Vietnamese – German University

COMPARE SEARCH ENGINE

NGUYEN TIEN HUNG – 9972

LE QUOC HUNG – 11084

DECEMBER 14, 2017

Compare Search Engine

Team 2 – Team name 2

Nguyễn Tiến Hưng – 9972

Lê Quốc Hưng – 11084

# 1. Topic introduction

As the beginning of the 21th century, traditional shopping ways has been gradually replaced by online stores. On the other hand, online products quality development was not meet as the product number pacea, this leading to difficulties for consumer in making choice in price, product, producer chaos.

To join hands in online shopping quality control, a third-party software for supply consumers a widen view in price chaos is necessary. That also our project purpose, a java software for collecting prices and specification from online products from the most three reputable online retailers which are Lazada, Thegioididong and FPTshop, which will help client makes a better choice.

# 2. Topic description

Compare Search Engine is a java software class project help online customers have a better view in the product price chaos in the e-commerce.

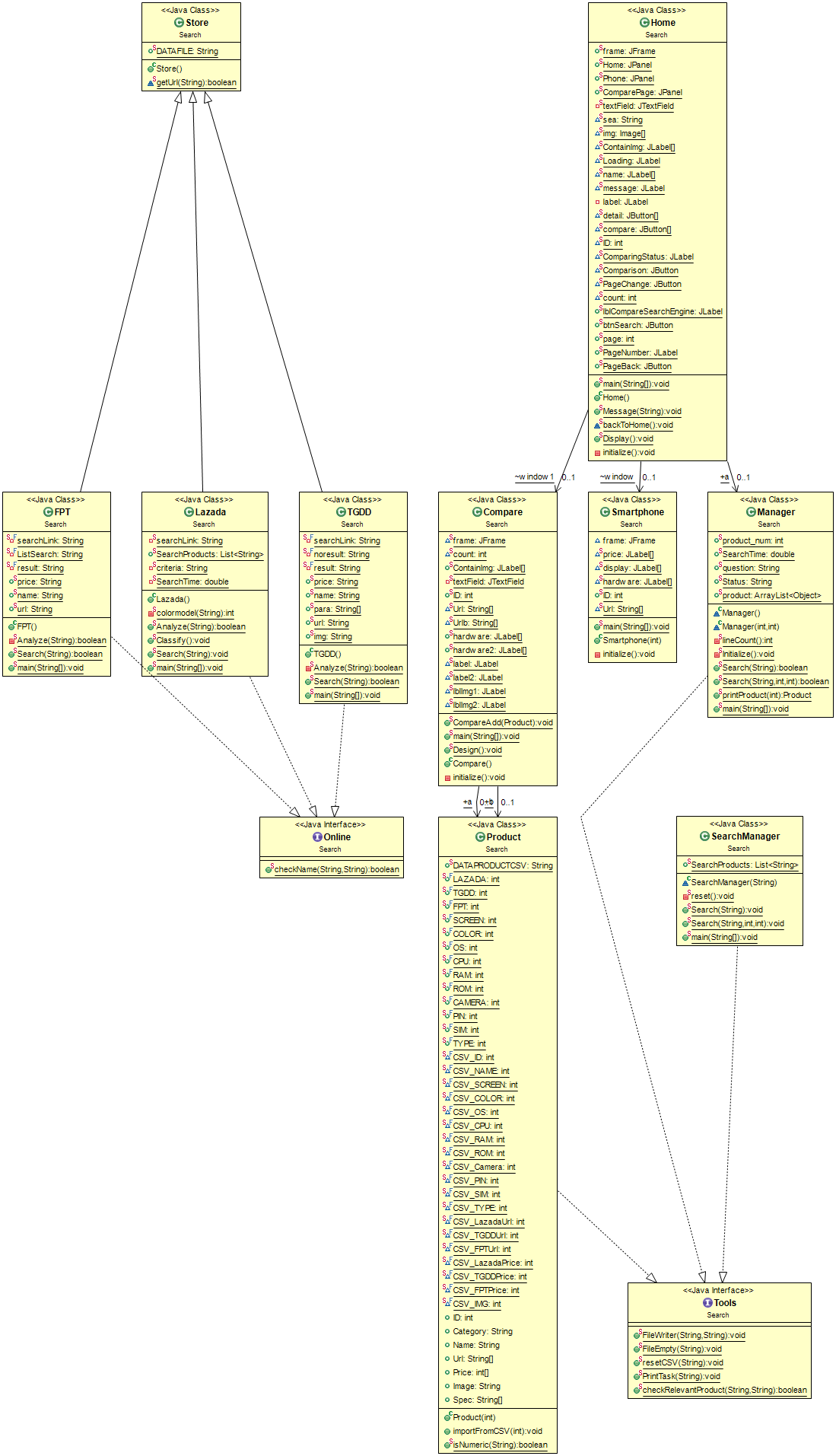
Our project detail goals:

* Collecting price and specifications correctly from Lazada, Thegioididong and FPTshop.
* Displaying prices and specifications in a obvious interface.
* Search engine must work effectively in oder to supply client needs.
* Compare engine provides obvious products information.
* Scope on Electronic Device category only.
* Software run correctly, speed is not required.

Our project futher goals:

* Can be use for all categories.
* Support other retailers.
* Increase search speed.
* Apply AI for Search Engine.
* More responsive and attractive User Interface.

# 3. Class diagram

.

*Home* class which is the main user interface have three frames for containing application funciton controllers. Home frame is for display list of products and funtional buttons. Second frame displays comparison information from *Compare* class between 2 chosen smartphones, and last frame uses for displaying products specifications and prices from *Smartphone* class. *Home* also can manage and search products by associating with *Manager* class which keep the product data and control the Search function.

Furthermore, *SearchManager* is used for control the cursor of advange search effectively to provide correct and quick responsive products from user needs.Firstly, *SearchManager* sends user need to *Lazada* class to get related products and also classify it into different kinds, then sends found Lazada’s products to *TGDD* and *FPT* in oder to finding corresonding products and get price.

*FPT*, *TGDD*, and *Lazada*, which inherit from Store class when it downloads and classified data from Lazada, FPT, and TGDD websites. These classes also implement Online class to check corresponding product as user need.

# 4. Details

## 4.1. Classes

1. **Home**

**Field Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Field and Description** |
| public static JPanel | Home  It represents the Home graphic user interface. |
| public static JPanel | Phone  It represents the detail of each smartphone. |
| public static JPanel | ComparePage  It represents the comparison between two chosen smartphones. |
| public static JTextField | textField  It display a empty box to input strings for searching. |
| static Image | img[]  It sets up images of smartphones. |
| static JLabel | ContainImg[]  It sets places for smartphones’ images. |
| static JLabel | name[]  Name of each product is set up. |
| static JLabel | message  It displays the number of products found. |
| static JButton | detail[]  It demonstrates technical specifications of each product. |
| static JButton | compare[]  It compares technical specifications between two smartphones. |
| static int | ID  It gets the specific smartphone. |
| static Smatphone | window  It creates a Smartphone class called window. |
| static Compare | window1  It creates a Compare class called window1. |
| static JLabel | ComparingStatus  It illustates how many smartphones chosen for comparison. |
| static JButton | Comparison  It changes to Compare class after choosing 2 smartphones. |
| static JButton | PageChange  It changes to another page to display more products found. |
| static JButton | PageBack  It goes backward to the previous pages. |
| static int | count  It counts the number of chosen smartphones. |
| public static JLabel | lblCompareSearchEngine  It displays a label called CompareSearchEngine. |
| pubic static JButton | btnSearch  It creates a button for searching. |
| public static int | page  It displays the order of pages. |
| public static JLabel | PageNumber  It displays which pages we are at. |

**Constructor Summary**

|  |  |
| --- | --- |
| **Constructor and Description** |  |
| public Home()  It is the main user interface, demonstrates number of products found, and contains other functions. | |

**Method Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| void | Message(String a)  Displays message about number of found smartphones. |
| void | Display()  Shows 6 smartphones displayed in a page, and also create *Search*, *Compare,Detail,Forward,* and *Backward* function for buttons. |

**public** **static** JFrame *frame*;

**public** **static** JPanel *Home*;

**public** **static** JPanel *Phone*;

**public** **static** JPanel *ComparePage*;

**public** **static** Manager *a*;

**private** **static** JTextField *textField*;

**static** String *sea*;

**static** Image *img*[] = **null**;

**static** JLabel *ContainImg*[] = **new** JLabel[6];

**static** JLabel *name*[] = **new** JLabel[6]; // product array

**static** JLabel *message*;

**private** JLabel label;

**static** JButton *detail*[] = **new** JButton[6];

**static** JButton *compare*[] = **new** JButton[6];

**static** **int** *ID*;

**static** Smartphone *window*;

**static** Compare *window1*;

**static** JLabel *ComparingStatus*;

**static** JButton *Comparison*;

**static** JButton *PageChange*;

**static** **int** *count* = 0;

**public** **static** JLabel *lblCompareSearchEngine*;

**public** **static** JButton *btnSearch*;

**public** **static** **int** *page* = 1;

**public** **static** JLabel *PageNumber*;

**public** **static** JButton *PageBack*;

These are all important variables needed in Home class. While Jpanel Home, Phone, and ComparePage are three graphic interface users to show the home page, the detail of each smartphone, and the comparison between two chosen smartphones respectively. This links Home to Smartphone or Compare class using Phone or ComparePage Jpanel in turn.

Home also provides other variables such as Jbutton detail[], compare[], Comparison, PageChange, and btnSearch help clients access other user interfaces. For instance, “Detail” accesses to the page displaying detailed information, “>” comes to other next pages to see more other smartphones, “Search” finds the smartphone clients are looking for as soon as possible.

It is necessary to have Jlabel ContainImg, name, message, ComparingStatus, PageNumber in order to have labels to contain images or names. There is a message that announces how many smartphones have been found, which is not the same at the right top corner of the interface, and a status notifying how many smartphones have been chosen (at least 1 and at most 2) at the left corner. At the two bottom corners, there are buttons to move forward or backward.

**public** Home() {

initialize();

*Message*("Loading");

*a* = **new** Manager();

*Display*();

Into the Display function

**for** ( **int** i = 0; i < 6; i++) {

*name*[i].setText("");

*ContainImg*[i].setIcon(**null**);

*detail*[i].setVisible(**false**);

*compare*[i].setVisible(**false**);

}

This sets every components becoming unvisible.

**int** size;

**if** (Manager.*product\_num* > 6)

size = 6;

**else**

size = Manager.*product\_num*;

The number of smarphones found is unknown and the home page should be neat or easy-to-follow, so at most there are only 6 smartphones displayed at a page, the others are displayed 3 at top and 3 at bottom in other pages, which is linked by “>” or “<” buttons.

**for** (**int** i = 0; i < size; i++) {

*ID* = i + 6\*(*page*-1)+1;

*name*[i].setText(Manager.*printProduct*(*ID*).Name);

*detail*[i].setVisible(**true**);

“ID” variable distinguishes smartphones themselves, and name[] variable can take the name of each product by using prinProduct function in Manager. After that, the detail[] is set visible to display “Detail” button.

**for**( ActionListener al : *detail*[i].getActionListeners() ) {

*detail*[i].removeActionListener( al );

}

This deletes all information displayed in the previous smartphones you have chosen. If not, if we have chosen the detail of the smartphone at the top left hand-side, then we go to the next page, we choose the phone at the same location, then the detail of the previous phone will be under that of the new one.

*detail*[i].addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

window = **new** Smartphone(id);

*Home*.setVisible(**false**);

**int** i = window.frame.getContentPane().getComponentCount();

i--;

**for** (; i > -1; i--) {

*Phone*.add(window.frame.getContentPane().getComponent(i));

}

*detail*[i].addActionListener(**new** ActionListener() allows clients to press “Detail” button to make some actions. Then,

static value window turns on Smartphone page, while *Home*.setVisible(**false**) hides all values of Home page. This method helps developers use the same Jframe for many other pages, just need to change Jpanel.

JButton btnExit = **new** JButton("Exit");

btnExit.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

*Phone*.removeAll();

*Phone*.setVisible(**false**);

*Home*.setVisible(**true**);

window = **null**;

}

});

btnExit.setBounds(290, 625, 89, 23);

*Phone*.add(btnExit);

*Phone*.setVisible(**true**);

We also creates “Exit” button to come back to the Home page when remove all information and set invisible for the Smartphone, and set visible for Home class. btnExit.setBounds(290, 625, 89, 23) to place “Exit” button.

*compare*[i].setVisible(**true**);

*compare*[i].addActionListener(**new** ActionListener() {

**int** id = *ID*;

**public** **void** actionPerformed(ActionEvent e) {

Compare.*CompareAdd*(Manager.*printProduct*(id));

*count*++;

**if** (*count* == 1) {

*ComparingStatus*.setText("There is one product");

*Comparison*.setEnabled(**false**);

}

**if** (*count* == 2) {

*ComparingStatus*.setText("There is two products");

*Comparison*.setEnabled(**true**);

}

**if** (*count* > 3) {

*ComparingStatus*.setText("There is two products already. Cannot add more");

*Comparison*.setVisible(**true**);

}

}

});

This is the same for Compare function. The “Add to compare” button is placed next to “Detail” one. There is an integer *Count* variable to count the number of products which has been chosen. If 1, there is a message at the left top corner and ‘Compare’ button still does not work, while if we have 2 options, ‘Compare’ is already enabled. If there are more than 3, then a message “Cannot add more” is executed, and when we press ‘Compare’, only two first products are compared.

*Comparison*.addActionListener(**new** ActionListener() {

**int** id=*ID*;

**public** **void** actionPerformed(ActionEvent e) {

*window1* = **new** Compare();

Compare.*Design*();

*Home*.setVisible(**false**);

**int** i = *window1*.*frame*.getContentPane().getComponentCount();

i--;

**for** (; i > -1; i--) {

*ComparePage*.add(*window1*.*frame*.getContentPane().getComponent(i));

}

JButton btnExit = **new** JButton("Exit");

btnExit.addActionListener(**new** ActionListener() **public** **void** actionPerformed(ActionEvent e) {

*ComparePage*.removeAll();

*ComparePage*.setVisible(**false**);

*Home*.setVisible(**true**);

*window1* = **null**;

*count*=0;

*ComparingStatus*.setText("There is no product chosen");

*Comparison*.setVisible(**true**);

}

});

The same process has been executed while we compare. We need to call Design function from Compare class, then hide Home page and display every component in Compare class. Following that, we also design ‘Exit’ when removing all components and hiding data of Compare class. Then, we turn on the Home page, and set the *count* variable to be 0, meaning non product has been chosen yet.

**if** (i < 3) {

*ContainImg*[i].setBounds(90 + 460 \* i, 120, 200, 200);

*name*[i].setBounds(100 + 460 \* i, 330, 200, 50);

*detail*[i].setBounds(100 + 460 \* i, 365, 70, 23);

*compare*[i].setBounds(180 + 460 \* i, 365, 125, 23);

} **else** {

*ContainImg*[i].setBounds(90 + 460 \* (i % 3), 400, 200, 200);

*name*[i].setBounds(100 + 460 \* (i % 3), 610, 200, 50);

*detail*[i].setBounds(80 + 480 \* (i % 3), 650, 70, 23);

*compare*[i].setBounds(160 + 480 \* (i % 3), 650, 125, 23);

}

This sets the place to put images, names, ‘Detail’ or ‘Add to compare’ buttons dynamically. With i%3, we get remainders, for example, 4%3, we get the value of 1. Therefore, the place of the bottom line will be exactly the same compared to the top line.

URL url = **new** URL(Manager.*printProduct*(*ID*).Image);

img = ImageIO.*read*(url);

Continuing, we design an URL to get the image of a specific product by printProduct function of Manager, and a Image img variable to contain this url.

*ContainImg*[i].setIcon(**new** ImageIcon(img));

*Home*.add(*ContainImg*[i]);

*Home*.add(*name*[i]);

*Home*.add(*detail*[i]);

*Home*.add(*compare*[i]);

Another important point is that, we need to add these variables to Home page if we want to show it on the screen.

JButton btnSearch = **new** JButton("SEARCH");

btnSearch.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

Manager.*Search*(*textField*.getText(), 1, 6);

*Display*()’

}});

JButton PageChange = **new** JButton(">");

PageChange.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

*page*++;

*Display*();

}

});

JButton PageBack = **new** JButton("<");

PageBack.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

*page*--;

**if**(*page*==1) {

PageBack.setEnabled(**false**);

}

*Display*();

}

});

We have 3 buttons for searching smartphones needed, coming forward to the next page, or backward to the previous page. When we are at the first page of the class, then the “<” will be set unenabled.

**for** (**int** i = 0; i < 6; i++) {

*name*[i] = **new** JLabel("");

*ContainImg*[i] = **new** JLabel("");

*detail*[i] = **new** JButton("Detail");

*compare*[i] = **new** JButton("Add to Compare");

}

This create new JLabel or Jbutton when needed, when we create many products ‘ name or images dynmically.

1. **Smartphone**

**Field Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Field and Description** |
| static JLabel | price[]  Shows three kinds of price from 3 retailers/ |
| static JLabel | display[]  Where to put 3 costs is set up. |
| static JLabel | hardware[]  It displays all technical specifications of each smartphone. |

**Constructor Summary**

|  |
| --- |
| **Constructor and Description** |
| public Smartphone(int id)  Constructs a Smartphone from providing products’s ID. |

**Method Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| private void | initialize()  Contains all detail of a chosen smartphone and display data into user interface. |

**static** JLabel *price*[] = **new** JLabel[3];

**static** JLabel *display*[] = **new** JLabel[3];

**static** JLabel *hardware*[] = **new** JLabel[9];

These variables are to display prices from 3 online suppliers and their technical specifications respectively.

*price*[0]=**new** JLabel("Lazada Price: ");

*price*[1]=**new** JLabel("TGDD Price: ");

*price*[2]=**new** JLabel("FPT Price: ");

**for**(**int** i=0;i<3;i++) {

*price*[i].setBounds(500,150+150\*i, 100, 50);

frame.getContentPane().add(*price*[i]);

}

**for**(**int** i=0;i<3;i++) {

*display*[i]=**new** JLabel(Integer.*valueOf*((Manager.*printProduct*(*ID*).Price[i])) +" VND");

*display*[i].setBounds(650,150+150\*i,200,50);

*display*[i].setFont(**new** Font("Footlight MT Light",Font.***BOLD***,20));

*display*[i].setForeground(Color.***RED***);

frame.getContentPane().add(*display*[i]);

}

The price[] array variables demonstrate places of three kinds of price from Lazada, TGDD, and FPT, meanwhile display[] array variables illustrate their prices. The setForeground function highlights the important data of each Smartphone in Red. After that, we also need getContentPane().add() function to add all what we have done into the Smartphone class.

String[] hard = **new** String[9];

**for**(**int** j=0;j<9;j++) {

hard[j]=Manager.*printProduct*(*ID*).Spec[j];

}

These lines create a string array *hard*  to comprise all technical specifications gotten from prinProduct function of Manager class, while *ID* plays an important role in specifying each product.

*hardware*[0]=**new** JLabel("<html> Màn Hình: "+" <font color='red'>"+hard[0]+"</font></html>");

*hardware*[1]=**new** JLabel("<html> Màu: "+" <font color='red'>"+hard[1]+"</font></html>");

*hardware*[2]=**new** JLabel("<html> OS: "+" <font color='red'>"+hard[2]+"</font></html>");

*hardware*[3]=**new** JLabel("<html> CPU: "+" <font color='red'>"+hard[3]+"</font></html>");

*hardware*[4]=**new** JLabel("<html> RAM: "+" <font color='red'>"+hard[4]+"</font></html>");

*hardware*[5]=**new** JLabel("<html> ROM: "+" <font color='red'>"+hard[5]+"</font></html>");

*hardware*[6]=**new** JLabel("<html> CAMERA: "+" <font color='red'>"+hard[6]+"</font></html>");

*hardware*[7]=**new** JLabel("<html> PIN: "+" <font color='red'>"+hard[7]+"</font></html>");

*hardware*[8]=**new** JLabel("<html> SIM: "+" <font color='red'>"+hard[8]+"</font></html>");

The hardware array variables places updated information into Jlabel separately, while, for example, ("<html> RAM: "+" <font color='red'>"+hard[4]+"</font></html>") change the basic color of each row into the expected one.

JLabel lblName= **new** JLabel(Manager.*printProduct*(*ID*).Name,SwingConstants.***CENTER***);

lblName.setForeground(Color.***BLACK***);

lblName.setFont(**new** Font("Times New Roman",Font.***BOLD***,20));

lblName.setBounds(70,50,500,300);

frame.getContentPane().add(lblName);

These codes display the name of each smartphone at the middle of chosen smartphone by SwingConstants.CENTER, change the color into *black*  by setForeground(Color.BLACK), set its size by 20 and place the branch at setBounds(70,50,500,300). Following that, we also need to add it into Jframe by getContentPane().add().

1. **Compare**

**Field Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Field and Description** |
| public static Product | a  The first smartphone is chosen for comparison. |
| public static Product | b  The second smartphone is chosen for comparison. |
| static int | count  Counts the number of products are chosen. |
| public static JLabel | ContainImg  Sets up label for containg images. |
| public static int | ID  It gets the correct smartphones. |
| static String | Url[], Urlb[]  They store the smartphones’ images gotten from the Internet. |
| public static JLabel | hardware[], hardware2[]  They display the specifications of chosen smartphones. |

**Constructor Summary**

|  |
| --- |
| **Constructor and Description** |
| public class Compare()  Creates new Compare funtion. |

**Method Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| public static void | CompareAdd(Product SpCanAdd)  Adds product to *Compare.* |
| public static void | Design()  Sets up places to put information of the two smartphones. |

The process for developing codes in *Compare* is similar with Smartphone one, but we make some differences

**static** JFrame frame;

**public** **static** Product a;

**public** **static** Product b;

**static** **int** count = 0;

**public** **static** JLabel ContainImg[] = **new** JLabel[2];

**public** **static** **int** ID;

**static** String Url[] = **new** String[3];

**static** String Urlb[] = **new** String[3];

**public** **static** JLabel hardware[] = **new** JLabel[9];

**public** **static** JLabel hardware2[] = **new** JLabel[9];

While **public** **static** Product a and **public** **static** Product b are used to set up data for the two product, **static** **int** count = 0 is used to count the number of products compared, and other variables are the same as those in *Smartphone* class. For example, Jlabel *hardware* or *hardware2* array still display all necessary specifications into *Compare* class.

**public** **static** **void** CompareAdd(Product SpCanAdd) {

**if** (count == 0) {

a = SpCanAdd;

count++;

} **else** **if** (count == 1) {

b = SpCanAdd;

count++;

} **else**

System.out.println("There are 2 products already");

}

This function chooses products based on SpCanAdd. If there is one, then *count* increase 1. Continuing, if the second one has been chosen, then no more products are allowed to compare.

Image img = **null**;

**try** {

URL url = **new** URL(Manager.printProduct(a.ID).Image);

img = ImageIO.read(url);

} **catch** (IOException e) {

e.printStackTrace();

}

These lines add image from specific product a.

Similar, the method to add image of product b:

Image imgb = **null**;

**try** {

URL urlb = **new** URL(Manager.printProduct(b.ID).Image);

imgb = ImageIO.read(urlb);

} **catch** (IOException e) {

e.printStackTrace();

}

String[] hard = **new** String[9];

**for** (**int** j = 0; j < 9; j++) {

hard[j] = Manager.printProduct(a.ID).Spec[j];

}

Then , we create a string *hard* array variable to contain technical specification of product a, the same for string *hard2* array variable.

**for**(**int** i=0;i<9;i++) {

hardware[i]=**new** JLabel("");

hardware2[i]=**new** JLabel("");

hardware2[i].setBounds(600 + 50 \* i, 700, 50, 50);

hardware[i].setBounds(400 + 50 \* i, 600, 50, 50);

frame.getContentPane().add(hardware[i]);

frame.getContentPane().add(hardware2[i]);

}

After that, we create new Jlabel for hardware and hardware2 when needed or images of 2 smartphones:

frame.getContentPane().add(lblImg1);

frame.getContentPane().add(lblImg2)

and display them into user interface.

1. **Product**

**Field Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Field and Description** |
| public static final int | LAZADA, TGDD, FPT, SCREEN, COLOR, OS, CPU, RAM, ROM, CAMERA, BATTERY, SIM  Define column number in array of CVS file. |
| public int | ID  Contains product identifier |
| public string | Name  Contains product name. |
| public String[] | Url  Contains product urls from three retailers. |
| public int[] | Price  Contains product prices from three retailers. |
| public String | Image  Contains product image url. |
| public Sring[] | Spec  Contains product’s specifications. |

**Constructor Summary**

|  |
| --- |
| **Constructor and Description** |
| public Product (int id)  Creates a new product with an unique ID. |

**Method Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| void | importFromCSV(int id)  Buffers data of one product from CSV file with corresponding ID. |

1. **Store**

**Field Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Field and Description** |
| public static String | **DATAFILE**  Webpage HTML download directory. |

**Constructor Summary**

|  |
| --- |
| **Constructor and Description** |
| public Store()  Creates a new Store. |

**Method Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| public void | getUrl(String urlstring)  Downloads webpage from url and saves to disk. |

1. **Lazada**

**Field Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Field and Description** |
| private static String | searchLink  Contains Lazada Search Smartphone only header link. |
| public static List<String>. | SearchProducts  Contains product infomation in String |
| private static String | criteria  Contains html search string for fiding product information section in html file. |
| private static double | SearchTime  Contains Lazada Search time. |

**Constructor Summary**

|  |
| --- |
| **Constructor and Description** |
| Lazada()  Creates new Lazada store. |

**Method Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| private static int | colormodel(String color)  Classify product color. |
| public static boolean | Analyze (String question)  Analyzes downloaded HTML file for getting list of products. Returns true if product found. |
| public static void | Classify()  Classify type, color, name, rom of product and save as CSV type in SearchProducts list. |
| public static void | Search(string question)  Control Analyze, Classify function and also count search time |

1. **FPT**

**Field Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Field and Description** |
| private static final Sting | searchLink  Contains FPTshop search header link. |
| public static final String | ListSearch, result  Contains html search string for fiding product information section in html file. |
| public static String | price, name, url  Contains product infomation in String |

**Constructor Summary**

|  |
| --- |
| **Constructor and Description** |
| FPT()  Creates new FPT store. |

**Method Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| public static boolean | Analyze (String question)  Analyzes downloaded HTML file for getting list of products. Returns true if product found. |
| public static void | Search(string question)  Control Analyze function. Return true if correspond product found. |

1. **TGDD**

**Field Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Field and Description** |
| private static final Sting | searchLink  Contains FPTshop search header link. |
| public static final String | noresult, result  Contains html search string for fiding product information section in html file. |
| public static String | price, name, url, img, para[]  Contains product infomation in String |

**Constructor Summary**

|  |
| --- |
| **Constructor and Description** |
| TGDD()  Creates new TGDD store. |

**Method Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| public static boolean | Analyze (String question)  Analyzes downloaded HTML file for getting list of products. Returns true if product found. |
| public static void | Search(string question)  Control Analyze function. Return true if correspond product found. |

1. **SearchManager**

**Decription**

Control and check corresponding product in search funtion.

**Constructor Summary**

|  |
| --- |
| **Constructor and Description** |
| SearchManager(string a)  Creates new search request |

**Method Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| private static void | reset()  Clears all information of previous searching. |
| public static void | Search(String question)  Searchs product prices and urls from Lazada and FPT, then all technical specifications of a smartphone from TGDD. |
| public static void | Search(String question, int start, end)  Searchs product prices and urls from Lazada and FPT, then all technical specifications of a smartphone from TGDD. Only get product have ID between start and end ID. |

1. **Tools**

**Decription**

Contain tools to read and write to file.

**Method Summary**

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| public static void | FileWriter(String text, String file)  Inputs data into file. |
| static boolean | checkRelevantProduct(String name, String question)  Checks relevant products. Returns true if product corresponding with user need. |

## 4.2. Database

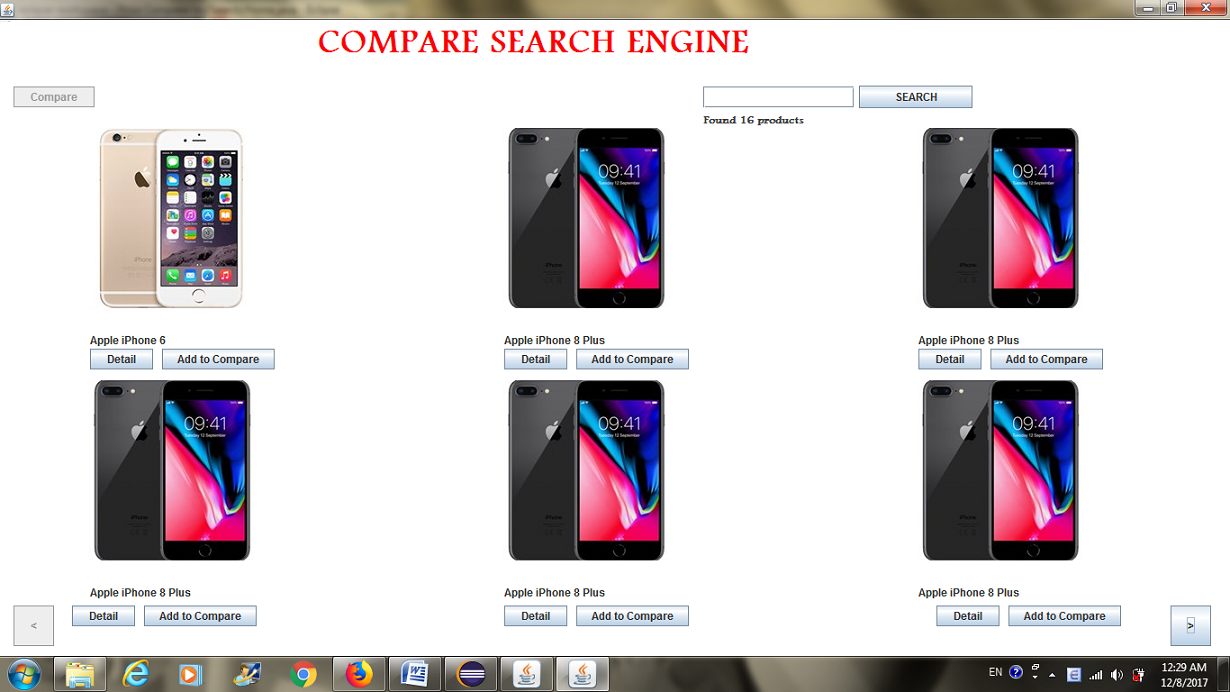
**Description:** Contains product infomation

**Type:** CSV

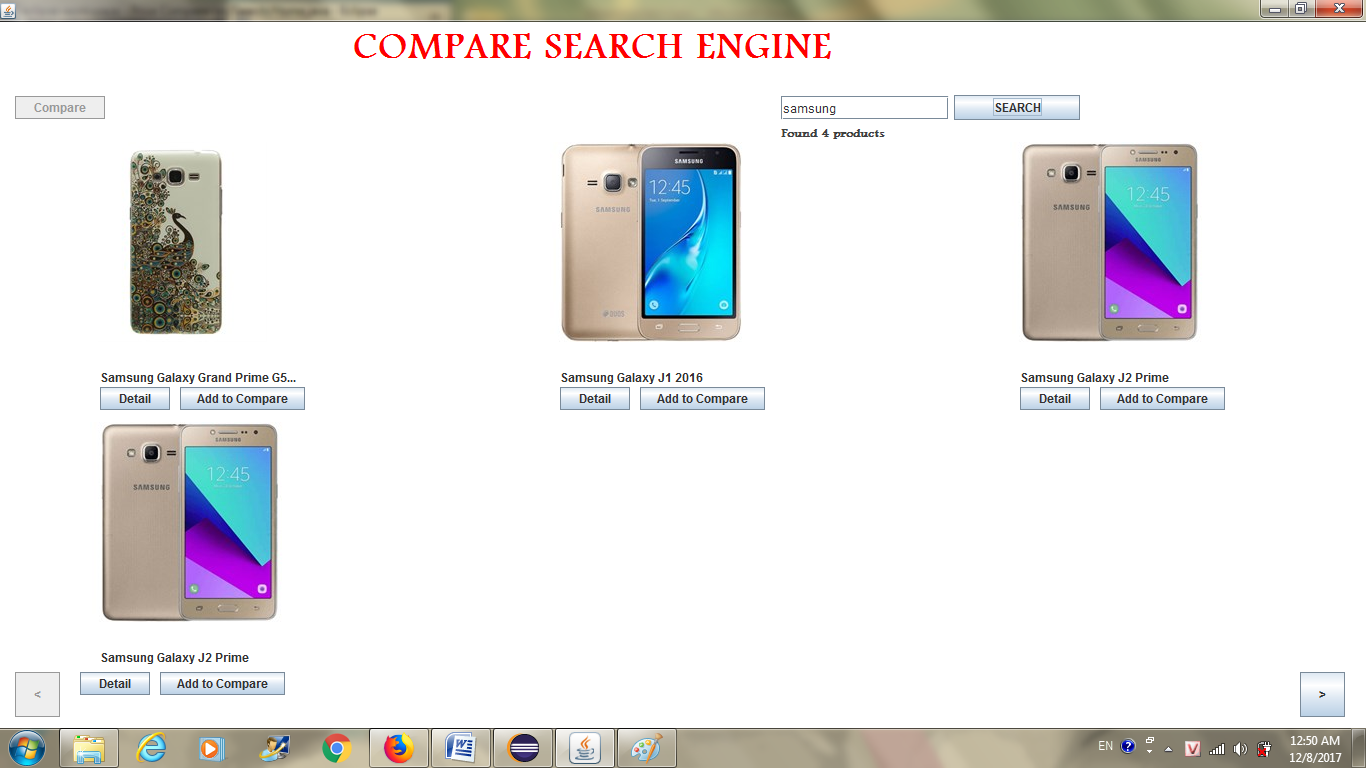
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column** | **Data Type** | **Primary Key** | **Nullable** | **Description** |
| ID | INT | 1 | - | Identifier number |
| Name | VARCHAR | - | - | Name |
| Màn hình | VARCHAR | - | ✓ | Screen size |
| Màu | VARCHAR | - | ✓ | Color |
| OS | VARCHAR | - | ✓ | Operating System |
| CPU | VARCHAR | - | ✓ | CPU type |
| RAM | VARCHAR | - | ✓ | RAM capacity |
| ROM | VARCHAR | - | ✓ | ROM capacity |
| Camera | VARCHAR | - | ✓ | Camera specification |
| PIN | VARCHAR | - | ✓ | Battery capacity |
| SIM | VARCHAR | - | ✓ | Number of SIM |
| Loại | VARCHAR | - | ✓ | Product Type |
| Lazada Url | VARCHAR | - | - | Lazada product link |
| TGDD Url | VARCHAR | - | - | Thegioididong product link |
| FPT Url | VARCHAR | - | - | FPT product link |
| Lazada Price | VARCHAR | - | - | Lazada product price |
| TGDD Price | VARCHAR | - | - | Thegioididong product price |
| FPT Price | VARCHAR | - | - | FPT product price |
| ImgUrl | VARCHAR | - | - | Thegioididong image link |

# 5. Interface user guide

At first, we have this Home page demonstrated:

There is a “Compare Search Engine” at the top of the Home page, an unenabled *Compare* button at the left hand-side, while the *Search* one is at the opposite side. Below it, there is a status displaying the number of digital products which have been gotten data from all three largest suppliers. In this example, there are 16 products found. Furthermore, six smartphones are displayed with three at the top and three at the bottom. Below each smartphone, there are 2 buttons, involving *Detail* and *Add to Compare.* At the end of the left of the page, a backward button is to come back to the former page, while at the opposite side, the forward button is to come to the next page.

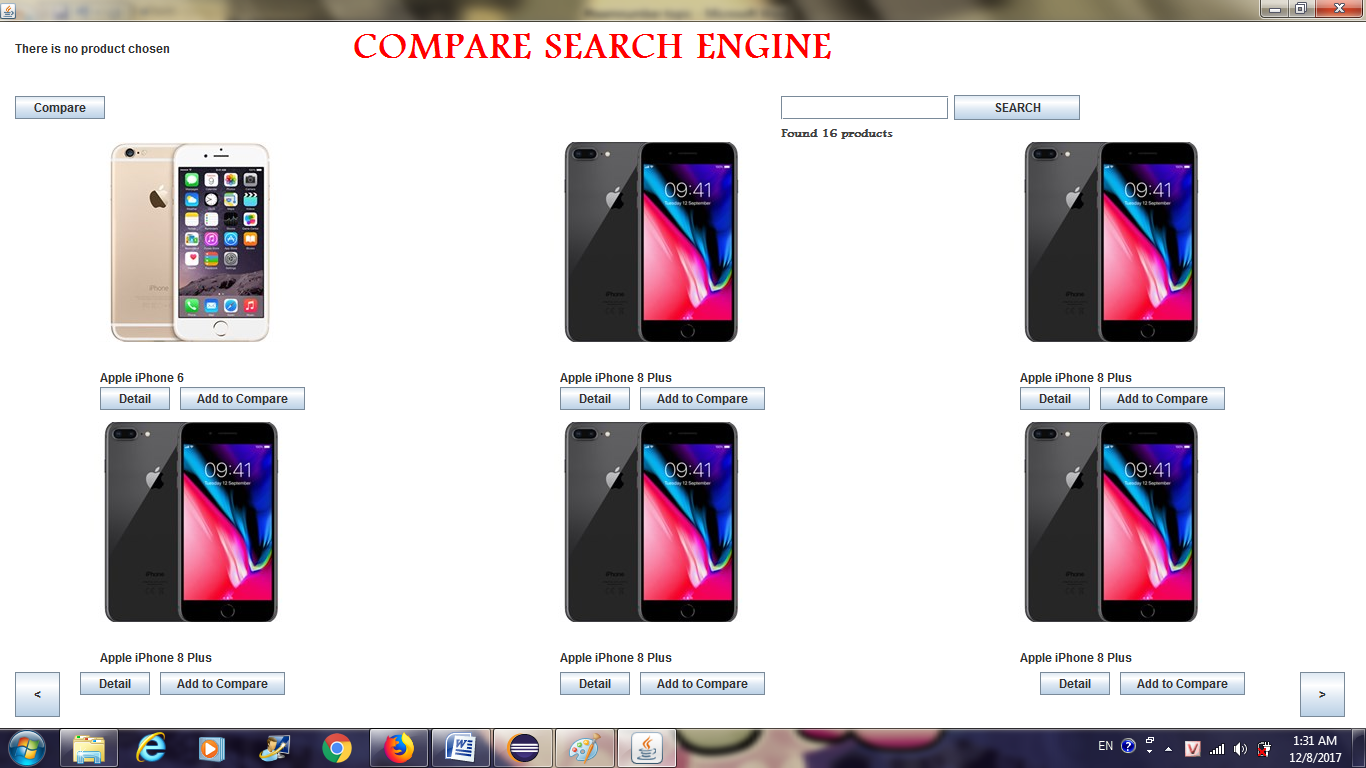
If we want to find some smartphones, just type the name of the product expected, then click to the *Search* and then the system will display all suitable results. For example, if we type ‘Samsung’:



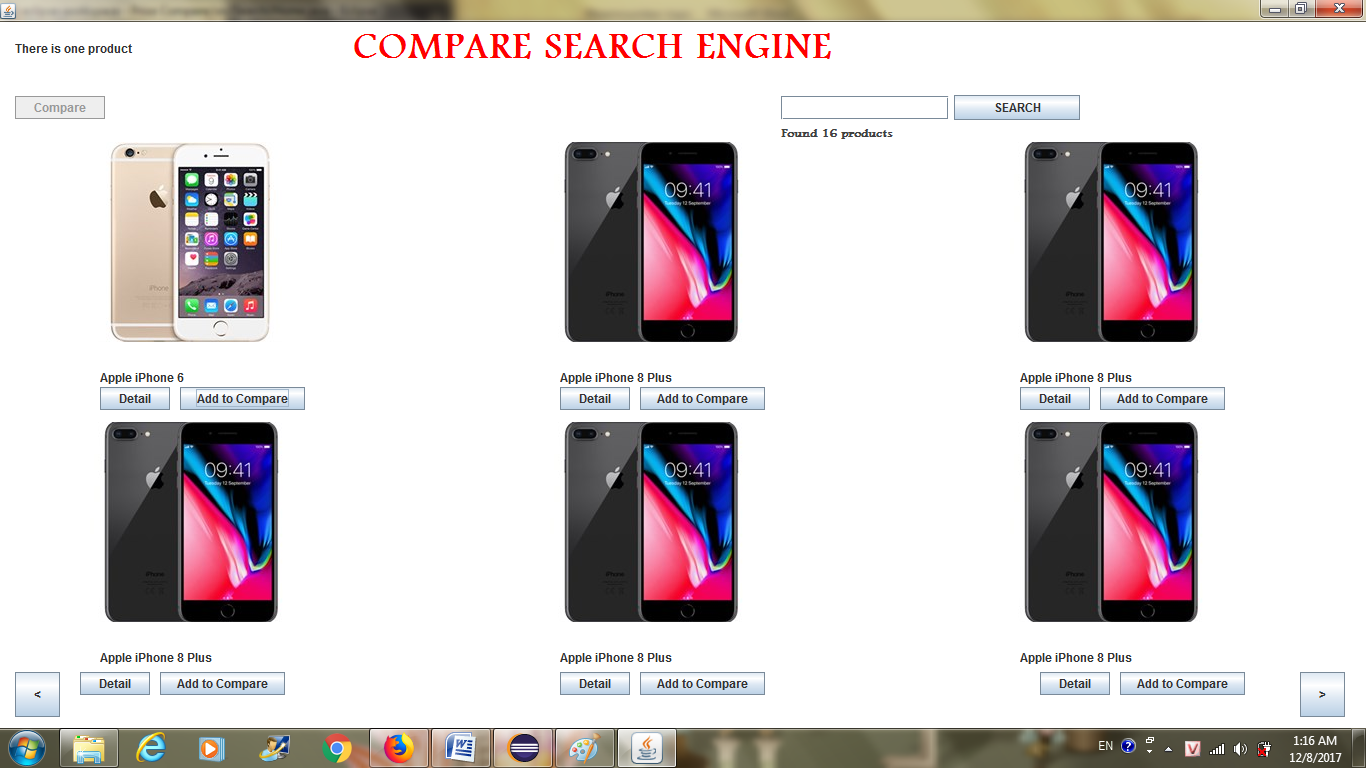
Then, the system collects data from 3 retailers, based on the Internet, the Home page shows all possible smartphones; in this example, there are 4 products found. After that, based on what we have searched, we click *Detail* to get the technical specification of that product. Now, we click to *Samsung Galaxy J2 Prime:*

In addition, the system displays the name of the phone above the image, while at the middle of the page, there will be clear 3 different prices from 3 online stores, at the right side of the page, there are technical specifications such as screen resolution, color, operating system, CPU, RAM, ROM, Camera, Battery, and Sim. In this Samsung Galaxy J2 Prime, the operating system is Android 6.0 or the Battery is about 2000 mAh.

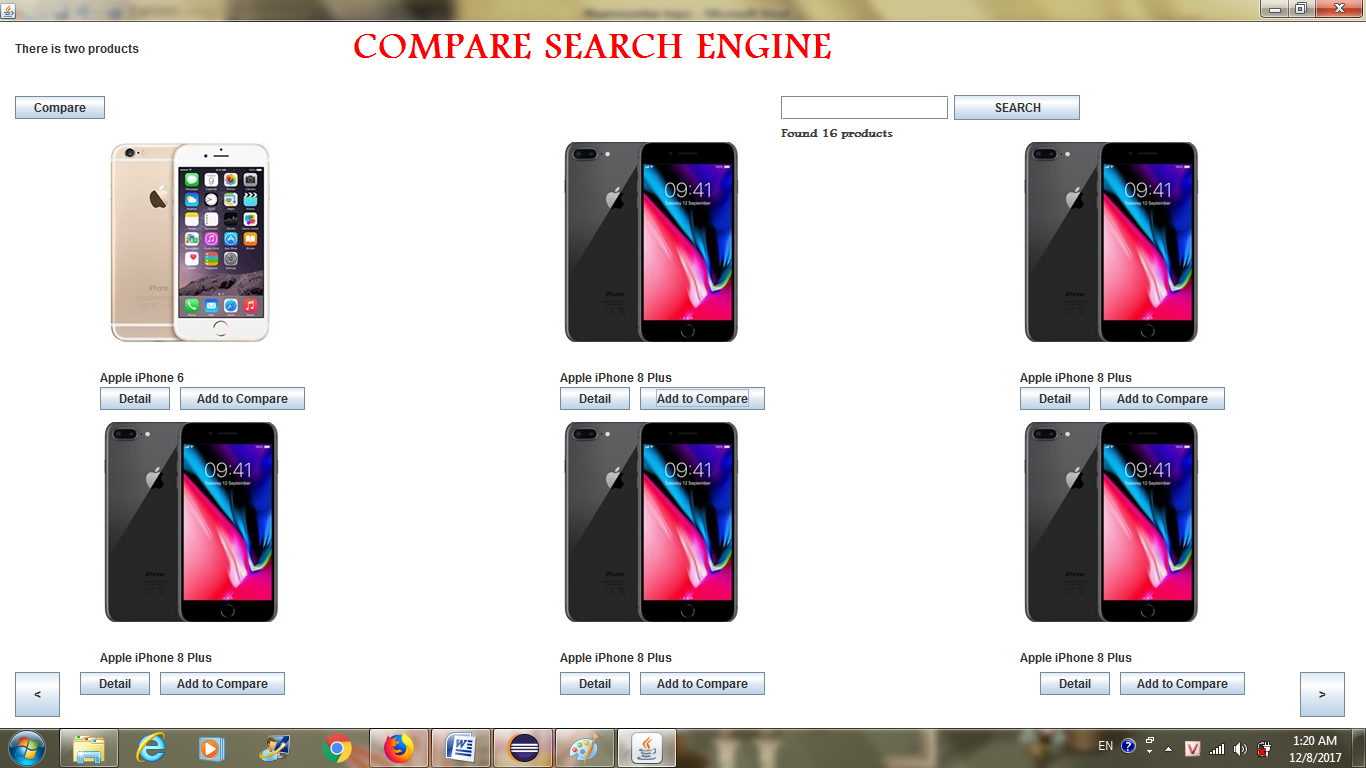
The next function is that if we want to compare 2 particular products, then we click to the *Add to Compare* of each smartphone below, and if no products have been chosen:



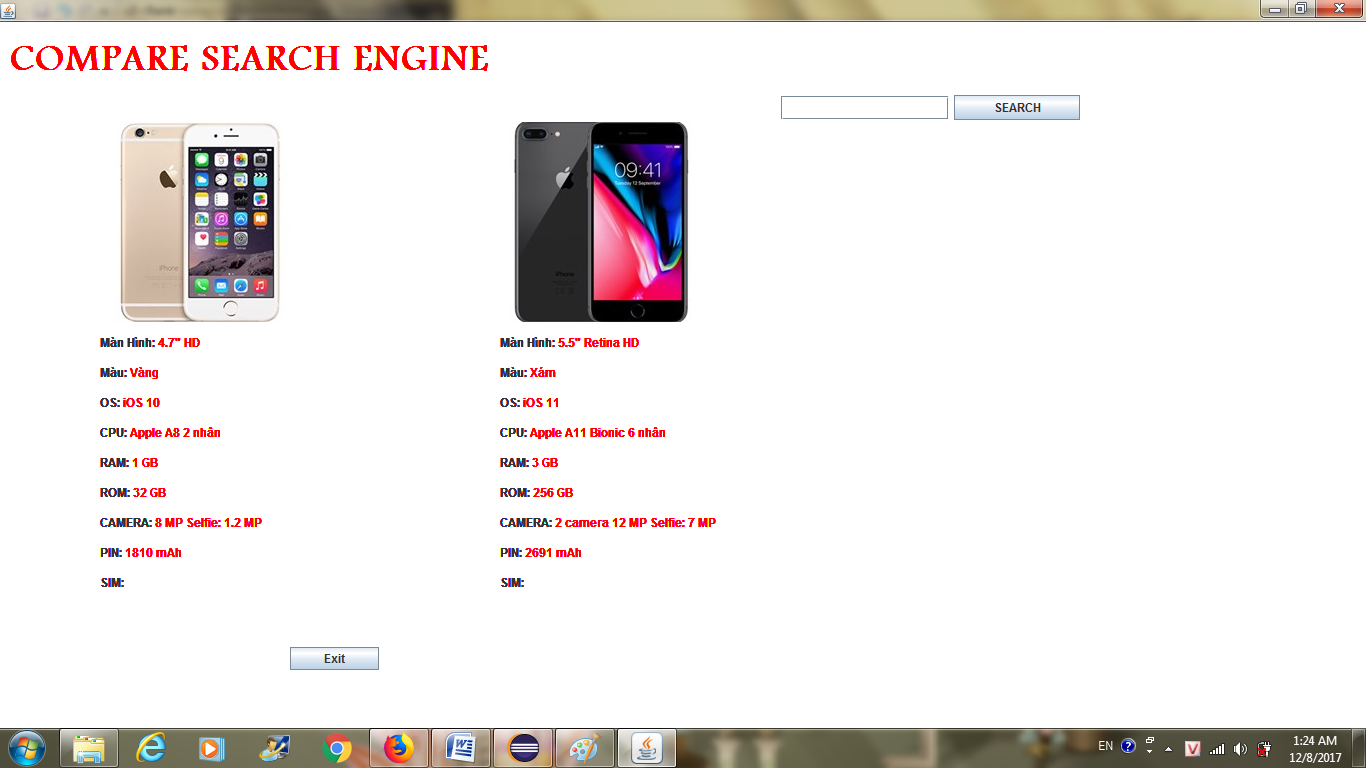
After being chosen the first smartphone, the system illustrates a status at the left top of the page: “There is one product” and the *Compare* still does not enabled.



Then, we have the second option:



After the second choice is set up, the status represents “There are 2 products” and the *Compare* is enabled; thus, we click to the *Compare:*

**

Therefore, the technical specifications of 2 smartphones are shown. Clients can have a better view to which smartphone is more suitable. We still have a *Exit*  to come back to the Home page. However, if the smartphone choices is over 2, then the status displays “There are 2 products already. Cannot add more”

# c3.png

# Since then, the system only compares two first smartphones which have been chosen.

# 6. Future development

1. **Product Diversity**

As the start of our project, we just basically focus on smartphone comparison, particularly Apple and Samsung branches. In future, we want to extend the number of product, not only in smartphone field, but also on other fields such as washing machines, air-conditioners, and so on.

1. **Graphic User Interface**

Our Java project is still a little simple, we want to have more effect to attract more users.

1. **More Compared Products**

Our research just allow users to choose at most 2 smartphones to compare; moreover, we want to upgrade options so that users can choose how many they would like to have comparison.

# 7. Acknowledge

1. If we want to understand more about JavaTM  Platform, we find in <https://docs.oracle.com/javase/7/docs/api/overview-summary.html>
2. If Java objects and classes are easily misunderstanding, <https://www.tutorialspoint.com/java/java_object_classes.htm> is helped.
3. Java Objects and Classes can be understood easily in <https://www.javatpoint.com/object-and-class-in-java>.

---------------