Module Code & Module Title

CS5004NA Emerging Programming Platforms and Technologies

Assessment Weightage & Type

30% Group Coursework

Home Appliances Inventory System

Year and Semester

2020-21Autumn

Group Name: Home Appliances group		
SN	**Student Name**	**College ID**
1	**Bijay Bharati**	**NP01CP4A190041**
2	**Pukar Neupane**	**NP01CP4A190272**
3	**Sushant Pant**	**NP01CP4A190331**

Table of Contents

Table of Figures

1 Proposal

1.1 Brief Introduction

This is an information system for home appliances, the title is "**Home Appliances Inventory System**", which gets straight to the point and is clear to the user about what it is. It will store details about different home appliances and display those details in a table and allow users to add more details. The system will also have search functionality with intuitive GUI which is clean and simple to use. The main purpose of the system is to provide users with easy way to access data for home appliances.

1.2 List of Data

The table below gives the information about the data that is stored in the table by our system.

Title	Purpose	Type
Model Number	Model Number uniquely identifies each product/ appliance.	String
Name	Name is the name of the product.	String
Category	Category for the product (entertainment, security, etc.).	String
Discount	How much discount is available? (we are assuming maximum discount cannot be more than 33% with minimum discount being 0%)	Integer
Price	Price will be the marked price for the product.	Integer
Rating	Rating will give the general idea about the quality of the product. (we are assuming the maximum rating a product can have is 5 and minimum is 1)	Integer

1.2 Features

The features of the program are listed below.

- The GUI for the application is without distractions and everything is labelled so that the user can easily use the system. Help file is also provided for better understanding of the system.
- CSV file with data about different products can be imported from the file menu bar, the csv file is checked before import so that only correct files may be imported.
- Text fields to input model number, name and price, a combo box to select category, a slider to input discount, radio buttons for rating are provided so that users can easily add data.
- There is an update button to update or change data in the table once they have been added, the same fields that are used to add data to the table are used to update data.
- There is a delete button so that users can easily delete unwanted data.
- There is a combo box with price and category, which allows the users to choose if they want to search items in the table based on price or based on category. If the users select price, a text box will appear where they can search for price, if the users select category, a combo box with categories will appear and the users will select which category they want to search for.
- Buttons will be there to search, add data, update and delete data and reset all the input fields.
- The users can also clear the table through the appropriate menu bar item.
- Code for the project is descriptive with proper names for all the variables, comments have been made where necessary and the code is easy to read and understand.

1.3 Tools used

NetBeans was used for this project. NetBeans is one of the most popular and feature rich IDEs for java development, it is an open-source project managed by the Apache foundation. We picked NetBeans because it allows us to drag and drop the GUI components which gives us more time to work on the backbone of the project i.e. work on the logic of our program.

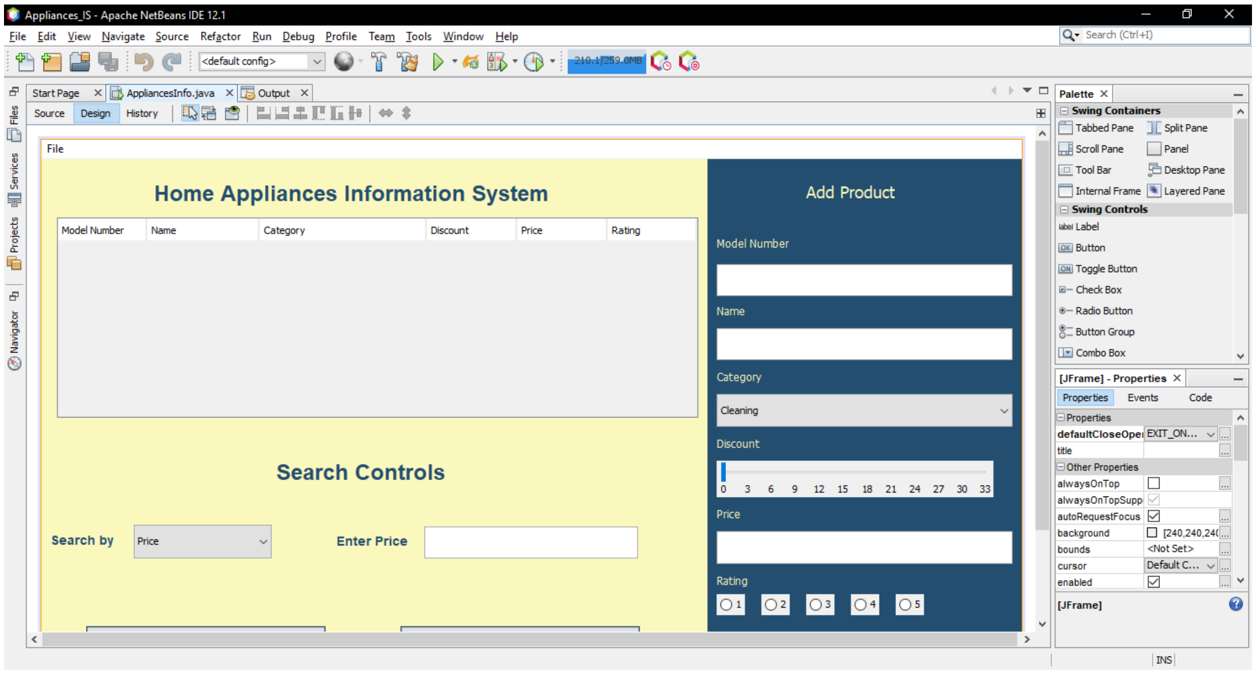


Figure 1:Working in NetBeans

As seen in the figure, the system was completely developed in NetBeans. GUI components can be easily added and modified. The coding was also done in NetBeans IDE.

The program is powered by java i.e. coding is done in java. Java version 15 was used for the project because of the updates and improvements from the previous versions, therefore it is highly recommended for users to use a version of java that is equal to or higher than version 15 to minimize the chances of encountering errors.

2 Individual Tasks

Bijay Bharati:

- GUI design, explanatory comments for code
- Data validations during add and update duplication checks, key typed, empty fields check
- Implemented functionality for update, add, search buttons, created help file
- Implementation of selection sort for 2d array and binary search in program for price and search for category
- Report: Proposal, Introduction, Binary Search 4,4.1,4.2, method description 6.1-6.12, Conclusion, References 1,2,3, Appendix

Pukar Neupane:

- GUI design, explanatory comments for code
- Implementation of csv file import, delete, clear table, validations for file import in table
- Implementation in 2d array creation in search
- Report: Introduction, Selection sort 5,5.1,5.2, method description 6.13-6.16, testing 7.7-7.12 Conclusion, References 4,5, Appendix

Sushant pant:

- GUI design, explanatory comments for code
- Creation of help file
- Implementation for data validations, creation of 2D array
- Implementation for help menu option and opening help pdf file
- Report: Introduction, method description 6.17-6.20, testing 7.1-7.6, Conclusion, Appendix

3 Introduction

This group coursework demonstrates the understanding on sorting and search algorithms to manipulate data. The core of the coursework is based on using binary search algorithm and sorting algorithm to take the data from our table and display the data according to certain parameters like price and category.

The sorting algorithm used in this project and the implementation on binary search is further explained in the coming sections of the report. The coursework also contains a GUI that makes it easier for users to navigate the system.

4 Binary Search Algorithm

Let us start by knowing what an algorithm is before we talk about Binary Search.

Informally, an algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output. An algorithm is thus a sequence of computational steps that transform the input into the output. We can also view an algorithm as a tool for solving a well-specified computational problem. The statement of the problem specifies in general terms the desired input/output relationship. The algorithm describes a specific computational procedure for achieving that input/output relationship. (Cormen, 2009)

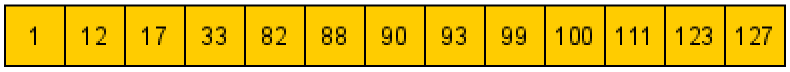
Binary search is also an algorithm. When we are handling any data, we will eventually require to find a specific part of the data. Searching algorithms are used in this case.

Binary search is a searching algorithm which works on a sorted table by testing the middle of an interval, eliminating the half of the table in which the key cannot lie, and then repeating the procedure iteratively. (Wessistene, 2020)

Binary search will return the position of the element in the array i.e. the index of the searched value.

4.1 Working Mechanism of Binary Search Algorithm

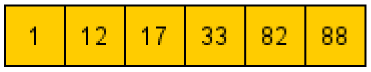
Binary search works on a sorted array by dividing the array into halves and comparing the values. The example below explains this process.



1	12	17	33	82	88	90	93	99	100	111	123	127

Figure 2: Example of sorted array

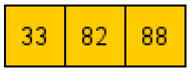
Suppose we want to search for 88. Then first, the binary search algorithm finds the mid-point for the array, in this case, midpoint is 90. Now 88 is compared to 90, 88 is smaller than 90 so, in a sorted array, 88 would be in the left side, now the array is further broken down and we get the following array.



1	12	17	33	82	88

Figure 3: Example 2 of sorted array

New midpoint now is 17. 88 is greater than 17 so, right-side of the array is taken and we get the following result.



33	82	88

Figure 4: Example 3 of sorted array.

Again, the step is repeated. 82 is the midpoint and since 88 is greater than 82, right-side of the array is taken and since 88 is the only value left, 88 is returned for our search result.

Binary search is very efficient for searching through large amount of data, instead of looking at a data from beginning to end, we essentially eliminate half of the data that we have to search through in each iteration of the search process, this saves a lot of time and resource.

Binary search is especially fast when we perform search on very large data because unlike search algorithms, where each element of the data is examined, binary search will only look for relevant data according to the value we are searching for.

If we start with n number of data, in first iteration we reduce it to $\frac{n}{2}, \frac{n}{4}, \frac{n}{8}$ and so on. The maximum number of times we have to divide the data is given by the following logarithmic expression: O(log n). The best case scenario is when the middle element is the element which we are searching for in which case, the expression is O(1).

Books in the references section can be referred for additional mathematical verifications on binary search.

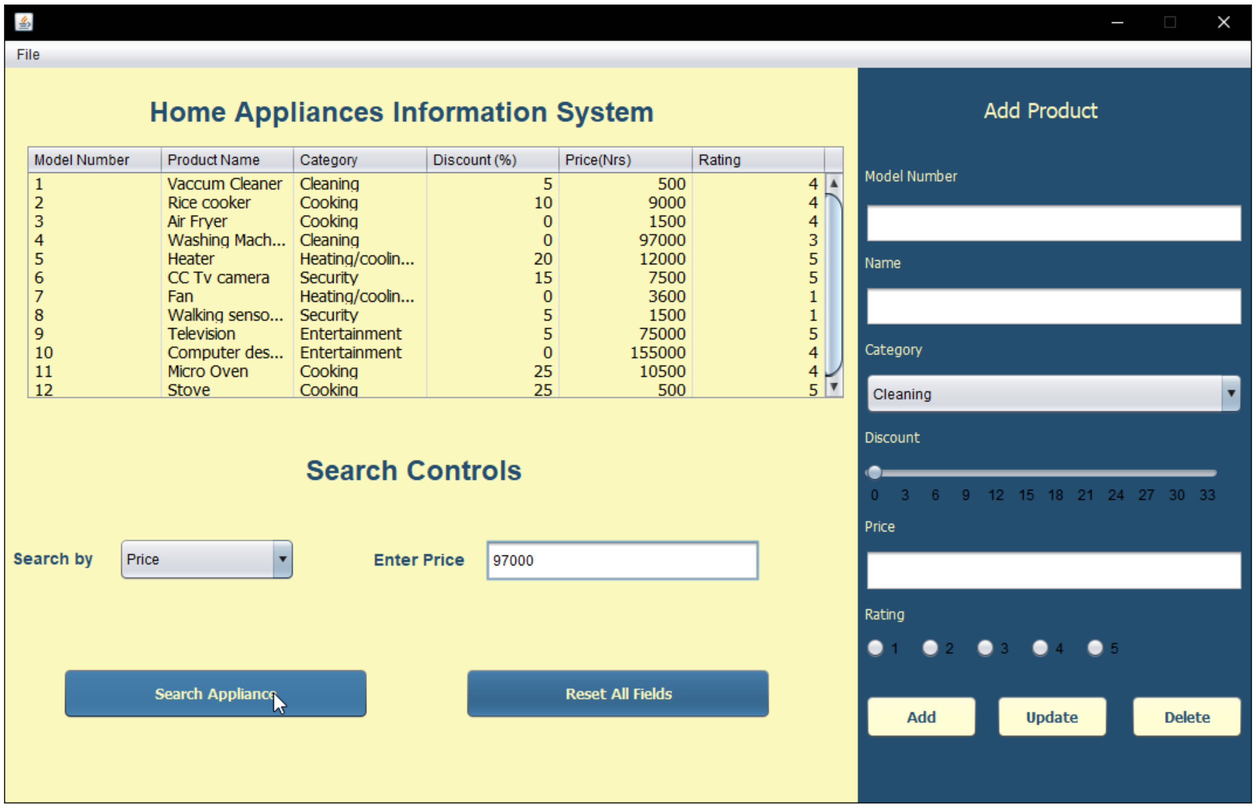
4.2 Implementation of Binary Search in program

This section of the report explains how binary search has been implemented in the program. We have a java class called BinarySearch which contains method called search the search method accepts 4 parameters an array of sorted integers, low point, high point and the value we are searching for.

Binary search is implemented for searching price in our program. Data is taken from the table and sorted via selection sort. The data from the whole table is taken. This data includes model number, name, category, discount and rating along with price. The data is stored in two-dimensional array as string.

The data that we take is consistent with the fifth element, 4th index of the array being data for price which is stored as string. We are sorting the arrays within the array based on the fourth index. Once the array is sorted, we extract only the price from the sorted list in a separate array. Then we pass the pricelist to our search method in BinarySearch class along with low being 0, high the length of the sorted price list and the value we are searching for.

Since the index in the pricelist and our sorted array with all the table data will match with each other, once we get the index for the value we searched in the pricelist, we can use the same index to extract the value from the 2 dimensional sorted array and list other details about the product based on price. The figures below will help visualize the points discussed here.



Home Appliances Information System

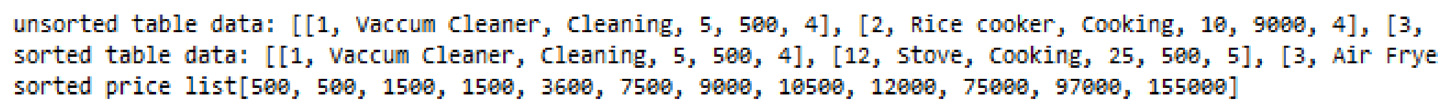
Model Number	Product Name	Category	Discount (%)	Price(Nrs)	Rating
1	Vaccum Cleaner	Cleaning	5	500	4
2	Rice cooker	Cooking	10	9000	4
3	Air Fryer	Cooking	0	1500	4
4	Washing Mach...	Cleaning	0	97000	3
5	Heater	Heating/coolin...	20	12000	5
6	CC Tv camera	Security	15	7500	5
7	Fan	Heating/coolin...	0	3600	1
8	Walking senso...	Security	5	1500	1
9	Television	Entertainment	5	75000	5
10	Computer des...	Entertainment	0	155000	4
11	Micro Oven	Cooking	25	10500	4
12	Stove	Cooking	25	500	5

Search by: Price
Enter Price: 97000

Model Number:
Name:
Category: Cleaning
Discount: 0 3 6 9 12 15 18 21 24 27 30 33
Price:
Rating: 1 2 3 4 5

Figure 5: Searching by price

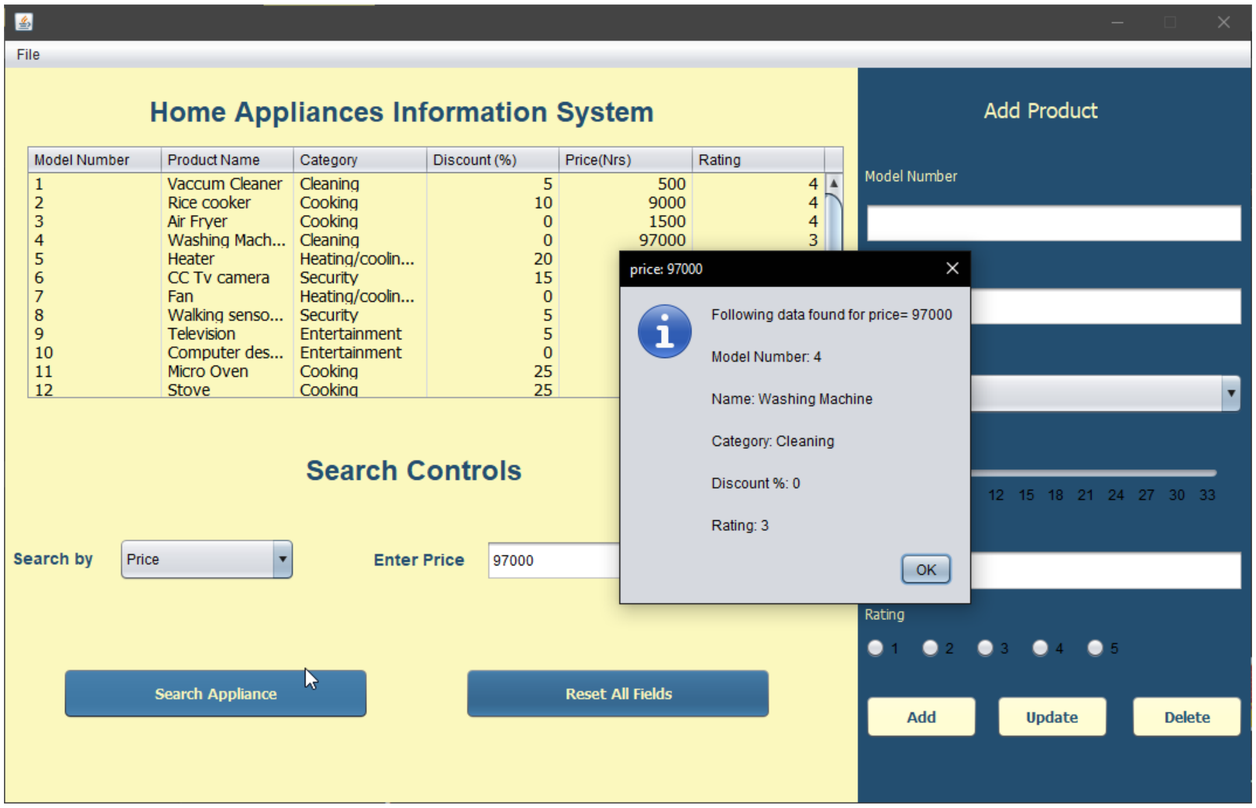
In the figure we can see that user is searching for items with price 97000. The data in table is not arranged according to price.



```
unsorted table data: [[1, Vaccum Cleaner, Cleaning, 5, 500, 4], [2, Rice cooker, Cooking, 10, 9000, 4], [3,
sorted table data: [[1, Vaccum Cleaner, Cleaning, 5, 500, 4], [12, Stove, Cooking, 25, 500, 5], [3, Air Frye
sorted price list[500, 500, 1500, 1500, 3600, 7500, 9000, 10500, 12000, 75000, 97000, 155000]
```

Figure 6: Working mechanism

In figure 6 we can see the steps described above. Only a part of the data is shown because the data was too large to display properly in A4 size. But we can see first data from whole table is extracted, then it is sorted according to price and list of prices is also extracted from the sorted table data.



Model Number	Product Name	Category	Discount (%)	Price(Nrs)	Rating
1	Vaccum Cleaner	Cleaning	5	500	4
2	Rice cooker	Cooking	10	9000	4
3	Air Fryer	Cooking	0	1500	4
4	Washing Mach...	Cleaning	0	97000	3
5	Heater	Heating/coolin...	20		
6	CC Tv camera	Security	15		
7	Fan	Heating/coolin...	0		
8	Walking senso...	Security	5		
9	Television	Entertainment	5		
10	Computer des...	Entertainment	0		
11	Micro Oven	Cooking	25		
12	Stove	Cooking	25		

price: 97000

Following data found for price= 97000

Model Number: 4

Name: Washing Machine

Category: Cleaning

Discount %: 0

Rating: 3

OK

Figure 7: Result for searching by price

And finally result is displayed to the user. All this happens very fast and it happens in the same order as described above.

5 Selection Sort Algorithm

Sort algorithms or functions are used to iterate through every elements in a data structure; like list, array, array list and so on which has a collection of elements, with the purpose of achieving sorted sequence of the elements in ascending or descending order. Numerous sort algorithms are available in the computing world, but selection sort is the one opted for this project. Some other algorithms are bubble sort, merge sort, insertion sort and so on. The reason behind selecting selection sort is its simplicity of the working mechanism.

5.1 Working Mechanism of the Selection Sort

Selection sort is one sorting algorithm which is based on the principal of swapping lowest element available to the first position of the unsorted array. The algorithm starts from the first element and search for the lowest element available in the array to swap to the position of first element. When the swap is complete, then it moves to the second element and again searches for the lowest element starting from second element. The function then does not manipulate the first element which is sorted part of the array. Likewise, the process continues till the last element where sorting is completed as all the positions before are already sorted. The sorted arrays keeps on moving to the right with unsorted part becoming smaller. The loop ends at the last element of the array.

For example: Lets take a list [7,5,8,3,2] of five elements.

In step 1;

2 is swapped with 7 at first position after which the list becomes [2,5,8,3,7].

In step 2 ;

3 is swapped with 5 at second position after which the list becomes [2,3,8,5,7].

In step 3;

5 is swapped with 8 at third position after which the list becomes [2,3,5,8,7].

In step 4;

7 is swapped with 8 at fourth position after which the list becomes sorted as [2,3,5,7,8].

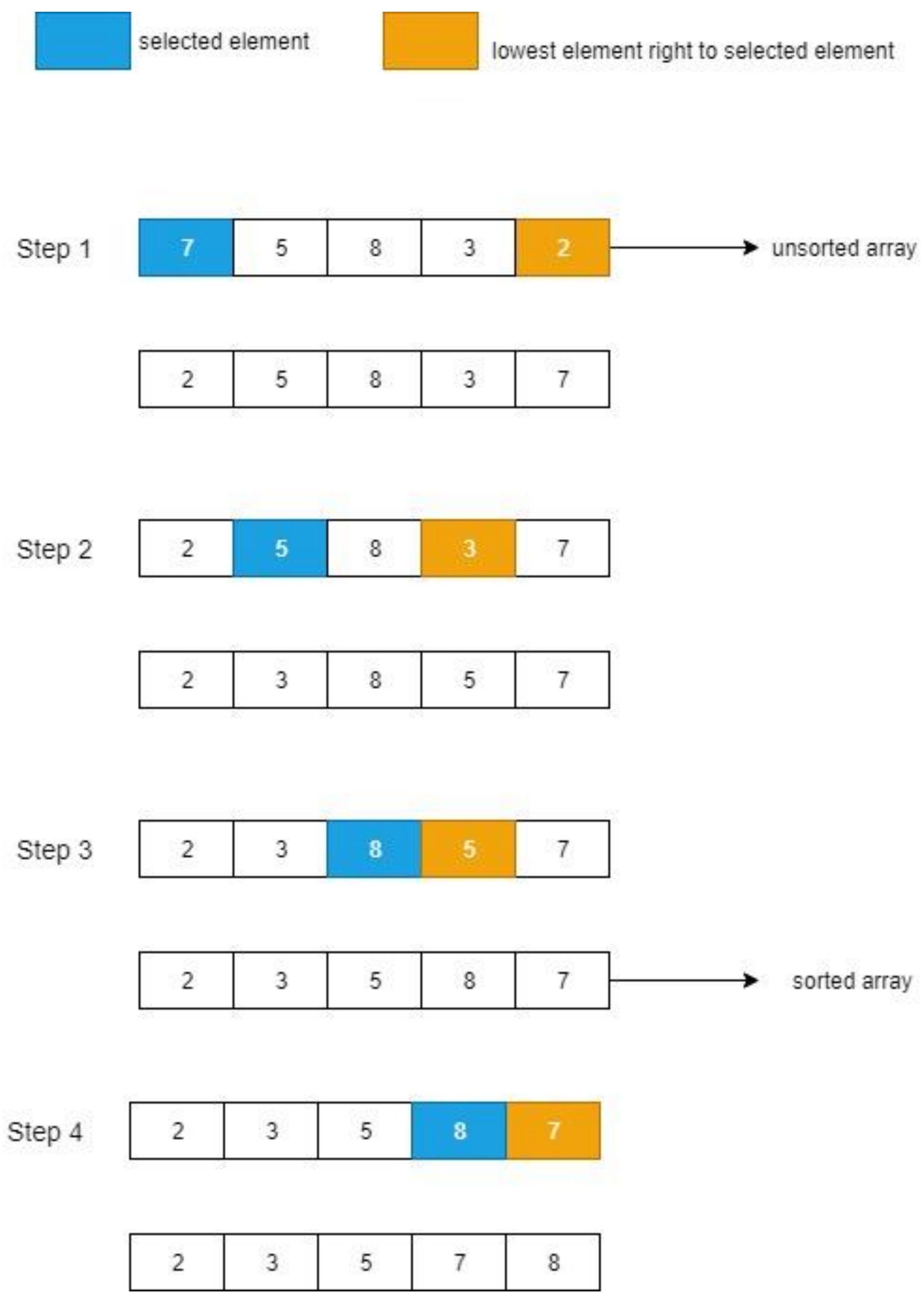
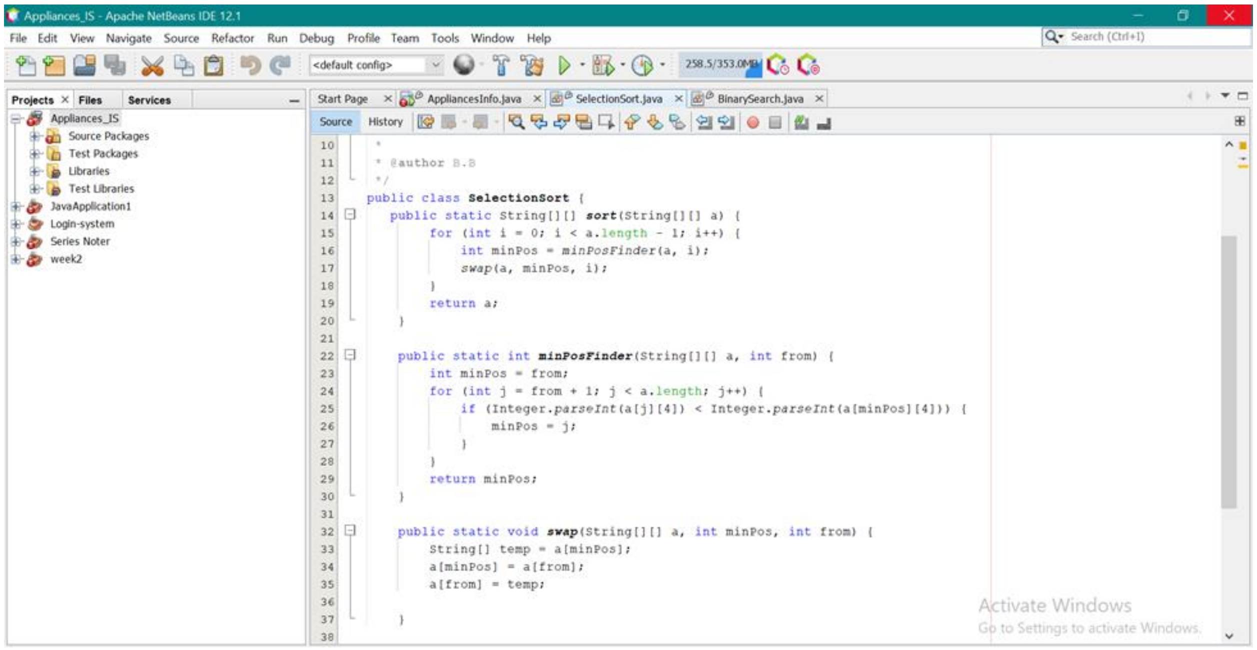


Figure 8: Selection sort example

5.2 Implementation of Selection Sort in program



```java
 *
 * @author B.B
 */
public class SelectionSort {
   public static String[][] sort(String[][] a) {
        for (int i = 0; i < a.length - 1; i++) {
            int minPos = minPosFinder(a, i);
            swap(a, minPos, i);
        }
        return a;
    }

    public static int minPosFinder(String[][] a, int from) {
        int minPos = from;
        for (int j = from + 1; j < a.length; j++) {
            if (Integer.parseInt(a[j][4]) < Integer.parseInt(a[minPos][4])) {
                minPos = j;
            }
        }
        return minPos;
    }

    public static void swap(String[][] a, int minPos, int from) {
        String[] temp = a[minPos];
        a[minPos] = a[from];
        a[from] = temp;

    }
```

Figure 9:Implementing Selection Sort

The selection is defined in a new class with in the same source package of the application. The major part is the first function sort() which accepts a 2d array of string. Since the program has different data types, string array is selected to store all data of the table including integers as well. The integers are also stored as string but later converted to integers as per need. The input array is obtained from data of table. Two separate functions minPosFinder() and swap() are called within this main method.

the function, minPosFinder() is responsible for iterating through the array but only at the 4th index of each row as price column lies in the respective index. It starts from the position next to the selected position and move to its right searching for lower element than the element at selected position. Whenever a lower value is found , the index of row of that element is set as minimum position and is again compared with rest of the values. Likewise, the lowest element is found for a certain index in one loop of this function.

After the minimum position(row with lowest value of price) is found, it needs to be swapped to the selected index which is performed by the last function, swap(). It takes

2d string array, position to swap and position to be swapped. A variable is created to store the value to be swapped. Then the values are swapped.

The sort() function calls these two methods on an array and returns the same array but with changed row indices as a sorted array based on the value present in price column of the table.

6 Method Descriptions

First, we will give short descriptions for the methods in AppliancesInfo followed by BinarySearch and SelectionSort.

6.1 addDataBtnActionPerformed

This method handles the task for add button. When add button is pressed data is added to the table and user is given a message about the data being successfully entered and the number ow rows in the table, but before that following conditions are checked:

- Module number text field is not empty, and the model number entered by the user is unique i.e. not used in the table another method called **modelNumberExixts** is used to verify model number is unique.
- The name text field is not empty.
- Price text field is not empty.
- Rating is selected.

(note: we do not check for category and discount because by default they have values Cooking and 0 %)

If all the conditions are satisfied, then data is entered and **resetDataFields** method is called to clear the input fields else, user is prompted to enter the data by a warning message.

6.2 modelNumberExists

This method returns a boolean value. If this method returns true, it means that the model number is already in use in the table. If the table contains one or more data, this method extracts data from the 0 index of each row of the table which is the model number. Then the model number entered by user in model number text field is compared with the data extracted from the table if a match is found, true is returned else false is returned.

6.3 updateBtnActionPerformed

This method handles the task for update button. The same fields used to enter data are used to update/ modify the data in the table. First it checks if the user has selected a row, if not a warning is given reminding the user to select a row.

When a row is selected, values from the input fields are taken. A warning is given if a user tries to update model number, model numbers are permanent and thus are not allowed to be changed. The method also keeps track of how many changes have been made and what fields have been updated. At the end, a summary of the changes made is given to the user and **resetDataFields** method is called to clear the input fields.

If empty fields are left in the input fields when updating data, they are ignored.

6.4 deleteDataBtnActionPerformed

This method handles the functionality of delete button. When the delete button is pressed, it is checked of a row is selected, if a row is not selected, the user is reminded to select a row. When the user selects a row and presses the delete button, the user is presented with a dialog box reminding the user that deleted data cannot be retrieved if the user selects the yes option, the row is deleted and user is updated about the total number of rows in the table if no is selected by the user, the dialog box disappears and nothing happens.

6.5 searchAppliancesBtnActionPerformed

This method is responsible for searches. All the data from the tables are extracted into a 2D array in this method and the data is sorted using **sort** method from SelectionSort.

Users can search according to price or according to category. There is a combo box which allows users to choose on what basis they want to perform the search. This method determines which value is selected for the combo box.

If user has selected price in the combo box, a text field will appear where the users can enter the price and press search button to search button to get results. If the users have selected category, then another combo box will appear where users can select a category to get results for the search.

when the user is searching according to price, **search** method from BinarySearch is used to search for price and only a single product's details are displayed whose price matches with the price that the user has entered. The details of how this is performed is mentioned in the previous section on the report (see section 4.2).

when the user is searching according to category, all the data is examined from the sorted list and where category is same as the one selected by the user, the data is displayed for the products in the table with their name, price and model number. String builders are used for this. If search results do not find any data, a suitable message is displayed.

6.6 ResetAllFieldsBtnActionPerformed

This method handles the function for the button to clear all fields. When the reset all fields button is pressed, all the input fields and the search options are reset to empty or default, this method is invoked which further calls **resetDataFields** and **resetSearchFields** methods.

6.7 resetDataFields

This method sets the model number text field, name text field, price field to empty, category combo box selected index to 0; discount slider to 0 and removes selection from the radio buttons.

6.8 resetSearchFields

This method sets the select index of search options combo box and select category combo box to 0 and price search text field to empty.

6.9 discountSliderStateChanged

This method is responsible for changing the value of the label present beside the discount slider to the corresponding value of the discount slider for the user. If the discount slider is at 0, this label has empty string as value else the value of the label is the string value of the discount slider.

6.10 priceTxtFldKeyTyped and priceSearchTxtFldKeyTyped

These methods make sure that only numeric values are entered in text fields for price. When a key is pressed in keyboard, it will only be processed, registered if the key is a numeric key (0,1,2,3,4,5,6,7,8,9) else the key press will be ignored.

6.11 searchOptionsComboBoxActionPerformed

This method shows the correct input fields when performing search. When the user wants to search according to price, price is selected in the search by combo box, this will hide select category and only display price search text field. When search by category is selected, input field for price search is hidden and combo box to choose category is made visible.

6.12 nameTxtFldKeyPressed

This method validates the input field for product name. This method makes sure that no special characters or numbers can be entered in the text field for product name. This field is only editable if the key pressed by the user is not special character like @,#,*etc.

6.13 Jmenuitem1 action performed (Open file)

This method is executed when the open file menu is clicked from the menubar for the purpose of importing data from a csv file to the table. A variable named imported is initialized as 0 which ensures no file is imported into the table of the GUI. If the value of imported is 0, an object of jfilechoooser is created so that user can select file to be uploaded from the device. Filter is applied to file chooser, so that only csv type files can be imported into the table. The file selected by the user is stored in variable f of object type file. Then, buffered reader object is initialized to read the file. The first line of the document is converted into the column headers of the table and rest lines are inserted as the values of that table in respective columns. All possible exceptions are avoided with the help of try and catch statements. After the table is updated with data from the chosen file, the imported is updated as 1 which disables the further importing of the files into the table.

6.14 Jmenuitem3 action performed (Clear table)

This method is executed when the user clicks on the clear table menu in the menubar. The purpose of this method is to clear all the data present in the table. But firstly, it takes confirmation from the user with a warning as the data once cleared cannot be retrieved again. If the user selects “Yes” option, the table is cleared. For that, number of rows are acquired through getRowCount() function and with a for loop, the rows are cleared one by one and totally at the end of the loop. And the imported variable is again set to 0 which enables to import file again into the table.

6.15 Jmenuitem4 action performed (Help)

When the user selects the Help menu, a help file is opened which is contained with the source package of the application. The open() method is called on the desktop class which takes a file as an argument. If no file is found, it throws an error with a message saying “File not found.”

6.16 Jmenuitem2 action performed (Exit)

This purpose of this menu is similar to the close button located at the top corner of the application. The only difference is that is takes confirmation from user before disposing the application.

6.17 Sort(String[][] a)

This method takes a two dimensional array as a collection of data from rows of table and returns the same array sorted in terms of values of price column. It calls two other methods minPosFinder() and swap() for this purpose. The sorting starts from first element and ends when the loop reaches last element resulting in a sorted array.

6.18 minPosFinder(String[][] a, int from)

This function takes a 2d array , same array as of sort() method. The loop starts from element next to the selected element. In each iteration, the value of price is compared to value of selected element. It returns the index value which holds the lowest value after the selected element.

6.19 (String[][] a, int minPos, int from)

As its name, it is solely responsible for swapping the row value of an array. It takes inputs as 2d array which is same array as above, row number with lowest value and selected row. It exchanges the index value of these two rows and uses a temporary variable for storing values to swap

6.20 search

This method handles binary search for an integer value in an array. This method returns an integer value which is the index in which we find the value we searched for. The array is divided to get the mid value if the mid value matches with the value we are looking for, we have found the value we are searching for and the operation is complete. If the mid value is greater than the value we searched for, we look towards the left of the mid value to the remaining array, if mid value was smaller, we look towards the right. This process is repeated until we find our value or only 1 value is left. If we did not find the value, -1 is returned.

7 Testing

7.1 open existing file

Objective:	To add past records (CSV file) in the table.
Action:	• File menu was opened and open file option was selected. • Then a valid csv file was selected via JFile chooser window.
Expected result:	Open/Shows existing csv file in the table.
Actual result:	Shows existing csv file in the table.
Conclusion:	Test is successful.

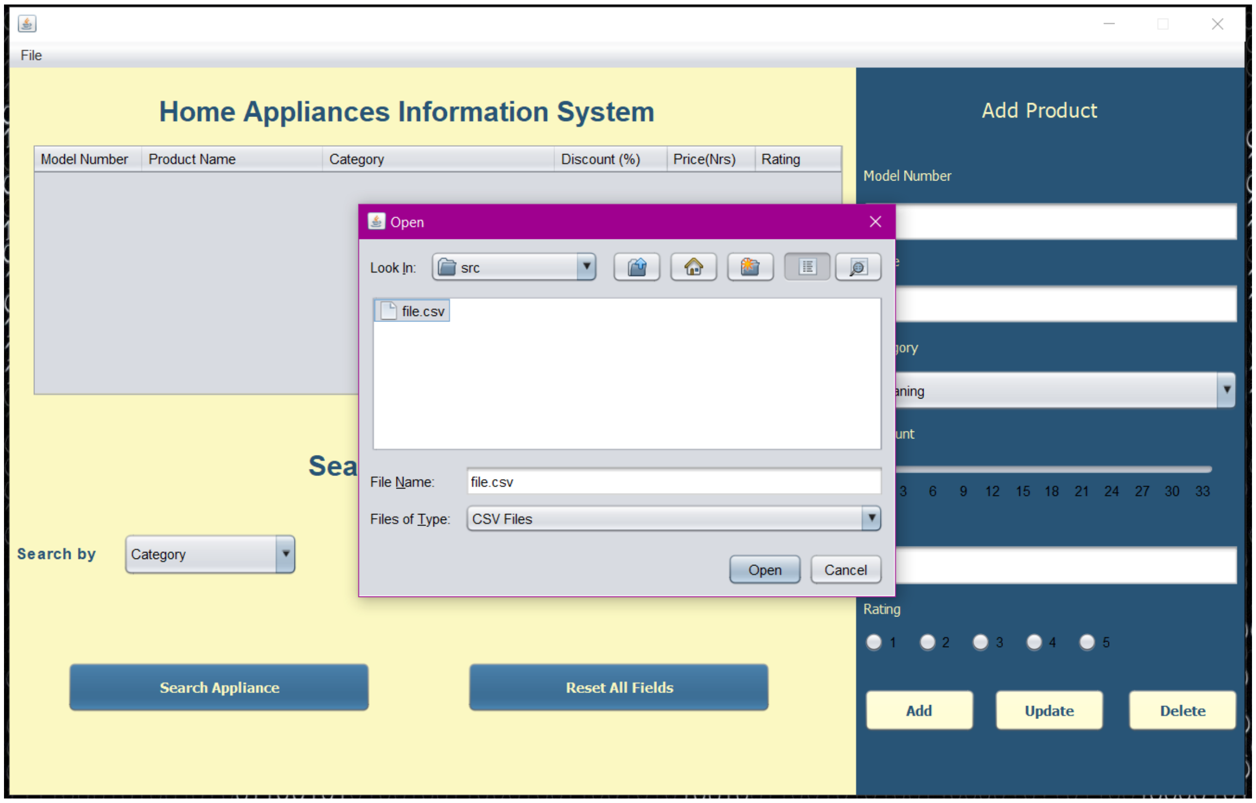
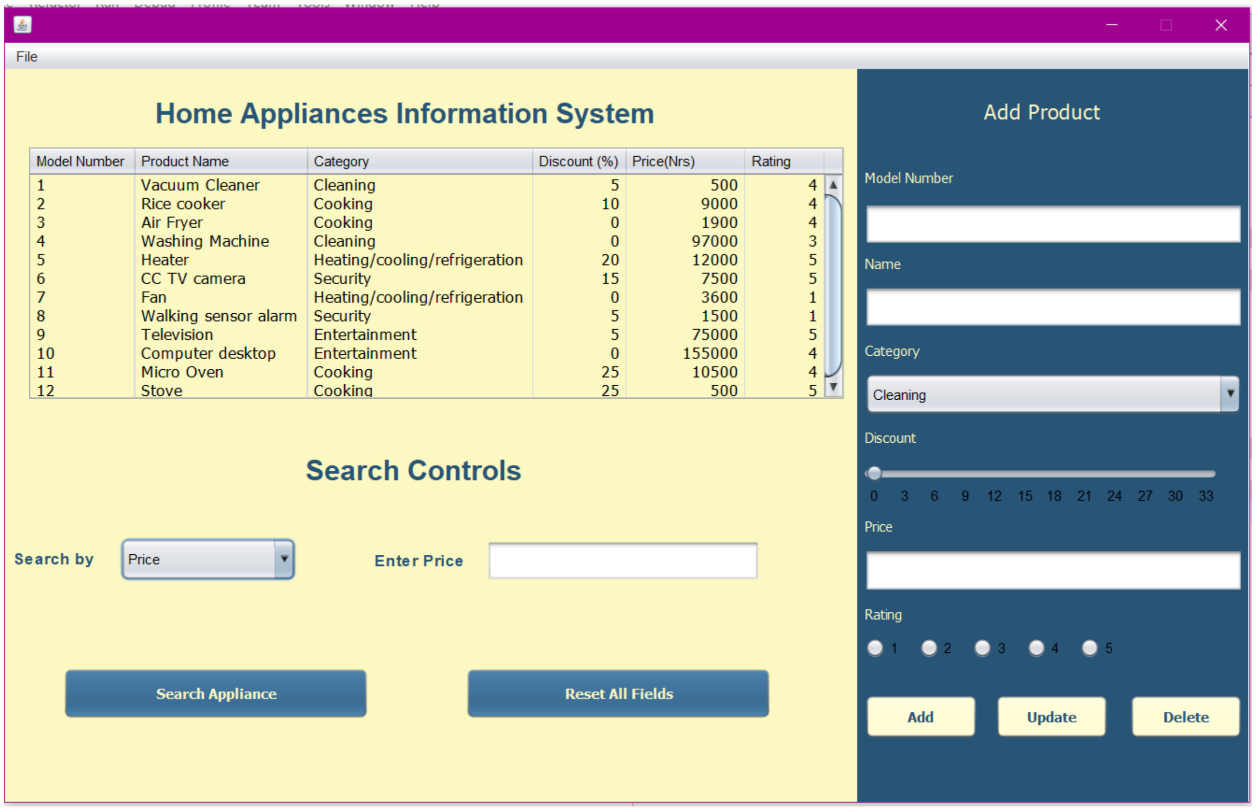


Figure 10: Test 1 opening csv file



Home Appliances Information System

Model Number	Product Name	Category	Discount (%)	Price(Nrs)	Rating
1	Vacuum Cleaner	Cleaning	5	500	4
2	Rice cooker	Cooking	10	9000	4
3	Air Fryer	Cooking	0	1900	4
4	Washing Machine	Cleaning	0	97000	3
5	Heater	Heating/cooling/refrigeration	20	12000	5
6	CC TV camera	Security	15	7500	5
7	Fan	Heating/cooling/refrigeration	0	3600	1
8	Walking sensor alarm	Security	5	1500	1
9	Television	Entertainment	5	75000	5
10	Computer desktop	Entertainment	0	155000	4
11	Micro Oven	Cooking	25	10500	4
12	Stove	Cooking	25	500	5

Search Controls

Search by: Price

Enter Price:

Model Number:

Name:

Category: Cleaning

Discount: 0 3 6 9 12 15 18 21 24 27 30 33

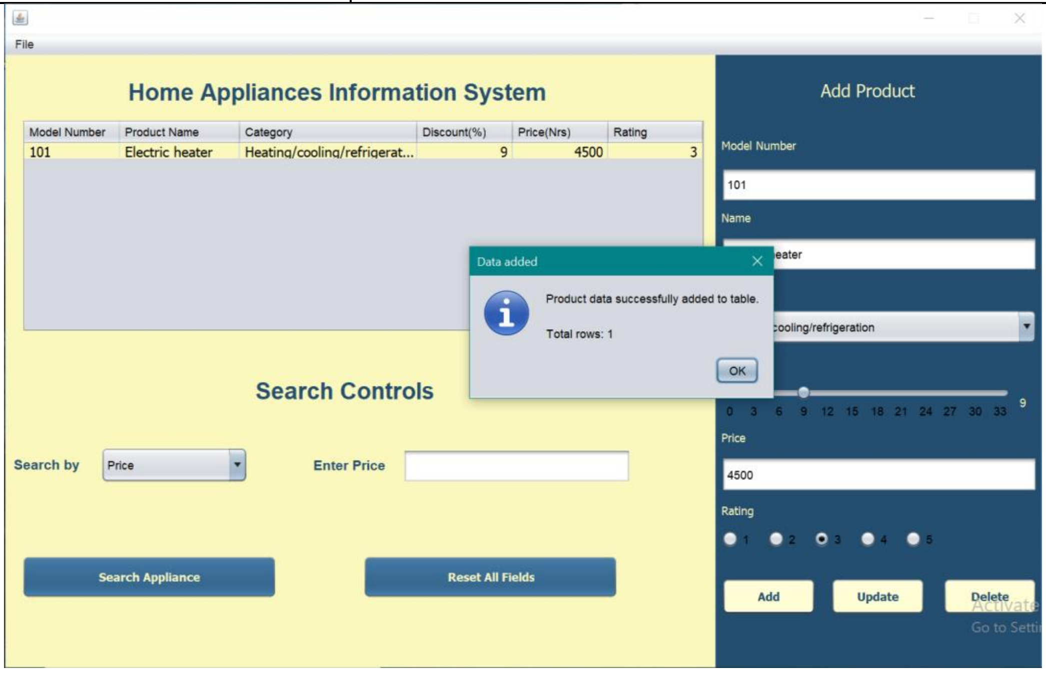
Price:

Rating: 1 2 3 4 5

Figure 11:Test 1 successful

7.2 Add data

Objective:	To add data in the table from "Add Product" section.
Action:	Following details are entered in the add product panel and add button is clicked. • Model number=101 • Name=Electric heater • Category=Heating/cooling/refrigeration • Discount=9 • Price=4500 • Rating=3
Expected result:	Product detail will be added in the table.
Actual result:	Details are added.
Conclusion:	Test is successful.



Model Number	Product Name	Category	Discount(%)	Price(Nrs)	Rating
101	Electric heater	Heating/cooling/refrigerat...	9	4500	3

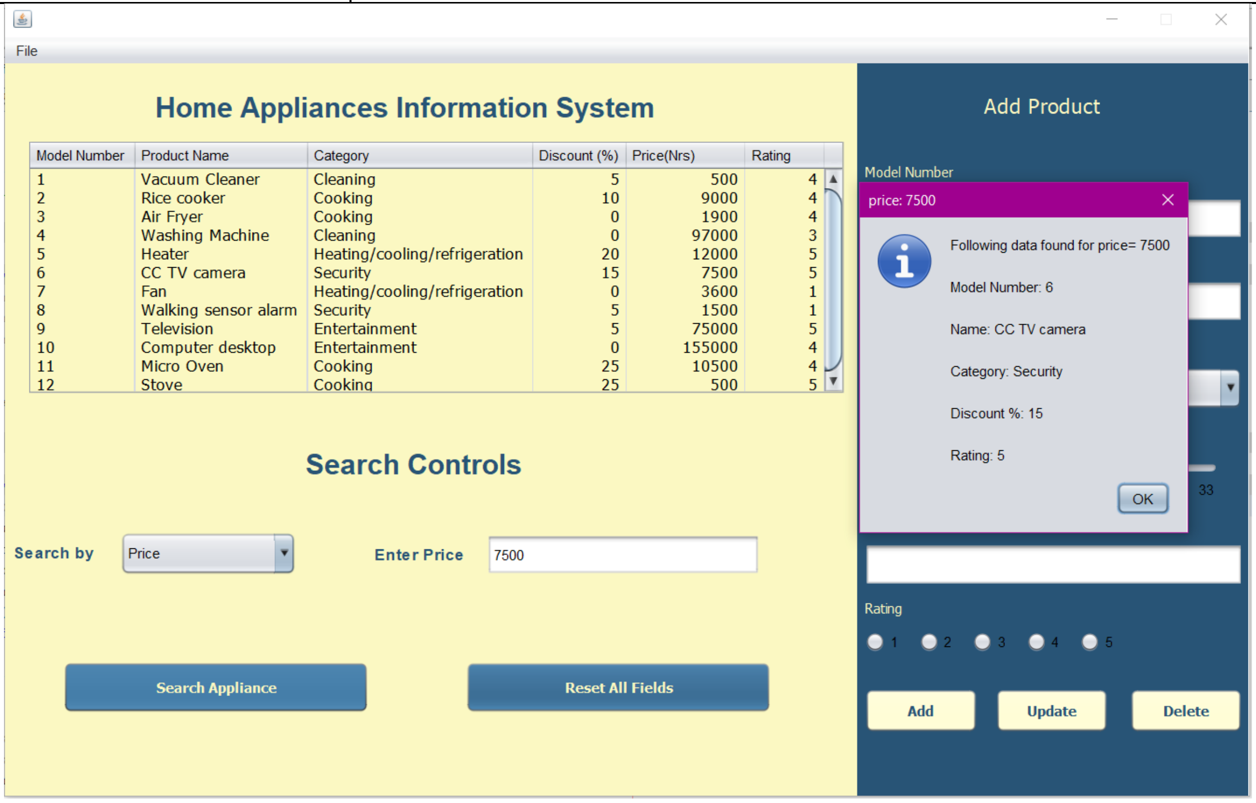
Model Number: 101
Name: eater
cooling/refrigeration
0 3 6 9 12 15 18 21 24 27 30 33 9
Price: 4500
Rating: 1 2 3 4 5

Data added
Product data successfully added to table.
Total rows: 1
OK

Figure 12: adding data

7.3 Searching price

Objective:	To search product according to their prices.
Action:	Price search was performed, 7500 was entered in the correct field and search appliance button was pressed
Expected result:	The details of the product with price 7500 would be displayed.
Actual result:	The details of the product with price 7500 is be displayed in message dialog box
Conclusion:	Test is successful.



Model Number	Product Name	Category	Discount (%)	Price(Nrs)	Rating
1	Vacuum Cleaner	Cleaning	5	500	4
2	Rice cooker	Cooking	10	9000	4
3	Air Fryer	Cooking	0	1900	4
4	Washing Machine	Cleaning	0	97000	3
5	Heater	Heating/cooling/refrigeration	20	12000	5
6	CC TV camera	Security	15	7500	5
7	Fan	Heating/cooling/refrigeration	0	3600	1
8	Walking sensor alarm	Security	5	1500	1
9	Television	Entertainment	5	75000	5
10	Computer desktop	Entertainment	0	155000	4
11	Micro Oven	Cooking	25	10500	4
12	Stove	Cooking	25	500	5

price: 7500

Following data found for price= 7500

Model Number: 6

Name: CC TV camera

Category: Security

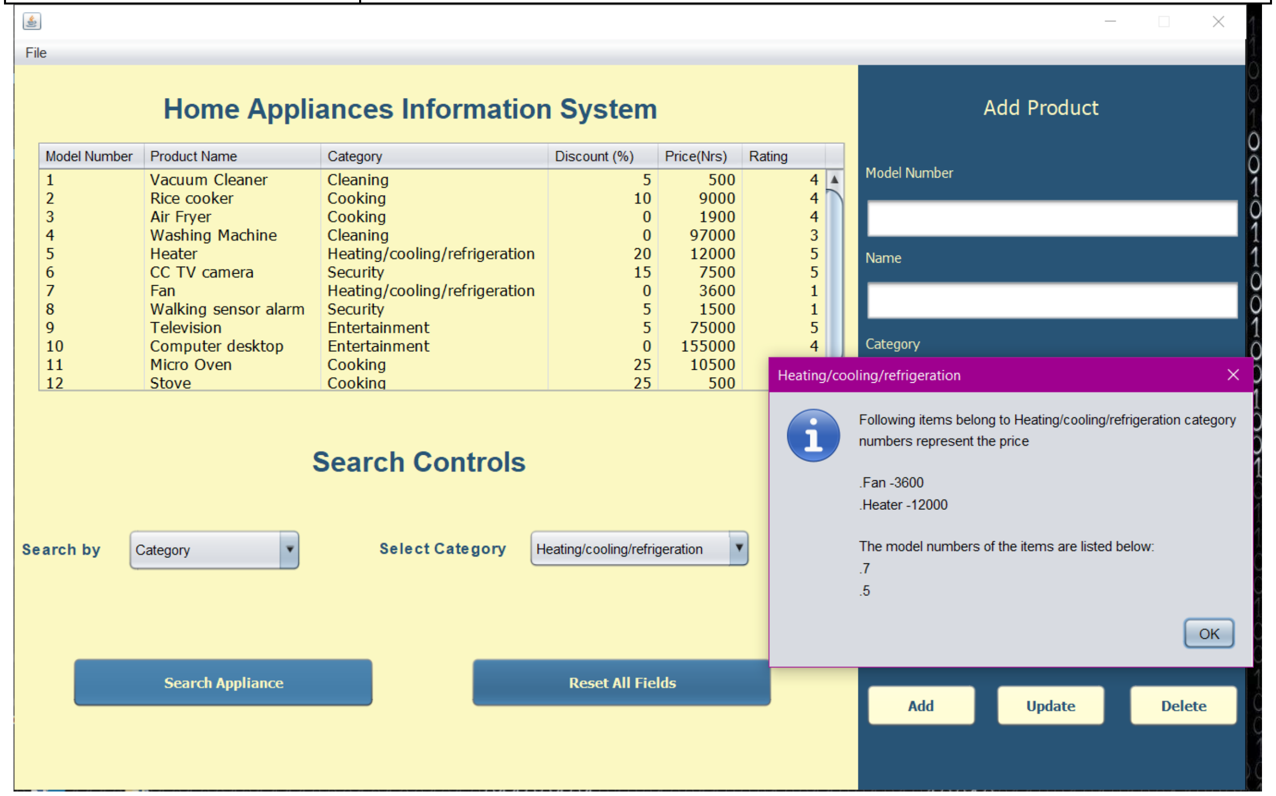
Discount %: 15

Rating: 5

Figure 13: Price search

7.4 Searching according to the category of the product

Objective:	To search product according to their category.
Action:	Search according to category was performed, desired category was selected and search appliance button was pressed,
Expected result:	Details of product of category, heating/cooling/refrigeration, should be displayed.
Actual result:	Details of product of category, heating/cooling/refrigeration, is displayed with product name, price and model number in an information dialog box.
Conclusion:	Test is successful.



Model Number	Product Name	Category	Discount (%)	Price(Nrs)	Rating
1	Vacuum Cleaner	Cleaning	5	500	4
2	Rice cooker	Cooking	10	9000	4
3	Air Fryer	Cooking	0	1900	4
4	Washing Machine	Cleaning	0	97000	3
5	Heater	Heating/cooling/refrigeration	20	12000	5
6	CC TV camera	Security	15	7500	5
7	Fan	Heating/cooling/refrigeration	0	3600	1
8	Walking sensor alarm	Security	5	1500	1
9	Television	Entertainment	5	75000	5
10	Computer desktop	Entertainment	0	155000	4
11	Micro Oven	Cooking	25	10500	
12	Stove	Cooking	25	500	

Heating/cooling/refrigeration

Following items belong to Heating/cooling/refrigeration category
numbers represent the price

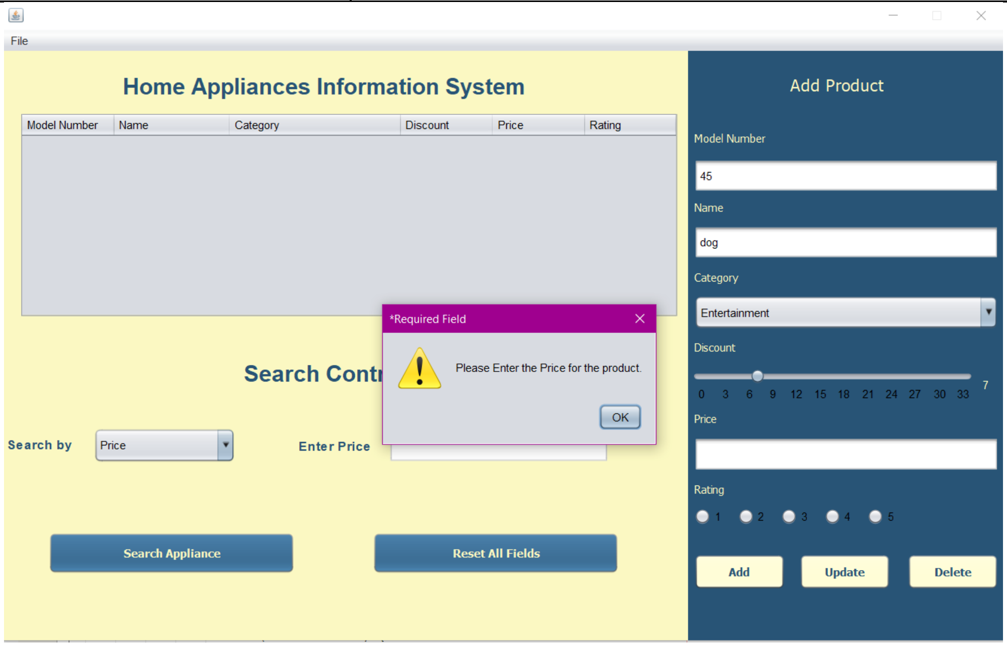
.Fan -3600
.Heater -12000

The model numbers of the items are listed below:
.7
.5

Figure 14: Category search

7.5 Validation

Objective:	To alert user.
Action:	Following value were inserted. • Model number=98 • Name= • Category=cleaning • Discount=12 • Price=45000 • Rating=4
Expected result:	A suitable message should be displayed warning user to input missing fields.
Actual result:	A dialog box appeared warning to enter the name of the product.
Conclusion:	Test is successful.

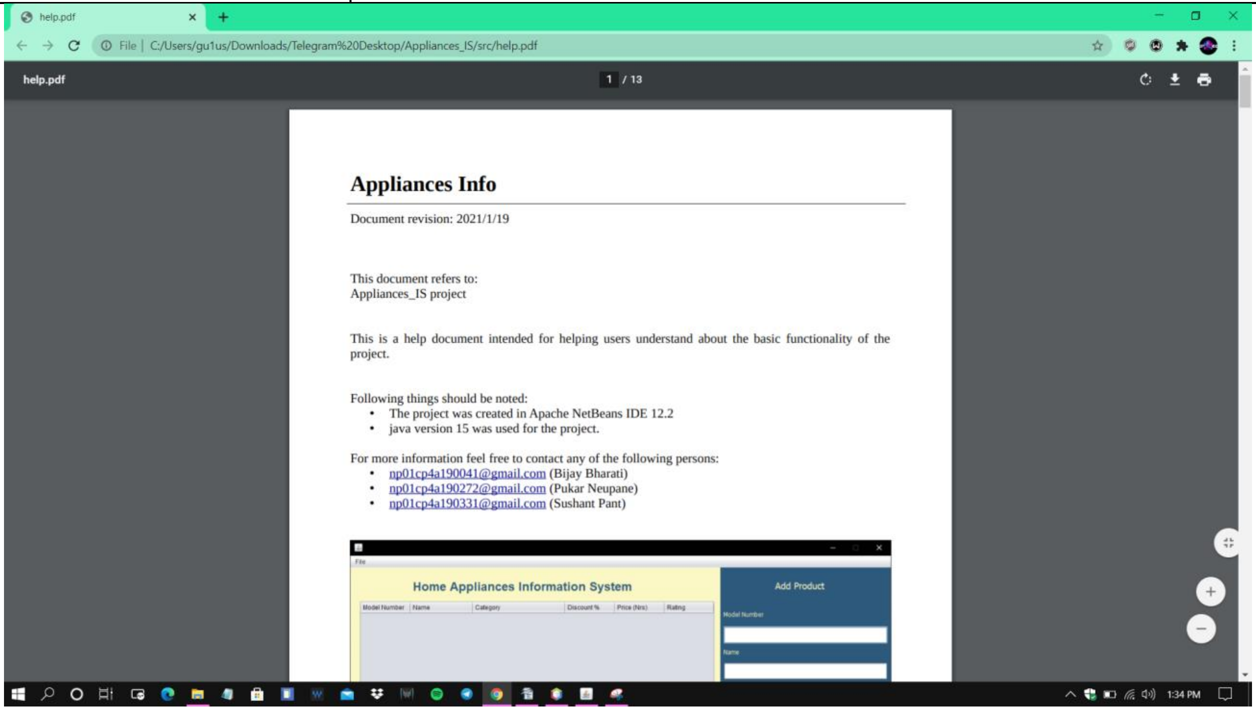


Please Enter the Price for the product.

Figure 15: Warning when incorrectly adding data

7.6 Help menu

Objective:	Whenever user are having problem understanding the program, 'Help' menu has all the solutions.
Action:	From file menu, help was pressed.
Expected result:	help.pdf file should open, which contains basic instructions and information about the program.
Actual result:	help.pdf file opened which can be read by the user.
Conclusion:	Test is successful.


help.pdf
File | C:/Users/gu1us/Downloads/Telegram%20Desktop/Appliances_IS/src/help.pdf
help.pdf
1 / 13
Home Appliances Information System
Model Number
Name
Category
Discount %
Price (Nrs)
Rating
Add Product
Model Number
Name
1:34 PM

Appliances Info

Document revision: 2021/1/19

This document refers to:
Appliances_IS project

This is a help document intended for helping users understand about the basic functionality of the project.

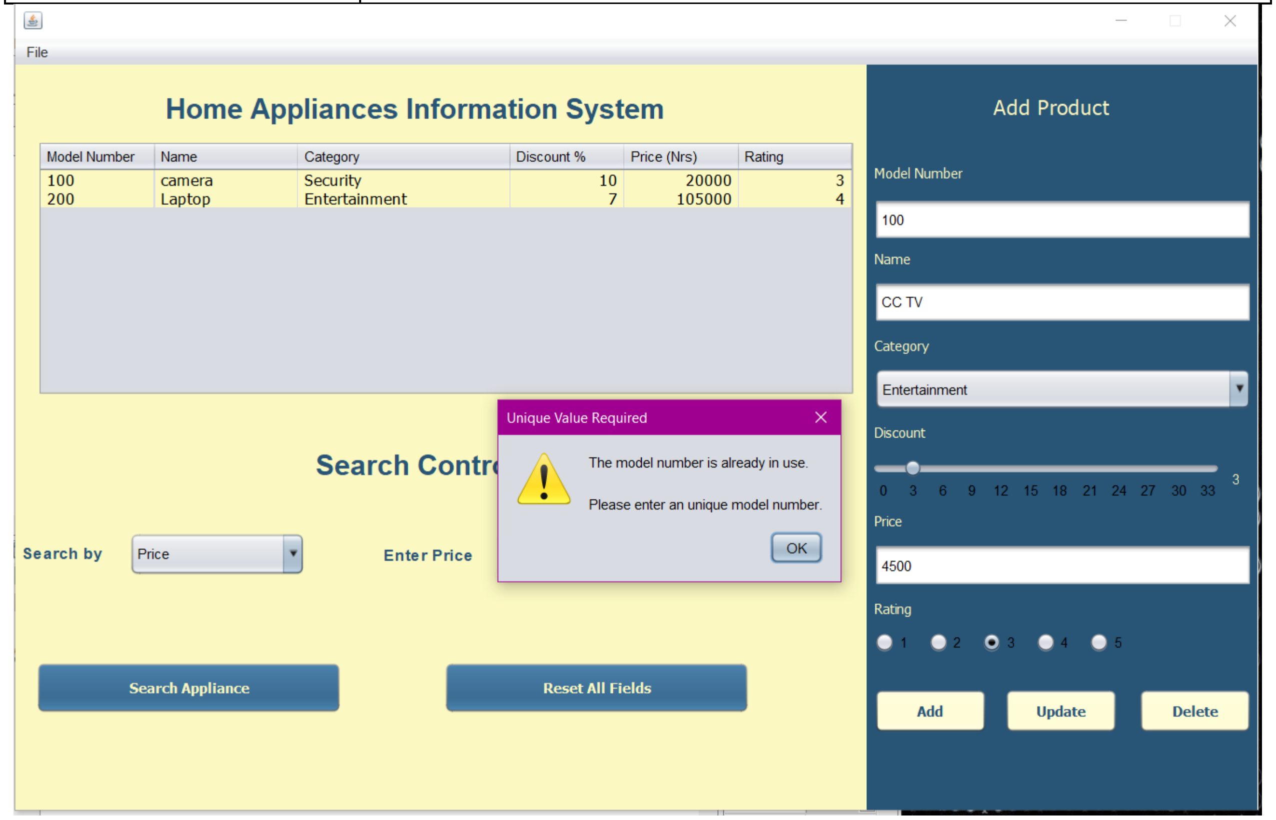
Following things should be noted:
- The project was created in Apache NetBeans IDE 12.2
- java version 15 was used for the project.

For more information feel free to contact any of the following persons:
- np01cp4a190041@gmail.com (Bijay Bharati)
- np01cp4a190272@gmail.com (Pukar Neupane)
- np01cp4a190331@gmail.com (Sushant Pant)

Figure 16: help.pdf opened

7.7 Entering duplicate product (same model number)

Objective:	Whenever same or different product with duplicate model number are entered, user are alerted. Program throws validation for such slog.
Action:	Model number 100 already exists in table, another product with model number 100 was added to table
Expected result:	Data should not be added and a warning message should be displayed.
Actual result:	Data was not added and user was warned that there needs to be an unique model number.
Conclusion:	Test is successful.



Model Number	Name	Category	Discount %	Price (Nrs)	Rating
100	camera	Security	10	20000	3
200	Laptop	Entertainment	7	105000	4

Unique Value Required

The model number is already in use.

Please enter an unique model number.

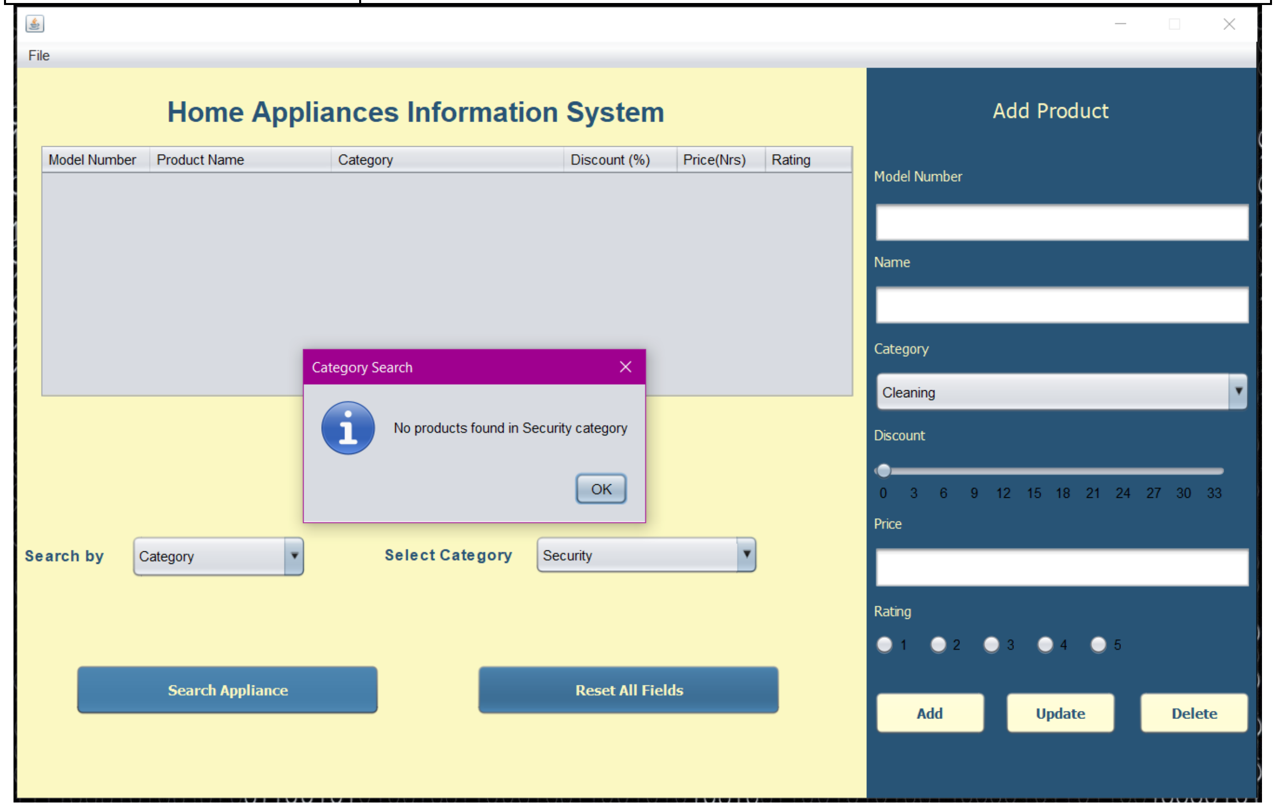
OK

Model Number: 100
Name: CC TV
Category: Entertainment
Discount: 3
Price: 4500
Rating: ○ 1 ○ 2 ● 3 ○ 4 ○ 5

Figure 17: Cannot add existing model number

7.8 Updating/deleting data without selecting row

Objective:	To alert user to select a row in order to update or delete
Action:	• Update/delete button is clicked without selecting any row.
Expected result:	User would be alerted to select a row that needs to be updated or deleted.
Actual result:	The program alerted with a message to select a row first to perform update or deletion.
Conclusion:	Test is successful.



Category Search

No products found in Security category

Category: Cleaning

Search by: Category

Select Category: Security

Figure 18: result for operations when no data in table

7.9 Inserting digit values in name and letter values in price

Objective:	To restrict user to enter special characters and digit values in name field and alphabet values in price field.
Action:	• Random digit values (0-9) and special characters (#, @) were inserted in name field. • Random letter values (a-z) and special characters (#, @) were inserted in price field.
Expected result:	The values entered would be consumed and not accepted by the respective text fields.
Actual result:	The fields would appear empty for restricted values and only valid types appeared in the field.
Conclusion:	Test is successful.

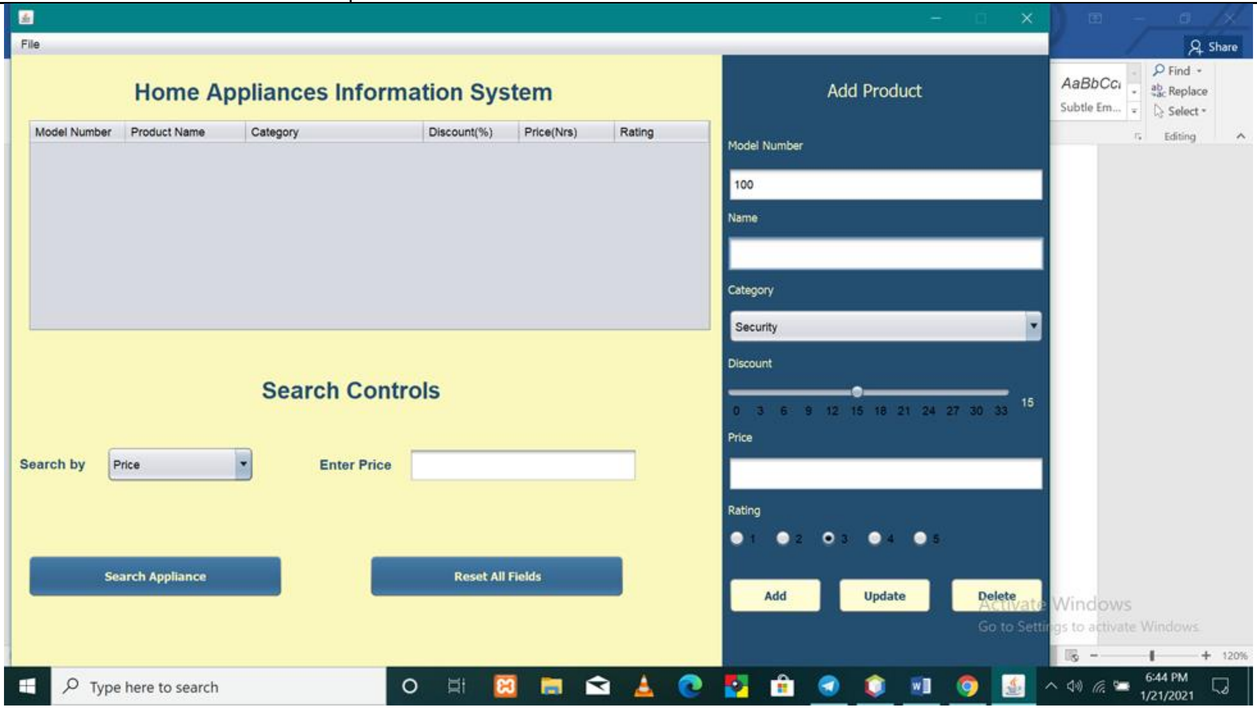
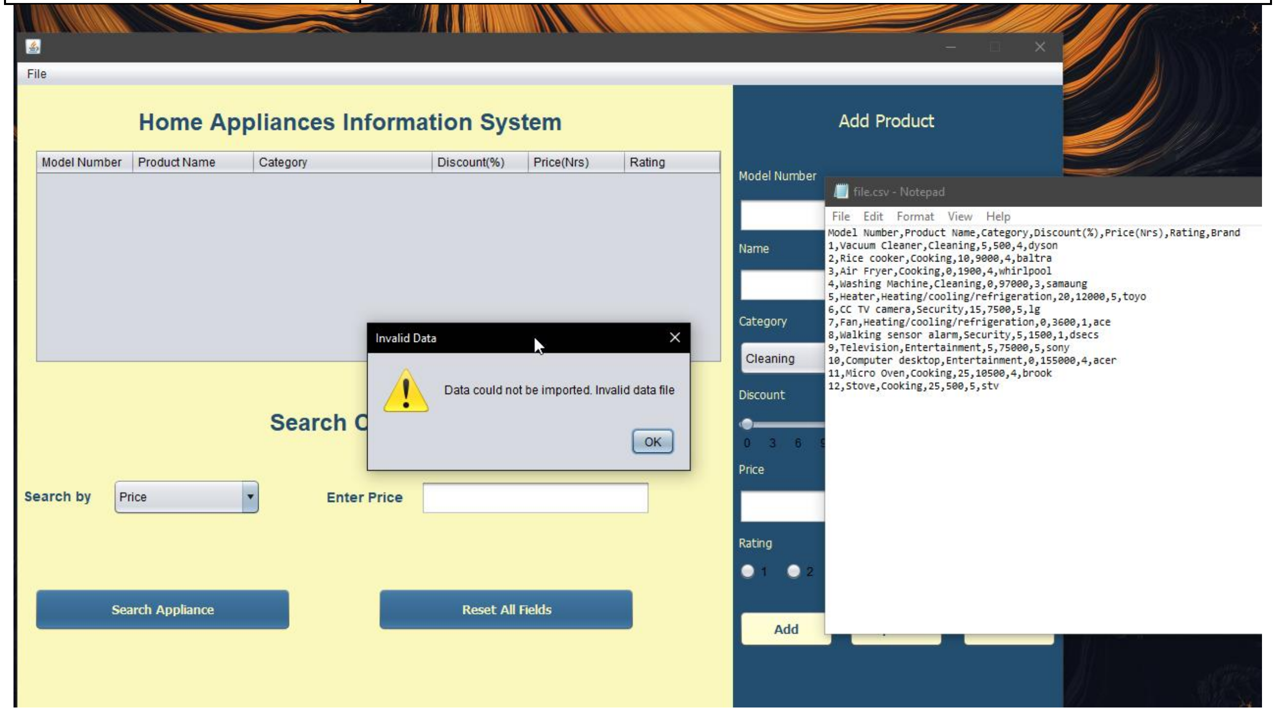


Figure 19: Empty fields when improper values are inserted

7.10 Importing file with different data structure

Objective:	To restrict user to import data from file which have more or less columns or different columns as that of GUI table columns.
Action:	• Open file menu is clicked from file menu. • A csv file is selected which have an extra column named Brand with values in all rows.
Expected result:	User should be warned about wrong data structure and import should be cancelled.
Actual result:	A warning message is displayed saying the data file is invalid.
Conclusion:	Test is successful.



Invalid Data

Data could not be imported. Invalid data file

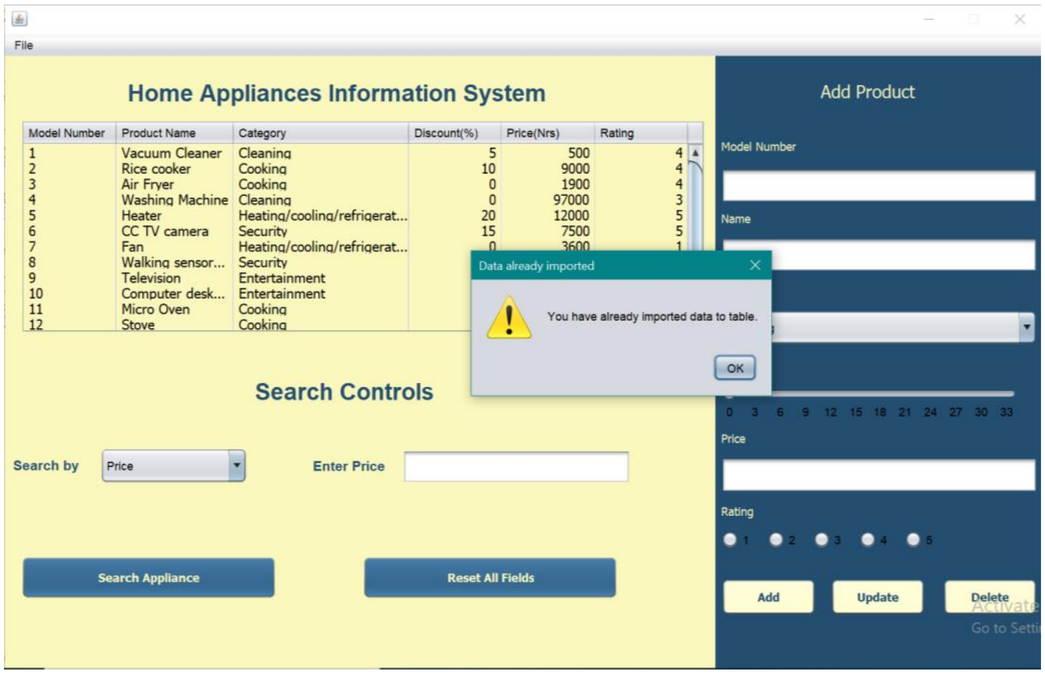
OK

```
Model Number,Product Name,Category,Discount(%),Price(Nrs),Rating,Brand
1,Vacuum Cleaner,Cleaning,5,500,4,dyson
2,Rice cooker,Cooking,10,9000,4,baltra
3,Air Fryer,Cooking,0,1900,4,whirlpool
4,Washing Machine,Cleaning,0,97000,3,samaung
5,Heater,Heating/cooling/refrigeration,20,12000,5,toyo
6,CC TV camera,Security,15,7500,5,lg
7,Fan,Heating/cooling/refrigeration,0,3600,1,ace
8,Walking sensor alarm,Security,5,1500,1,dsecs
9,Television,Entertainment,5,75000,5,sony
10,Computer desktop,Entertainment,0,155000,4,acer
11,Micro Oven,Cooking,25,10500,4,brook
12,Stove,Cooking,25,500,5,stv
```

Figure 20: Importing file with more columns than GUI table

7.11 Importing file when one file is already imported

Objective:	To restrict user to import data from file when a file is already opened in the table .
Action:	• A csv file is selected to import data. • After successful importing of data, again another file is selected to import data.
Expected result:	User should not be able to import/open another file when a file has been imported, another file can be opened when table is cleared from clear menu.
Actual result:	File cannot be imported if another file has already been opened.
Conclusion:	Test is successful.



Model Number	Product Name	Category	Discount(%)	Price(Nrs)	Rating
1	Vacuum Cleaner	Cleaning	5	500	4
2	Rice cooker	Cooking	10	9000	4
3	Air Fryer	Cooking	0	1900	4
4	Washing Machine	Cleaning	0	97000	3
5	Heater	Heating/cooling/refrigerat...	20	12000	5
6	CC TV camera	Security	15	7500	5
7	Fan	Heating/cooling/refrigerat...	0	3600	1
8	Walking sensor...	Security			
9	Television	Entertainment			
10	Computer desk...	Entertainment			
11	Micro Oven	Cooking			
12	Stove	Cooking			

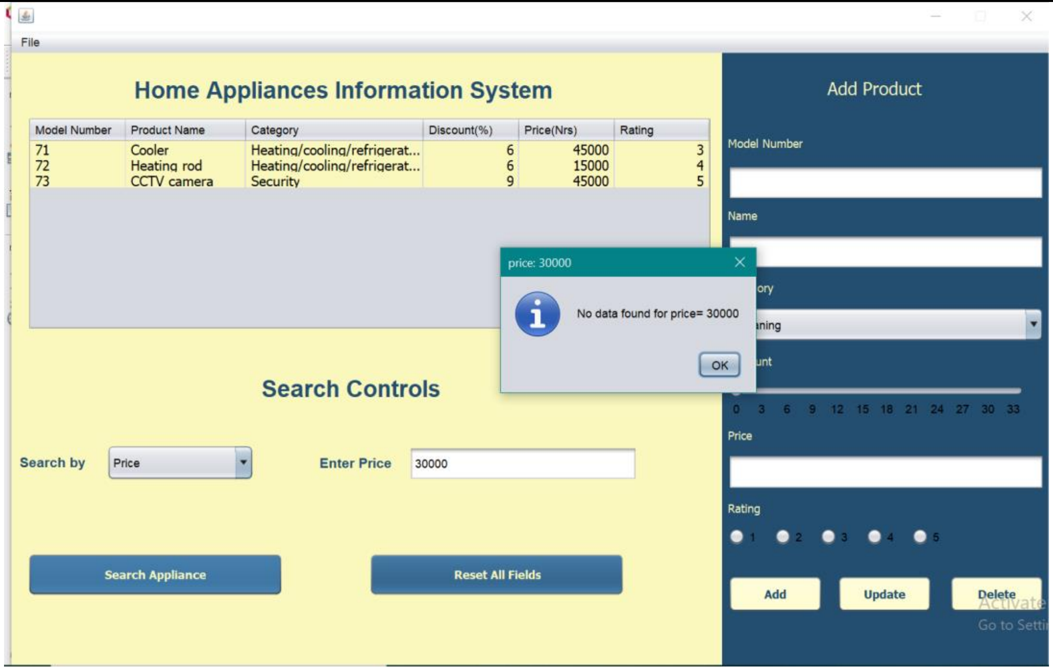
Data already imported

You have already imported data to table.

Figure 21: Alerting user that file has been imported

7.12 Searching for unavailable price

Objective:	To display suitable message when search price is not found.
Action:	• Some random data are inserted. • 30000 is inserted is search box. • Search button is clicked.
Expected result:	A suitable message would be displayed saying no data found.
Actual result:	A suitable message is displayed saying no data found for given price.
Conclusion:	Test is successful.


Home Appliances Information System

Model Number	Product Name	Category	Discount(%)	Price(Nrs)	Rating
71	Cooler	Heating/cooling/refrigerat...	6	45000	3
72	Heating rod	Heating/cooling/refrigerat...	6	15000	4
73	CCTV camera	Security	9	45000	5

price: 30000

No data found for price= 30000

Search by: Price

Enter Price: 30000

Figure 22:Searching for unavailable price

8 Conclusion

We have successfully demonstrated the use of sorting algorithm and binary search algorithm to manipulate data. This project was great practice and research on algorithms which is fundamental for any computing student. The project was also a group work and has helped us to understand the importance of working properly in a team.

The project was completed without any major problems considering we had to work from home and take online classes without being present in college to discuss our ideas properly. There were some communication issues, but we sorted them early and well and the project was completed within the time limit.

This project has been immensely helpful in developing our programming skills and introducing us to what real life problems we may have to face when working on any project.

9 References

9.1 References

Cormen, T.H. (2009) *Introduction to Algorithms*. 3rd ed. Massachusetts Institute of Technology.

Knuth, D.E. (1998) *The Art Of Programming Volume 3/ Sorting and Searching*. 2nd ed.

Wessistene, E. (2020) *MathWorld* [Online]. Available from: https://mathworld.wolfram.com/BinarySearch.html [Accessed 10 January 2021].

http://ijibm.site666.com/IJIBM_Vol9No1_Feb2017.pdf#page=282

https://www.researchgate.net/profile/Jesus-Albert/publication/39426117_An_analysis_of_selection_sort_using_recurrence_relations/links/548196120cf263ee1adfcbc5/An-analysis-of-selection-sort-using-recurrence-relations.pdf

10 Appendix

This section contains the code for the project.

10.1 AppliancesInfo.java

```
import java.awt.Component;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import static java.lang.Integer.parseInt;
import java.util.Arrays;
import java.awt.Desktop;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableModel;

public class AppliancesInfo extends javax.swing.JFrame {

   int imported = 0;

   public AppliancesInfo() {
      initComponents();

   }
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
      boolean isImported = false;
      DefaultTableModel model = (DefaultTableModel) dataTable.getModel();
      String columnHeaders[] = {"Model Number", "Product Name", "Category",
"Discount(%)", "Price(Nrs)", "Rating"};

      if (imported == 0) {
         JFileChooser chooser = new JFileChooser();
         chooser.setAcceptAllFileFilterUsed(false);
         chooser.addChoosableFileFilter(new FileNameExtensionFilter("CSV Files",
"csv"));
         int choice = chooser.showOpenDialog(this);
         File f = chooser.getSelectedFile();

         if (choice == JFileChooser.APPROVE_OPTION) {
            try {
                BufferedReader br = new BufferedReader(new FileReader(f));
                String firstLine = br.readLine().trim();
                String[] columnsName = firstLine.split(",");
                System.out.println(Arrays.toString(columnsName));
                int columnNum = columnsName.length;

                if (Arrays.equals(columnsName, columnHeaders)) {
                    Object[] tableLines = br.lines().toArray();

                    for (int i = 0; i < tableLines.length; i++) {
                        String line = tableLines[i].toString().trim();
                        String[] dataRow = line.split(",");
                        int colLength = dataRow.length;
                        if (colLength == columnNum) {
                            model.addRow(dataRow);
                            isImported = true;
                        } else {
                            JOptionPane.showMessageDialog(null, "Data in file invalid.",
"Incomplete data", JOptionPane.WARNING_MESSAGE);
                            isImported = false;
                            for (int a = 0; a < model.getRowCount(); a++) {
                                model.removeRow(a);
                            }
                        }

                    }
                } else {
                    JOptionPane.showMessageDialog(null, "Data could not be imported.
Invalid data file", "Invalid Data  ", JOptionPane.WARNING_MESSAGE);
                }

            } catch (FileNotFoundException ex) {
                Logger.getLogger(AppliancesInfo.class.getName()).log(Level.SEVERE,
null, ex);
            } catch (IOException ex) {
                Logger.getLogger(AppliancesInfo.class.getName()).log(Level.SEVERE,
null, ex);
            }
            if (isImported == true) {
                imported = 1;
                JOptionPane.showMessageDialog(null, "Successfully imported data.",
"Data Import", JOptionPane.INFORMATION_MESSAGE);
            } else {
                imported = 0;
            }
        }
    } else {
        JOptionPane.showMessageDialog(null, "You have already imported data to table.", "Data already imported", JOptionPane.WARNING_MESSAGE);
    }
}

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

    if (searchOptionsComboBox.getSelectedIndex() == 0) {
        priceSearchPanel.setVisible(true);
        priceSearchTxtFld.setVisible(true);
        categorySearchPanel.setVisible(false);
        selectCategoryComboBox.setVisible(false);
        selectCategoryComboBox.setSelectedIndex(0);
    }
    if (searchOptionsComboBox.getSelectedIndex() == 1) {
        priceSearchPanel.setVisible(false);
        priceSearchTxtFld.setVisible(false);
        selectCategoryComboBox.setVisible(true);
        categorySearchPanel.setVisible(true);
        priceSearchTxtFld.setText("");

    }


}

private void discountSliderStateChanged(javax.swing.event.ChangeEvent evt) {

    if (discountSlider.getValue() == 0) {
        miniDiscountLbl.setText("");
    } else {
        miniDiscountLbl.setText(Integer.toString(discountSlider.getValue()));
    }
}

private void ResetAllFieldsBtnActionPerformed(java.awt.event.ActionEvent evt) {
    resetDataFields();
    resetSearchFields();
}
public void resetDataFields() {
    modelNumberTxtFld.setText("");
    nameTxtFld.setText("");
    categoryComboBox.setSelectedIndex(0);
    discountSlider.setValue(0);
    priceTxtFld.setText("");
    ratingBtnGroup.clearSelection();
}

public void resetSearchFields() {
    searchOptionsComboBox.setSelectedIndex(0);
    selectCategoryComboBox.setSelectedIndex(0);
    priceSearchTxtFld.setText("");
}
private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
    int yes_no = JOptionPane.showConfirmDialog(this, "Do you want to exit?" + "\n\n"
            + "Yes will exit the application, No will close this window.", "Exit",
JOptionPane.YES_NO_OPTION);
    if (yes_no == JOptionPane.YES_OPTION) {
        this.dispose();
    } else {
        System.out.println("Returned to main page.");
    }
}

private void addDataBtnActionPerformed(java.awt.event.ActionEvent evt) {
    String modNum = modelNumberTxtFld.getText();
    String productName = nameTxtFld.getText();
    String category = categoryComboBox.getSelectedItem().toString();
    int discount = discountSlider.getValue();


    int radioVal = 0;

    if (radio1.isSelected()) {
        radioVal = 1;
    } else if (radio2.isSelected()) {
        radioVal = 2;
    } else if (radio3.isSelected()) {
        radioVal = 3;
    } else if (radio4.isSelected()) {
        radioVal = 4;
    } else if (radio5.isSelected()) {
        radioVal = 5;
    }
    if (!modNum.isEmpty()) {
        if (!productName.isEmpty()) {
            if (!priceTxtFld.getText().isEmpty()) {
                if (radioVal != 0) {
                    int price = Integer.parseInt(priceTxtFld.getText());
                    DefaultTableModel model = (DefaultTableModel) dataTable.getModel();
                    if (modelNumberExists()) {
                        JOptionPane.showMessageDialog(null, "The model number is already in use."
                                + "\n\n" + "Please enter an unique model number.", "Unique Value Required", JOptionPane.WARNING_MESSAGE);
                        modelNumberTxtFld.grabFocus();
                    } else {
                        model.addRow(new Object[]{modNum, productName, category, discount, price, radioVal});
                        JOptionPane.showMessageDialog(null, "Product data successfully added to table."
                                + "\n\n" + "Total rows: " + model.getRowCount(), "Data added", JOptionPane.INFORMATION_MESSAGE);
                        resetDataFields();

                    }

                } else {
                    JOptionPane.showMessageDialog(null, "Please select a Rating for the product.", "*Required Field", JOptionPane.WARNING_MESSAGE);
                }
            } else {
                JOptionPane.showMessageDialog(null, "Please Enter the Price for the product.", "*Required Field", JOptionPane.WARNING_MESSAGE);
                priceTxtFld.grabFocus();
            }
        } else {
            JOptionPane.showMessageDialog(null, "Please Enter Name of the product.", "*Required Field", JOptionPane.WARNING_MESSAGE);
            nameTxtFld.grabFocus();
        }
    } else {
        JOptionPane.showMessageDialog(null, "Please Enter the product's Model Number.", "*Required Field", JOptionPane.WARNING_MESSAGE);
        modelNumberTxtFld.grabFocus();
    }

}
public boolean modelNumberExists() {
    DefaultTableModel model = (DefaultTableModel) dataTable.getModel();
    int rows = model.getRowCount();
    if (rows > 0) {
        String modelNumber = "";
        for (int i = 0; i < rows; i++) {
            modelNumber = model.getValueAt(i, 0).toString();
            if (modelNumber.equalsIgnoreCase(modelNumberTxtFld.getText())) {
                return true;
            }
        }
    }
    return false;
}

private void updateBtnActionPerformed(java.awt.event.ActionEvent evt) {
    DefaultTableModel model = (DefaultTableModel) dataTable.getModel();
    int selectedRowIndex = dataTable.getSelectedRow();
    int changes = 0;
    StringBuilder fieldsUpdated = new StringBuilder();
    if (selectedRowIndex >= 0) {
        if (modelNumberTxtFld.getText().isBlank()) {
            System.out.println("no need to update field.");
        } else {
            JOptionPane.showMessageDialog(null, "Model Number is final." + "\n\n" +
"Cannot modify model number.", "Model Number",
JOptionPane.WARNING_MESSAGE);
        }
        if (nameTxtFld.getText().isBlank()) {
            System.out.println("no need to update field.");
        } else {
            model.setValueAt(nameTxtFld.getText(), selectedRowIndex, 1);
            changes += 1;
            fieldsUpdated.append(".Name" + "\n");
        }
        model.setValueAt(categoryComboBox.getSelectedItem().toString(),
selectedRowIndex, 2);
        changes += 1;
        fieldsUpdated.append(".Category" + "\n");
        model.setValueAt(discountSlider.getValue(), selectedRowIndex, 3);
        changes += 1;
        fieldsUpdated.append(".Discount" + "\n");
        if (priceTxtFld.getText().isBlank()) {
            System.out.println("no need to update field.");
        } else {
            model.setValueAt(parseInt(priceTxtFld.getText()), selectedRowIndex, 4);
            changes += 1;
            fieldsUpdated.append(".Price" + "\n");
        }
        int radioVal = 0;
        if (radio1.isSelected()) {
            radioVal = 1;
        } else if (radio2.isSelected()) {
            radioVal = 2;
        } else if (radio3.isSelected()) {
            radioVal = 3;
        } else if (radio4.isSelected()) {
            radioVal = 4;
        } else if (radio5.isSelected()) {
            radioVal = 5;
        }
        if (radioVal == 0) {
            System.out.println("no need to update field.");
        } else {
            model.setValueAt(radioVal, selectedRowIndex, 5);
            changes += 1;
            fieldsUpdated.append(".Rating" + "\n");
        }
        JOptionPane.showMessageDialog(null, "Row updated." + "\n\n" + "Changes
made: " + changes
                    + "\n\n" + "Fields Updated:" + "\n" + fieldsUpdated, "Data Updated",
JOptionPane.INFORMATION_MESSAGE);
        resetDataFields();
    } else {
        JOptionPane.showMessageDialog(null, "Please select a row you wish to
update.", "Update", JOptionPane.WARNING_MESSAGE);
    }
}

private void deleteDataBtnActionPerformed(java.awt.event.ActionEvent evt) {
    DefaultTableModel model = (DefaultTableModel) dataTable.getModel();
    int selectedRowIndex = dataTable.getSelectedRow();
    if (selectedRowIndex >= 0) {
        int yes_no = JOptionPane.showConfirmDialog(null, "Do you want to delete the
selected row?" + "\n\n"
                    + "This action cannot be undone.", "Delete",
JOptionPane.YES_NO_OPTION);
        if (yes_no == JOptionPane.YES_OPTION) {
            model.removeRow(selectedRowIndex);
            JOptionPane.showMessageDialog(null, "Data deleted." + "\n\n" + "Total rows:
" + model.getRowCount(), "Data Deleted", JOptionPane.INFORMATION_MESSAGE);
        } else {
            System.out.println("Returned to main page.");
        }

    } else {
        JOptionPane.showMessageDialog(null, "Please select a row you wish to
delete.", "Delete", JOptionPane.WARNING_MESSAGE);
    }
}

private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
    int result = JOptionPane.showConfirmDialog(null, "Sure? You want to remove all
data from the table?", "Empty Table",
            JOptionPane.YES_NO_OPTION,
            JOptionPane.QUESTION_MESSAGE);

    if (result == JOptionPane.YES_OPTION) {
        DefaultTableModel model = (DefaultTableModel) dataTable.getModel();
        model.getRowCount();
        while (model.getRowCount() > 0) {
            for (int i = 0; i < model.getRowCount(); i++) {
                    model.removeRow(i);
            }
        }
        imported = 0;
    }
}

private void priceTxtFldKeyTyped(java.awt.event.KeyEvent evt) {
    char c = evt.getKeyChar();
    if (!Character.isDigit(c)) {
        evt.consume();
    }
}

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

    DefaultTableModel model = (DefaultTableModel) dataTable.getModel();
    int r = model.getRowCount();
    int c = model.getColumnCount();
    String tableData[][] = new String[r][c];
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            tableData[i][j] = model.getValueAt(i, j).toString();
        }
    }
    String[][] sortedTableData = SelectionSort.sort(tableData);

    if (searchOptionsComboBox.getSelectedIndex() == 0) {
        if (!priceSearchTxtFld.getText().isEmpty()) {
            int price = Integer.parseInt(priceSearchTxtFld.getText());
            int priceList[] = new int[r];
            for (int i = 0; i < r; i++) {
                    priceList[i] = Integer.parseInt(sortedTableData[i][4]);
                }
                int index = BinarySearch.search(priceList, 0, (priceList.length - 1), price);
                if (index == -1) {
                    JOptionPane.showMessageDialog(null, "No data found for price= " + price,
"price: " + price, JOptionPane.INFORMATION_MESSAGE);
                } else {
                    String modelNumber = sortedTableData[index][0];
                    String name = sortedTableData[index][1];
                    String category = sortedTableData[index][2];
                    int discount = Integer.parseInt(sortedTableData[index][3]);
                    int rating = Integer.parseInt(sortedTableData[index][5]);
                    JOptionPane.showMessageDialog(null, "Following data found for price= " +
price
                            + "\n\n" + "Model Number: " + modelNumber
                            + "\n\n" + "Name: " + name
                            + "\n\n" + "Category: " + category
                            + "\n\n" + "Discount %: " + discount
                            + "\n\n" + "Rating: " + rating,
                            "price: " + price, JOptionPane.INFORMATION_MESSAGE
                    );
                }
            } else {
                JOptionPane.showMessageDialog(null, "Please enter a value for price",
"Price Search", JOptionPane.WARNING_MESSAGE);
                priceSearchTxtFld.grabFocus();
            }
        }
        if (searchOptionsComboBox.getSelectedIndex() == 1) {
            String category = selectCategoryComboBox.getSelectedItem().toString();
            StringBuilder modelNumberList = new StringBuilder();
            StringBuilder nameList = new StringBuilder();
            for (int i = 0; i < r; i++) {
                if (category.equals(sortedTableData[i][2])) {
                    nameList.append("." + sortedTableData[i][1] + " -" + sortedTableData[i][4] +
"\n");
                    modelNumberList.append("." + sortedTableData[i][0] + "\n");
                }

            }
            if (modelNumberList.length() == 0 || nameList.length() == 0) {
                JOptionPane.showMessageDialog(null, "No products found in " + category + "
category", "Category Search", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Following items belong to " +
category + " category" + "\n"
```

```
                    + "numbers represent the price" + "\n\n" + nameList
                    + "\n" + "The model numbers of the items are listed below: " + "\n" +
modelNumberList, category, JOptionPane.INFORMATION_MESSAGE);
          }

        }

    }

    private void priceSearchTxtFldKeyTyped(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        if (!Character.isDigit(c)) {
            evt.consume();
        }
    }

    private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {

        File file = new File("src\\help.pdf");
        try {
            Desktop.getDesktop().open(file);
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "File not found", "Help File",
JOptionPane.ERROR_MESSAGE);
        }
    }

    private void nameTxtFldKeyPressed(java.awt.event.KeyEvent evt) {
        char c = evt.getKeyChar();
        if (Character.isLetter(c) || Character.isWhitespace(c) || Character.isISOControl(c)) {
            nameTxtFld.setEditable(true);
        } else {
            nameTxtFld.setEditable(false);
        }
    }
    public static void main(String args[]) {

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

                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(AppliancesInfo.class
                    .getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(AppliancesInfo.class
                    .getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(AppliancesInfo.class
                    .getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(AppliancesInfo.class
                    .getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new AppliancesInfo().setVisible(true);
            }
        });
    }
    private javax.swing.JButton ResetAllFieldsBtn;
    private javax.swing.JButton addDataBtn;
    private javax.swing.JPanel addDataPanel;
    private javax.swing.JLabel addDataTitleLbl;
    private javax.swing.JComboBox<String> categoryComboBox;
    private javax.swing.JPanel categorySearchPanel;
    private javax.swing.JLabel catehoryLbl;
    private javax.swing.JTable dataTable;
    private javax.swing.JButton deleteDataBtn;
    private javax.swing.JLabel discountLbl;
    private javax.swing.JSlider discountSlider;
    private javax.swing.JLabel enterPriceLbl;
    private javax.swing.JMenu fileMenu;
    private javax.swing.JLayeredPane jLayeredPane1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem2;
    private javax.swing.JMenuItem jMenuItem3;
    private javax.swing.JMenuItem jMenuItem4;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JPanel mainPanel;
    private javax.swing.JLabel mainTitleLbl;
    private javax.swing.JMenuBar menuBar;
    private javax.swing.JLabel miniDiscountLbl;
    private javax.swing.JLabel modelNumberLbl;
    private javax.swing.JTextField modelNumberTxtFld;
    private javax.swing.JLabel nameLbl;
    private javax.swing.JTextField nameTxtFld;
    private javax.swing.JLabel priceLbl;
    private javax.swing.JPanel priceSearchPanel;
    private javax.swing.JTextField priceSearchTxtFld;
    private javax.swing.JTextField priceTxtFld;
    private javax.swing.JRadioButton radio1;
    private javax.swing.JRadioButton radio2;
    private javax.swing.JRadioButton radio3;
    private javax.swing.JRadioButton radio4;
    private javax.swing.JRadioButton radio5;
    private javax.swing.ButtonGroup ratingBtnGroup;
    private javax.swing.JLabel ratingLbl;
    private javax.swing.JLabel searcgByLbl;
    private javax.swing.JButton searchAppliancesBtn;
    private javax.swing.JComboBox<String> searchOptionsComboBox;
    private javax.swing.JLabel searchTitleLbl;
    private javax.swing.JComboBox<String> selectCategoryComboBox;
    private javax.swing.JLabel selectCategoryLbl;
    private javax.swing.JButton updateBtn;

}
```

10.2 SelectionSort.java

```
public class SelectionSort {

    public static String[][] sort(String[][] a) {
        for (int i = 0; i < a.length - 1; i++) {
            int minPos = minPosFinder(a, i);
            swap(a, minPos, i);
        }
        return a;
    }

    public static int minPosFinder(String[][] a, int from) {
        int minPos = from;
        for (int i = from + 1; i < a.length; i++) {
            if (Integer.parseInt(a[i][4]) < Integer.parseInt(a[minPos][4])) {
                minPos = i;
            }
        }
        return minPos;
    }

    public static void swap(String[][] a, int minPos, int from) {
        String[] temp = a[minPos];
        a[minPos] = a[from];
        a[from] = temp;

    }

}
```

10.3 BinarySearch.java

```
public class BinarySearch {
    static int search(int arr[], int low, int right, int key){
if (right >= low) {
          int mid = low + (right - low) / 2;
if (arr[mid] == key)
             return mid;
if (arr[mid] > key)
             return search(arr, low, mid - 1, key);
  return search(arr, mid + 1, right, key);
      }
     return -1;
      }
}
```