LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS5004NA Emerging Programming Platforms and Technologies

Assessment Weightage & Type

30% Group Coursework

Year and Semester

2018-19 Autumn

Group Name: Laptop GanG			
SN	**Student Name**	**College ID**	**University ID**
1	**Rajat Shrestha**	**np01cp4a170021**	**17030954**
2	**Pranav Khatri**	**np01cp4a170210**	**17031046**
3	**Nikolas Shrestha**	**np01cp4a170150**	**17031243**

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 mark of zero will be awarded.

Proposal

This proposal is to address the first coursework assigned as group work in Emerging Programming Platforms and Technologies module which involves designing and developing an Information System. For this project, an Information System dealing with laptops was finalized as it seemed more reasonable as a laptop will have various different attributes which user might be searching for. This information system will implement the concept of binary search algorithm to find the specific product within its inventory. This project will also include GUI for dealing with adding and searching the items for better user experience. The completed information system will have a various set of features and how the user can interact with various pieces of information regarding the laptop. The list of features and data dealt by this system are as follows:

List of data:

1. **Serial Number** *(Integer)* It assigns the laptop according to its entry.
2. **Manufacturer** *(String)* It will describe the manufacturer of the laptop.
3. **Model Number** *(String)* It will give the unique model number of the device.
4. **Screen Size** *(String)* This determines the display size of the laptop.
5. **Processor** *(String)* This determines the processor of the laptop.
6. **RAM Size** *(String)* This determines Memory size of the laptop.
7. **Storage Size** *(Integer)* This determines Storage capacity of the laptop.
8. **Price** *(Integer)* This will give the price of the laptop.

List of features:

- GUI is designed to enhance the user experience,
- The system will contain 12 unique items at the beginning,
- New products can be added to the table,
- Object-Oriented Programming technique is used,
- The array list is used as a data structure,
- A selection sorting algorithm is used to sort products according to price,
- Laptops can be sorted in the GUI,
- Data of each product will be stored in a table and Array lists,
- The items are sorted while stored in an Array lists,
- Products can be searched according to category or price,
- The binary search algorithm is used to find products by the price.
- Invalid inputs are not accepted and users are notified about it,
- The code will have implemented exception handling feature for seamless running.

Tools used

To create this project various tools are needed to be used for designing, planning, and developing which are:

- Sublime text: A simplistic text editor to edit texts and codes.

- Microsoft Word: Word processing software to create professional and formal documentation of the designing, planning, and reporting the development process.

- Net-Beans IDE: A well-integrated development environment of Java which simplifies the task of designing and creating the GUI.

- Affinity Designer: A simple but powerful tool for creating elegant vector graphics and editing pictures.

Table of Contents

Table of Figures:

Table of tables:

Individual Task

To complete the following project, every group member has contributed equally while designing, developing, testing and documenting the entire process. To complete this project the following steps are essential to be done:

1. Analyzing the requirement of the project,
2. Creating a design prototype consisting of GUI elements,
3. Making the components function,
4. Using a data structure to store the data,
5. Sorting the data stored,
6. Implementing Binary Search on the sorted data.

The group consists of three members and the task load was divided equally among three members. The responsibilities divided into each individual while working on this project is listed as follows:

Rajat Shrestha:

Analyzed the requirements of the project and identified the key functionality. Suggested to store the data in an array list composed of custom object. Researched about binary search and created a flowchart to represent the logic clearly. Specified and documented the testing process.

Nikolas Shrestha:

Created a GUI structure to make platform for the functionality. Made the custom object with all the required accessor methods. Discussed how the binary search algorithm could be used in the program and written a detailed pseudocode. Did the testing in the actual program.

Pranav Khatri:

Initializing the document by writing a proposal and listing all the data used and features in the program. Implemented the linear sort algorithm to store the object as data in array list. Converted the pseudocode into code and made the search button functional. Added various small features in the program.

Introduction

This document clarifies the information System developed which deals with laptops in which a user can add, store or find the desired product. This program contains various elements and programming techniques to function properly and is created in a modular manner so that it can be configured or debugged easily. The modularity of this program will allow this program to reuse the same code at multiple instances, which is much more efficient. The program will also store the product data for each laptop as a laptop object in array lists for storing, sorting and searching. The GUI is created by implementing included java library dealing with all sorts of GUI component called “Java Swing”. The selection sorting algorithm is used to sort the data individually while being added to the program. The sorted information will be crucial to search the product with respect to its price.

Graphical User Interface (GUI) is simply a method of delivery of data to the user in a user-friendly manner (Hopping, 2018). User Experience (UX) design deals with creating user interfaces (GUI) that provides easy and relevant experience to users who will use that program (The Interaction Design Foundation, 2018). Java Swing package is a pre-installed java library which provides a basic set of light-weight GUI elements to create small to big projects that implement GUI and often call ActionListener class to ensure functionality (Oracle , 2019). To create the project including buttons and other interactable components the main class is extended from the ActionListener class which will handle different user-defined actions. Various components such as buttons, labels, text-fields, combo-boxes, menu-items, radio-buttons, panel and table are used for the creation of GUI and pop-up boxes linked to different actions to interact with the user.

The object-oriented method of programming involves detailed specification of objects rather than logic and commands so that these objects can have defined attributes and functions which will make it easier to manipulate and handle these objects to solve various scenarios. Data Structures is simply a format for storing data (Rouse, 2014). The data structure used for this project is Array-list because it can also store various objects, simple syntax, easier methods and adjustable size (Jacobson & Thornton, 2004). Sorting is simply arranging a set of similar data in a specific order the sorting algorithm used in this system is Selection Sort which is a linear sorting algorithm (Mahmoud, 2011). The program will utilize this sorted data to search any product with respect to its price using Binary Sort Algorithm.

Binary Search Algorithm

Also known as log arithmetic search or half-interval search requires a sorted data-structure to work on unlike a linear searching algorithm (Williams Jr., 1976). The binary search algorithm directly compares the middle index to the search term and if the value is equal it will return the index of the middle term. If the search term is bigger or smaller than the search term then it will divide the sorted data into smaller parts and begins the whole procedure again on the divided half (Weisstein, 2018). *(The limitations are discussed in the appendix.)*

1. Flow-Chart:

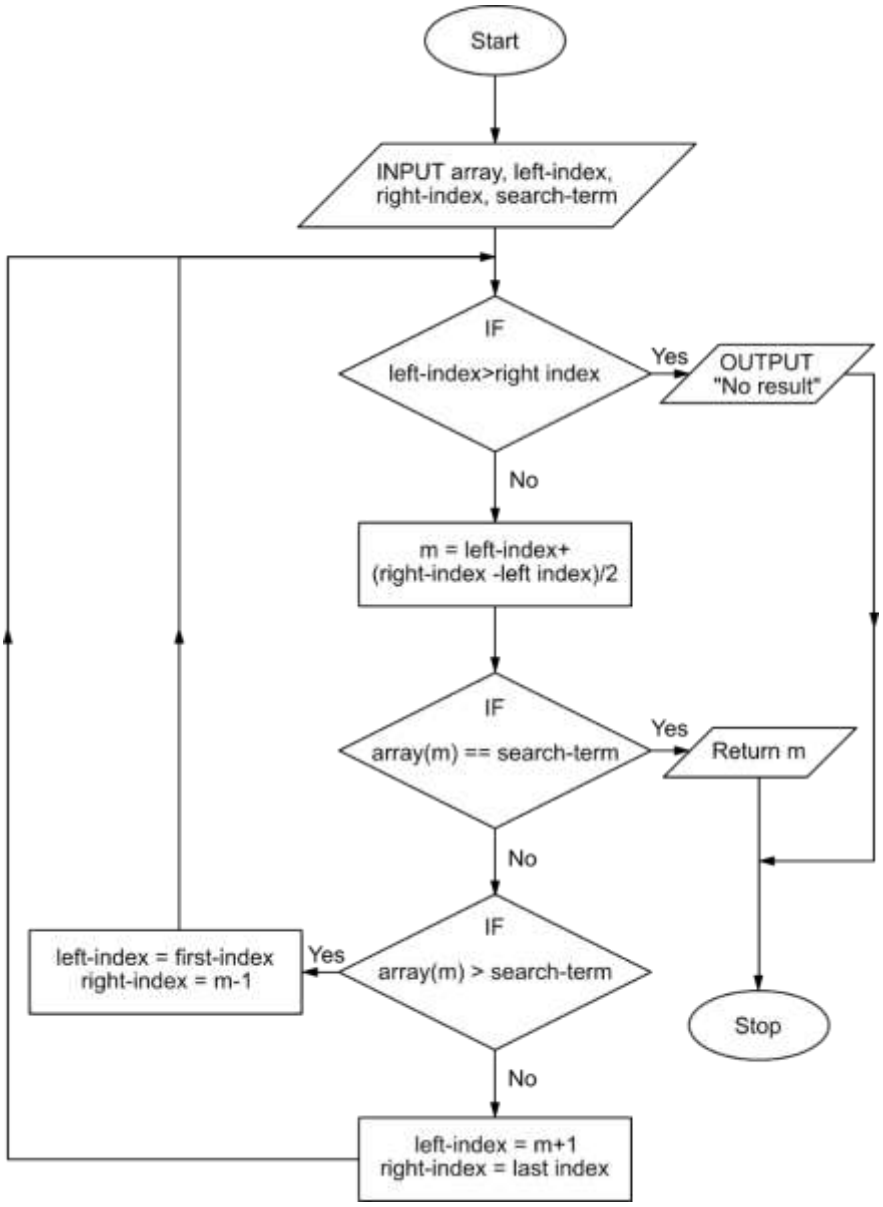

Figure 1: Flow-Chart for Binary Search Algorithm

2. Implementation of the program:

To use the principle of binary search in the program to search any product according to their price, two sorted array lists were created in which the products were sorted according to their price. In the first array list, price sorted laptops were stored while the other array list contained only the prices of the products in it. These two array lists have the same product stored within them so that when we take the index from the price array list to the product array list the product is mapped. *(The java method is discussed further in the appendix.)*

Then to search the product according to their price the price array list is taken with only sorted data in form of an integer and is put through the binary search algorithm along with the search term. The first and last index info is also passed as a parameter so that at the program will know the range in which it will have to search for the term. If the product with that price doesn't exist the program will notify the user about the error. The pseudocode for the method and method calling is as follows:

Pseudocode:

```
METHOD binarySearcher(ArrayList, FirstIndex, LastIndex, SearchTerm)
  IF (FirstIndex <= LastIndex)
    MiddleIndex = (LastIndex - FirstIndex)/2 + FirstIndex;
    MiddleValue = VALUE(MiddleIndex);
    IF (MiddleValue == SearchTerm){
      RETURN(MiddleIndex);
    ELSE IF(MiddleValue > SearchTerm)
      RETURN(binarySearcher(ArrayList, FirstIndex, MiddleIndex-1, SearchTerm));
    ELSE
      RETURN(binarySearcher(ArrayList, MiddleIndex+1, LastIndex, SearchTerm));
  NOTIFY ("Not found")
CALL METHOD binarySearcher(price, 0, price.length(), searchPrice)
```

Method Description

A Java method contains statements that are put together to perform an operation in java code (Jenkov, 2015). It helps to make the code modular and reuse the code. The java program to manage the information regarding laptops were created with two separate java class. The method described for the main java class that implements the GUI and the "Laptop" class is given below with the method, arguments, summary and return datatype.

Table 1: Method Summary for FindYourLaptop class

Method Summary	
Constructor	**FindYourLaptop()** The constructor of the FindYourLaptop class.
void, main *Static*	**main(String args[])** The main method for FindYourLaptop class.
Void	**initComponents()** The autogenerated method that manages The GUI.
Boolean	**not duplicate(String productModel, int index)** Method to check for duplicate values from a certain index in the table and return a Boolean value to indicate duplicate entry.
void	**displayAll()** The method that will add 12 laptop items to the system.
void	**addProduct(Laptop product)** The method that will take a laptop object and store it in tables and ArrayList(s).
void	**void productIterator()** The method will iterate through various ArrayList and print them in the console.
int	**binarySearcher(ArrayList<Integer> priceArray, int leftIndex, int rightIndex, int x)** The method that uses a binary search algorithm to find a specific item from a sorted array and return the index.
void	**clearInput()** Method That will clear all the user inputs

void	**addActionPerformed(java.awt.event.ActionEvent e)** The method that will take user input for a product and store it when “Add” button is pressed.
void	**clearActionPerformed(java.awt.event.ActionEvent evt)** Method to clear the form after pressing the “Clear” Button.
void	**jbtnPriceActionPerformed(java.awt.event.ActionEvent evt)** The method that will take user input for a product with specific price and search the product using binary search when the “Search” Button is pressed.
void	**jbtnScreenActionPerformed(java.awt.event.ActionEvent evt)** The method that will take user input for a product with specific screen-size and search the product when the “Search” Button is pressed.
void	**jmiDocActionPerformed(java.awt.event.ActionEvent evt)** Opens documentation file when the “Open Documentation” menu item is pressed
void	**void jmiExitActionPerformed(java.awt.event.ActionEvent evt)** Closes the program when the “Exit” menu item is pressed
void	**jmiClearActionPerformed(java.awt.event.ActionEvent evt)** Clears all the user input in the program when the “Clear” menu item is pressed
void	**jmiALCActionPerformed(java.awt.event.ActionEvent evt)** Prints all the array list content in the console when “Array List Content” item is pressed.
void	**jmiExitKeyPressed(java.awt.event.KeyEvent evt)** To link the “Ctrl+x” button to exit the program.
void	**jmiClearKeyPressed(java.awt.event.KeyEvent evt)** To link the “Ctrl+c” button to clear the user inputs.

Table 2: Method Summary for laptop object

Method Summary	
Constructor	**Laptop(int sN, String manufacturer, String modelNumber, String processor, String screenSize, String ram, int storage, int price)** The constructor of the Laptop object.
String[ ] *(Array)*	**getLaptop()** The method that returns all the laptop detail in form of an array of string.
Void	**displayModel()** Displays the laptop information about model number and price in the console
int	**getSN()** Accessor method to get the Serial No of the product.
String	**getManufacturer()** Accessor method to get the Manufacturer of the product.
String	**getModelNumber()** Accessor method to get the Model Number of the product.
String	**getProcessor()** Accessor method to get the Processor info of the product.
String	**getScreenSize()** Accessor method to get the screen size of the product.
String	**getRam()** Accessor method to get the memory-size of the product.
int	**getStorage()** Accessor method to get the storage Size of the product.
int	**getPrice()** Accessor method to get the price of the product.

Testing

1. Running in NetBeans:

Table 3: (Test 1) Running in NetBeans

Objective	Checking whether the program can be imported and be run from NetBeans or not
Action	Opening NetBeans and importing the project. Running the FindYourLaptop class.
Expected Result	The file gets compiled and GUI opens.
Actual Result	File compiled and GUI is opened after giving the path where the file is located.
Conclusion	Test Successful.

(*Note: This project will have different themes on different OS The first testing done on the Windows While the rest is done on Mac OS X)*

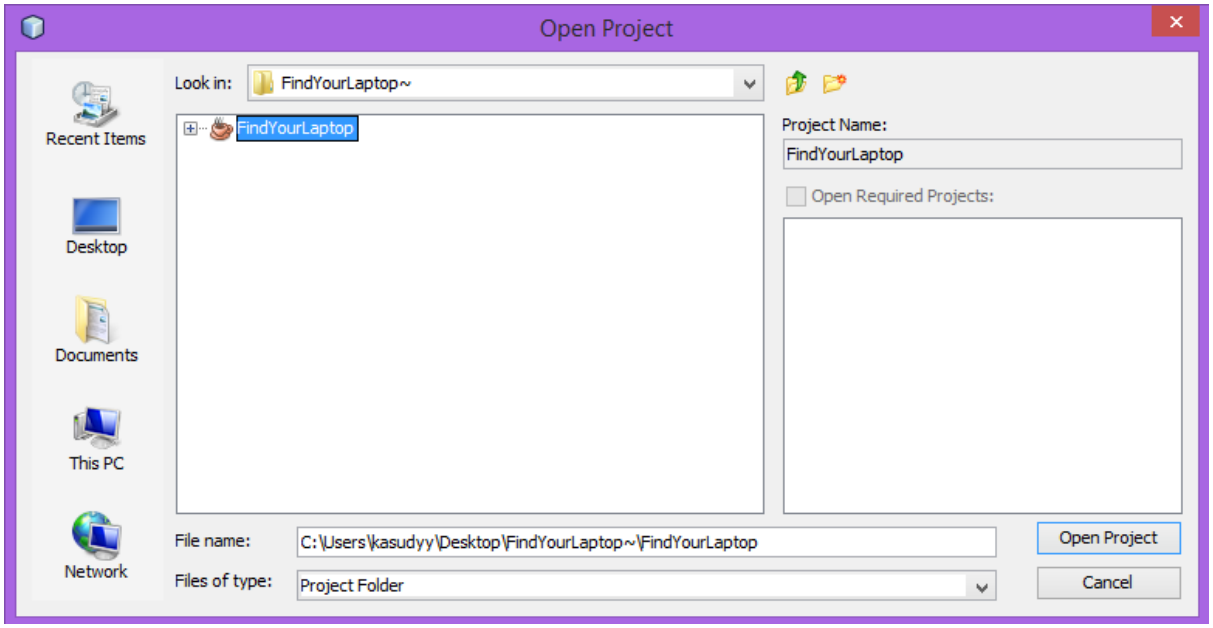

Figure 2: (Test 1): Importing Project File

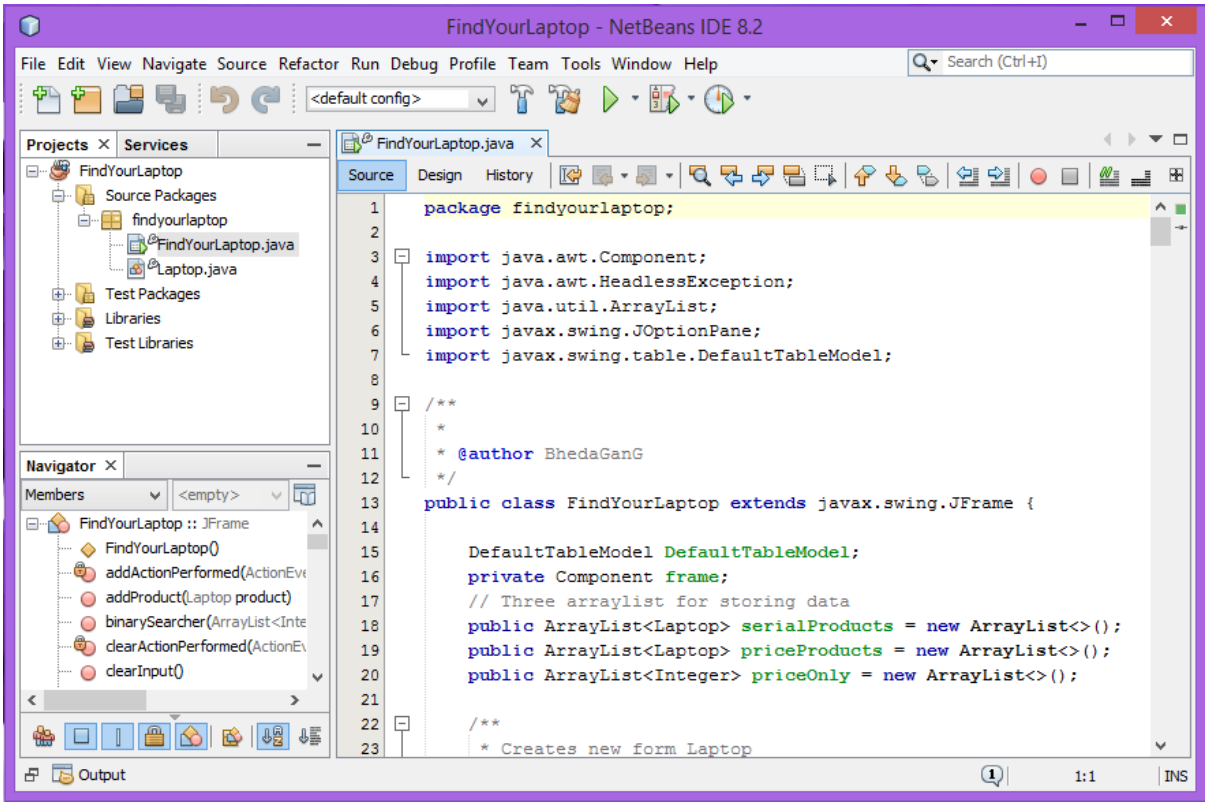

Figure 3: (Test 1): Running the GUI class

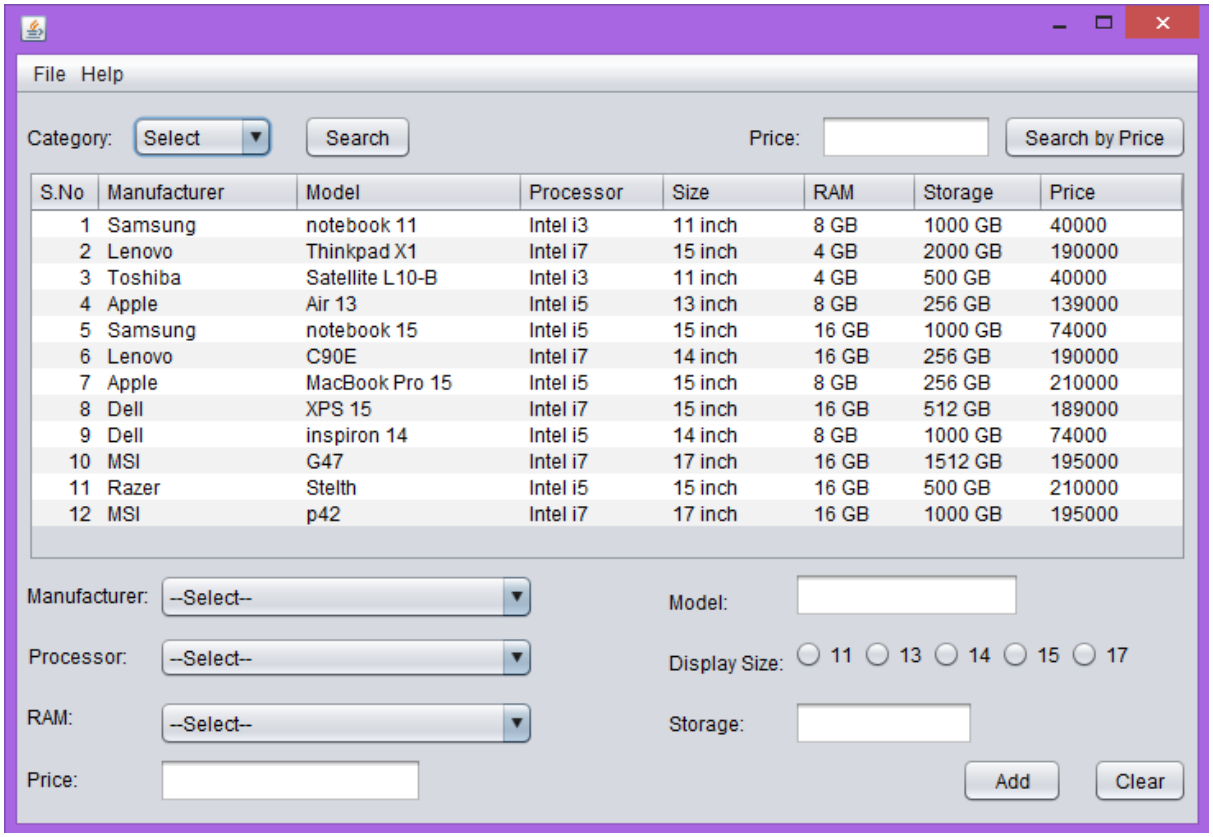

Figure 4: (Test 1): Find Your Laptop Running

2. Adding an Item:

Table 4: (Test 2) Adding an item

Objective	Checking whether the program adds new items or not. Also if all the information gets added or not.
Action	Adding new products with a unique Model number Checking the added data in the table
Expected Result	The product will get added, Serial Number will be generated, The product is displayed on the table.
Actual Result	The product was added, New Serial Number for the product was generated, The product was displayed at the table.
Conclusion	Test Successful.

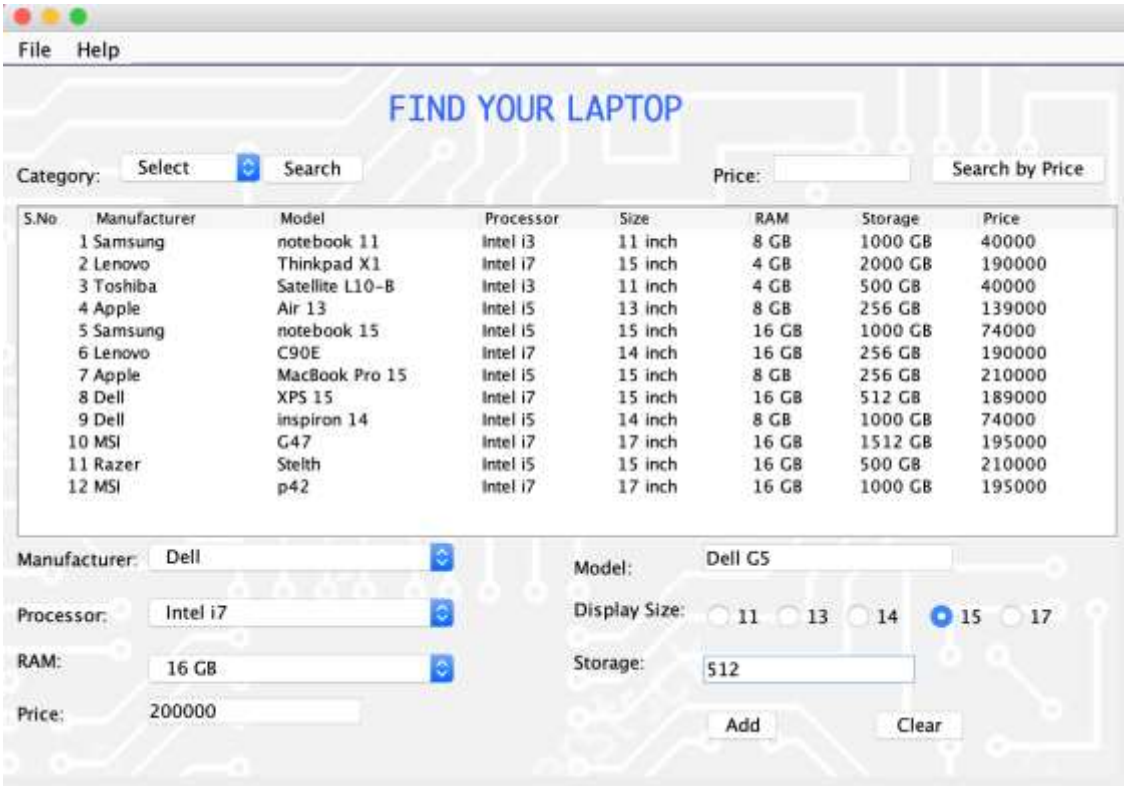

Figure 5: (Test 2): Entering data

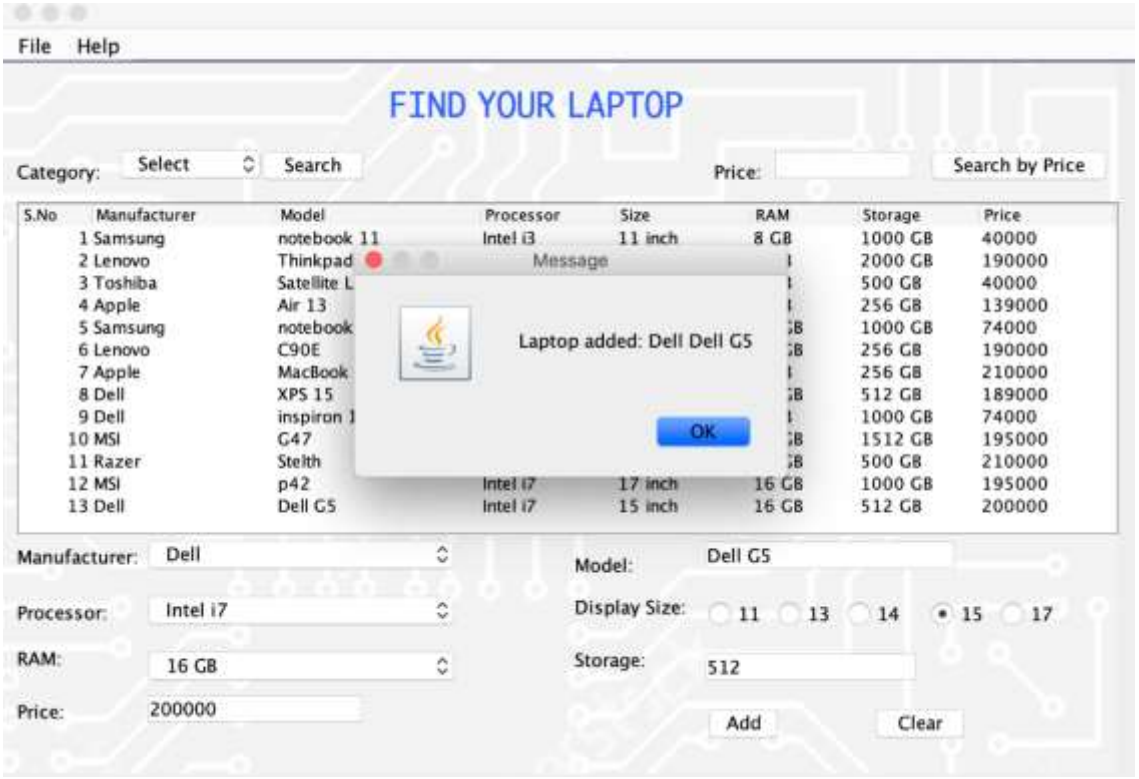

Figure 6: (Test 2): Message pops up after adding

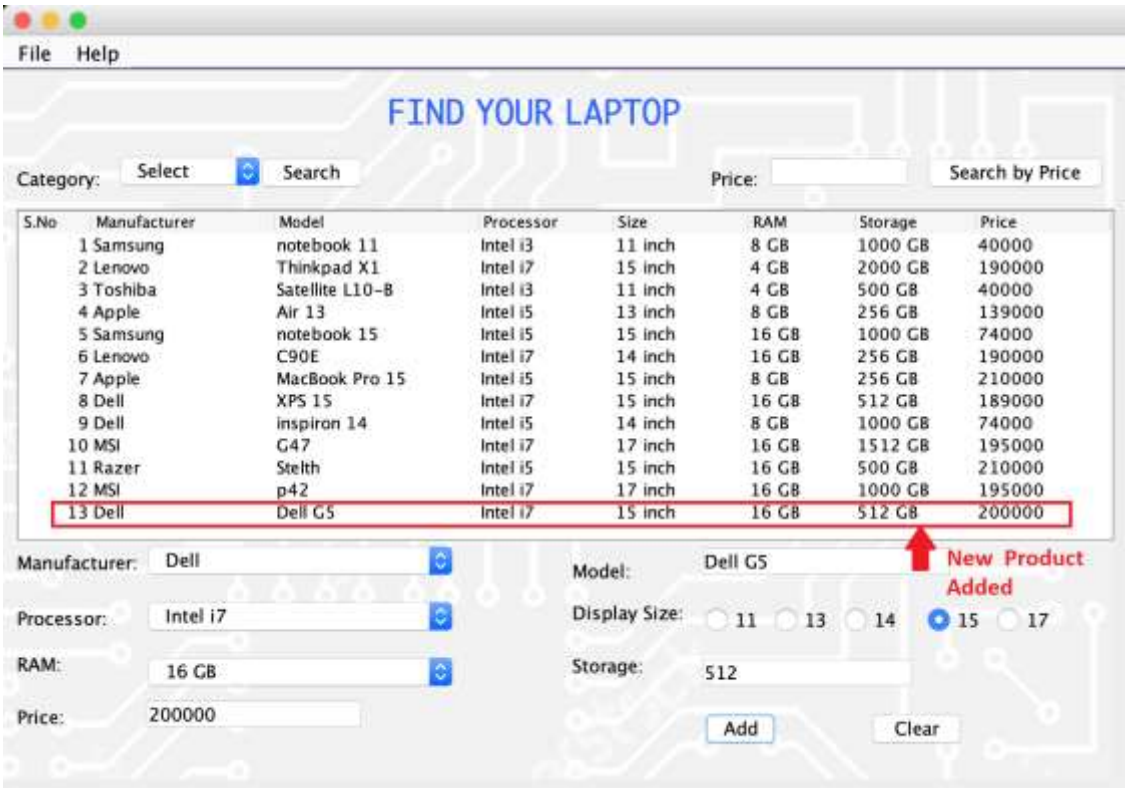

Figure 7: (Test 2): Item added in table

3. Searching by Category (Screen-size):

Table 5: (Test 3) Searching by category

Objective	Checking whether the program can find products according to the specific screen-size category.
Action	Searching laptops with 15-inch screen. Searching laptop with a 17 inch screen.
Expected Result	The program will show all the laptops with the selected screen-size when the search button will be pressed.
Actual Result	The program showed all the 15-inch laptops and 17-inch laptops when searching by category
Conclusion	Test Successful.

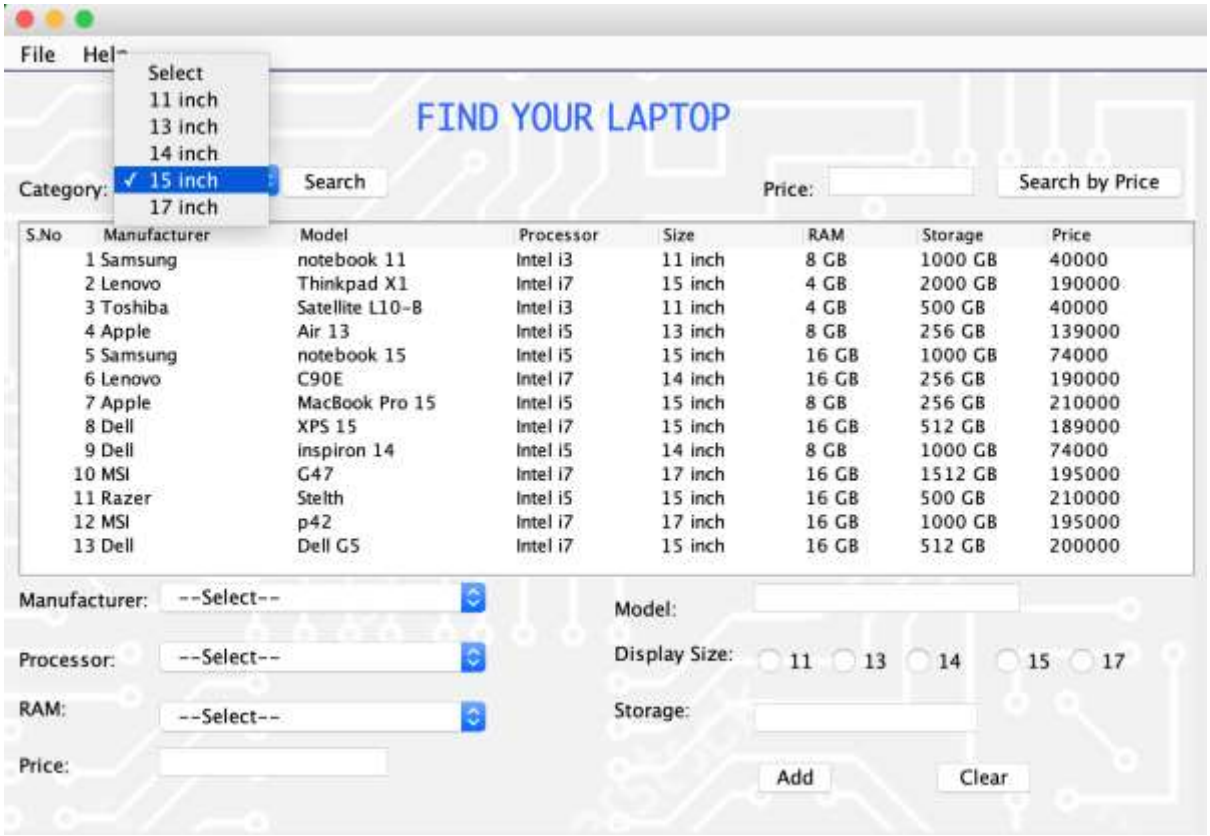

Figure 8: (Test 3): Searching by category, selecting 15 inch

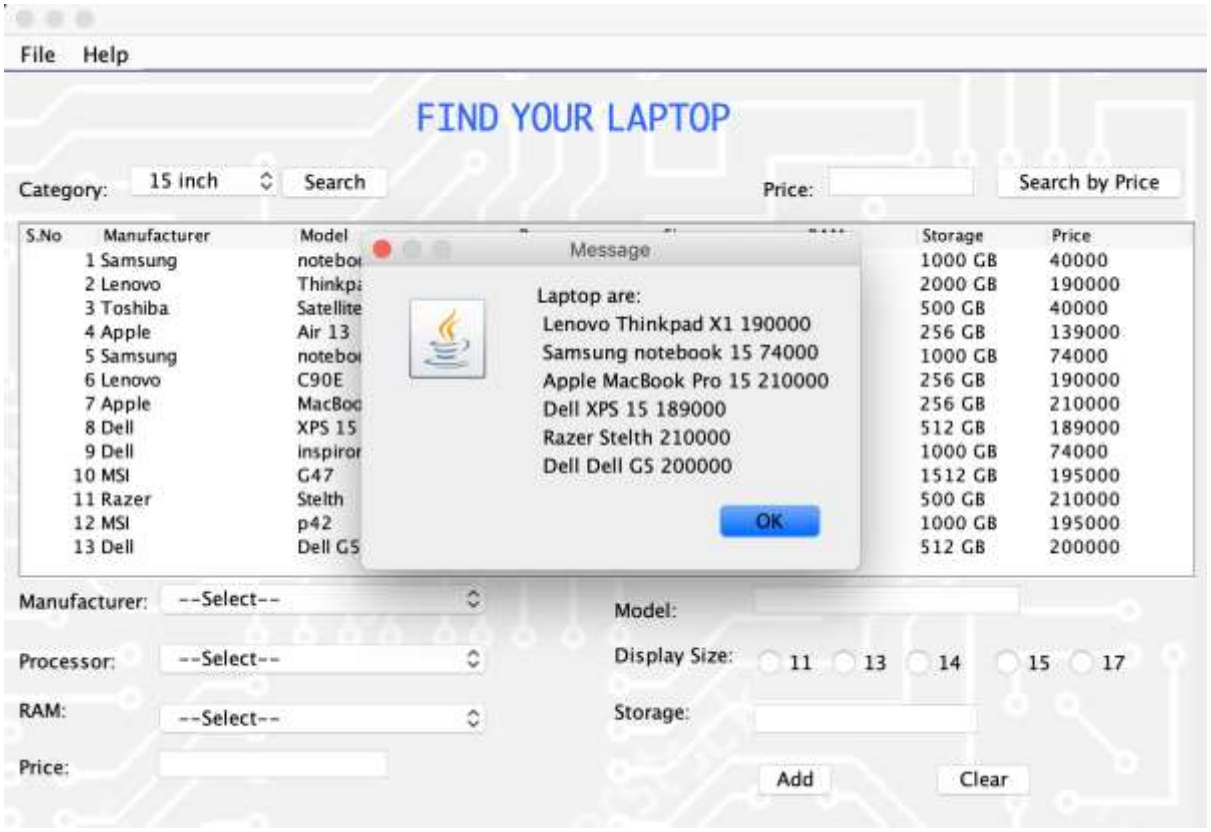

Figure 9: (Test 3): Displaying all 15 inch laptops

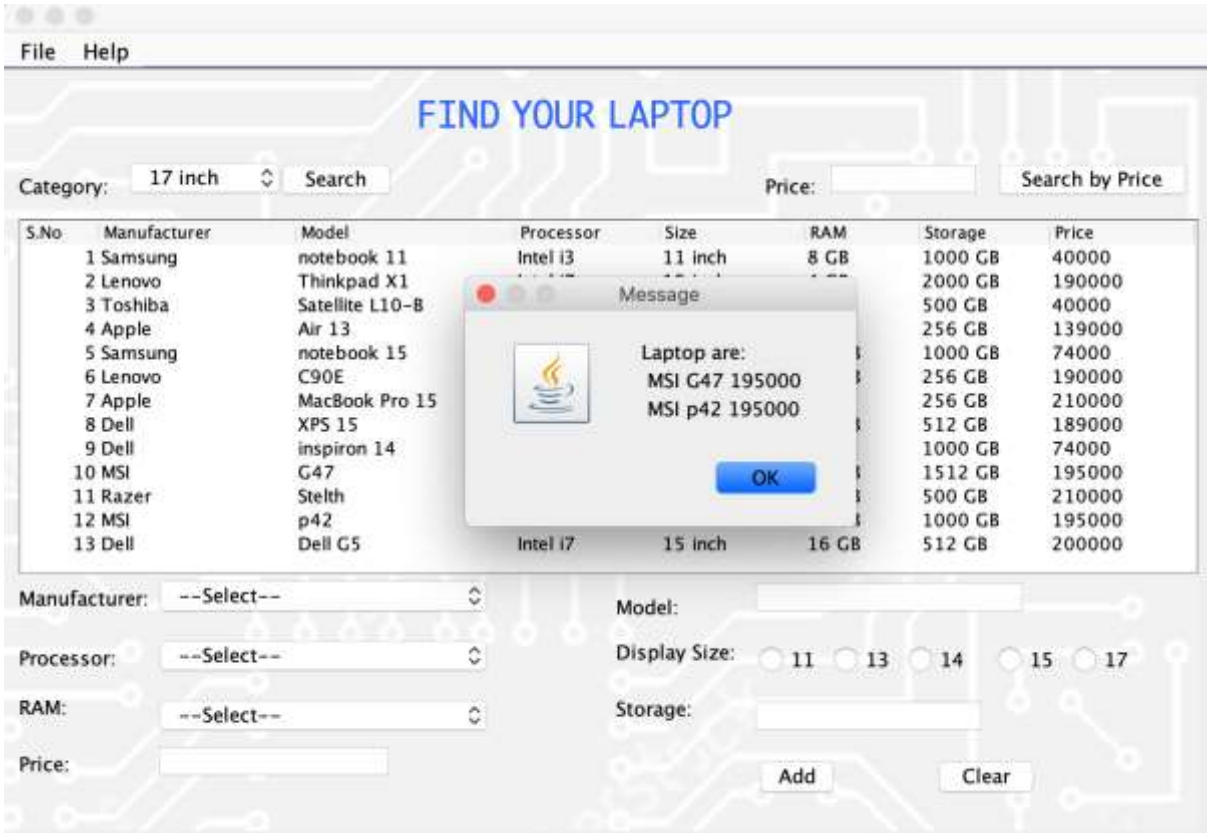

Figure 10: (Test 3): Similarly, displaying all 17 inch laptops

4. Searching by Price:

Table 6: (Test 4) Searching by price

Objective	Checking whether the program can find products according to price.
Action	Searching a laptop with a price: 210000 (only one device has this price Searching for a laptop with a price: 40000 (multiple devices have this price)
Expected Result	The program will show one item when the search button is pressed if the system contains a product with that price. Else the program will notify the user that the product with that specific price doesn't exist in the system.
Actual Result	The program showed a laptop when there was only one item, Program notified the user about the absence of a laptop when the price didn't match to any entry, The program showed the first laptop when there was multiple numbers of products with the same price point
Conclusion	Test Successful.

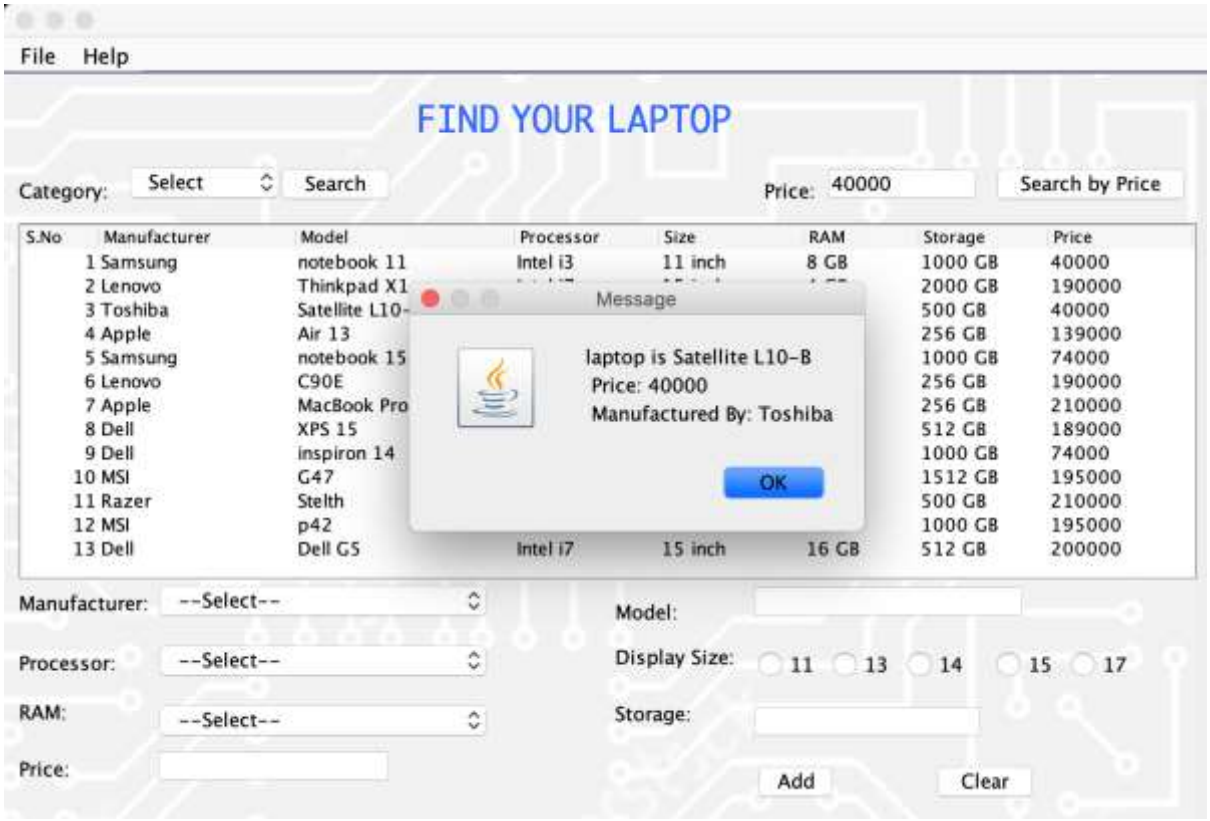

Figure 11: (Test 4): Searching a laptop by price 40000 (With multiple entries)

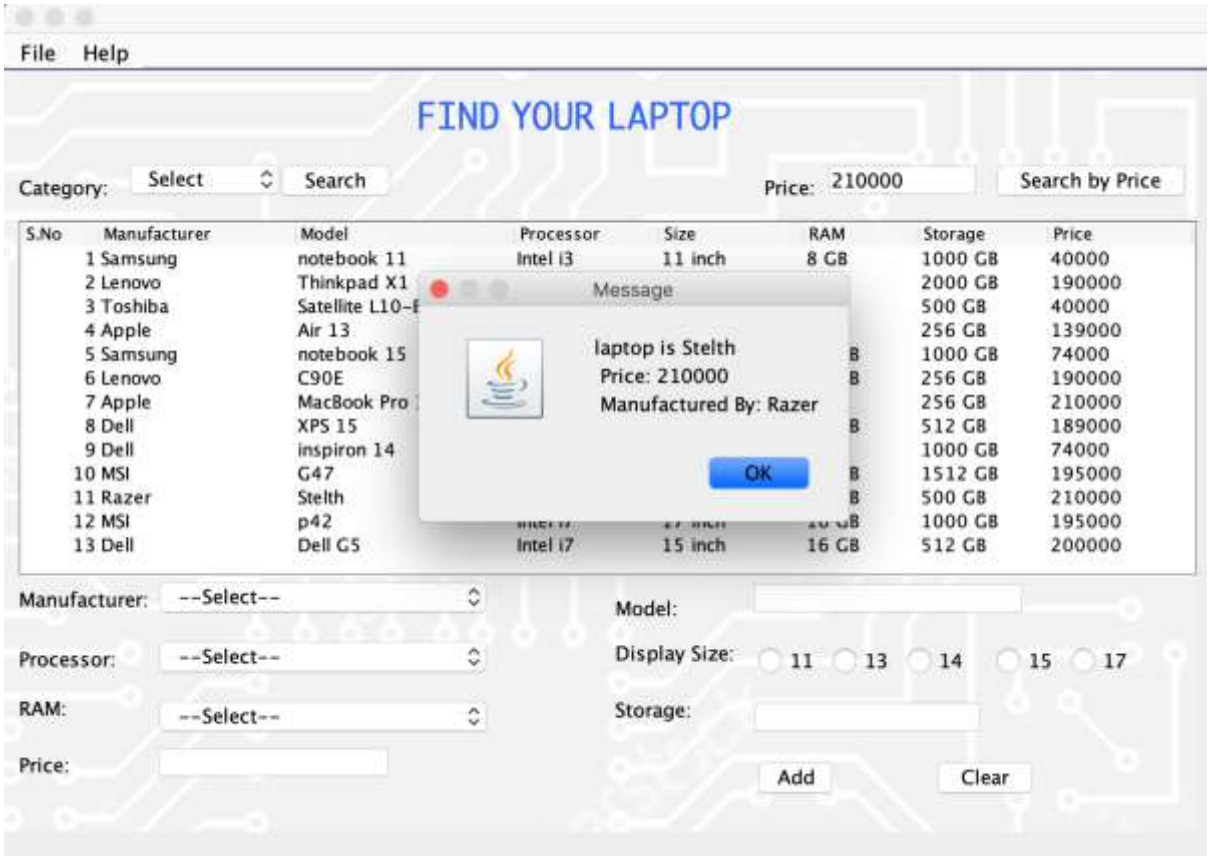

Figure 12: (Test 4): Searching a laptop by price 210000 (With Single entry)

5. Exception Handling:

Table 7: (Test 5) Exception Handling

Objective	Checking whether the program can handle various exceptions or not
Action	Inputting String where integer needs to be taken, Inputting number greater than the highest product price at the search by price.
Expected Result	The program will display error in a pop-up box to notify about the errors that have occurred due to improper use of the system.
Actual Result	The program specified exact reason for the error for the user suggesting they alter their actions for the right result.
Conclusion	Test Successful.

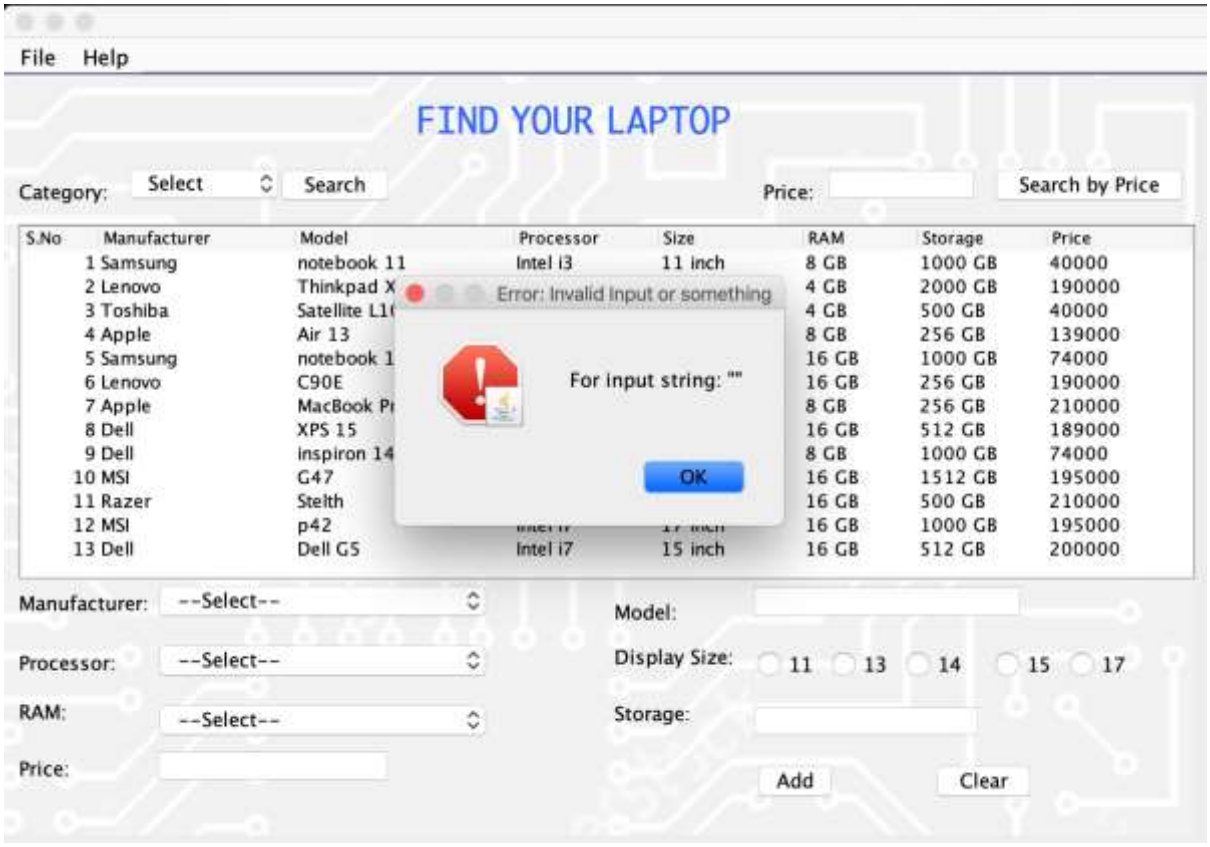

Figure 13: (Test 5): Error due to empty fields which require integer input

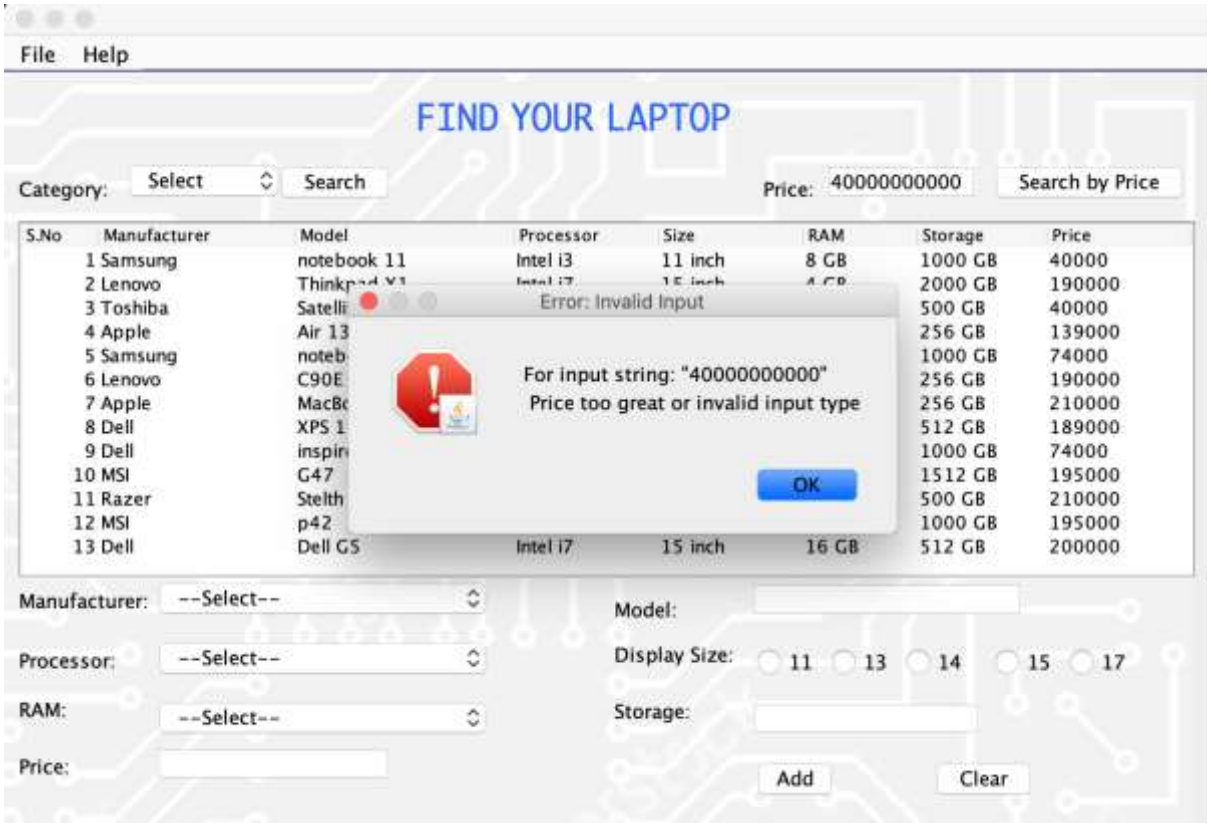

Figure 14: (Test 5): Error due to higher than the highest price in search by price

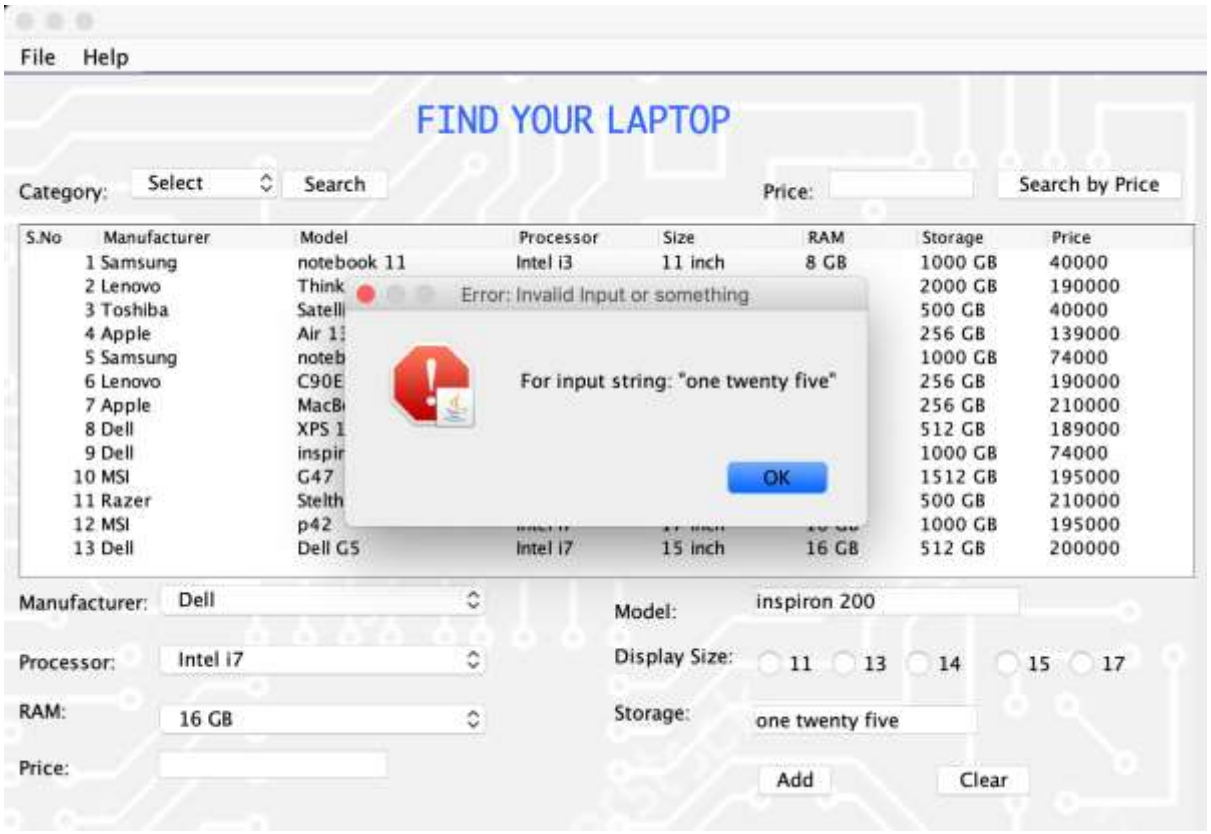

Figure 15: (Test 5): Error due to string input in place of integer

6. User Input Error Handling

Table 8: (Test 6) User Input error handling

Objective	Checking whether the program can detect empty fields or repeated model number in the program.
Action	Add button was pressed with all the fields empty, Add button was pressed with no screen size selected(radio-button), Add button was pressed with no price (text-field), Add button was pressed with no processor (combo-box), Add button was pressed with duplicate model-no(duplication check)
Expected Result	The product doesn't get added and the user gets notified about the errors in an understandable context in a pop-up box.
Actual Result	The product wasn't added and the user was notified about the incomplete or duplicate entries.
Conclusion	Test Successful.

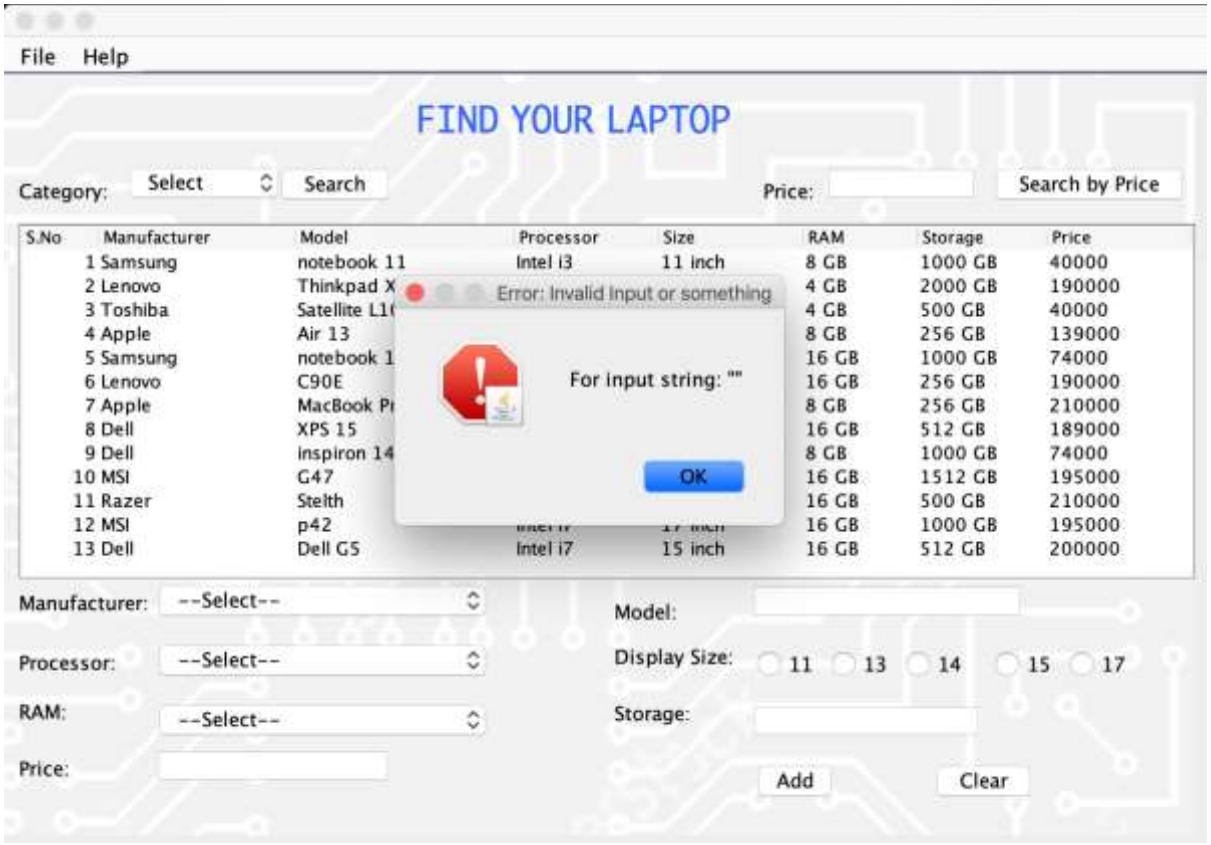

Figure 16: (Test 6): Error due to Empty Fields

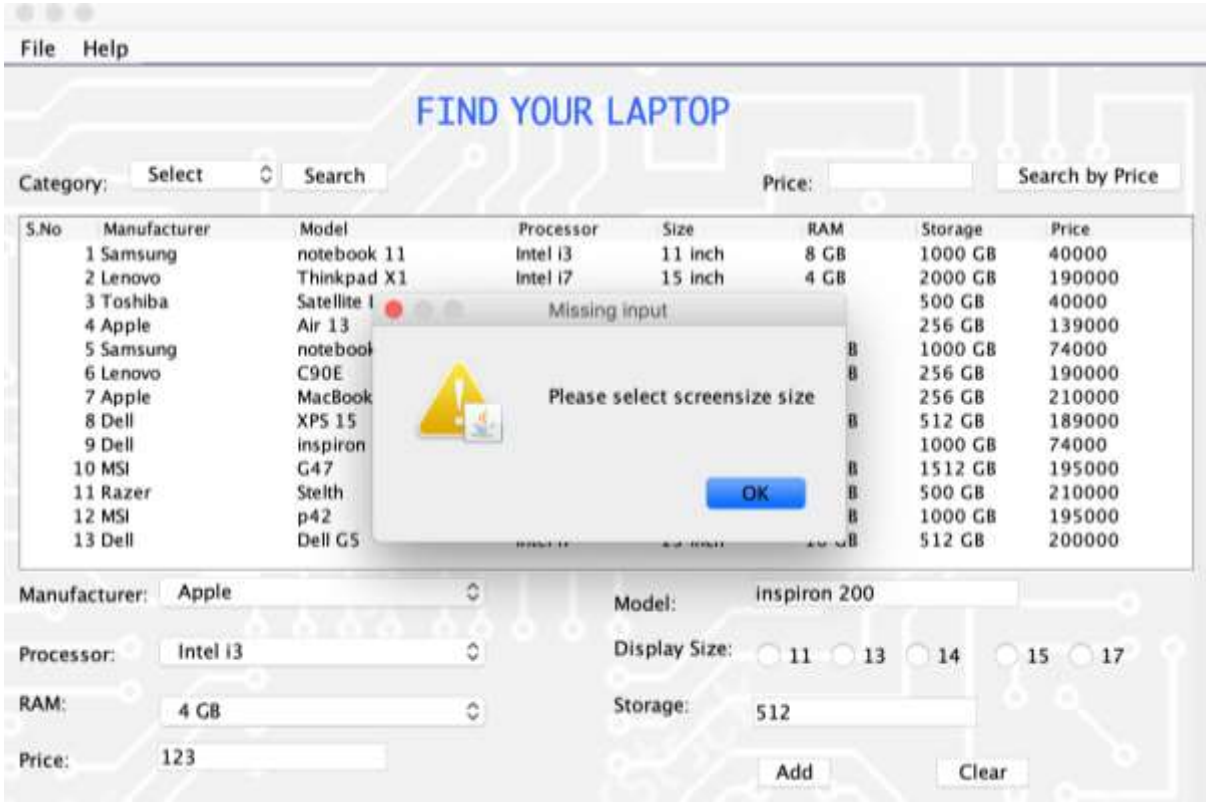

Figure 17: (Test 6): Error due to missing screen-size details

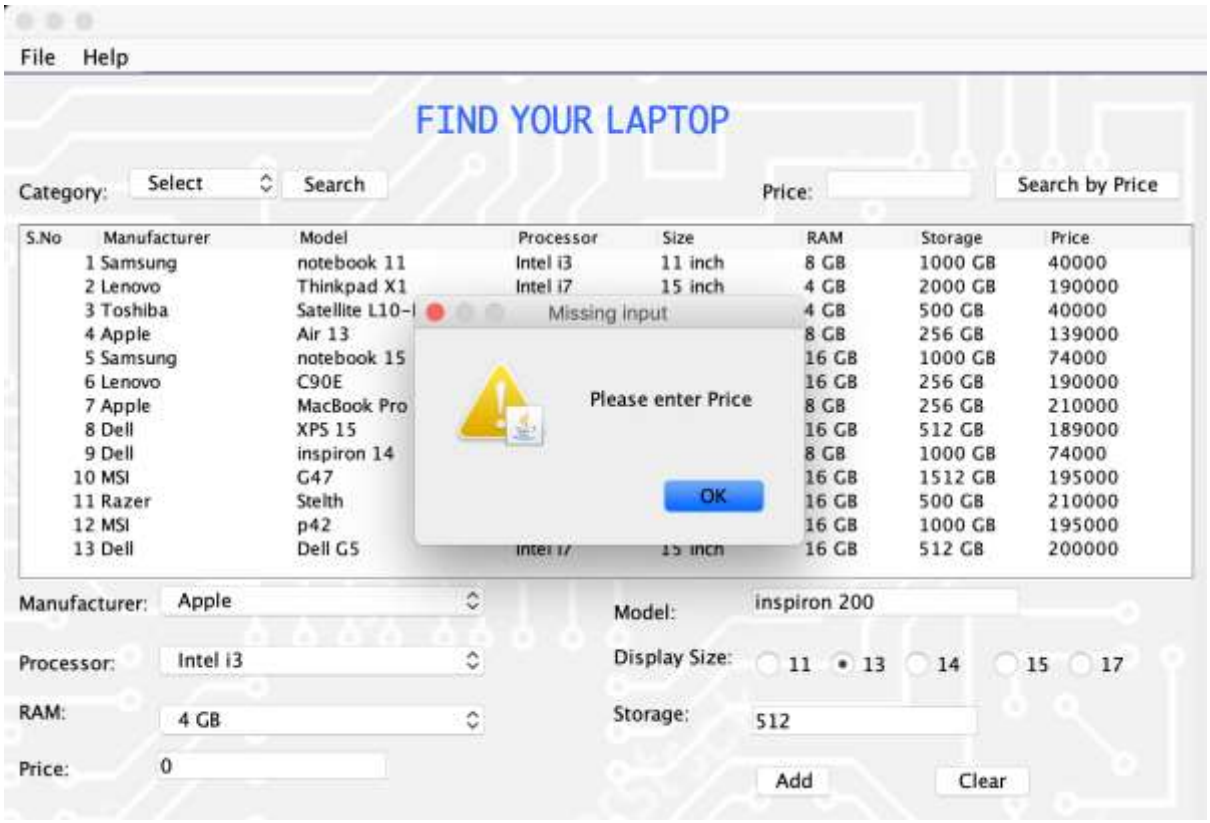

Figure 18: (Test 6): Error due to missing price

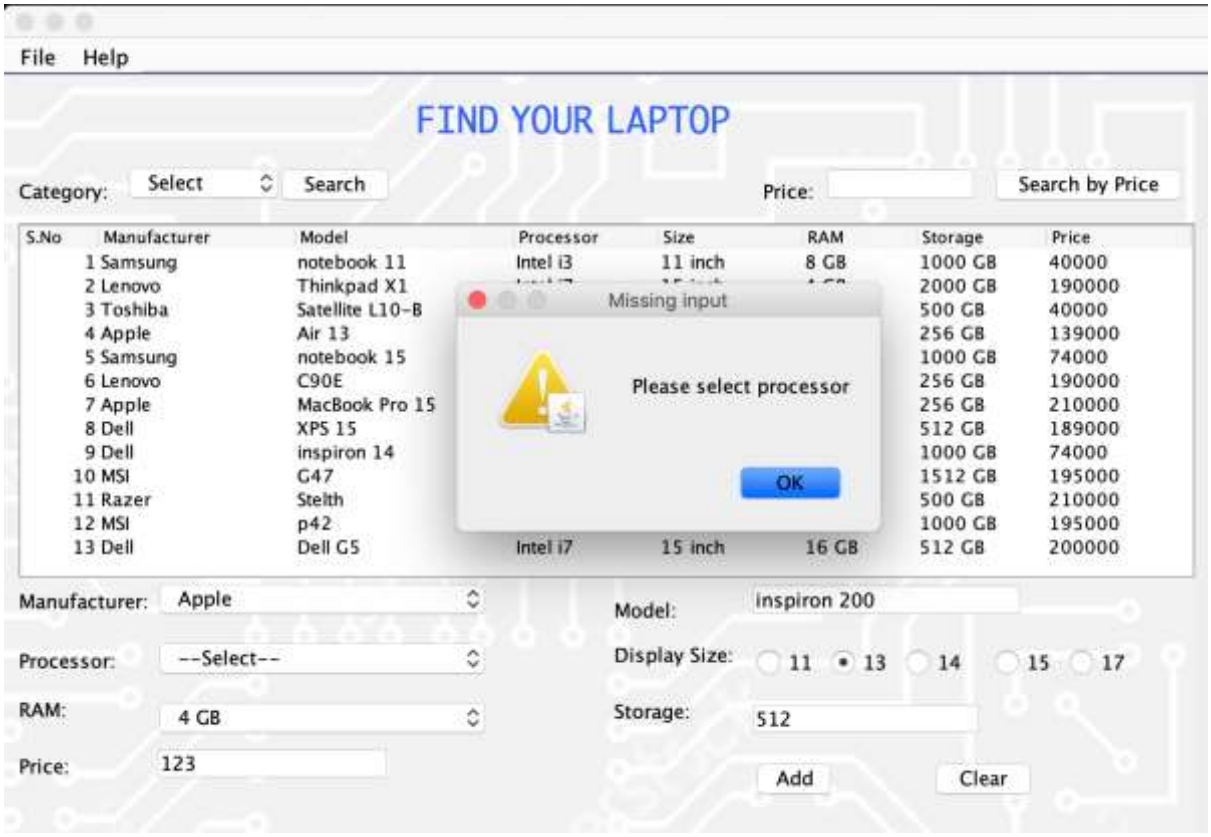

Figure 19: (Test 6): Error due to missing processer details

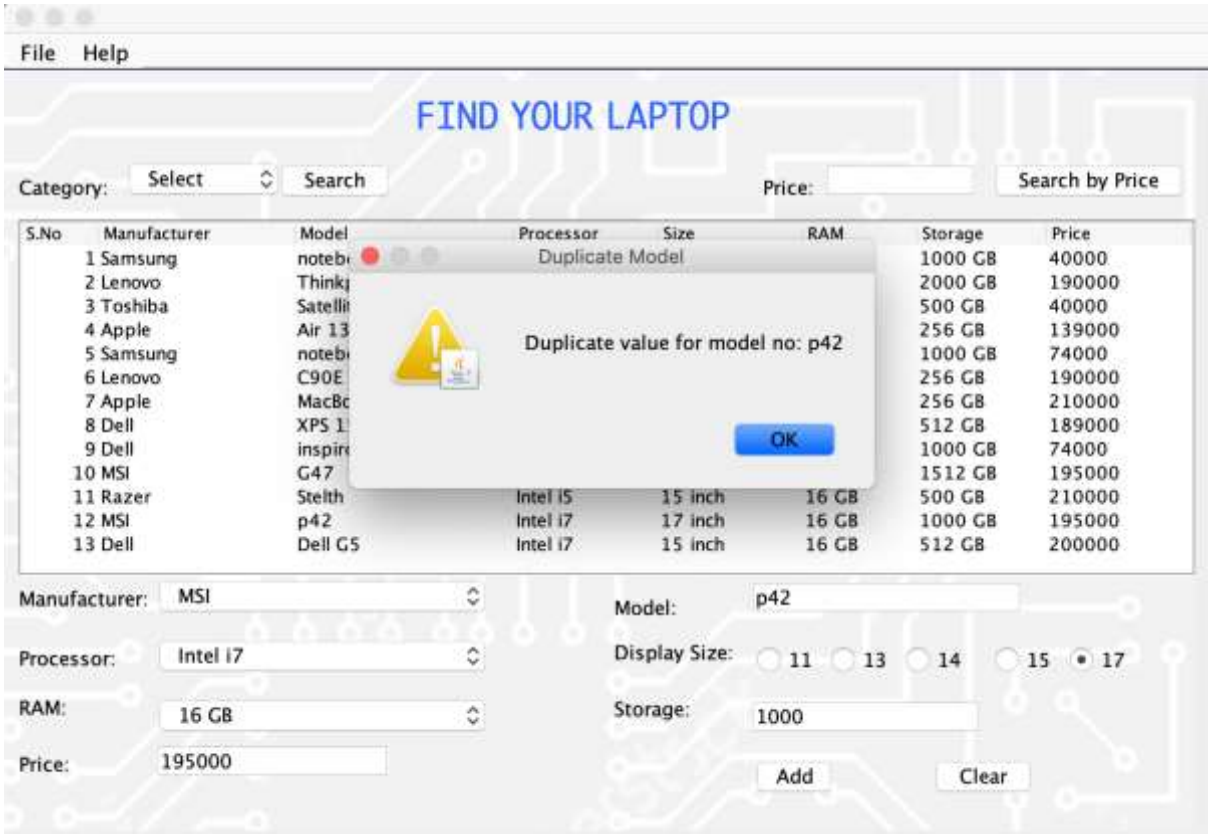

Figure 20: (Test 6): Error due to duplicate value for model number

7. Checking the functionality of other features:

Table 9: (Test 7) Other Features checking

Objective	Checking whether the menu items, clear button and sorting the table works
Action	Clearing the input via the “clear” button, Clearing the input via the “clear” menu item in the file, Clicking the column indicator to sort the product, Exiting the program. Trying to open the documentation from the “view documentation” option on the menu item within help.
Expected Result	Input fields will be cleared by the button and the menu item, The items will be sorted according to the column, The program will exit if the exit menu item is pressed, Documentation file gets opened
Actual Result	The input fields were cleared, Items were sorted, Documentation file was opened, The program was terminated.
Conclusion	Test Successful.

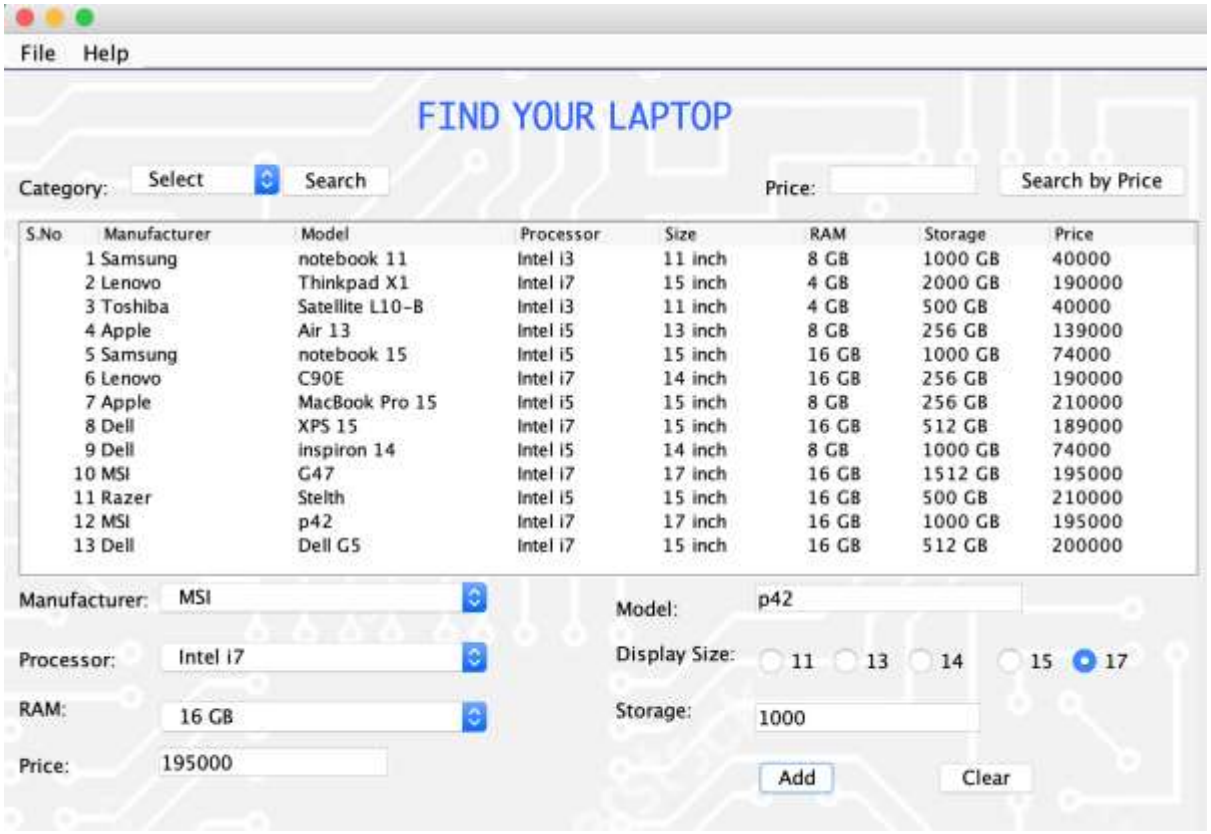

Figure 21: (Test 7): Unwanted user input to be cleared

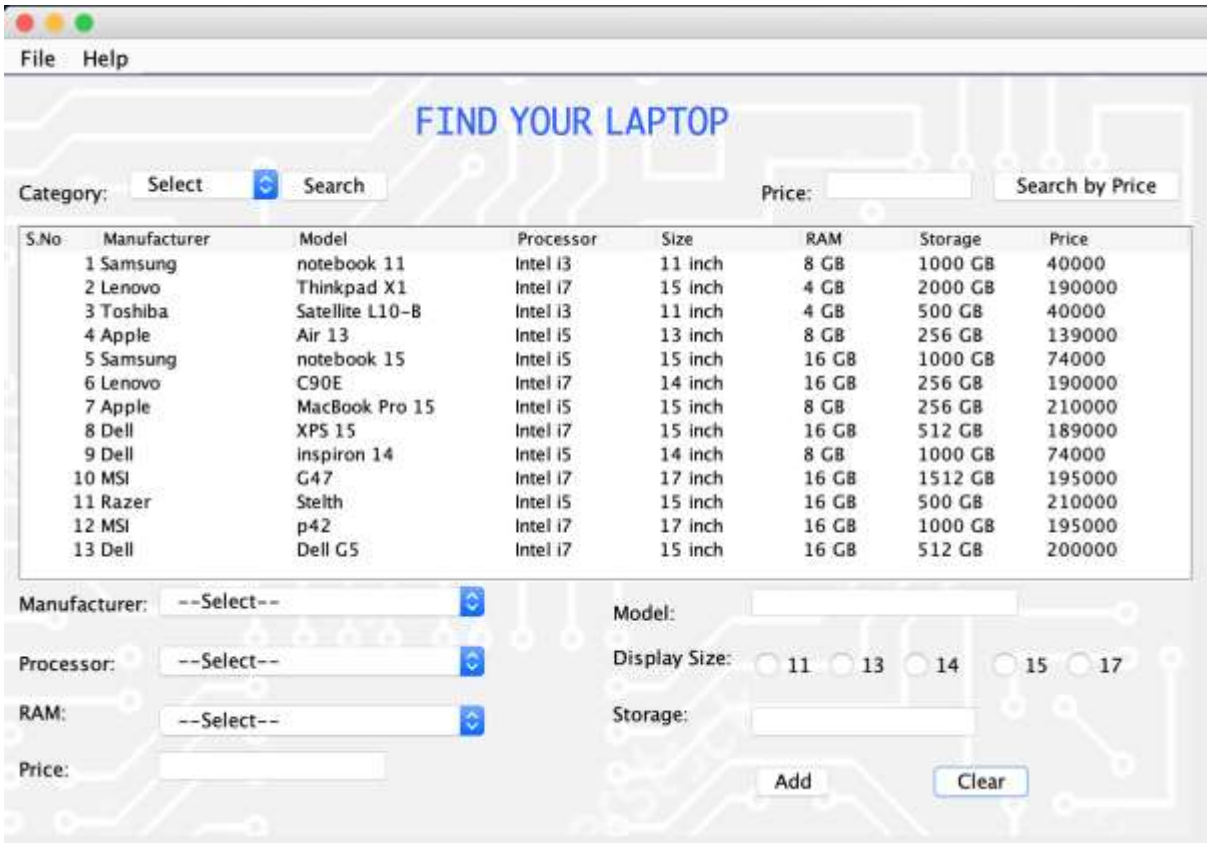

Figure 22: (Test 7): Pressing Clear Button

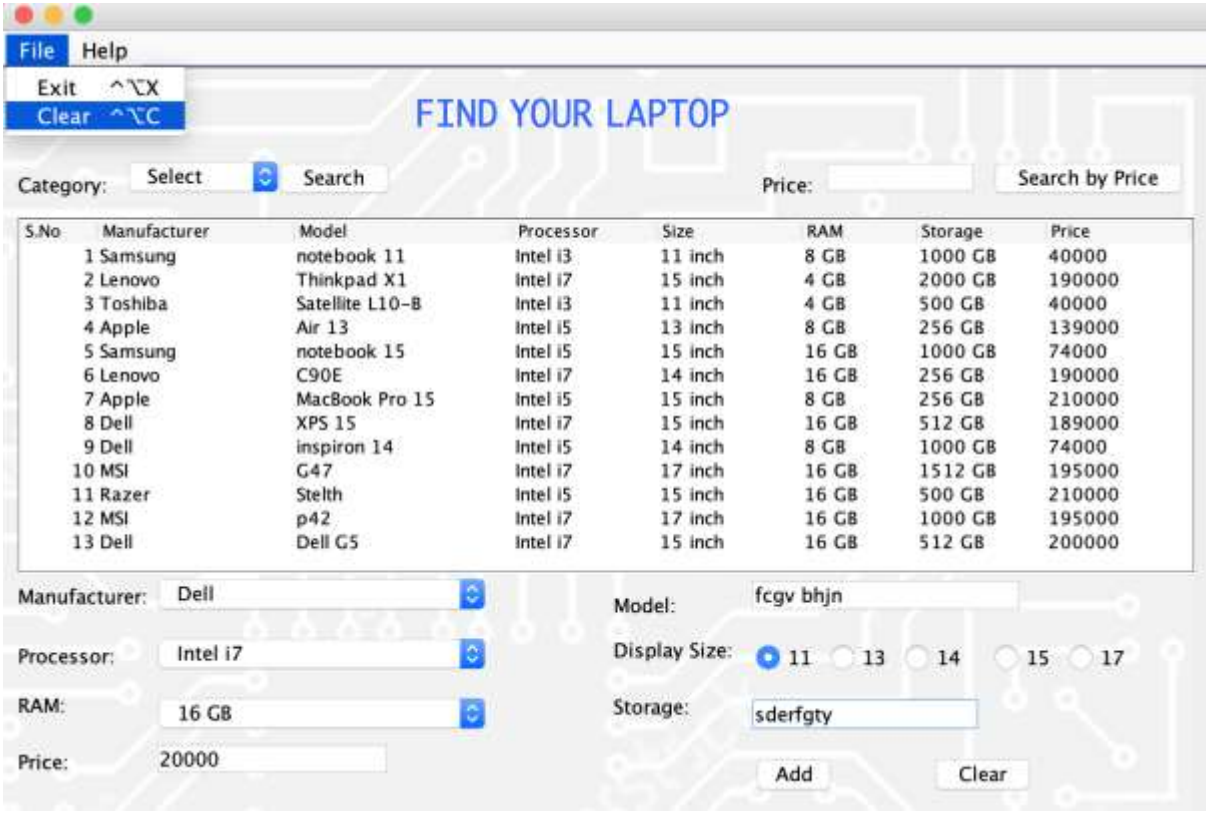

Figure 23: (Test 7): Clearing using Menu item Clear option

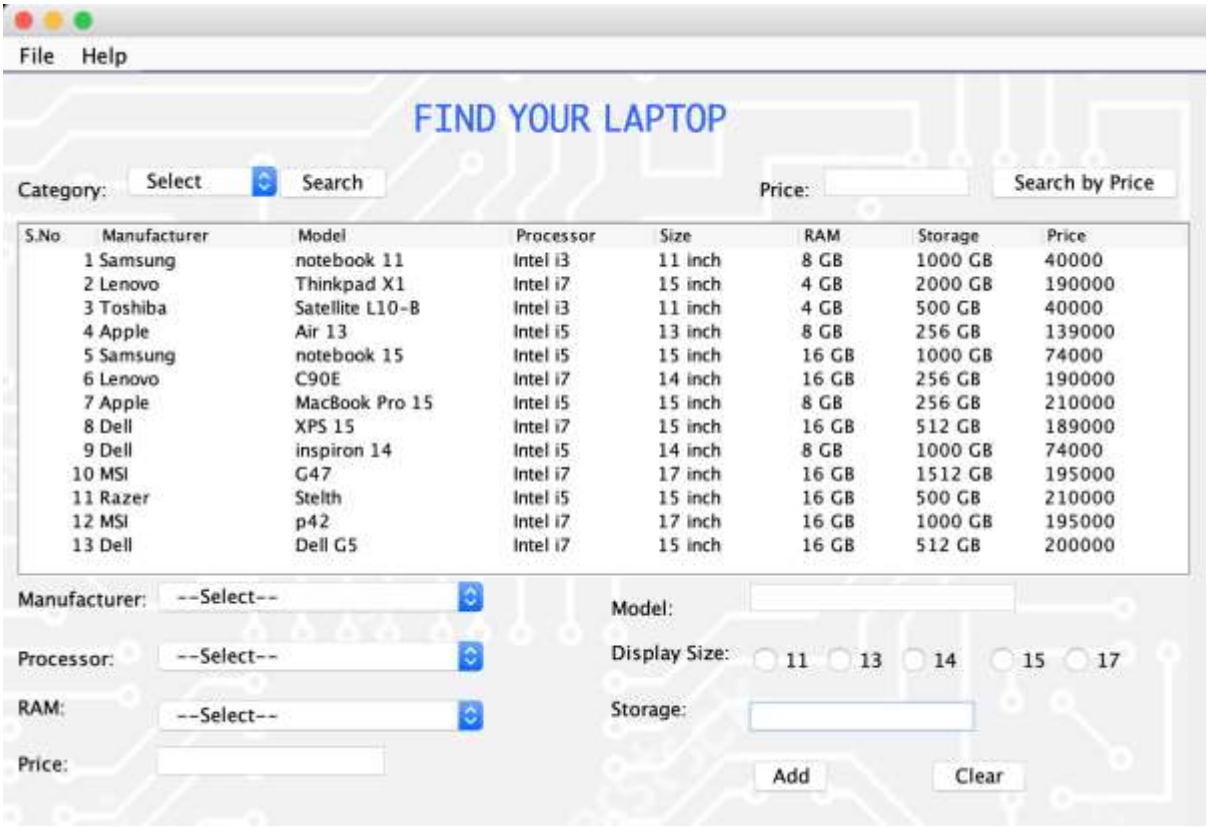

Figure 24: (Test 7): Cleared User input due to the Clear action

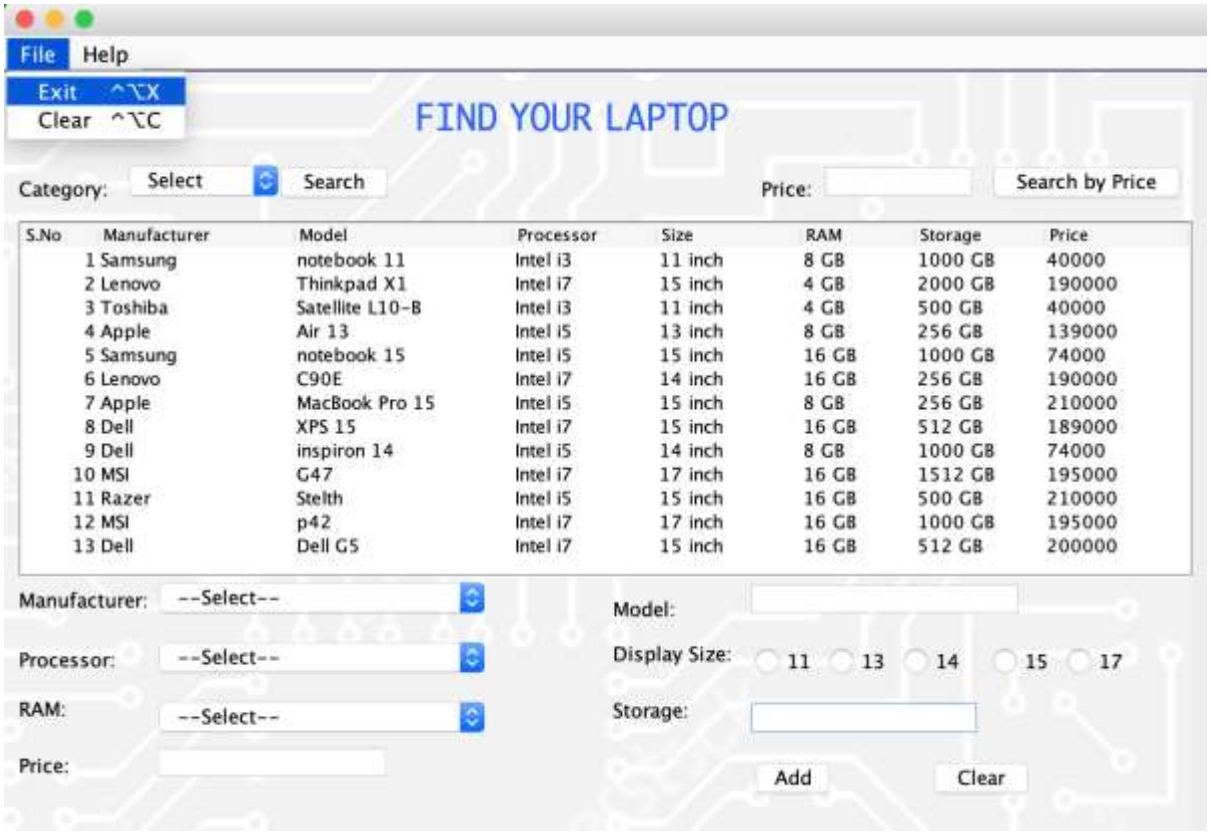

Figure 25: (Test 7): Checking the close option

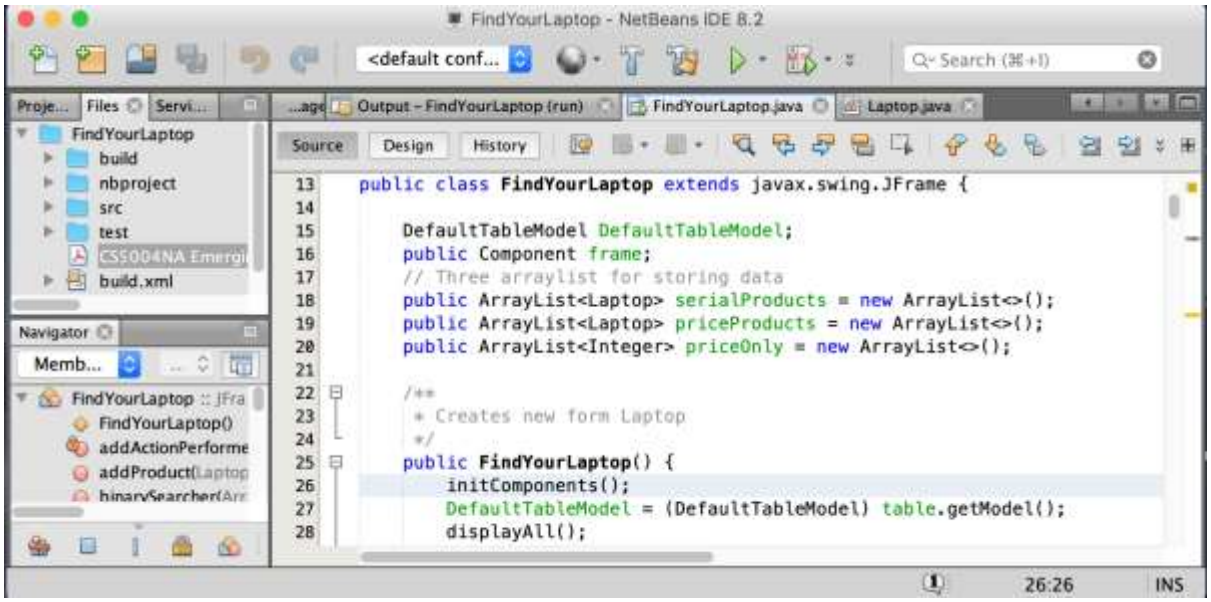

Figure 26: Program terminated due to close action

Conclusion

While developing any kind of information systems, data must be created, stored, appended/modified or deleted. So to make this task easier data structures must be implemented. And to process any piece of information, to sort or to search various algorithms are required to be implemented in the program. The sorting algorithm used in this program is a variant of Selection sort, in this modified selection sort will automatically sort the user input in the ArrayList according to their price. Even though linear sort is quite inefficient, this version of selection sort was used because it is easier to understand and works well on a small amount of data. The sorting was required to sort the data according to their prices because the searching algorithm used in this program is a binary search which can work only on sorted data. Binary search is an efficient searching algorithm as it can be expressed in logarithmic functions. This means that this searching algorithm will work really well when there is a huge amount of data from which an item needs to be searched. A separate java class was also created, which handled all of the processes and methods required to compare and retrieve information about any laptop object in the main java class which provided a platform for the GUI.

The user interface is also a key factor that needs to be analyzed and work upon as it greatly influences the actions that a user can do in an information system. So to enhance the user experience the GUI is made as simple and attractive as possible so that it will affect the user positively while using this program. While testing, the software was required to perform certain task which it has done effortlessly. The GUI can be easily opened by importing the project and adding new data is relatively easy. The program will sort the data as it is entered according to its price so when the user searches any product the product can directly use binary search algorithm to search for that item. The program also features many small other features like clearing all the input fields, exiting the program and viewing the user guide for the program. The help button also contains a function that will list out all the items stored in the ArrayList which can be observed in the console.

References

Hopping, C., 2018. *What is a graphical user interface?.* [Online]
Available at: https://www.itpro.co.uk/operating-systems/30248/what-is-a-graphical-user-interface
[Accessed 18 1 2019].

Jacobson, N. & Thornton, A., 2004. It is Time to Emphasize ArrayLists over Arrays. *inroads – The SIGCSE Bulletin ,* 36(4), pp. 88-92.

Jenkov, J., 2015. *Java Methods.* [Online]
Available at: http://tutorials.jenkov.com/java/methods.html
[Accessed 17 Jan 2018].

Knoernschild, K., 2002. *Java Design: Objects, UML, and Process.* 1st ed. Boston: Addison-Wesley Professional.

Mahmoud, H. M., 2011. *Sorting: A Distribution Theory.* 1st ed. New Jersey: John Wiley & Sons.

Oracle , 2019. *Package javax.swing.* [Online]
Available at: https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html
[Accessed 18 1 2019].

Rouse, M., 2014. *What is object-oriented programming (OOP)? - Definition from WhatIs.com.* [Online]
Available at: http://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP
[Accessed 18 Jan 2018].

The Interaction Design Foundation, 2018. *User Experience (UX) Design.* [Online]
Available at: https://www.interaction-design.org/literature/topics/ux-design
[Accessed 18 1 2019].

Appendix

1. The limitations of the binary search algorithm:

1. When there are two items which fulfill the description of the search term the binary search will only return the first term it gets matched with.

2. When the search term doesn't exist within the sorted data list, the binary search will not break the process but continues to search at the probable proximity of the search term until the algorithm realizes that the term doesn't exist.

3. The binary search algorithm requires a sorted data to start searching. If the data is not sorted the binary search will not function at all.

4. While working on a small amount of data, the binary search will first have to sort the data then perform a binary search. This will take more time to sort and search than by using just linear searching algorithm to search the term.

5. The binary search algorithm used in this program will only return the index of the first item it comes across.

6. The other items with duplicate value are rejected or neglected according to this algorithm.

2. Detailed method description for the binary search:

The method that implements the principle of the binary sort is mentioned below:

```
public int binarySearcher(ArrayList<Integer> priceArray, int leftIndex,
int rightIndex, int x) {
  if (rightIndex >= leftIndex) {
    System.out.println("Searching...");
    int middleIndex =  ((rightIndex - leftIndex)/2) + leftIndex;
    if (priceArray.get(middleIndex) == x){
      System.out.println("found");
      return middleIndex;
    }
    else if (priceArray.get(middleIndex) > x){
      System.out.println("left");
      return binarySearcher(priceArray, leftIndex, middleIndex-1, x);
    }
    else{
      System.out.println("right");
      return binarySearcher(priceArray, middleIndex+1, rightIndex, x);
    }
  }
  return -1;
  }
}
```

This method returns the index of the found term which will make it easier to call up the whole detail on the product. As the sorted price list is mapped with the sorted list containing the laptop objects. When the method is invoked when the search button is pressed the index is stored as an integer variable which is then used as an index to display the product details.

User Guide

Kasari chalaune app?