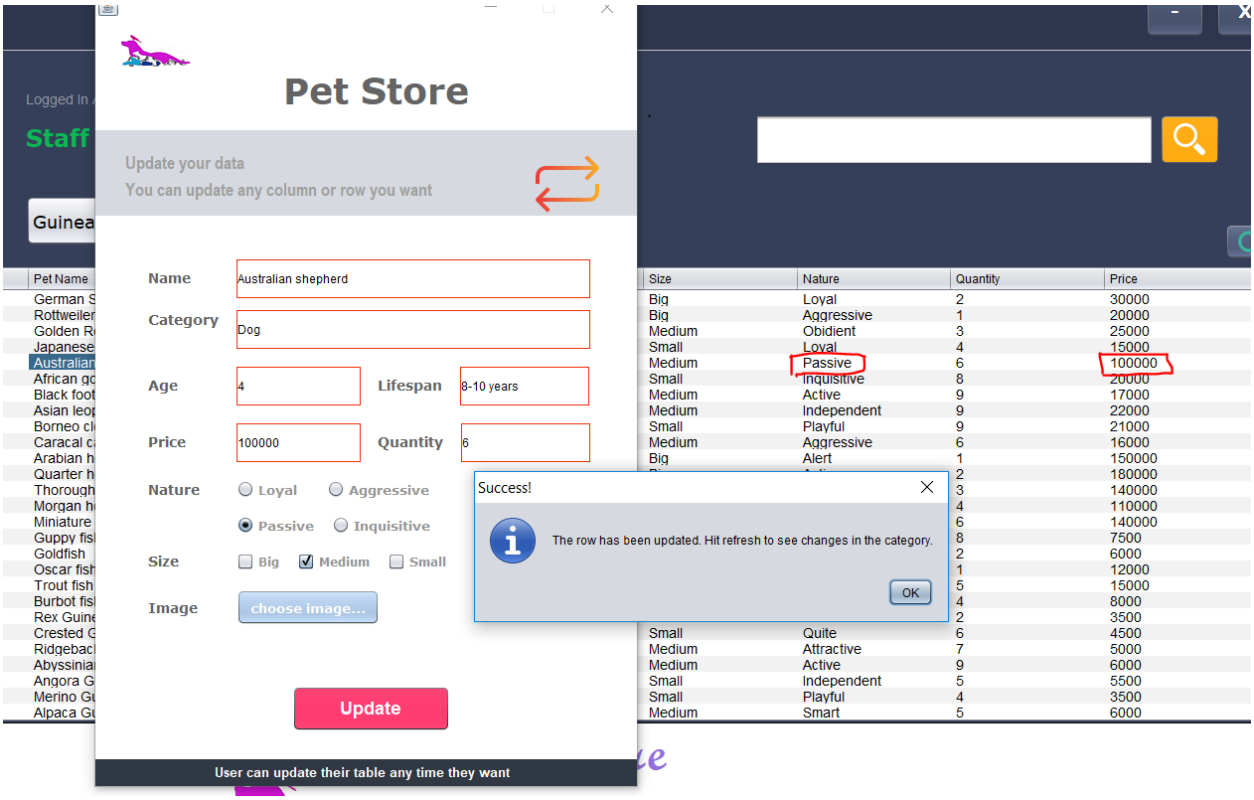


Figure 23:Updated value in the table

Test 5

Objective	To view the details of the row from the table
Action	The row to be viewed in detail was selected then the view button was pressed.
Expected Result	A new frame should open displaying all the details of the row along with the pictures.
Actual Result	The row was successfully displayed in a new frame to be viewed.
Test Remark	Success.

Table 6: Testing the view feature of the application

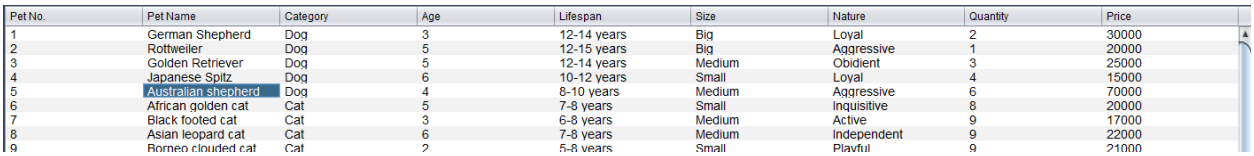


Figure 24: Selecting a row to view the details

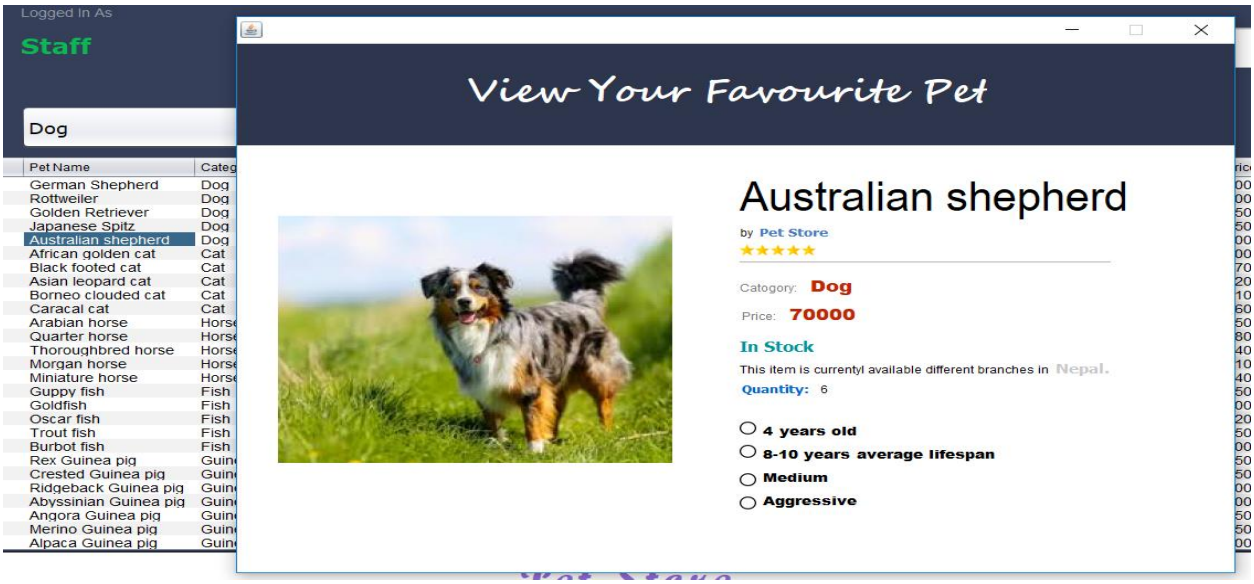


Figure 25: Viewing the selected row

Test 6

Objective	To delete a specific row from the table.
Action	The delete button was pressed and the pet id of the pet to be deleted was inserted into an input dialog box.
Expected Result	The row with the same pet id should be deleted from the table along with an appropriate message in a dialog box.
Actual Result	The row was successfully deleted.
Test Remark	Success.

Table 7: Testing the delete feature of the application

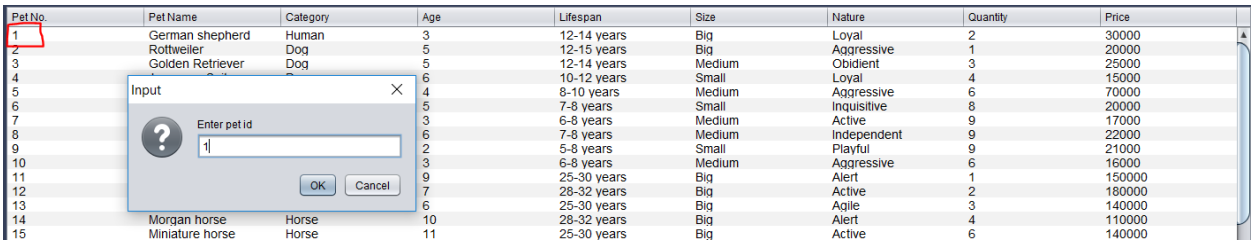


Figure 26: Entering the pet id for the corresponding row to be deleted

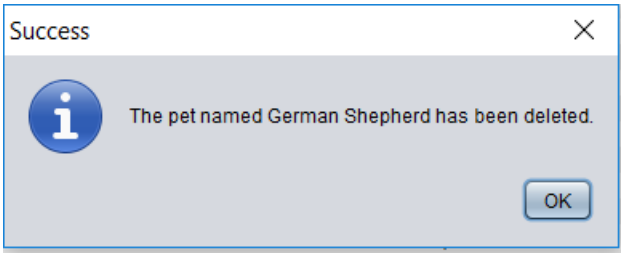


Figure 27: Dialog box printing the name of the deleted pet

Pet No.	Pet Name	Category	Age	Lifespan	Size	Nature	Quantity	Price
2	Rottweiler	Dog	5	12-15 years	Big	Aggressive	1	20000
3	Golden Retriever	Dog	5	12-14 years	Medium	Obidient	3	25000
4	Japanese Spitz	Dog	6	10-12 years	Small	Loyal	4	15000
5	Australian shepherd	Dog	4	8-10 years	Medium	Aggressive	6	70000
6	African golden cat	Cat	5	7-8 years	Small	Inquisitive	8	20000
7	Black footed cat	Cat	3	6-8 years	Medium	Active	9	17000
8	Asian leopard cat	Cat	6	7-8 years	Medium	Independent	9	22000
9	Borneo clouded cat	Cat	2	5-8 years	Small	Playful	9	21000
10	Caracal cat	Cat	3	6-8 years	Medium	Aggressive	6	16000
11	Arabian horse	Horse	9	25-30 years	Big	Alert	1	150000
12	Quarter horse	Horse	7	28-32 years	Big	Active	2	180000
13	Thoroughbred horse	Horse	6	25-30 years	Big	Agile	3	140000
14	Morgan horse	Horse	10	28-32 years	Big	Alert	4	110000

Figure 28: The row has been deleted from the table

Test 7

Objective	To search for a pet with a specific price tag.
Action	The desired price was entered in the text field and the pet for the corresponding price was searched by pressing the search button.
Expected Result	A dialog box should appear that shows the detail of the pet to meets the price tag.
Actual Result	A search result was successfully displayed on searching for a pet with a certain price tag.
Test Remark	Success.

Table 8: Testing the search feature of the application

Pet Name	Category	Age	Lifespan	Size	Nature	Quantity	Price
German Shepherd	Dog	3	12-14 years	Big	Loyal	2	30000
Rottweiler	Dog	5	12-15 years	Big	Aggressive	1	20000
Golden Retriever	Dog	5	12-14 years	Medium	Obidient	3	25000
Japanese Spitz	Dog	6	10-12 years	Small	Loyal	4	15000
Australian shepherd	Dog	4	8-10 years	Medium	Aggressive	6	70000
African golden cat	Cat	5	7-8 years	Small	Inquisitive	8	20000
Black footed cat	Cat	3	6-8 years	Medium	Active	9	17000
Asian leopard cat	Cat	6	7-8 years	Medium	Independent	9	22000
Borneo clouded cat	Cat	2	5-8 years	Small	Playful	9	21000
Caracal cat	Cat	3	6-8 years	Medium	Aggressive	6	16000
Arabian horse	Horse	9	25-30 years	Big	Alert	1	150000
Quarter horse	Horse	7	28-32 years	Big	Active	2	180000
Thoroughbred horse	Horse	6	25-30 years	Big	Agile	3	140000
Morgan horse	Horse	10	28-32 years	Big	Alert	4	110000
Miniature horse	Horse	11	25-30 years	Big	Active	6	140000
Guppy fish	Fish	1	3-5 years	Medium	Reactive	8	7500
Goldfish	Fish	2	5-8 years	Small	Attractive	2	6000
Oscar fish	Fish	3	3-5 years	Medium	Quick to react	1	12000
Trout fish	Fish	4	4-7 years	Medium	Less-active	5	15000
Burbot fish	Fish	3	3-5 years	Small	Attractive	4	8000
Rex Guinea pig	Guinea pig	2	4-8 years	Small	Social	2	3500

Figure 29: Searching for a pet with the price of '180000'

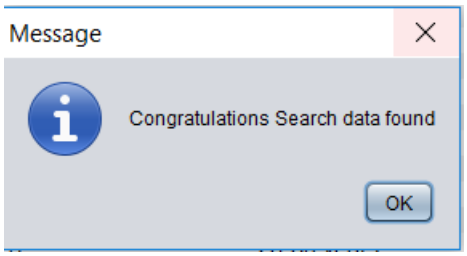


Figure 30: Dialog box informing for the data that has been found

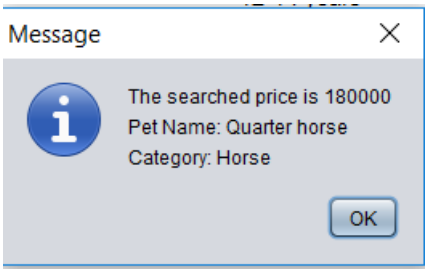


Figure 31: Details of the search result

Test 8

Objective	To search for a category of pet.
Action	A desired category of pet was selected from the JComboBox. Then the button to check the availability was clicked.
Expected Result	A dialog box should appear that displays the total number of pets for that category. Also, the table should get filtered with the right result.
Actual Result	The table was filtered with the correct results and a dialog box appeared stating the total number of pets for the category.
Test Remark	Success.

Table 9: Testing the check availability feature of the application

Pet No.	Pet Name	Category	Age	Lifespan	Size	Nature	Quantity	Price
1	German Shepherd	Dog	3	12-14 years	Big	Loyal	2	30000
2	Rottweiler	Dog	5	12-15 years	Big	Aggressive	1	20000
3	Golden Retriever	Dog	5	12-14 years	Medium	Obidient	3	25000
4	Japanese Spitz	Dog	6	10-12 years	Small	Loyal	4	15000
5	Australian shepherd	Dog	4	8-10 years	Medium	Aggressive	6	70000
6	African golden cat	Cat	5	7-8 years	Small	Inquisitive	8	20000
7	Black footed cat	Cat	3	6-8 years	Medium	Active	9	17000
8	Asian leopard cat	Cat	6	7-8 years	Medium	Independent	9	22000
9	Borneo clouded cat	Cat	2	5-8 years	Small	Playful	9	21000
10	Caracal cat	Cat	3	6-8 years	Medium	Aggressive	6	16000
11	Arabian horse	Horse	9	25-30 years	Big	Alert	1	150000
12	Quarter horse	Horse	7	28-32 years	Big	Active	2	180000

Figure 32: Checking availability for the category 'Dog'

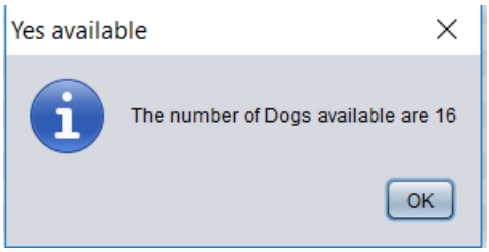


Figure 33: Dialog box showing the number of pets for the category

Pet No.	Pet Name	Category	Age	Lifespan	Size	Nature	Quantity	Price
1	German Shepherd	Dog	3	12-14 years	Big	Loyal	2	30000
2	Rottweiler	Dog	5	12-15 years	Big	Aggressive	1	20000
3	Golden Retriever	Dog	5	12-14 years	Medium	Obidient	3	25000
4	Japanese Spitz	Dog	6	10-12 years	Small	Loyal	4	15000
5	Australian shepherd	Dog	4	8-10 years	Medium	Aggressive	6	70000

Figure 34: Showing the filtered result in the table

Test 9

Objective	To refresh the table to show all the data.
Action	The refresh button was clicked to refresh the data of the table and recover all the unfiltered data.
Expected Result	All the data from the table should be shown.
Actual Result	The table was refreshed, and all the data was shown in the table.
Test Remark	Success.

Table 10: Testing the refresh functionality

Figure 35: Filtered value in the table

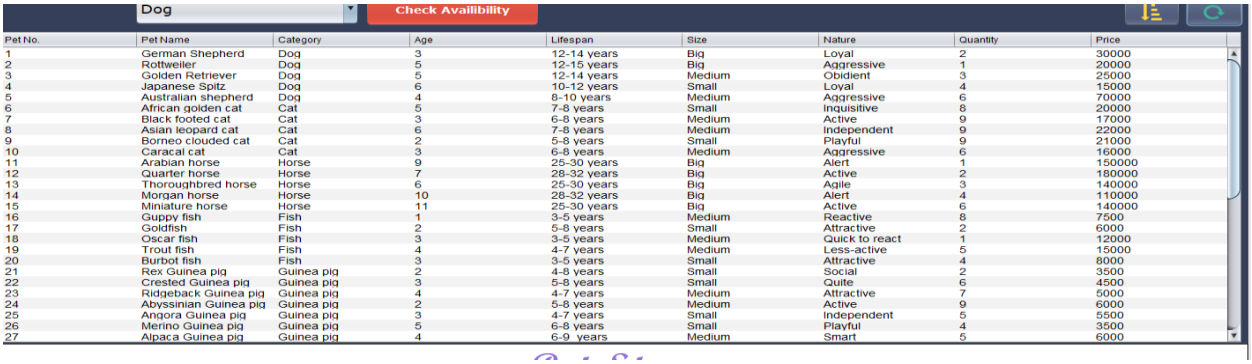
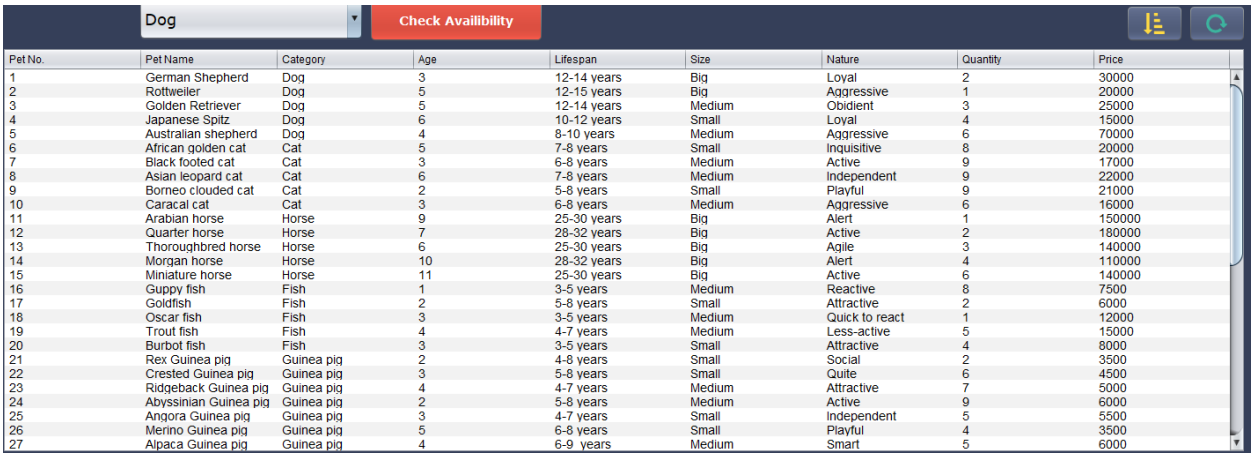


Figure 36: Refreshing the table to see all the value

Test 10

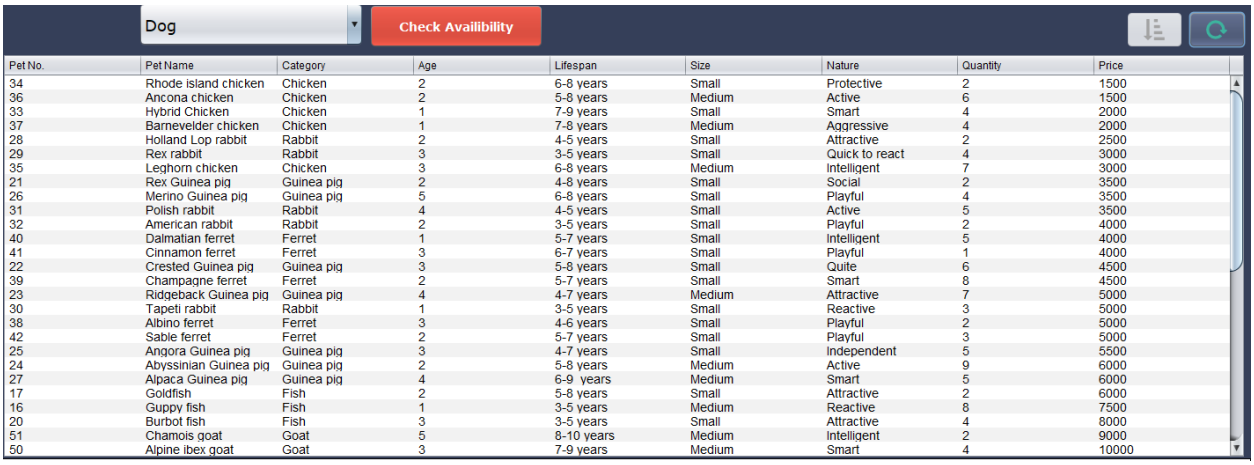
Objective	To sort the data in the table in ascending order with respect to the price of the pet.
Action	The button to sort the table was clicked to sort the data in ascending order.
Expected Result	On clicking the sort button, the table should sort the data in ascending order with respect to the price of the pet.
Actual Result	All the data was successfully sorted.
Test Remark	Success.

Table 11: Testing the sort feature of the application



Pet No.	Pet Name	Category	Age	Lifespan	Size	Nature	Quantity	Price
1	German Shepherd	Dog	3	12-14 years	Big	Loyal	2	30000
2	Rottweiler	Dog	5	12-15 years	Big	Aggressive	1	20000
3	Golden Retriever	Dog	5	12-14 years	Medium	Obidient	3	25000
4	Japanese Spitz	Dog	6	10-12 years	Small	Loyal	4	15000
5	Australian shepherd	Dog	4	8-10 years	Medium	Aggressive	6	70000
6	African golden cat	Cat	5	7-8 years	Small	Inquisitive	8	20000
7	Black footed cat	Cat	3	6-8 years	Medium	Active	9	17000
8	Asian leopard cat	Cat	6	7-8 years	Medium	Independent	9	22000
9	Borneo clouded cat	Cat	2	5-8 years	Small	Playful	9	21000
10	Caracal cat	Cat	3	6-8 years	Medium	Aggressive	6	16000
11	Arabian horse	Horse	9	25-30 years	Big	Alert	1	150000
12	Quarter horse	Horse	7	28-32 years	Big	Active	2	180000
13	Thoroughbred horse	Horse	6	25-30 years	Big	Agile	3	140000
14	Morgan horse	Horse	10	28-32 years	Big	Alert	4	110000
15	Miniature horse	Horse	11	25-30 years	Big	Active	6	140000
16	Guppy fish	Fish	1	3-5 years	Medium	Reactive	8	7500
17	Goldfish	Fish	2	5-8 years	Small	Attractive	2	6000
18	Oscar fish	Fish	3	3-5 years	Medium	Quick to react	1	12000
19	Trout fish	Fish	4	4-7 years	Medium	Less-active	5	15000
20	Burbot fish	Fish	3	3-5 years	Small	Attractive	4	8000
21	Rex Guinea pig	Guinea pig	2	4-8 years	Small	Social	2	3500
22	Crested Guinea pig	Guinea pig	3	5-8 years	Small	Quite	6	4500
23	Ridgeback Guinea pig	Guinea pig	4	4-7 years	Medium	Attractive	7	5000
24	Abyssinian Guinea pig	Guinea pig	2	5-8 years	Medium	Active	9	6000
25	Angora Guinea pig	Guinea pig	3	4-7 years	Small	Independent	5	5500
26	Merino Guinea pig	Guinea pig	5	6-8 years	Small	Playful	4	3500
27	Alpaca Guinea pig	Guinea pig	4	6-9 years	Medium	Smart	5	6000

Figure 37: Before the table is sorted



Pet No.	Pet Name	Category	Age	Lifespan	Size	Nature	Quantity	Price
34	Rhode island chicken	Chicken	2	6-8 years	Small	Protective	2	1500
36	Ancona chicken	Chicken	2	5-8 years	Medium	Active	6	1500
33	Hybrid Chicken	Chicken	1	7-9 years	Small	Smart	4	2000
37	Barnevelder chicken	Chicken	1	7-8 years	Medium	Aggressive	4	2000
28	Holland Lop rabbit	Rabbit	2	4-5 years	Small	Attractive	2	2500
29	Rex rabbit	Rabbit	3	3-5 years	Small	Quick to react	4	3000
35	Leghorn chicken	Chicken	3	6-8 years	Medium	Intelligent	7	3000
21	Rex Guinea pig	Guinea pig	2	4-8 years	Small	Social	2	3500
26	Merino Guinea pig	Guinea pig	5	6-8 years	Small	Playful	4	3500
31	Polish rabbit	Rabbit	4	4-5 years	Small	Active	5	3500
32	American rabbit	Rabbit	2	3-5 years	Small	Playful	2	4000
40	Dalmatian ferret	Ferret	1	5-7 years	Small	Intelligent	5	4000
41	Cinnamon ferret	Ferret	3	6-7 years	Small	Playful	1	4000
22	Crested Guinea pig	Guinea pig	3	5-8 years	Small	Quite	6	4500
39	Champagne ferret	Ferret	2	5-7 years	Small	Smart	8	4500
23	Ridgeback Guinea pig	Guinea pig	4	4-7 years	Medium	Attractive	7	5000
30	Tapeti rabbit	Rabbit	1	3-5 years	Small	Reactive	3	5000
38	Albino ferret	Ferret	3	4-6 years	Small	Playful	2	5000
42	Sable ferret	Ferret	2	5-7 years	Small	Playful	3	5000
25	Angora Guinea pig	Guinea pig	3	4-7 years	Small	Independent	5	5500
24	Abyssinian Guinea pig	Guinea pig	2	5-8 years	Medium	Active	9	6000
27	Alpaca Guinea pig	Guinea pig	4	6-9 years	Medium	Smart	5	6000
17	Goldfish	Fish	2	5-8 years	Small	Attractive	2	6000
16	Guppy fish	Fish	1	3-5 years	Medium	Reactive	8	7500
20	Burbot fish	Fish	3	3-5 years	Small	Attractive	4	8000
51	Chamois goat	Goat	5	8-10 years	Medium	Intelligent	2	9000
50	Alpine ibex goat	Goat	3	7-9 years	Medium	Smart	4	10000

Figure 38: After sorting the table

Test 11

Objective	To successfully login into the application.
Action	The login button was clicked and then the password was entered in the input dialog box.
Expected Result	After entering the password for the authorization, the user is granted with all the features of the application.
Actual Result	The user was successfully logged in.
Test Remark	Success.

Table 12: Testing the privilege system of the application

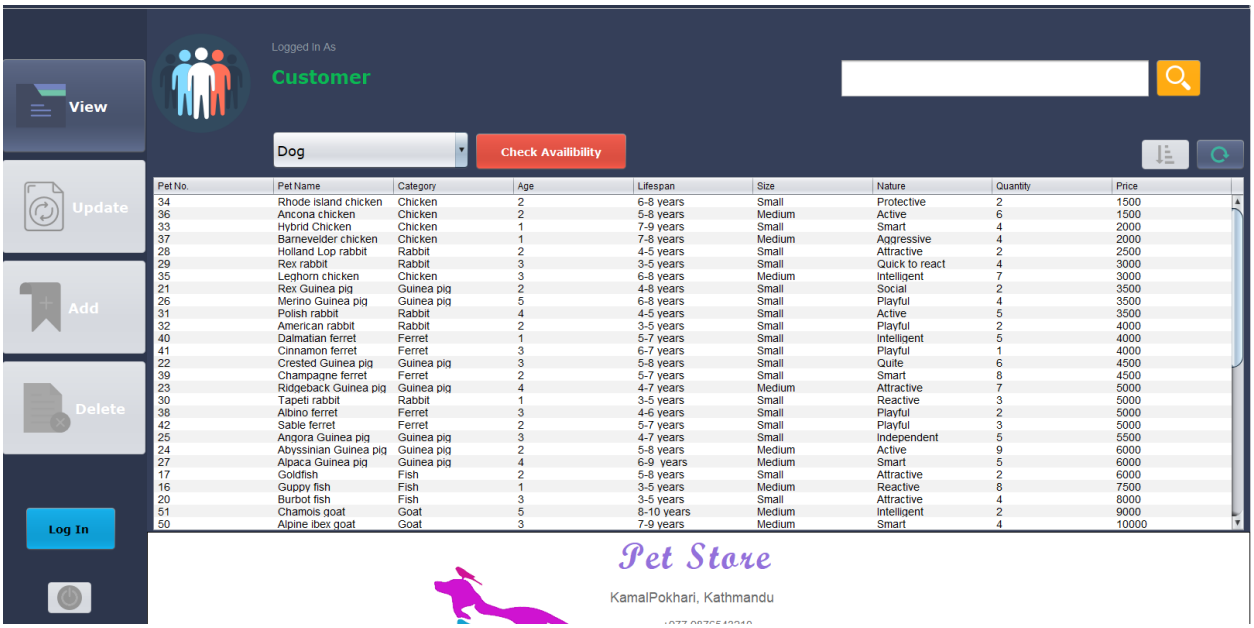


Figure 39: Dashboard before logging in

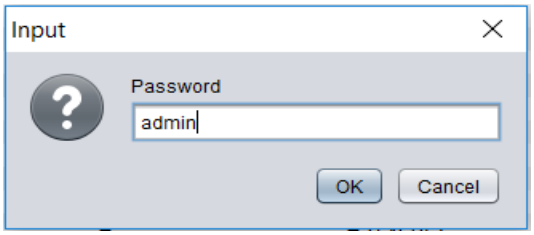
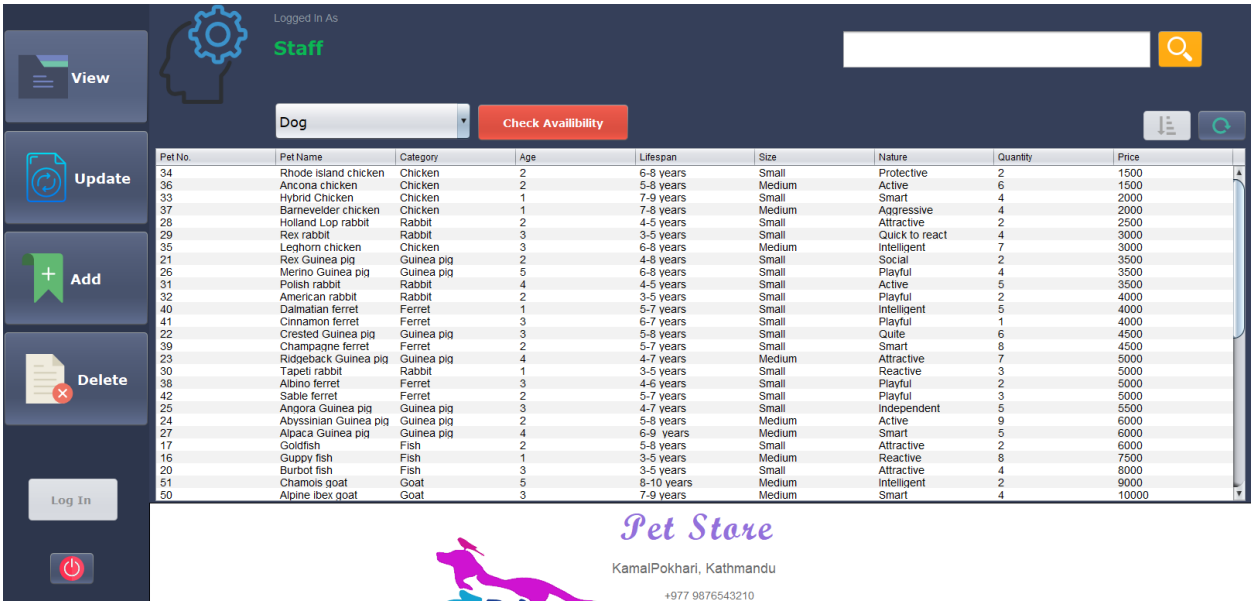


Figure 40: Putting the password for authorization



Pet No.	Pet Name	Category	Age	Lifespan	Size	Nature	Quantity	Price
34	Rhode island chicken	Chicken	2	6-8 years	Small	Protective	2	1500
36	Ancona chicken	Chicken	2	5-8 years	Medium	Active	6	1500
33	Hybrid Chicken	Chicken	1	7-9 years	Small	Smart	4	2000
37	Barnevelder chicken	Chicken	1	7-8 years	Medium	Aggressive	4	2000
28	Holland Lop rabbit	Rabbit	2	4-5 years	Small	Attractive	2	2500
29	Rex rabbit	Rabbit	3	3-5 years	Small	Quick to react	4	3000
35	Leghorn chicken	Chicken	3	6-8 years	Medium	Intelligent	7	3000
21	Rex Guinea pig	Guinea pig	2	4-8 years	Small	Social	2	3500
26	Merino Guinea pig	Guinea pig	5	6-8 years	Small	Playful	4	3500
31	Polish rabbit	Rabbit	4	4-5 years	Small	Active	5	3500
32	American rabbit	Rabbit	2	3-5 years	Small	Playful	2	4000
40	Dalmatian ferret	Ferret	1	5-7 years	Small	Intelligent	5	4000
41	Cinnamon ferret	Ferret	3	6-7 years	Small	Playful	1	4000
22	Crested Guinea pig	Guinea pig	3	5-8 years	Small	Quite	6	4500
39	Champagne ferret	Ferret	2	5-7 years	Small	Smart	8	4500
23	Ridgeback Guinea pig	Guinea pig	4	4-7 years	Medium	Attractive	7	5000
30	Tapeti rabbit	Rabbit	1	3-5 years	Small	Reactive	3	5000
38	Albino ferret	Ferret	3	4-6 years	Small	Playful	2	5000
42	Sable ferret	Ferret	2	5-7 years	Small	Playful	3	5000
25	Angora Guinea pig	Guinea pig	3	4-7 years	Small	Independent	5	5500
24	Abyssinian Guinea pig	Guinea pig	2	5-8 years	Medium	Active	9	6000
27	Alpaca Guinea pig	Guinea pig	4	6-9 years	Medium	Smart	5	6000
17	Goldfish	Fish	2	5-8 years	Small	Attractive	2	6000
16	Guppy fish	Fish	1	3-5 years	Medium	Reactive	8	7500
20	Burbot fish	Fish	3	3-5 years	Small	Attractive	4	8000
51	Chamois goat	Goat	5	8-10 years	Medium	Intelligent	2	9000
50	Alpine ibex goat	Goat	3	7-9 years	Medium	Smart	4	10000

Figure 41: Dashboard after logging in

Test 12

Objective	To check if the empty text field gives an error.
Action	A text field was left empty for the add/update form and the input dialog box for the password then the respective buttons were pressed.
Expected Result	A dialog box should appear with an appropriate message.
Actual Result	A dialog box appeared with an appropriate message.
Test Remark	Success.

Table 13: Testing the error handling of the input dialog box of login

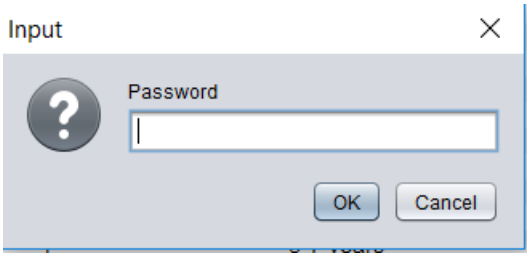


Figure 42: Leaving the input dialog box empty

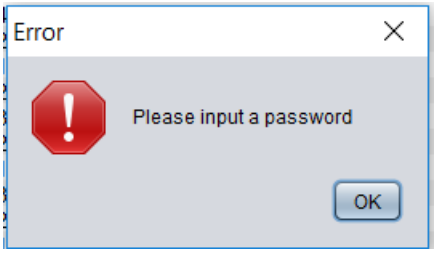


Figure 43: Error message for the empty input dialog box

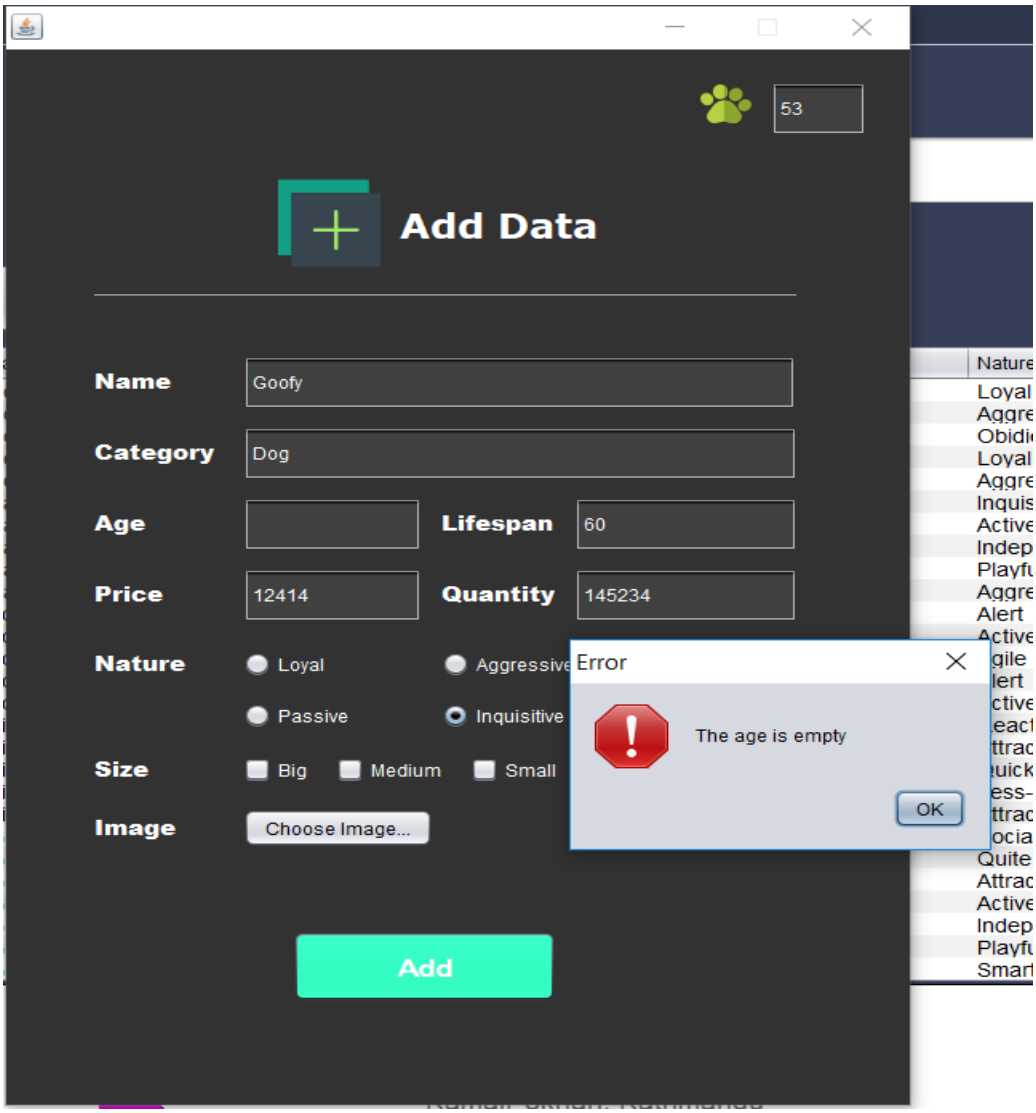


Figure 44: Error message for an empty text field in the add form

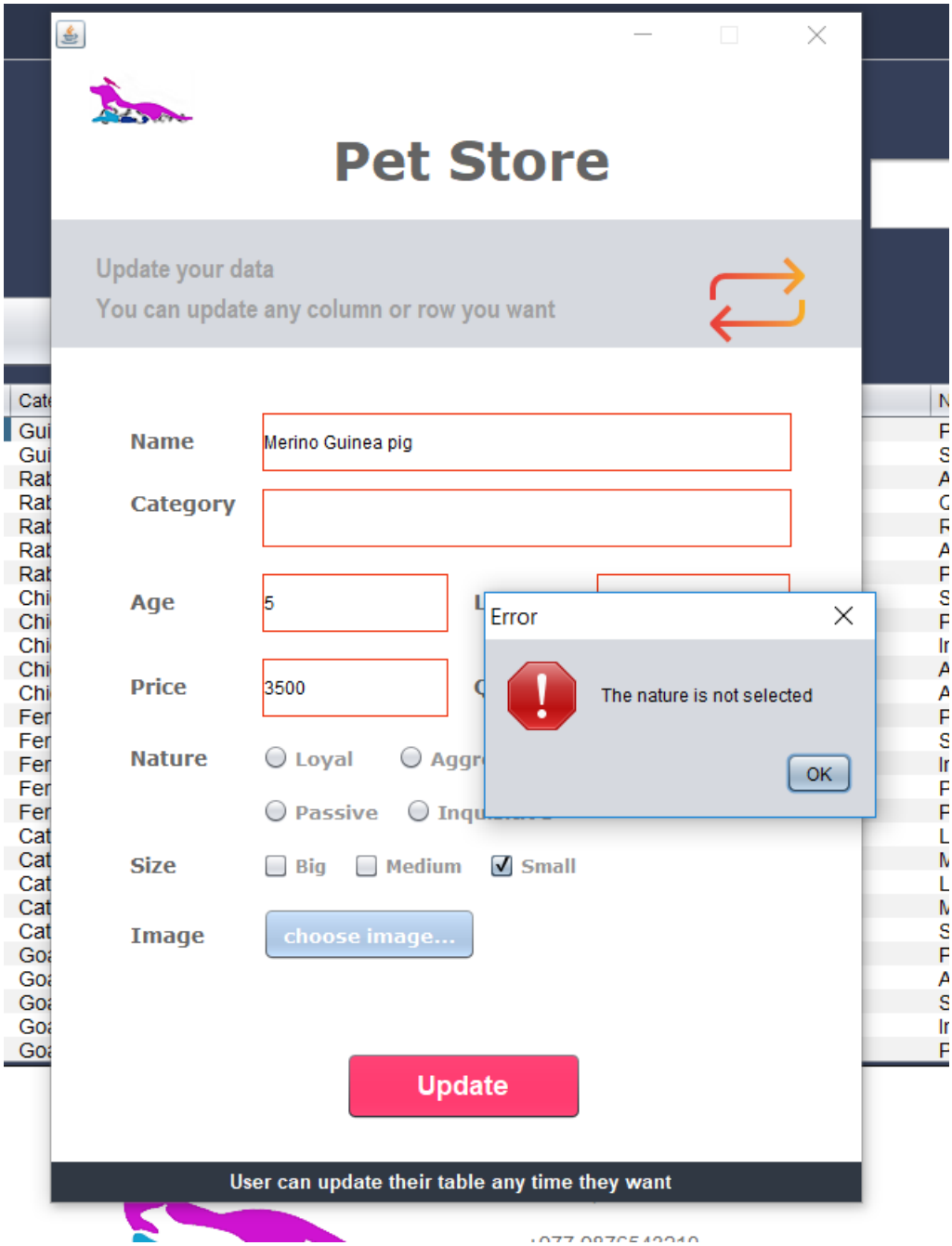


Figure 45: Error message for an empty text field in the update form

Test 13

Objective	Adding a value that already exists.
Action	A name of the pet that already exists was tried to be added.
Expected Result	A dialog box should appear with an appropriate message.
Actual Result	A dialog box appeared with an appropriate message.
Test Remark	Success.

Table 14: Testing if the add form takes in duplicate values

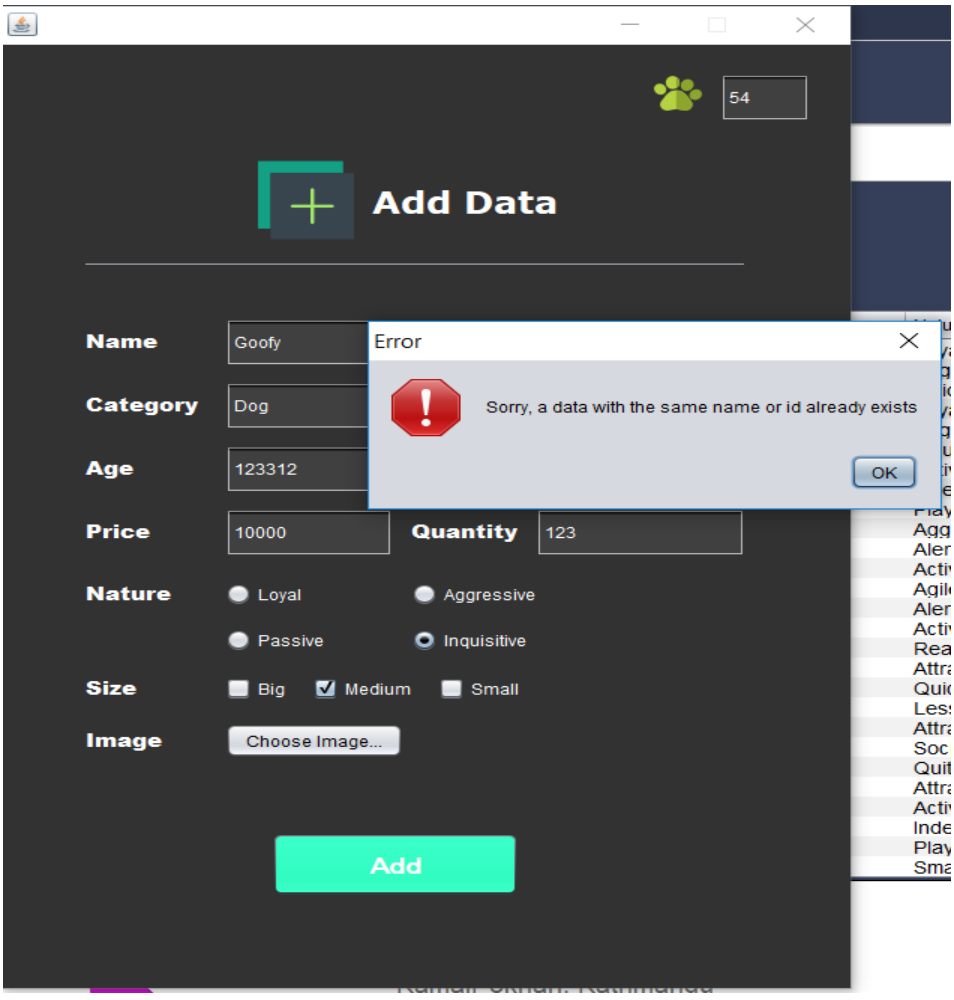


Figure 46: Error message for a duplicate value

Test 14

Objective	Leaving the search field empty
Action	The search button was pressed without entering any value.
Expected Result	A dialog box should appear with an appropriate message.
Actual Result	A dialog box appeared with an appropriate message.
Test Remark	Success.

Table 15: Testing the error handling of the search field

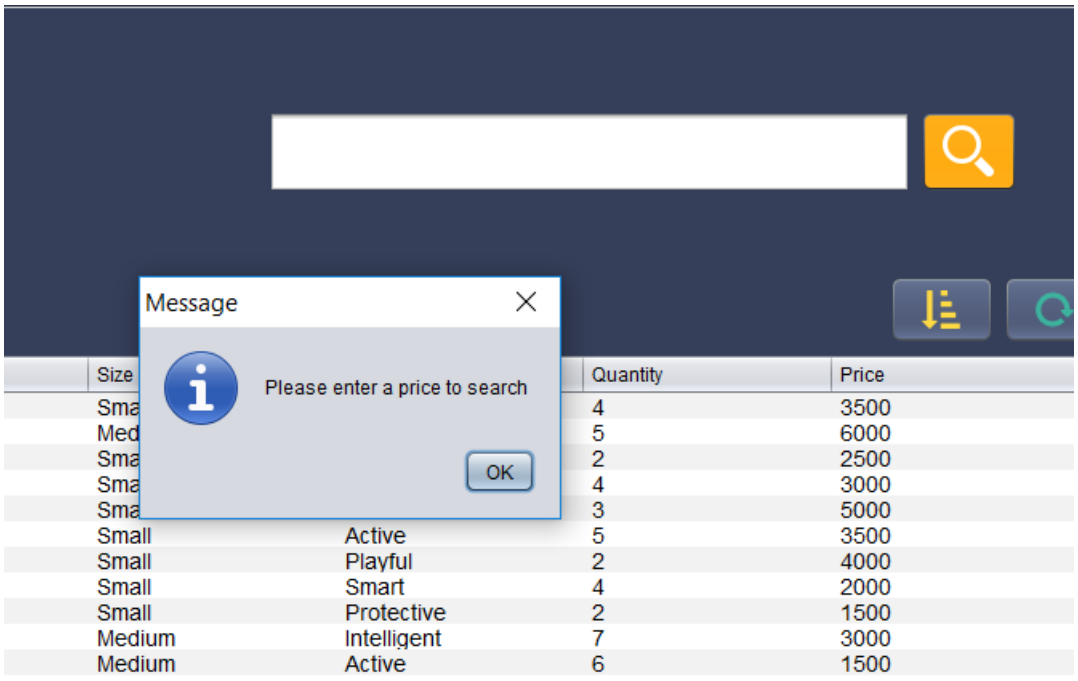


Figure 47: Dialog message for the empty search field

Test 15

Objective	Trying to search for a category that does not exists
Action	A category was selected that does not exists.
Expected Result	A dialog box should appear with an appropriate message.
Actual Result	A dialog box appeared with an appropriate message.
Test Remark	Success.

Table 16: Testing if the non-existent category is checked in the application

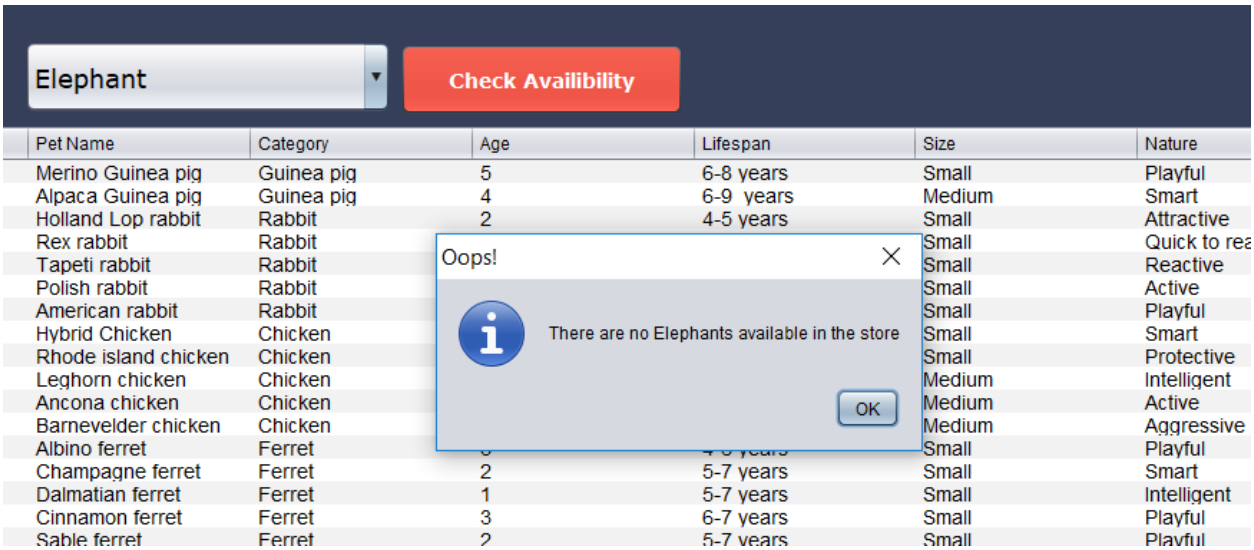


Figure 48: Dialog message for the non-existing category

All these testing shows the robust and reliable nature of the program. This test may, however, lack some of the data validation during user input. This is because the application has restricted certain keystrokes while entering the values in the text fields. Additionally, this testing helped us recognize some of the flaws which lead to further understanding of why they occurred.

7. Tools used

a. Java

We used Java to make this application mainly because it is an independent platform. Java is robust, portable and is secured with the use of JVM. Since it is an object-oriented language the codes are easier to maintain. It is quite simple to use due to the lack of pointers and has an automatic garbage collector. Java also has a rich set of libraries for common utilities. Moreover, Java also has plenty of documentation that helps to explore different possibilities in the language.

b. NetBeans

NetBeans was used to develop this application because it is one of the primary IDE for Java development in the industry. Creating GUI using swing component in NetBeans is easier due to Swing Application Framework and Beans Binding. Therefore, less time is consumed in the designing a modern and powerful user interface. NetBeans also includes a vibrant community that has active and expert members available for help.

c. Illustrator

We used illustrator to create the logo for the application. Illustrator is a vector-based designing application which allows for creating clean and attractive logos with great accuracy and precision. Since Illustrator is a vector-based application, the logos do not lose its' quality that helps to produce high-quality logos and other graphical content. This also makes the application more extensible than others.

d. Mock Flow

Mock Flow is an online tool that was used to design the prototype of the GUI. Mock Flow includes built-in features allowing us to select pack design more of a clear-cut wireframe in just a few minutes. All the elements are drag and drop making it more convenient to visualize the wireframe.

Conclusion

The delivered application demonstrates an information system that uses the components of extracting raw data, processing the data and the displaying the processed data for the user to work with. Besides that, this application allows the user to search and filter that helps to organize and distribute the data. The entire development of the application is concentrated on the realization of small scaled operation that wants to manage and organize its data in a simple way.

During the tenure of this project, the whole group has contributed a lot in making this application robust and scalable. In this project, we have worked into key concepts of using the quick and easy use of the NetBeans swing components, object-oriented-programming and modular design for the application. This group work has led us to a better understanding of using comprehensive data structures used in our application such as arrays, a two-dimensional array, and other primitive arrays. Learning and executing different data structures ultimately helped us a lot. We have used a binary search and selection sort algorithm which was one of the primary focus of the group work and the development process. The implementation of such algorithms helps to make the task more robust and quicker.

Throughout the course of the group work, the team has also faced many challenges where the situation cropped up and tested us with many difficulties. One of the many advantages of the group is that individual can consult and help one other whenever necessary which helped to solve most of the problems. The divisions of the work have helped exponentially in completing the tasks quickly and more efficiently. The development and the reporting for this group work were divided with mutual consent and considering the capacity of each individual. This has immensely helped in the development of this group work.

In today's work culture especially in the field of programming group work is valued a lot. Interacting and coordinating with another individual can be difficult and hard to manage. However, this group work has helped us to identify the importance of collaboration and working together. Working together has not only helped us to complete our work easily but has also helped us to learn from one another in an interactive learning environment. Therefore, this group work has helped us to enhance our knowledge and competencies level among ourselves.

References

Anon. (n.d.) *coderanch* [Online]. Available from: https://coderanch.com/t/379046/java/Image-smoothing [Accessed 9 January 2019].

Anon. (n.d.) *Stack Overflow* [Online]. Available from: https://stackoverflow.com/questions/11951123/button-opens-new-jframe-multiple-times-how-do-i-stop-this [Accessed 10 January 2019].

Bourgeois, D. & Bourgeois, D.T. (2014) *Information Systems for Business and Beyind*. ebook ed. Saylor Foundation.

Khan Academy. (2019) *selection-sort-pseudocode* [Online]. Available from: https://www.khanacademy.org/computing/computer-science/algorithms/sorting-algorithms/a/selection-sort-pseudocode [Accessed 11 January 2019].

Limiye, M.G. (2009) *Software Testing*. Tata McGraw-Hill Education, 2009.

Poo, D., Kiong, D. & Ashok, S. (2007) *Object-Oriented Programming and Java*. 2nd ed. Springer Science & Business Media.

Shah, A. (2017) *Crunchify* [Online]. Available from: https://crunchify.com/how-to-read-convert-csv-comma-separated-values-file-to-arraylist-in-java-using-split-operation/?fbclid=IwAR2FNZi86kP94uSC7GIqMXAxi6Q_i7rEQmbnXPSzoQ2hMrFvg8UQVdBL2fQ [Accessed 10 January 2019].

Study Tonight. (2019) *selection-sorting* [Online]. Available from: https://www.studytonight.com/data-structures/selection-sorting [Accessed 11 January 2019].

Bibliography

Anon. (n.d.) *coderanch* [Online]. Available from: https://coderanch.com/t/379046/java/Image-smoothing [Accessed 9 January 2019].

Anon. (n.d.) *Stack Overflow* [Online]. Available from: https://stackoverflow.com/questions/11951123/button-opens-new-jframe-multiple-times-how-do-i-stop-this [Accessed 10 January 2019].

Bourgeois, D. & Bourgeois, D.T. (2014) *Information Systems for Business and Beyind*. ebook ed. Saylor Foundation.

Khan Academy. (2019) *selection-sort-pseudocode* [Online]. Available from: https://www.khanacademy.org/computing/computer-science/algorithms/sorting-algorithms/a/selection-sort-pseudocode [Accessed 11 January 2019].

Limiye, M.G. (2009) *Software Testing*. Tata McGraw-Hill Education, 2009.

Poo, D., Kiong, D. & Ashok, S. (2007) *Object-Oriented Programming and Java*. 2nd ed. Springer Science & Business Media.

Shah, A. (2017) *Crunchify* [Online]. Available from: https://crunchify.com/how-to-read-convert-csv-comma-separated-values-file-to-arraylist-in-java-using-split-operation/?fbclid=IwAR2FNZi86kP94uSC7GIqMXAxi6Q_i7rEQmbnXPSzoQ2hMrFvg8UQVdBL2fQ [Accessed 10 January 2019].

Study Tonight. (2019) *selection-sorting* [Online]. Available from: https://www.studytonight.com/data-structures/selection-sorting [Accessed 11 January 2019].

Appendix

MenuInfo Class Code

```
/*
 * 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 Dashboard;

import java.awt.event.KeyEvent;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ComboBoxModel;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author Dell
 */
public class MenuInfo extends javax.swing.JFrame {

    /**
     * Creates new form Main
     */
    public static ArrayList<ArrayList<String>> array;
    private int mousepX, mousepY;
    public static DefaultTableModel model;
    private String filePath;
    public static ArrayList<String> newCategory = new ArrayList<>();
    public static String drive;

    public MenuInfo() {
        initComponents();
    }

```

```
43      /**
44       * return the object of the table
45       *
46       * @return
47       */
48      public int getRowCount() {
49
50          return model.getRowCount();
51      }
52
53      /**
54       * This method is called from within the constructor to initialize the form.
55       * WARNING: Do NOT modify this code. The content of this method is always
56       * regenerated by the Form Editor.
57       */
58      @SuppressWarnings("unchecked")
59      Generated Code
608
609     /**
610      * Closes the application
611      *
612      * @param evt
613      */
614     private void exitActionPerformed(java.awt.event.ActionEvent evt) {
615         //Exiting the application
616         System.exit(0);
617     }
618
619     /**
620      * This method minimizes the application
621      *
622      * @param evt
623      */
624     private void minimizeActionPerformed(java.awt.event.ActionEvent evt) {
625         //minmize
626         this.setState(MenuInfo.ICONIFIED);
627     }
```

```
629     /**
630      * This method accepts the category of a pet that is newly added or update
631      * It then checks if the category already exists If not it adds the new
632      * category into an array to be added to JComboBox.
633      *
634      * @param category_
635      */
636     public void checkCategory(String category_) {
637
638         int count = 0;
639         try {
640             for (int i = 0; i < array.size(); i++) {
641                 if (category_.equals(array.get(i).get(2))) {
642                     count += 1;
643                 }
644             }
645
646             if (count == 1) {
647                 MenuInfo.newCategory.add(category_);
648
649             }
650
651         } catch (NullPointerException e) {
652             JOptionPane.showMessageDialog(null, "Wow", "Success!!", JOptionPane.INFORMATION_MESSAGE);
653         }
654     }
655
656     /**
657      * This method is called from AddData class After the user inserts value
658      * into the add form all the values has be to be added into the table It
659      * creates a new row and adds the new row
660      */
661     public void addData() {
662
663         model.addRow(new String[]{AddData.data.get(0), AddData.data.get(1), AddData.data.get(2), AddData.data.get(3), AddData.data.get(4), AddData.data.get(5), AddData.data.get(6), AddData.data.get(7), AddData.data.get(8)});
664
665         JOptionPane.showMessageDialog(null, "The new row has been added. Please refresh the table to see the changes in  category", "Success!!", JOptionPane.INFORMATION_MESSAGE);
666         AddData.getObj().setVisible(false);
667     }
```

```
669     /**
670      * This is the add button On clicking this method the add form is visible
671      *
672      * @param evt
673      */
674     private void addActionPerformed(java.awt.event.ActionEvent evt) {
675
676         try {
677
678             AddData.getObj().setVisible(true);
679             AddData.getObj().setIdDefault();
680
681             refresh();
682             model.setRowCount(0);
683             addToTable();
684
685         } catch (NullPointerException e) {
686             JOptionPane.showMessageDialog(null, "The table is empty.", "Error", JOptionPane.ERROR_MESSAGE);
687             AddData.getObj().setVisible(false);
688         } catch (FileNotFoundException ex) {
689             Logger.getLogger(MenuInfo.class.getName()).log(Level.SEVERE, null, ex);
690         }
691
692     }
693
694     /**
695      * this method calls the object of the Details class and makes it visible
696      * then it calls the viewDetails() method which accepts the pet id and the
697      * two dimensional arrayList
698      *
699      * @param evt
700      */
```

```
701     private void view_detailsActionPerformed(java.awt.event.ActionEvent evt) {
702         //View details
703         int rowIndex_ = 1;
704         int id = 0;
705         try {
706
707             if (petTable.getCellSelectionEnabled()) {
708                 //petTable.setSelectionModel(ListSelectionModel.SINGLE_SELECTION);
709                 rowIndex_ = petTable.getSelectedRow();
710                 id = Integer.parseInt(model.getValueAt(rowIndex_, 0).toString());
711             }
712
713             Details.getObj().viewDetails(id, array);
714             Details.getObj().setVisible(true);
715
716         } catch (NullPointerException ex) {
717             JOptionPane.showMessageDialog(null, "No records found.", "Error", JOptionPane.ERROR_MESSAGE);
718         } catch (ArrayIndexOutOfBoundsException ex) {
719             JOptionPane.showMessageDialog(null, "You have to select a row to view the details.", "Error", JOptionPane.INFORMATION_MESSAGE);
720         }
721     }
722
723     /**
724      * this methods takes in the pet id to be deleted then it deletes all the
725      * values for the corresponding id from the arrayList and the table
726      *
727      * @param evt
728      */
```

```
729     private void delActionPerformed(java.awt.event.ActionEvent evt) {
730         String id = JOptionPane.showInputDialog("Enter pet id");
731         //try {
732         int i;
733         boolean match = false;
734
735         for (i = 0; i < array.size(); i++) {
736             String petId = array.get(i).get(0);
737             if (petId.equals(id)) {
738                 JOptionPane.showMessageDialog(null, "The pet named " + array.get(i).get(1) + " has been deleted.", "Success", JOptionPane.INFORMATION_MESSAGE);
739                 array.remove(i);
740                 refresh();
741                 model.removeRow(getRowCount() - 1);
742                 match = true;
743                 break;
744             }
745
746         }
747         if (match == false) {
748             JOptionPane.showMessageDialog(null, "please enter proper data to delete", "Error", JOptionPane.INFORMATION_MESSAGE);
749         }
750
751         //} //catch (NullPointerException e) {
752         //JOptionPane.showMessageDialog(null, "Looks like the table is empty. Please import a csv file into the table", "Error", JOptionPane.ERROR_MESSAGE);
753         //}
754     }
```

```
755     /**
756      * This is for the privilege system At default the add, delete and update
757      * function in the application is disabled When trying to login a
758      * JOptionPane follows up that asks for the password On providing with the
759      * correct password the aforementioned functionalities are unlocked This is
760      * for the super-user or the staff
761      *
762      * @param evt
763      */
764     private void adminActionPerformed(java.awt.event.ActionEvent evt) {
765         //Login as Admin
766         try {
767             String pass = JOptionPane.showInputDialog("Password");
768 
769             if (pass.equals("admin")) {
770 
771                 add.setEnabled(true);
772                 del.setEnabled(true);
773                 update.setEnabled(true);
774                 log_out.setEnabled(true);
775                 user_logo.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Dashboard/images/admin.png")));
776 
777                 user_name.setText("Staff");
778                 admin.setEnabled(false);
779 
780             } else if (pass.length() <= 0) {
781                 JOptionPane.showMessageDialog(null, "Please input a password", "Error", JOptionPane.ERROR_MESSAGE);
782             } else {
783                 JOptionPane.showMessageDialog(null, "Wrong Password", "Error", JOptionPane.ERROR_MESSAGE);
784             }
785         } catch (NullPointerException e) {
786             JOptionPane.showMessageDialog(null, "Please input a password", "Error", JOptionPane.ERROR_MESSAGE);
787 
788         }
```

```
792     /**
793      * The following series of methods (exitMouseMoved and exitMouseExited) is
794      * for the exit button This is to create a hover effect When the mouse is
795      * moved over the button the color changes to red and the when the mouse is
796      * moved the color changes back
797      *
798      * @param evt
799      */
800     private void exitMouseMoved(java.awt.event.MouseEvent evt) {
801         exit.setBackground(new java.awt.Color(255, 34, 56));
802     }
803 
804     private void exitMouseDragged(java.awt.event.MouseEvent evt) {
805         // TODO add your handling code here:
806     }
807 
808     private void exitMouseEntered(java.awt.event.MouseEvent evt) {
809 
810     }
811 
812     private void exitMouseExited(java.awt.event.MouseEvent evt) {
813         exit.setBackground(new java.awt.Color(45, 54, 76));
814     }
```

```
815     /**
816      * This method is for the staff to logout After logging out the customer
817      * user cannot update, delete or add to the data
818      *
819      * @param evt
820      */
821     private void log_outActionPerformed(java.awt.event.ActionEvent evt) {
822         //logout
823 
824         // asks the user if he/she wants to logout with a dialog box
825         Object[] options = {"Yes", "No"};
826         int n = JOptionPane.showOptionDialog(null,
827                 "Are you sure, you want to log out?",
828                 "Are you sure?",
829                 JOptionPane.YES_NO_CANCEL_OPTION,
830                 JOptionPane.QUESTION_MESSAGE,
831                 null,
832                 options,
833                 options[1]);
834 
835         //when the user clicks on yes it returns a 0
836         //On clicking no it return 1
837         if (n == 0) {
838             add.setEnabled(false);
839             del.setEnabled(false);
840             log_out.setEnabled(false);
841             update.setEnabled(false);
842 
843             admin.setEnabled(true);
844             user_logo.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Dashboard/images/user.png")));
845             user_name.setText("Customer");
846         }
847     }
```

```
850      * Utility which converts CSV to ArrayList using Split Operation
851      *
852      * @param CSV
853      * @return
854      */
855     private ArrayList<String> CSVtoArrayList(String CSV) {
856         ArrayList<String> Result = new ArrayList<String>();
857 
858         if (CSV != null) {
859             String[] splitData = CSV.split("\\s*,\\s*");
860             for (int i = 0; i < splitData.length; i++) {
861                 if (!(splitData[i] == null) || !(splitData[i].length() == 0)) {
862                     Result.add(splitData[i].trim());
863                 }
864 
865             }
866         }
867 
868         return Result;
869     }
```

```
871     /**
872      * this method adds all the data from the CSV into the main two dimensional
873      * arrayList @array it reads the csv file using the bufferedReader class
874      * which reads each of the line in the csv file then each of the line is
875      * added to the arrayList
876      */
877     private void addToArray() {
878         BufferedReader Buffer = null;
879 
880         try {
881             String Line;
882             Buffer = new BufferedReader(new FileReader(filePath));
              MenuInfo.array = new ArrayList<ArrayList<String>>();
884             Buffer.readLine();
885             // How to read file in java line by line?
886             while ((Line = Buffer.readLine()) != null) {
887                 MenuInfo.array.add(CSVtoArrayList(Line));
888             }
889 
890 
891         } catch (IOException e) {
892         } finally {
893             try {
894                 if (Buffer != null) {
895                     Buffer.close();
896                 }
897             } catch (IOException Exception) {
898             }
899         }
900     }
```

```
902     /**
903      * This method adds the data from the two dimensional array @array into the
904      * table When a CSV file is imported for the first time the sort button,
905      * searchPrice button, search text field are enabled
906      *
907      * @throws FileNotFoundException
908      */
909     private void addToTable() throws FileNotFoundException {
910         for (int i = 0; i < array.size(); i++) {
911             //The delimeter in a csv file is ","
912             String[] row = new String[]{array.get(i).get(0),
913                 array.get(i).get(1),
914                 array.get(i).get(2),
915                 array.get(i).get(3),
916                 array.get(i).get(4),
917                 array.get(i).get(5),
918                 array.get(i).get(6),
919                 array.get(i).get(7),
920                 array.get(i).get(8)};
921             model.addRow(row);
922 
923         }
924 
925         if (model.getRowCount() > 0) {
926             sortPricel.setEnabled(true);
927             search.setEnabled(true);
928             searchPrice.setEnabled(true);
929         }
930     }
```

```
932     /**
933      * This method adds all the unique category from a CSV file in the JComboBox
934      */
935     private void addCategory() {
936         String value1, value2;
937 
938         int i, j;
939         for (i = 0; i < model.getRowCount(); i++) {
940             value1 = model.getValueAt(i, 2).toString();
941             for (j = 0; j < i; j++) {
942                 value2 = model.getValueAt(j, 2).toString();
943                 if (value1.equals(value2)) {
944                     break;
945                 }
946             }
947 
948             if (i == j) {
949                 category.addItem(value1);
950 
951             }
952         }
953 
954         ComboBoxModel model = category.getModel();
955         int size = model.getSize();
956 
957         for (int k = 0; k < size; k++) {
958             Object element = model.getElementAt(k);
959 
960         }
961     }
```

```
963     /**
964      * This method is allows the user to import a CSV file. File explorer is
965      * used to import the CSV file Also checks for the extension type Doesn't
966      * allow user to import file of any other type except for CSV. An object for
967      * table is created and all the values are set in the table
968      *
969      * @param evt
970      */
971     private void import_csvActionPerformed(java.awt.event.ActionEvent evt) {
972         // TODO add your handling code here:
973         model = (DefaultTableModel) petTable.getModel();
974         category.removeAllItems();
975         if (model != null) {
976             model.setRowCount(0);
977         }
978         try {
979             if (array != null) {
980                 array.clear();
981             }
982 
983             //Using file explorer to get the file path
984             JFileChooser file = new JFileChooser();
985             file.showOpenDialog(null);
986             File f = file.getSelectedFile();
987             filePath = f.getAbsolutePath();
988 
989 
990             int iend = filePath.indexOf("MenuIS"); //this is done so that the image can be viewed no matter where the
991             drive = filePath.substring(0, iend);
992 
993 
994             String extension = "";
995             //checking if the file format is csv or not
996             int j = filePath.lastIndexOf('.');
997             if (j >= 0) {
998                 extension = filePath.substring(j + 1);
999             }
```

```
            if (extension.equals("csv")) {
                //String filePath = "C:\\Users\\Dell\\Desktop\\a.csv";

                //adding all values to table
                addToArray();
                addToTable();
                addCategory();

            } else {
                JOptionPane.showMessageDialog(null, "Please open a proper csv file");
                import_csvActionPerformed(evt);
            }
        } catch (NullPointerException ex) {
            JOptionPane.showMessageDialog(null, "No file selected.", "Info", JOptionPane.INFORMATION_MESSAGE);
            category.addItem("<None>");
        } catch (FileNotFoundException ex) {
            JOptionPane.showMessageDialog(null, "File not found.", "Error", JOptionPane.ERROR_MESSAGE);
            category.addItem("<None>");
        } catch (StringIndexOutOfBoundsException e) {
            JOptionPane.showMessageDialog(null, "You can import the csv file only if it is inside the project file.", "Error", JOptionPane.ERROR_MESSAGE);
            category.addItem("<None>");
        }
    }

    /**
     * this method is called by the
     * availabilityActionPerformed(java.awt.event.ActionEvent evt) it filters
     * all the data for the category and then only shows the filtered result in
     * the table
     *
     * @param array
     */
    private void updateTable(String[][] array) {
        for (int i = 0; i < array.length; i++) {
            model.addRow(new Object[]{null});
            for (int j = 0; j < array[0].length; j++) {
                model.setValueAt(array[i][j], i, j);
            }
        }
        availability.setEnabled(false);
    }

    /**
     * This method is for exiting the application from the menu
     *
     * @param evt
     */
    private void menu_exitActionPerformed(java.awt.event.ActionEvent evt) {
        //Exiting (Alt+F4)
        System.exit(0);
    }

    /**
     * The method windowMouseDragged and windowMousePressed allows the user to
     * drag the undecorated JFrame freely
     *
     * @param evt
     */
    private void windowMouseDragged(java.awt.event.MouseEvent evt) {
        // Moving the undecorated JFrame
        int cordinateX = evt.getXOnScreen();
        int cordinateY = evt.getYOnScreen();

        this.setLocation(cordinateX - mousepX, cordinateY - mousepY);
    }

    /**
     * This method gives the coordinates of the working screen on clicking the
     * frame with the cursor
     *
     * @param evt
     */
    private void windowMousePressed(java.awt.event.MouseEvent evt) {
        // Moving the undecorated JFrame
        mousepX = evt.getX();
        mousepY = evt.getY();
    }
```

```java
    /**
     * this method is called by
     * availabilityActionPerformed(java.awt.event.ActionEvent evt) it checks the
     * category and then adds the quantity of all the pets for that category it
     * then returns a boolean flag
     *
     * @param checkItem
     * @param row
     * @param quantity
     * @param columnCount
     * @param rowCount
     * @param available
     * @return
     */
    private boolean categorySelection(String checkItem, int row, int quantity, int columnCount, int rowCount, String[][] available) {
        boolean flag;
        for (int i = 0; i < rowCount; i++) {
            if (model.getValueAt(i, 2).equals(checkItem)) {
                row++;
                quantity += Integer.parseInt(model.getValueAt(i, 7).toString());
                for (int j = 0; j < columnCount; j++) {
                    available[row][j] = model.getValueAt(i, j).toString();
                }
            }

        }

        if (quantity > 0) {
            JOptionPane.showMessageDialog(null, "The number of " + checkItem + "s available are " + quantity, "Yes available", JOptionPane.INFORMATION_MESSAGE);
            model.setRowCount(0);
            flag = true;
        } else {
            JOptionPane.showMessageDialog(null, "There are no " + checkItem + "s available in the store", "Oops!", JOptionPane.INFORMATION_MESSAGE);
            flag = false;
        }

        return flag;
    }
```

```java
    /**
     * this is the method for the check availability button this method checks
     * for all the category that is selected in the JComboBox for the all the
     * selected category the quantity of the pet is added then it calls the
     * categorySelection() if the categorySelection() return true boolean value
     * then the updateTable() is called
     *
     * @param evt
     */
    private void availabilityActionPerformed(java.awt.event.ActionEvent evt) {
        // Check for avaiable items
        try {
            String checkItem = category.getSelectedItem().toString();
            int rowCount = model.getRowCount();
            int columnCount = model.getColumnCount();

            if (!checkItem.equals("<None>")) {

                String[][] available = new String[rowCount][columnCount];

                if (categorySelection(checkItem, -1, 0, columnCount, rowCount, available)) {
                    updateTable(available);
                }

            } else {
                JOptionPane.showMessageDialog(null, "You need to select a category", "No option selected", JOptionPane.INFORMATION_MESSAGE);
            }
        } catch (NullPointerException e) {
            JOptionPane.showMessageDialog(null, "You need to import a csv file into the table ", "The table is empty!", JOptionPane.ERROR_MESSAGE);
        }
    }
```

```java
    /**
     * This method refreshes the table and brings the table to its original
     * states Also when a user adds to or updates the table it also adds any new
     * category to the JComboBox
     */
    public void refresh() {
        try {

            if (MenuInfo.newCategory != null) {
                for (int i = 0; i < MenuInfo.newCategory.size(); i++) {
                    category.addItem(MenuInfo.newCategory.get(i));
                    MenuInfo.newCategory.clear();
                }
            }
            for (int i = 0; i < array.size(); i++) {
                for (int j = 0; j < array.get(i).size() - 1; j++) {
                    petTable.setValueAt(array.get(i).get(j), i, j);
                }
            }
            availability.setEnabled(true);
        } catch (NullPointerException e) {
            JOptionPane.showMessageDialog(null, "You need to import a csv file into the table ", "The table is empty!", JOptionPane.ERROR_MESSAGE);
        } catch (ArrayIndexOutOfBoundsException e) {
            JOptionPane.showMessageDialog(null, "You need to import a csv file into the table ", "The table is empty!", JOptionPane.ERROR_MESSAGE);
        }
    }
```

```
1176      /**
1177       * this is the method for the refresh button this method calls the refresh()
1178       * method
1179       *
1180       * @param evt
1181       */
1182      private void refreshActionPerformed(java.awt.event.ActionEvent evt) {
1183          refresh();
1184      }
1185
1186      private void categoryActionPerformed(java.awt.event.ActionEvent evt) {
1187          // TODO add your handling code here:
1188      }
1189
1190      /**
1191       * this is the method for the update button this method checks for empty
1192       * table and sends error method if the table is not empty all the values
1193       * from the row that the user wants to update is sent to a constructor The
1194       * constructor belongs to the Update.java class Also the update form is
1195       * opened if there are no errors
1196       *
1197       * @param evt
1198       */
1199      private void updateActionPerformed(java.awt.event.ActionEvent evt) {
1200          String id, name, categori, age, lifespan, size, nature, quantity, price, image;
1201          image = "";
              int rowIndex = 0;
1203
1204          try {
1205              if (petTable.getCellSelectionEnabled()) {
1206                  //petTable.setSelectionModel(ListSelectionModel.SINGLE_SELECTION);
1207
1208                  rowIndex = petTable.getSelectedRow();
1209                  id = model.getValueAt(rowIndex, 0).toString();
1210
```

```
1211                  if (id != null) {
1212
1213                      name = String.valueOf(model.getValueAt(rowIndex, 1));
1214
1215                      categori = String.valueOf(model.getValueAt(rowIndex, 2));
1216
1217                      age = String.valueOf(model.getValueAt(rowIndex, 3));
1218
1219                      lifespan = String.valueOf(model.getValueAt(rowIndex, 4));
1220
1221                      size = String.valueOf(model.getValueAt(rowIndex, 5));
1222
1223                      nature = String.valueOf(model.getValueAt(rowIndex, 6));
1224
1225                      quantity = String.valueOf(model.getValueAt(rowIndex, 7));
1226
1227                      price = String.valueOf(model.getValueAt(rowIndex, 8));
1228
1229                      for (int i = 0; i < array.size(); i++) {
1230                          if (id.equals(array.get(i).get(0))) {
                                  image = array.get(i).get(9).toString();
1232                          }
1233                      }
1234
1235                      Update upd = new Update(rowIndex, id, name, categori, age, lifespan, size, nature, quantity, price, image);
1236                      upd.setVisible(true);
1237
1238                      upd.setValue();
1239                  }
1240              }
1241          } catch (NullPointerException e) {
1242              JOptionPane.showMessageDialog(null, "There are no values in the table.", "The table is empty!", JOptionPane.ERROR_MESSAGE);
1243              //upd.setVisible(false);
1244          } catch (ArrayIndexOutOfBoundsException ex) {
1245              JOptionPane.showMessageDialog(null, "You have to select a row you want to update.", "Information", JOptionPane.INFORMATION_MESSAGE);
1246          }
1247      }
```

```
1249    /**
1250     * this is the method for the search button: the rows of the model is
1251     * counted a local price array is made to store price data An ArrayList
1252     * copyarray is made to store the SelectionSort.sortArray() is called to
1253     * sort the array and arraylist searchchelist method is called to search the
1254     * data if data is found the data is displayed in dailogbox.
1255     *
1256     * @param evt
1257     */
1258    private void searchPriceActionPerformed(java.awt.event.ActionEvent evt) {
1259        // TODO add your handling code here:
1260
1261        if (search.getText().equals("")) {
1262            JOptionPane.showMessageDialog(null, "Please enter a price to search");
1263        } else {
1264            int rows = model.getRowCount();
1265            int[] price = new int[rows];
1266            for (int i = 0; i < petTable.getRowCount(); i++) {
1267                int value = Integer.parseInt(array.get(i).get(8));
1268                price[i] = value;
1269            }
1270            ArrayList<ArrayList<String>> copyarray = new ArrayList<>(array);
1271            SelectionSort.sortArray(price, copyarray);
1272
1273            int key = Integer.parseInt(search.getText());
1274            int low = 0;
1275            int high = price.length - 1;
1276            int found = searchList(price, low, high, key);
1277
1278            if (found == -1) {
1279                JOptionPane.showMessageDialog(null, "Search data not found");
1280            } else {
1281                JOptionPane.showMessageDialog(null, "Congratulations Search data found");
1282                String name = copyarray.get(found).get(1);
1283                String categori = copyarray.get(found).get(2);
1284                String prices = copyarray.get(found).get(8);
1285                JOptionPane.showMessageDialog(null, "The searched price is: " + prices
1286                        + "\nPet Name: " + name
1287                        + "\nCategory: " + categori);
1288            }
1289        }
1290
1291    }
1293    /**
1294     * this is the method for the search button: the rows of the model is
1295     * counted a local price array is made to store price data An ArrayList
1296     * copy array is made to store the SelectionSort.sortArray() is called to
1297     * sort the array and ArrayList .the table is cleared and
1298     * the sorted ArrayList is displayed  in the JTable
1299     *
1300     * @param evt
1301     */
1302    private void sortPrice1ActionPerformed(java.awt.event.ActionEvent evt) {
1303        // TODO add your handling code here:
1304        int rows = model.getRowCount();
1305        int[] price = new int[rows];
1306        for (int i = 0; i < petTable.getRowCount(); i++) {
1307            int value = Integer.parseInt(array.get(i).get(8));
1308            price[i] = value;
1309        }
1310        ArrayList<ArrayList<String>> copyarray = new ArrayList<>(array);
1311        SelectionSort.sortArray(price, copyarray);
1312        model.setRowCount(0);
1313
1314        for (int i = 0; i < copyarray.size(); i++) {
1315            //The delimeter in a csv file is ","
1316            String[] row = new String[]{copyarray.get(i).get(0),
1317                copyarray.get(i).get(1),
1318                copyarray.get(i).get(2),
1319                copyarray.get(i).get(3),
1320                copyarray.get(i).get(4),
1321                copyarray.get(i).get(5),
1322                copyarray.get(i).get(6),
1323                copyarray.get(i).get(7),
1324                copyarray.get(i).get(8)};
1325            model.addRow(row);
1326        }
1327
1328
1329    }
```

```
1331      /**
1332       * this method restricts the user to input any non numeric value in the
1333       * search field the search field only searches for the price of the pet
1334       *
1335       * @param evt
1336       */
1337      private void searchKeyTyped(java.awt.event.KeyEvent evt) {
1338          // TODO add your handling code here:
1339          char dn = evt.getKeyChar();
1340          if (!(Character.isDigit(dn) || (dn == KeyEvent.VK_BACK_SPACE) || dn == KeyEvent.VK_DELETE)) {
1341              evt.consume();
1342          }
1343      }
1344
1345      private void searchActionPerformed(java.awt.event.ActionEvent evt) {
1346          // TODO add your handling code here:
1347      }
1348
1349      private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
1350          AppHelp.getObj().setVisible(true);
1351      }
1352
1353      /**
1354       *
1355       * @param price
1356       * @param low
1357       * @param high
1358       * @param key
1359       * @return
1360       */
1361      public int searchList(int price[], int low, int high, int key) {
1362
1363          if (low <= high) {
1364              int mid = (low + high) / 2;
1365
1366              if (price[mid] == key) {
1367                  return mid;
1368              } else if (price[mid] > key) {
1369                  return searchList(price, low, mid - 1, key);
1370              } else if (price[mid] < key) {
1371                  return searchList(price, mid + 1, high, key);
1372              }
1373          }
1374
1375          return -1;
1376      }
1378      /**
1379       * @param args the command line arguments
1380       */
1381      public static void main(String args[]) {
1382          /* Set the Nimbus look and feel */
1383          Look and feel setting code (optional)
1404          //</editor-fold>
1405
1406          /* Create and display the form */
              java.awt.EventQueue.invokeLater(new Runnable() {
                  public void run() {
1409                  try {
1410                      Thread.sleep(3000);
                      } catch (Exception e) {
1412
1413                  }
1414                  new MenuInfo().setVisible(true);
1415              }
1416          });
1417
1418      }
1419
```

```
1420        // Variables declaration - do not modify
1421        private javax.swing.JButton add;
1422        private javax.swing.JButton admin;
1423        private javax.swing.JButton availability;
1424        private javax.swing.JComboBox<String> category;
1425        private javax.swing.JButton del;
1426        private javax.swing.JLabel denoter;
1427        private javax.swing.JButton exit;
1428        private javax.swing.JPanel footer;
1429        private javax.swing.JMenuItem import_csv;
1430        private javax.swing.JDialog jDialog1;
1431        private javax.swing.JLabel jLabel1;
1432        private javax.swing.JLabel jLabel2;
1433        private javax.swing.JLabel jLabel3;
1434        private javax.swing.JLabel jLabel4;
1435        private javax.swing.JLabel jLabel5;
1436        private javax.swing.JLabel jLabel6;
1437        private javax.swing.JLabel jLabel7;
1438        private javax.swing.JMenu jMenu1;
1439        private javax.swing.JMenu jMenu2;
1440        private javax.swing.JMenuBar jMenuBar1;
1441        private javax.swing.JMenuItem jMenuItem1;
1442        private javax.swing.JScrollPane jScrollPane1;
1443        private javax.swing.JButton log_out;
1444        private javax.swing.JPanel main;
1445        private javax.swing.JMenuItem menu_exit;
1446        private javax.swing.JButton minimize;
1447        private javax.swing.JTable petTable;
1448        private javax.swing.JButton refresh;
1449        private javax.swing.JTextField search;
1450        private javax.swing.JButton searchPrice;
1451        private javax.swing.JPanel side_panel;
1452        private javax.swing.JButton sortPrice1;
1453        private javax.swing.JButton update;
1454        private javax.swing.JLabel user_logo;
1455        private javax.swing.JLabel user_name;
1456        private javax.swing.JButton view_details;
1457        private javax.swing.JPanel window;
1458        // End of variables declaration
1459
1460    }
```

Details Class Code

```
/*
 * 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 Dashboard;

import java.awt.Image;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import javax.swing.ImageIcon;

/**
 *
 * @author Dell
 */
public class Details extends javax.swing.JFrame {

    private static Details b = null;
    /**
     * Creates new form Details
     */
    private Details() {
        initComponents();
    }

    /**
     * this initializes the object of the class if it is null
     * this prevents the frame from opening again and again.
     * @return
     */
    public static Details getObj() {
        if (b == null) {
            b = new Details();
        }
        return b;
    }

    /***
     * this value searches for the pet id in the two dimensional array @array
     * the row in the array that has the pet id is fetched
     * then the value of the row is set in the frame for the user to see
     * this method also makes sure the dimension of the image is fitted to the jLabel with smooth scaling
     * @param id
     * @param array
     */
    public void viewDetails (int id, ArrayList<ArrayList<String>> array) {
        MenuInfo obj = new MenuInfo();

        for (int i = 0; i < array.size(); i++) {
            for (int j = 0; j < array.get(0).size(); j++) {
                int petId = Integer.parseInt(array.get(i).get(0));
                if ( petId == id) {
                    ImageIcon img = new ImageIcon(MenuInfo.drive+array.get(i).get(9));

                    Image img1 = img.getImage();
                    Image img2 = img1.getScaledInstance(357, 287, Image.SCALE_SMOOTH);
                    ImageIcon icon = new ImageIcon(img2);

                    image.setIcon(icon);
                    title.setText(array.get(i).get(1));
                    category.setText(array.get(i).get(2));
                    price.setText(array.get(i).get(8));
                    quantity.setText(array.get(i).get(7));
                    age.setText(array.get(i).get(3));
                    lifespan.setText(array.get(i).get(4));
                    size.setText(array.get(i).get(5));
                    nature.setText(array.get(i).get(6));

                    break;
                }
            }
        }
    }
```

```
76      /**
77       * This method is called from within the constructor to initialize the form.
78       * WARNING: Do NOT modify this code. The content of this method is always
79       * regenerated by the Form Editor.
80       */
81      @SuppressWarnings("unchecked")
82      Generated Code
348
349     /**
350      * @param args the command line arguments
351      */
        public static void main(String args[]) throws FileNotFoundException {
353
354         /* Set the Nimbus look and feel */
355         Look and feel setting code (optional)
376
377         /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
380                 new Details().setVisible(true);
381             }
382         });
383
384     }
385
386     // Variables declaration - do not modify
387     private javax.swing.JLabel age;
388     private javax.swing.JLabel age1;
389     private javax.swing.JLabel category;
390     private javax.swing.JLabel image;
391     private javax.swing.JLabel jLabel1;
392     private javax.swing.JLabel jLabel10;
393     private javax.swing.JLabel jLabel11;
394     private javax.swing.JLabel jLabel12;
395     private javax.swing.JLabel jLabel13;
396     private javax.swing.JLabel jLabel19;
397     private javax.swing.JLabel jLabel2;
398     private javax.swing.JLabel jLabel20;
399     private javax.swing.JLabel jLabel21;
400     private javax.swing.JLabel jLabel3;
401     private javax.swing.JLabel jLabel4;
402     private javax.swing.JLabel jLabel5;
403     private javax.swing.JLabel jLabel6;
404     private javax.swing.JLabel jLabel7;
405     private javax.swing.JLabel jLabel9;
406     private javax.swing.JPanel jPanel1;
407     private javax.swing.JPanel jPanel2;
408     private javax.swing.JSeparator jSeparator1;
409     private javax.swing.JLabel lifespan;
410     private javax.swing.JLabel lifespan1;
411     private javax.swing.JLabel nature;
412     private javax.swing.JLabel price;
413     private javax.swing.JLabel quantity;
414     private javax.swing.JLabel size;
415     private javax.swing.JLabel title;
416     // End of variables declaration
417 }
418
```

AddData Class Code

```
1  package Dashboard;
2
3  import java.awt.event.KeyEvent;
4  import java.awt.event.WindowEvent;
5  import java.io.File;
6  import java.util.ArrayList;
7  import java.util.InputMismatchException;
8  import javax.swing.JFileChooser;
9  import javax.swing.JOptionPane;
10
11 /*
12  * To change this license header, choose License Headers in Project Properties.
13  * To change this template file, choose Tools | Templates
14  * and open the template in the editor.
15  */
16
17 /**
18  *
19  * @author Dell
20  */
21 public class AddData extends javax.swing.JFrame {
22
23     private static AddData obj1 = null;
24
25     public static ArrayList<String> data;
26     String imgPath;
27     MenuInfo obj = new MenuInfo();
28     /**
29      * Creates new form AddData
30      */
31     private AddData() {
32         initComponents();
33     }
34
35     /**
36      * this initializes the object of the class if it is null
37      * this prevents the frame from opening again and again.
38      * @return
39      */
40     public static AddData getObj() {
41         if (obj1 == null) {
42             obj1 = new AddData();
43         }
44         return obj1;
45     }
```

```
    /**
     * this method checks for the biggest pet id in the array
     * then sets the pet id in the text field of the add form incrementing it by one for the nest pet to be added.
     */
    public void setIdDefault(){
        int max = 0;
        for (int i = 0; i < MenuInfo.array.size(); i++) {
            int element = Integer.parseInt(MenuInfo.array.get(i).get(0));
            if (max < element) {
                max = element;
            }
        }
        max += 1;
        id.setText(String.valueOf(max));
    }

    /**
     * this method returns the selected radio button from the add form
     * @return
     */
    private String getRadioSelection() {
        String getNature ="";
        if (radioLoyal.isSelected()){
            getNature = radioLoyal.getText();

        } else if (radioAggressive.isSelected()) {
            getNature = radioAggressive.getText();

        } else if (radioPassive.isSelected()) {
            getNature = radioPassive.getText();

        } else if(radioInquisitive.isSelected()) {
            getNature = radioInquisitive.getText();

        }
        return getNature;
    }
    /**
     * this method return the selected check box from the add form
     * @return
     */
    private String getCheckSelection() {
        String getSize = "";

        if (checkBig.isSelected()){

            getSize = checkBig.getText();
            checkMed.setSelected(false);
            checkSmall.setSelected(false);

        } else if (checkMed.isSelected()) {

            getSize = checkMed.getText();
            checkBig.setSelected(false);
            checkSmall.setSelected(false);

        } else if (checkSmall.isSelected()) {

            getSize = checkSmall.getText();
            checkBig.setSelected(false);
            checkMed.setSelected(false);
        }
        return getSize;
    }

```

```
113     /**
114      * this method makes sure the integer value for the price, quantity, age, lifespan is non fractional and non negative
115      * @param id
116      * @param price
117      * @param quantity
118      * @return
119      */
120     private boolean checkForInt(String id, String price, String quantity) {
121         boolean flag ;
122         int price_, quantity_, id_;
123         try {
124                 price_ = Integer.parseInt(price);
125                 quantity_ = Integer.parseInt(quantity);
126                 id_ = Integer.parseInt(id);
127 
128                 if (price_ > 0 && quantity_ > 0 && id_ > 0) {
129                     flag = true;
130                 } else {
131                     flag = false;
132                     JOptionPane.showMessageDialog (null,"Make sure the id, quantity and price are non-negative", "Error", JOptionPane.ERROR_MESSAGE);
133                 }
134 
135         } catch(InputMismatchException | NumberFormatException e) {
136             JOptionPane.showMessageDialog (null,"Please input whole numbers only", "Error", JOptionPane.ERROR_MESSAGE);
137             flag = false;
138         }
139 
140         return flag;
141     }
142 
143     /**
144      * This method is called from within the constructor to initialize the form.
145      * WARNING: Do NOT modify this code. The content of this method is always
146      * regenerated by the Form Editor.
147      */
148     @SuppressWarnings("unchecked")
149     Generated Code
503 
504     private void lifespanActionPerformed(java.awt.event.ActionEvent evt) {
505         // TODO add your handling code here:
506     }
507 
508     private void categoryActionPerformed(java.awt.event.ActionEvent evt) {
509         // TODO add your handling code here:
510     }
511 
512     private void quantityActionPerformed(java.awt.event.ActionEvent evt) {
513         // TODO add your handling code here:
514     }
515 
516     private void priceActionPerformed(java.awt.event.ActionEvent evt) {
517         // TODO add your handling code here:
518     }
519 
520     /**
521      * this method clears all the value from the add form
522      */
523     public void clear () {
524         id.setText("");
525         name.setText("");
526         category.setText("");
527         age.setText("");
528         lifespan.setText("");
529         quantity.setText("");
530         price.setText("");
531         checkBox.clearSelection();
532         radioButton.clearSelection();
533     }
```

```
535     /**
536      * this is the add button in the add form
537      * this method takes all the user inputs and the adds the new values into the table.
538      * this method checks for validation by calling other methods
539      * @param evt
540      */
541     private void addActionPerformed(java.awt.event.ActionEvent evt) {
542         try{
544             data = new ArrayList<String>();
545 
546             String getId = id.getText();
547             String getName = name.getText();
548             String getCategory = category.getText();
549             String getAge = age.getText();
550             String getLifespan = lifespan.getText();
551             String getQuantity = quantity.getText();
552             String getPrice = price.getText();
553             String getNature = getRadioSelection();
554             String getSize = getCheckSelection();
555 
556             int counter = 0;
557 
558             if ("".equals(getName)){
559                 counter += 1;
560                 JOptionPane.showMessageDialog (null,"The name is empty", "Error", JOptionPane.ERROR_MESSAGE);
561             }
562 
563             if ("".equals(getId)){
564                 counter += 1;
565                 JOptionPane.showMessageDialog (null,"The id is empty", "Error", JOptionPane.ERROR_MESSAGE);
566             }
567 
568             if ("".equals(getCategory)){
569                 counter += 1;
570                 JOptionPane.showMessageDialog (null,"The category is empty", "Error", JOptionPane.ERROR_MESSAGE);
571             }
562             if ("".equals(getId)){
563                 counter += 1;
564                 JOptionPane.showMessageDialog (null,"The id is empty", "Error", JOptionPane.ERROR_MESSAGE);
565             }
566 
567             if ("".equals(getCategory)){
568                 counter += 1;
569                 JOptionPane.showMessageDialog (null,"The category is empty", "Error", JOptionPane.ERROR_MESSAGE);
570             }
571 
572             if ("".equals(getAge)){
573                 counter += 1;
574                 JOptionPane.showMessageDialog (null,"The age is empty", "Error", JOptionPane.ERROR_MESSAGE);
575             }
576 
577             if ("".equals(getLifespan)){
578                 counter += 1;
579                 JOptionPane.showMessageDialog (null,"The life span is empty", "Error", JOptionPane.ERROR_MESSAGE);
580             }
581 
582             if ("".equals(getQuantity)){
583                 counter += 1;
584                 JOptionPane.showMessageDialog (null,"The quantity is empty", "Error", JOptionPane.ERROR_MESSAGE);
585             }
586 
587             if ("".equals(getPrice)){
588                 counter += 1;
589                 JOptionPane.showMessageDialog (null,"The price is empty", "Error", JOptionPane.ERROR_MESSAGE);
590             }
591 
592             if ("".equals(getNature)){
593                 counter += 1;
594                 JOptionPane.showMessageDialog (null,"The nature is not selected", "Error", JOptionPane.ERROR_MESSAGE);
595             }
596 
597             if ("".equals(getSize)){
598                 counter += 1;
599                 JOptionPane.showMessageDialog (null,"The size has not been selected", "Error", JOptionPane.ERROR_MESSAGE);
600             }
601 
602             if ("".equals(imgPath) || imgPath == null){
603                 counter += 1;
604                 JOptionPane.showMessageDialog (null,"No image has been selected", "Error", JOptionPane.ERROR_MESSAGE);
605             }
```

```
            if (counter == 0){
                getName = getName.substring(0,1).toUpperCase() + getName.substring(1).toLowerCase();
                getCategory = getCategory.substring(0,1).toUpperCase() + getCategory.substring(1).toLowerCase();

                String newCat = getCategory;
                if (checkForInt(getId, getQuantity, getPrice)) {

                    int noZero = Integer.parseInt(getQuantity);
                    getQuantity = String.valueOf(noZero);

                    noZero = Integer.parseInt(getPrice);
                    getPrice = String.valueOf(noZero);

                    int check = 0;
                    for (int i = 0; i < MenuInfo.array.size(); i++) {
                        if ((MenuInfo.array.get(i).get(0)).equals(getId) || (MenuInfo.array.get(i).get(1)).equals(getName)) {
                            JOptionPane.showMessageDialog (null,"Sorry, a data with the same name or id already exists", "Error", JOptionPane.ERROR_MESSAGE);
                            check += 1;
                            break;

                        }
                    }
                     if (check == 0) {
                         data.add(0, getId);
                         data.add(1, getName);
                         data.add(2, getCategory);

                         data.add(3, getAge);
                         data.add(4,getLifespan);
                         data.add(5, getSize);
                         data.add(6, getNature);
                         data.add(7, getQuantity);
                         data.add(8, getPrice);
                         data.add(9, imgPath);

                         MenuInfo.array.add(data);
                         obj.addData();
                         obj.checkCategory(getCategory);
                         clear();
                         setIdDefault();
                     }

                }
            }
        } catch (NullPointerException e) {
            JOptionPane.showMessageDialog (null,"The table is empty. Make sure you import a csv file into the table.", "Error", JOptionPane.ERROR_MESSAGE);
        }

    }
    /**
     * this method opens the file selector for the user to add image for the new pet.
     * @param evt
     */
    private void addImageActionPerformed(java.awt.event.ActionEvent evt) {
        try {
                JFileChooser file=new JFileChooser();
                file.showOpenDialog(null);
                file.setCurrentDirectory(new File(System.getProperty("user.home")));
                File f = file.getSelectedFile();
                imgPath=f.getAbsolutePath();

                String extension = "";
                //checking if the file format is csv or not
                    int j = imgPath.lastIndexOf('.');
                    if (j >= 0) {
                        extension = imgPath.substring(j+1);
                    }
                if (extension.equals("png") || extension.equals("jpg") || extension.equals("jpeg")){
                    //Do nothing
                    //return the value
                }
                else {
                    JOptionPane.showMessageDialog (null,"Please select a proper image file. It should be either png, jpg or jpeg.", "Error", JOptionPane.ERROR_MESSAGE);
                    addImageActionPerformed(evt);
                }
        } catch (NullPointerException e) {
            JOptionPane.showMessageDialog (null,"No file selected.", "Error", JOptionPane.ERROR_MESSAGE);
        }

    }

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

```
690     private void ageActionPerformed(java.awt.event.ActionEvent evt) {
691         // TODO add your handling code here:
692     }
693 
694     /**
695      * this method restricts the user fro inputting non-numeric value for the age of the pet
696      * @param evt
697      */
698     private void ageKeyTyped(java.awt.event.KeyEvent evt) {
699         // TODO add your handling code here:
700         char dn= evt.getKeyChar();
701         if(!(Character.isDigit(dn) || (dn==KeyEvent.VK_BACK_SPACE) || dn==KeyEvent.VK_DELETE)){
702             evt.consume();
703         }
704     }
705 
706     /**
707      * this method restricts the user fro inputting non-numeric value for the quantity of the pet
708      * @param evt
709      */
710     private void quantityKeyTyped(java.awt.event.KeyEvent evt) {
711         // TODO add your handling code here:
712         char dn= evt.getKeyChar();
713         if(!(Character.isDigit(dn) || (dn==KeyEvent.VK_BACK_SPACE) || dn==KeyEvent.VK_DELETE)){
714             evt.consume();
715         }
716     }
717 
718     /**
719      * this method restricts the user fro inputting non-numeric value for the price of the pet
720      * @param evt
721      */
722     private void priceKeyTyped(java.awt.event.KeyEvent evt) {
723         // TODO add your handling code here:
724         char dn= evt.getKeyChar();
725         if(!(Character.isDigit(dn) || (dn==KeyEvent.VK_BACK_SPACE) || dn==KeyEvent.VK_DELETE)){
726             evt.consume();
727         }
728     }
729 
730     /**
731      * this method only allows character inputs for the name of the et
732      * @param evt
733      */
734     private void nameKeyTyped(java.awt.event.KeyEvent evt) {
735         // TODO add your handling code here:
736         char dn= evt.getKeyChar();
737         if((Character.isDigit(dn) || (dn==KeyEvent.VK_BACK_SPACE) || dn==KeyEvent.VK_DELETE)){
738             evt.consume();
739         }
740     }
741 
742     private void nameActionPerformed(java.awt.event.ActionEvent evt) {
743         // TODO add your handling code here:
744     }
```

```
746     /**
747      * @param args the command line arguments
748      */
749     public static void main(String args[]) {
750         /* Set the Nimbus look and feel */
751         Look and feel setting code (optional)
766
767         //</editor-fold>
768
769         /* Create and display the form */
770         java.awt.EventQueue.invokeLater(() -> {
771             new AddData().setVisible(true);
772         });
773
774         new AddData().addWindowListener(new java.awt.event.WindowAdapter() {
775             @Override
              public void windowClosed(java.awt.event.WindowEvent windowEvent) {
777             JOptionPane.showMessageDialog (null,"Don't go.", "Error", JOptionPane.ERROR_MESSAGE);
778             }
779             });
780     }
781
782     // Variables declaration - do not modify
783     private javax.swing.JButton add;
784     private javax.swing.JButton addImage;
785     private javax.swing.JTextField age;
786     private javax.swing.JTextField category;
787     private javax.swing.JCheckBox checkBig;
788     private javax.swing.ButtonGroup checkBox;
789     private javax.swing.JCheckBox checkMed;
790     private javax.swing.JCheckBox checkSmall;
791     private javax.swing.JTextField id;
792     private javax.swing.JLabel jLabel1;
793     private javax.swing.JLabel jLabel10;
794     private javax.swing.JLabel jLabel11;
795     private javax.swing.JLabel jLabel12;
796     private javax.swing.JLabel jLabel2;
797     private javax.swing.JLabel jLabel3;
798     private javax.swing.JLabel jLabel4;
799     private javax.swing.JLabel jLabel5;
800     private javax.swing.JLabel jLabel6;
801     private javax.swing.JLabel jLabel7;
802     private javax.swing.JLabel jLabel8;
803     private javax.swing.JLabel jLabel9;
804     private javax.swing.JPanel jPanel1;
805     private javax.swing.JSeparator jSeparator1;
806     private javax.swing.JTextField lifespan;
807     private javax.swing.JTextField name;
808     private javax.swing.JTextField price;
809     private javax.swing.JTextField quantity;
810     private javax.swing.JRadioButton radioAggressive;
811     private javax.swing.ButtonGroup radioButton;
812     private javax.swing.JRadioButton radioInquisitive;
813     private javax.swing.JRadioButton radioLoyal;
814     private javax.swing.JRadioButton radioPassive;
815     // End of variables declaration
```

Update Class Code

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

import java.io.File;
import java.util.InputMismatchException;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

/**
 *
 * @author Dell
 */
public class Update extends javax.swing.JFrame {

    private static Update upd1 = null;

    public String id;
    public String name;
    public String category;
    public String age;
    public String lifespan;
    public String size;
    public String nature;
    public String quantity;
    public String price;
    public String image;

    public int rowIndex;

    /**
     * Creates new form Update
     */
    private Update() {
        initComponents();
    }
    /**
     * this is the constructor of the class that takes the value of the selected row in the table and initializes them in the class
     * @param rowIndex
     * @param id
     * @param name
     * @param category
     * @param age
     * @param lifespan
     * @param size
     * @param nature
     * @param quantity
     * @param price
     * @param image
     */
    public Update(int rowIndex, String id, String name, String category, String age, String lifespan, String size, String nature, String quantity, String price, String image){
        initComponents();
        this.rowIndex = rowIndex;
        this.id = id;
        this.name = name;
        this.category = category;
        this.age = age;
        this.lifespan = lifespan;
        this.size = size;
        this.nature = nature;
        this.quantity = quantity;
        this.price = price;
        this.image = image;

    }

```

```
72      /**
73       * This method is called from within the constructor to initialize the form.
74       * WARNING: Do NOT modify this code. The content of this method is always
75       * regenerated by the Form Editor.
76       */
77      @SuppressWarnings("unchecked")
78      Generated Code
443
444
445     /**
446      * this checks the check box for the corresponding value of the elected row
447      * @param size
448      */
449     private void checkSize(String size) {
450
451         switch(size) {
452             case "Big":
453                 checkBig.setSelected(true);
454                 break;
455
456             case "Medium":
457                 checkMed.setSelected(true);
458                 break;
459
460             case "Small":
461                 checkSmall.setSelected(true);
462                 break;
463         }
464     }
465
466     /**
467      * this selects the corresponding radio button for the value of the selected row
468      * @param nature
469      * @return
470      */
471     private boolean checkNature(String nature) {
            boolean check = false;
473
474         switch(nature) {
475             case "Loyal":
476                 radioLoyal.setSelected(true);
477                 check = true;
478                 break;
479
480             case "Aggressive":
481                 radioAgg.setSelected(true);
482                 check = true;
483                 break;
484
485             case "Passive":
486                 radioPas.setSelected(true);
487                 check = true;
488                 break;
489
490             case "Inquisitive":
491                 radioInq.setSelected(true);
492                 check = true;
493                 break;
494
495             default:
496                 check = false;
497                 break;
498
499         }
500         return check;
501     }
502
```

```
503     /**
504      * this method makes sure the integer value for the price, quantity, age, lifespan is non fractional and non negative
505      * @param id
506      * @param price
507      * @param quantity
508      * @return
509      */
510     private boolean checkForInt(String price, String quantity) {
511         boolean flag ;
512         int price_, quantity_, id_;
513         try {
514                 price_ = Integer.parseInt(price);
515                 quantity_ = Integer.parseInt(quantity);
516 
517                 if (price_ > 0 && quantity_ > 0) {
518                     flag = true;
519                 } else {
520                     flag = false;
521                     JOptionPane.showMessageDialog (null,"Make sure the id, quantity and price are non-negative", "Error", JOptionPane.ERROR_MESSAGE);
522                 }
523 
524         } catch(InputMismatchException | NumberFormatException e) {
525             JOptionPane.showMessageDialog (null,"Please input whole numbers only", "Error", JOptionPane.ERROR_MESSAGE);
526             flag = false;
527         }
528 
529         return flag;
530     }
531 
533     /**
534      * this method updates the table and the array for the all the changes the user has made in the update form
535      * this method also checks for invalid inputs
536      *
537      * @param evt
538      */
539     private void updateActionPerformed(java.awt.event.ActionEvent evt) {
540         MenuInfo obj = new MenuInfo();
541         String name_, category_, age_, lifespan_, price_, quantity_, nature_, size_;
542 
543         int counter = 0;
544 
545 
546         name_ = name2.getText();
547         category_ = category2.getText();
548         age_ = age2.getText();
549         lifespan_ = lifespan2.getText();
550         price_ = price2.getText();
551         quantity_ = quantity2.getText();
552 
553         nature_ = radioSelection();
554 
555 
556         size_ = checkSelection();
557 
558             if ("".equals(name_)){
559                 counter += 1;
560                 JOptionPane.showMessageDialog (null,"The name is empty", "Error", JOptionPane.ERROR_MESSAGE);
561             }
562 
563             if ("".equals(category)){
564                 counter += 1;
565                 JOptionPane.showMessageDialog (null,"The category is empty", "Error", JOptionPane.ERROR_MESSAGE);
566             }
567 
568             if ("".equals(age_)){
569                 counter += 1;
570                 JOptionPane.showMessageDialog (null,"The age is empty", "Error", JOptionPane.ERROR_MESSAGE);
571             }
572 
573             if ("".equals(lifespan_)){
574                 counter += 1;
575                 JOptionPane.showMessageDialog (null,"The life span is empty", "Error", JOptionPane.ERROR_MESSAGE);
576             }
```

```
                if ("".equals(quantity_)){
                    counter += 1;
                    JOptionPane.showMessageDialog (null,"The quantity is empty", "Error", JOptionPane.ERROR_MESSAGE);
                }

                if ("".equals(price_)){
                    counter += 1;
                    JOptionPane.showMessageDialog (null,"The price is empty", "Error", JOptionPane.ERROR_MESSAGE);
                }

                if ("".equals(nature_)){
                    counter += 1;
                    JOptionPane.showMessageDialog (null,"The nature is not selected", "Error", JOptionPane.ERROR_MESSAGE);
                }

                if ("".equals(size_)){
                    counter += 1;
                    JOptionPane.showMessageDialog (null,"The size has not been selected", "Error", JOptionPane.ERROR_MESSAGE);
                }


                if (counter == 0) {
                    name_ = name_.substring(0,1).toUpperCase() + name_.substring(1).toLowerCase();
                    category_ = category_.substring(0,1).toUpperCase() + category_.substring(1).toLowerCase();
                    String newCat = category_;

                    if (checkForInt(price, quantity)) {

                        int noZero = Integer.parseInt(quantity_);
                        quantity = String.valueOf(noZero);

                        noZero = Integer.parseInt(price_);
                        price_ = String.valueOf(noZero);

                        for (int i = 0; i < MenuInfo.array.size(); i++) {
                            if (id.equals(MenuInfo.array.get(i).get(0))) {

                                MenuInfo.array.get(i).set(1,name_);
                                MenuInfo.model.setValueAt(name_, rowIndex, 1);

                                MenuInfo.array.get(i).set(2, category_);
                                MenuInfo.model.setValueAt(category_, rowIndex, 2);
                                MenuInfo.array.get(i).set(3, age_);
                                MenuInfo.model.setValueAt(age_, rowIndex, 3);

                                MenuInfo.array.get(i).set(4, lifespan_);
                                MenuInfo.model.setValueAt(lifespan_, rowIndex, 4);

                                MenuInfo.array.get(i).set(5, size_);
                                MenuInfo.model.setValueAt(size_, rowIndex, 5);

                                MenuInfo.array.get(i).set(6, nature_);
                                MenuInfo.model.setValueAt(nature_, rowIndex, 6);

                                MenuInfo.array.get(i).set(7, quantity_);
                                MenuInfo.model.setValueAt(quantity_, rowIndex, 7);

                                MenuInfo.array.get(i).set(8, price_);
                                MenuInfo.model.setValueAt(price_, rowIndex, 8);

                                MenuInfo.array.get(i).set(9,image);

                                obj.checkCategory(category_);
                                JOptionPane.showMessageDialog (null,"The row has been updated. Please update to see the changes in the category.", "Success!", JOptionPane.INFORMATION_MESSAGE);
                                this.setVisible(false);

                                break;
                            }
                        }
                    }
                }


    }
```

```
655     /**
656      * this method opens the file selector for the user to add image for the new pet.
657      * @param evt
658      */
659     private void imageChooseActionPerformed(java.awt.event.ActionEvent evt) {
660         try {
661                 JFileChooser file=new JFileChooser();
662                 file.showOpenDialog(null);
663                 file.setCurrentDirectory(new File(System.getProperty("user.home")));
664                 File f = file.getSelectedFile();
665                 image =f.getAbsolutePath();
666 
667                 String extension = "";
668                 //checking if the file format is csv or not
669                     int j = image.lastIndexOf('.');
670                     if (j >= 0) {
671                         extension = image.substring(j+1);
672                     }
673                 if (extension.equals("png") || extension.equals("jpg") || extension.equals("jpeg")){
674                     //Do nothing
675                     //return the value
676                 }
677                 else {
678                     JOptionPane.showMessageDialog (null,"Please select a proper image file. It should be either png, jpg or jpeg.", "Error", JOptionPane.ERROR_MESSAGE);
679                     imageChooseActionPerformed(evt);
680                 }
681         } catch (NullPointerException e) {
682             JOptionPane.showMessageDialog (null,"No file selected.", "Error", JOptionPane.ERROR_MESSAGE);
683         }
684     }
685 
686     private void radioPasActionPerformed(java.awt.event.ActionEvent evt) {
687         // TODO add your handling code here:
688     }
690     /**
691      * this method returns the selected radio button from the add form
692      * @return
693      */
694     private String radioSelection(){
695         String nature ="";
696         if (radioLoyal.isSelected()){
697             nature = radioLoyal.getText();
698 
699         } else if (radioAgg.isSelected()) {
700             nature = radioAgg.getText();
701 
702         } else if (radioPas.isSelected()) {
703             nature = radioPas.getText();
704 
705         } else if(radioInq.isSelected()) {
706             nature = radioInq.getText();
707 
708         }
709         return nature;
710     }
```

```
712     /**
713      * this method return the selected check box from the add form
714      * @return
715      */
716     private String checkSelection() {
            String size = "";
718
719         if (checkBig.isSelected()){
720
721             size = checkBig.getText();
722             checkMed.setSelected(false);
723             checkSmall.setSelected(false);
724
725         } else if (checkMed.isSelected()) {
726
727             size = checkMed.getText();
728             checkBig.setSelected(false);
729             checkSmall.setSelected(false);
730
731         } else if (checkSmall.isSelected()) {
732
733             size = checkSmall.getText();
734             checkBig.setSelected(false);
735             checkMed.setSelected(false);
736         }
737         return size;
738     }
739
740     public void setValue() {
741         name2.setText(name);
742         category2.setText(category);
743         age2.setText(age);
744         lifespan2.setText(lifespan);
745         checkSize(size);
746         checkNature(nature);
747         quantity2.setText(quantity);
748         price2.setText(price);
749     }
791     // Variables declaration - do not modify
792     private javax.swing.JTextField age2;
793     private javax.swing.JTextField category2;
794     private javax.swing.JCheckBox checkBig;
795     private javax.swing.ButtonGroup checkBox2;
796     private javax.swing.JCheckBox checkMed;
797     private javax.swing.JCheckBox checkSmall;
798     private javax.swing.JButton imageChoose;
799     private javax.swing.JLabel jLabel1;
800     private javax.swing.JLabel jLabel10;
801     private javax.swing.JLabel jLabel11;
802     private javax.swing.JLabel jLabel12;
803     private javax.swing.JLabel jLabel13;
804     private javax.swing.JLabel jLabel14;
805     private javax.swing.JLabel jLabel15;
806     private javax.swing.JLabel jLabel16;
807     private javax.swing.JLabel jLabel2;
808     private javax.swing.JLabel jLabel4;
809     private javax.swing.JLabel jLabel6;
810     private javax.swing.JLabel jLabel7;
811     private javax.swing.JLabel jLabel8;
812     private javax.swing.JLabel jLabel9;
813     private javax.swing.JPanel jPanel1;
814     private javax.swing.JPanel jPanel2;
815     private javax.swing.JPanel jPanel3;
816     private javax.swing.JTextField lifespan2;
817     private javax.swing.JLabel logo;
818     private javax.swing.JTextField name2;
819     private javax.swing.JTextField price2;
820     private javax.swing.JTextField quantity2;
821     private javax.swing.JRadioButton radioAgg;
822     private javax.swing.ButtonGroup radioButton2;
823     private javax.swing.JRadioButton radioInq;
824     private javax.swing.JRadioButton radioLoyal;
825     private javax.swing.JRadioButton radioPas;
826     private javax.swing.JButton update;
827     // End of variables declaration
828 }
829
```

SelectionSort Class Code

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

import java.util.ArrayList;
import java.util.Collections;

/**
 *
 * @author Dell
 */
public class SelectionSort {
    /**
     * the array value are swapped according to index
     * @param a
     * @param i
     * @param j
     */
    public static void swap(int[] a, int i, int j)
    {
        int temp = a[i];
        a[i] = a[j];
        a[j] = temp;
    }
```

```java
    /**
     *
     * @param a array t sort
     * @param array ArrayList to sort
     * minimumPosition method is called to find the minimum position
     * swap method is also called
     * Collections.swap is also called to sort ArrayList
     *
     */
public static void sortArray(int [] a,ArrayList array)
    {
        for (int i = 0; i < a.length-1; i++)
        {
            int minPos = minimumPosition(a, i);
            swap(a, minPos, i);
            Collections.swap(array,minPos,i);
        }

    }

/**
 * finding the minimum position of the array
 * @param a
 * @param from
 * @return
 */
    public static int minimumPosition(int[] a, int from)
    {
        int minPos = from;
        for (int i = from + 1; i < a.length; i++)
        {
            if (a[i] < a[minPos]) {
                minPos = i;
            }
        }
        return minPos;
    }

}
```

AppHelp Class Code

```
1   package Dashboard;
2
3   /*
4    * To change this license header, choose License Headers in Project Properties.
5    * To change this template file, choose Tools | Templates
6    * and open the template in the editor.
7    */
8
9   /**
10   *
11   * @author Dell
12   */
13  public class AppHelp extends javax.swing.JFrame {
14
15      private static AppHelp help = null;
16
17      /**
18       * Creates new form AppHelp
19       */
20      private AppHelp() {
21          initComponents();
22      }
23
24      public static AppHelp getObj(){
25          if (help == null) {
26              help = new AppHelp();
27          }
28
29          return help;
30      }
31
32      /**
33       * This method is called from within the constructor to initialize the form.
34       * WARNING: Do NOT modify this code. The content of this method is always
35       * regenerated by the Form Editor.
36       */
37      @SuppressWarnings("unchecked")
38      Generated Code
231
232     /**
233      * @param args the command line arguments
234      */
```

```
235     public static void main(String args[]) {
236         /* Set the Nimbus look and feel */
237         Look and feel setting code (optional)
258
259         /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
262                 new AppHelp().setVisible(true);
263             }
264         });
265     }
266
267     // Variables declaration - do not modify
268     private javax.swing.JLabel jLabel1;
269     private javax.swing.JLabel jLabel10;
270     private javax.swing.JLabel jLabel2;
271     private javax.swing.JLabel jLabel3;
272     private javax.swing.JPanel jPanel1;
273     private javax.swing.JPanel jPanel2;
274     private javax.swing.JPanel jPanel3;
275     private javax.swing.JScrollPane jScrollPane1;
276     private javax.swing.JScrollPane jScrollPane2;
277     private javax.swing.JSeparator jSeparator1;
278     private javax.swing.JSeparator jSeparator2;
279     private javax.swing.JTable jTable1;
280     private javax.swing.JTable jTable2;
281     // End of variables declaration
282 }
```