



**T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY of ENGINEERING
DEPARTMENT of COMPUTER ENGINEERING**

E-SHOPPING APPLICATION

CSE 222 DATA STRUCTURES AND ALGORITHMS HOMEWORK 6 REPORT

STUDENT

**Şeyda Nur DEMİR
12 10 44 042**

LECTURER

Prof. Dr. Fatih Erdoğan SEVİLGİN

TEACHING ASSISTANT

**Başak KARAKAŞ
Mehmet Burak KOCA**

KOCAELİ, 2021

1.DESCRPTION

This homework has 1 part.

Design and implement an e-shopping application that has two user roles as usual: traders and customers. Users authenticate to the system by using their user IDs and passwords. The user ID is an eight-digit unique number and the password consists of six characters. After the authentication process is done properly, the program shows a menu to the users with respect to their user role. The application meets the following requirements by these menus:

- Traders add, remove, and edit products on their shop. Each product has a unique ID, a hierarchical category, a name, a description, a price, and a discount amount.
- Traders see the list of orders from customers and can meet or cancel the orders.
- Customers can search and query the products by their names. The products that contain the search text in their name or description should be returned in the results.
- The search results are shown in decreasing order of the product name. Customers can sort the products in the search results by their prices, the percentage of the discount, or the name.
- The search results can be filtered by their category or the prices with respect to given upper or lower (or both) thresholds. The customer can filter the products not only in the lowest level category but also in any higher-level category.
- Customers can display all the products of a trader.

Use the following five data structures properly (do NOT use simple array structure) to implement functionalities of the application efficiently:

- ArrayList
- LinkedList
- Queue
- Tree
- Hash Table

You need to use each of them at least once, but you feel free to use more than once when you needed. Indicate which data structure is used to implement which part of the application and why you do it so in the report file. Use different sorting algorithms (your implementation) while sorting the search results by the price, the discount percentage, or the name.

There is a CSV file on the Moodle page which is named as “e-commerce-samples.csv”. Use the given csv file to create your data files. Keep the users, the orders, and the products data in your own text files. When a customer performs a search, you need to access to the products file, and return the suitable products as the result of the search. Don’t load all the products into the memory. You need to keep products in a well-organized data file for products to answer queries efficiently.

Restrictions

- Can be only one main class in project
- Don't use any other third part library

General Rules

- For any question firstly use course news forum in Moodle, and then the contact TA.
- You can submit assignment one day late and will be evaluated over sixty percent (%60).

Technical Rules

- You must write a driver function that demonstrates all possible actions in your homework. For example, if you are asked to implement an array list and perform an iterative search on the list then. The driver function should run when the code file is executed.
- Implement clean code standards in your code ;
 - Classes, methods and variables names must be meaningful and related with the functionality.
 - Your functions and classes must be simple, general, reusable and focus on one topic.
 - Use standard java code name conventions.

Report Rules

- Add all javadoc documentations for classes, methods, variables ...etc. All explanation must be meaningful and understandable.
- You should submit your homework code, Javadoc and report to Moodle in a "studentid_hw5.tar.gz" file.
- Use the given homework format including selected parts from the table below:
 - Problem solutions approach
 - Test cases
 - Running command and results

Grading

- No OOP design: -100
- No interface: -95
- No method overriding: -95
- No error handling: -50
- No inheritance: -95
- No polymorphism: -95
- No javadoc documentation: -50
- No report: -90
- Disobey restrictions: -100
- Cheating: -200
- Your solution is evaluated over 100 as your performance.

2.REPORT

I detailed here what I did in my homework.

2.1.Problem Solutions Approach

Note : I googled the way you said in the report to problem solving approach, but I could not any useful article, what I found was either paid or long. After all I found a useful post from medium. I prepared this part according to this post. I hope I got it right.

- **Clearly understanding and/or defining the problem :**

I understood and defined the problem clearly.

- We should implement e-shopping application using defined 5 data structures.
- First, I decided my structures where I use them.
- Then, I decided how I read the file and write the files.
- Last, I decided search/filter/sort algorithms.

- **Breaking down large problems into smaller problems :**

I broke down large problem e-shopping application, to small problems company-account-trader-customer-product-order classes, search-filter-sort algorithms and a Driver, a Test and a Main class.

- We have a large problem, e-shopping application.
- I have now small problems, read file, write file, keep data, search-filter-sort data, and trader-customer-product-order-company-account modules

- **Solving the problem at an abstract level first :**

I solved the problem at an abstract level first.

- I thought a lot about the problem.
- I scribbled something about this subject in the ledger.
- Something started to take shape in my head.
- I used my knowledge of data structures, and it is.

- **Using notes and pseudo-code :**

I noted what I thought and wrote permanently pseudo-codes mixed with java codes.

- If I understand or find something new, I noted.
- I wrote pseudo-codes mixed with java codes, not clear.

- **Running code early and often :**

I wrote real codes and run them often.

- I turned my pseudo-codes into real java codes.
- I coded them in ide and run them often.

Used Data Structures

ArrayList : used as *Java Util Generic ArrayList* for **TradersName**

```
private ArrayList<String> tradenames = new ArrayList<>();
```

LinkedList : used as *Java Util Generic LinkedList* for **Products**

```
private LinkedList<String> products = new LinkedList<>();
```

Queue : used as *Java Util Generic ArrayDeque* for **Sessions, Orders**

```
public ArrayDeque<String> sessions = new ArrayDeque<>();
```

```
private ArrayDeque<String> orders = new ArrayDeque<>();
```

Tree : used as *KW Generic Binary Search Tree* for **Categories**

```
private BinarySearchTree<String> category = new BinarySearchTree<>();
```

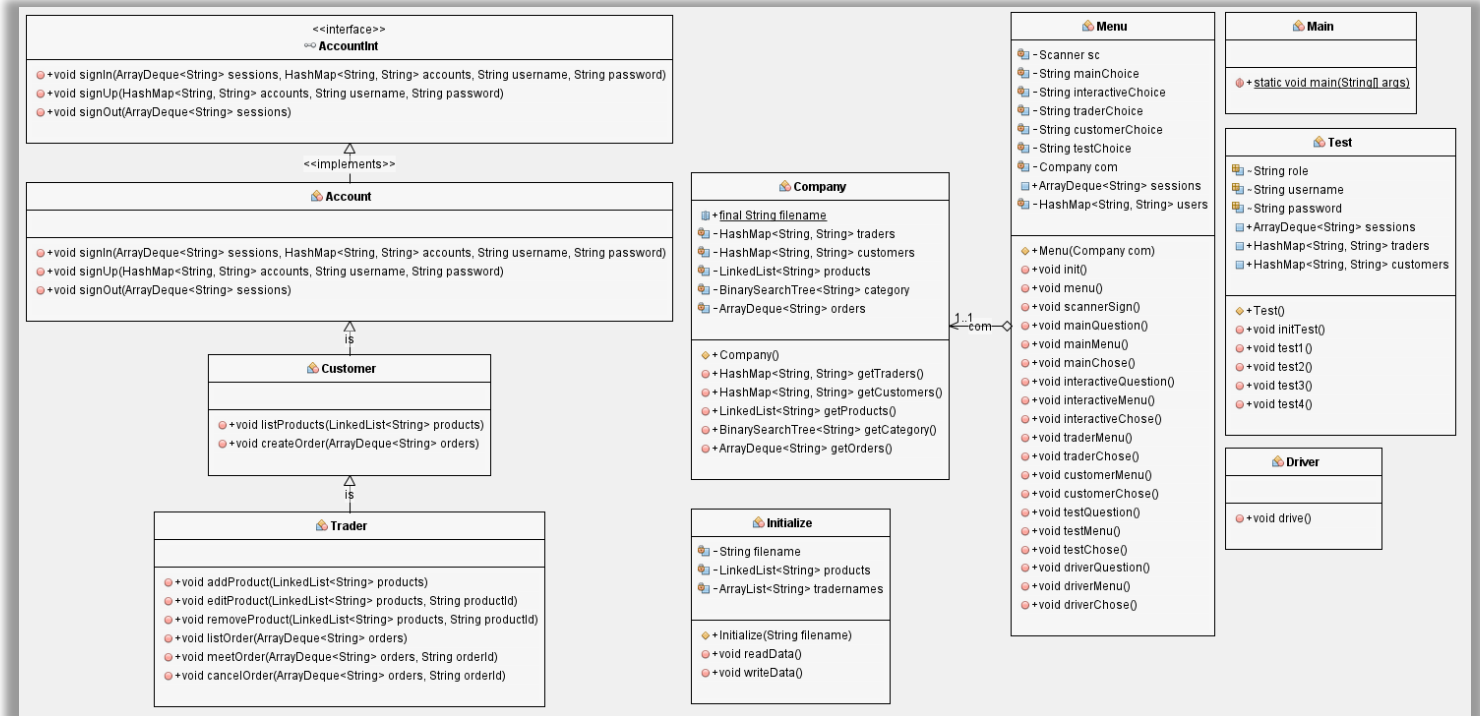
HashTable : used as *Java Util Generic HashMap* for **Users, Customers, Traders**

```
private HashMap<String, String> users = new HashMap<>();
```

```
private HashMap<String, String> customers = new HashMap<>();
```

```
private HashMap<String, String> traders = new HashMap<>();
```

2.2.Uml Diagram



You can see uml diagram in png format in “Report” directory.

2.3.Test Cases

Testing Requirements

Test Case	Pres / Posts	Done
Read File Write Files	<ul style="list-style-type: none">• Program initializes the system• Reads "e-commerce-samples.csv" file• Splits using ";" and "\n" as delimiters• Keeps each splitted product info• Keeps each trader info• Writes all products to "e-commerce-products.csv" file• Writes all traders to "e-commerce-traders.csv" file	Done
Sign Up Sign In Sign Out	<ul style="list-style-type: none">• User can sign up• Username must be unique• User can sign in• User must be signed up to sign in• User signs out when exit current menu• User must be signed in to sign out	Done
List Products by searching by filtering by sorting	<ul style="list-style-type: none">• Customer can see list product option on customer menu• Customer selects this option• Customer enters search-filter-sort options• Customer can enter "none" to pass any option• Then program lists products according to entered options	Undone
Order Product	<ul style="list-style-type: none">• Customer can buy any product by entering product id• File must contain the product with entered product id	Undone
Add Product Edit Product Remove Product	<ul style="list-style-type: none">• Trader can see product options on trader menu• Trader can select add-edit-remove options• Trader can add new product by entering product info• File must does not contain any product with entered product id• Trader can edit any product by entering product id• File must contains the product with entered product id• Trader can remove any product by entering product id• File must contains the product with entered product id	Undone

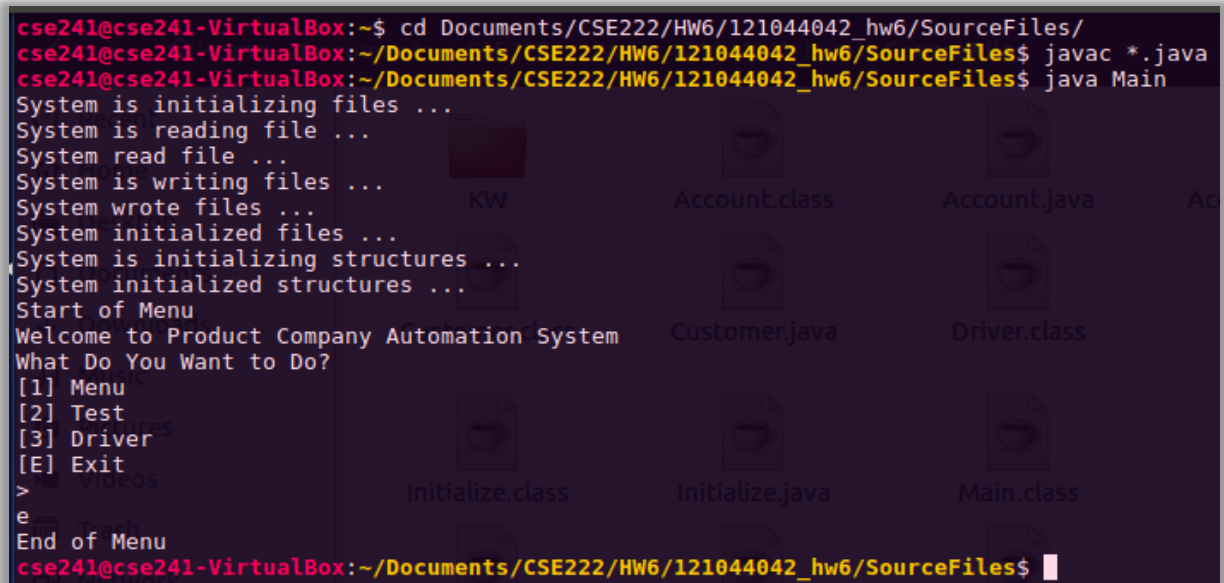
2.4. Running Commands and Results

Compile and Run Commands, Testing Steps

Usage of my program :

1. Compile program with “**javac *.java**” command
2. Run program with “**java Main**” command

Simply Follow This Screenshot



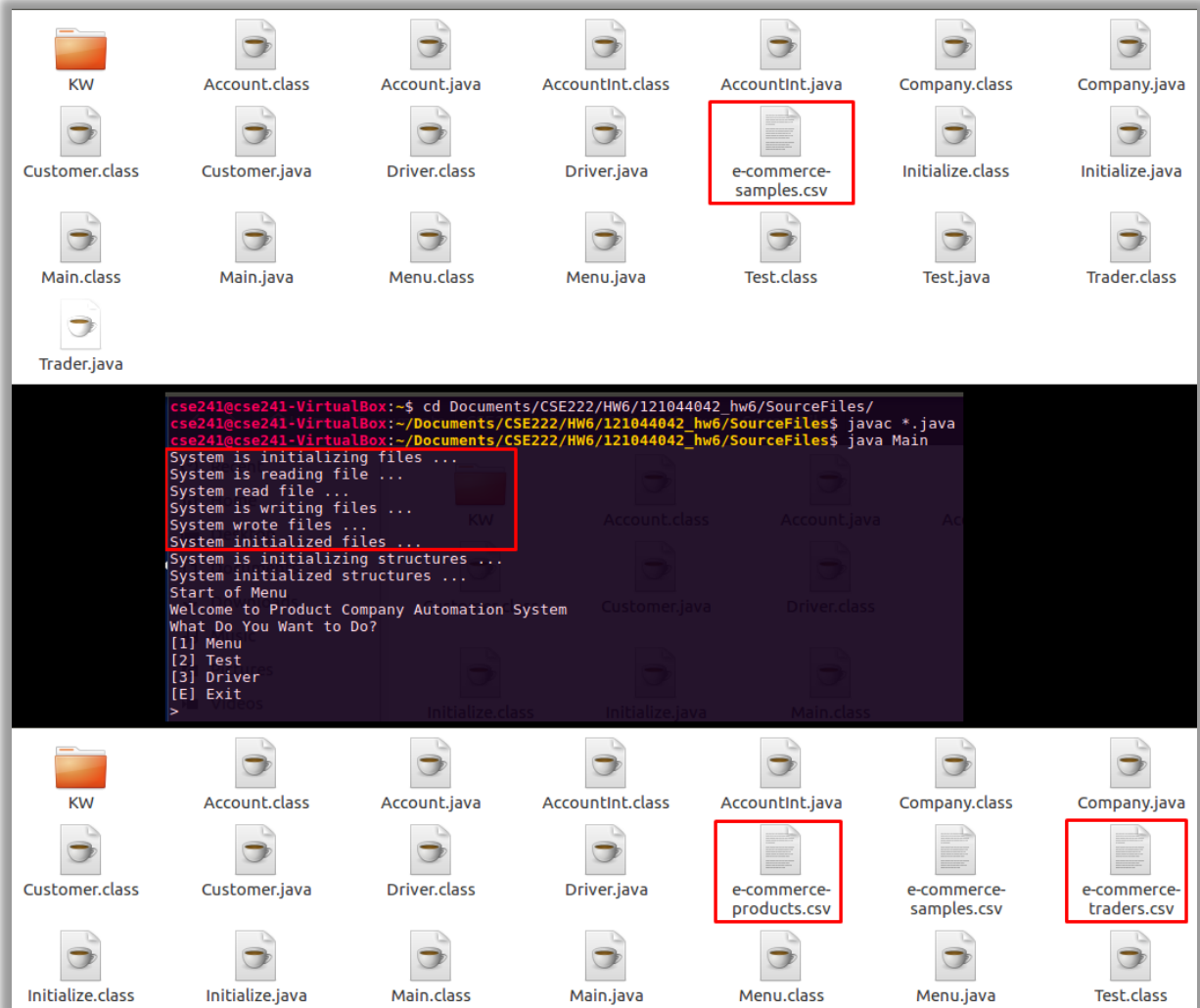
```
cse241@cse241-VirtualBox:~$ cd Documents/CSE222/HW6/121044042_hw6/SourceFiles/
cse241@cse241-VirtualBox:~/Documents/CSE222/HW6/121044042_hw6/SourceFiles$ javac *.java
cse241@cse241-VirtualBox:~/Documents/CSE222/HW6/121044042_hw6/SourceFiles$ java Main
System is initializing files ...
System is reading file ...
System read file ...
System is writing files ...
System wrote files ...
System initialized files ...
System is initializing structures ...
System initialized structures ...
Start of Menu
Welcome to Product Company Automation System
What Do You Want to Do?
[1] Menu
[2] Test
[3] Driver
[E] Exit
>
End of Menu
cse241@cse241-VirtualBox:~/Documents/CSE222/HW6/121044042_hw6/SourceFiles$
```

Commands and Results with Screenshots

Read File and Write Files Test

1. Reads data from “e-commerce-samples.csv” file
2. Writes data to “e-commerce-traders.csv” and “e-commerce-products.csv” files

Test Results are Here



Sign Up, Sign In and Sign Out Users Test

1. User can sign up
2. User can sign in
3. User can sign out

Test Results are Here

```
Make a Choise
[1] Sign In
[2] Sign Up
[E] Exit
> 1
2
Role : Videos
> 2
> 1
Customer
Username :
> 2
SeydaNur
Password :
> 1
xyz Connect to Server
User signed up.
```

```
Make a Choise
[1] Sign In
[2] Sign Up
[E] Exit
> 1
1
Role :
> 1
Customer
Username :
> 2
SeydaNur
Password :
> 1
xyz
User signed in.
```

```
Make a Choise
[1] List Products (Search, Filter, Sort)
[E] Exit
> 1
> 1
e
User signed out.
Customer signed out ...
System is returning to interactive menu ...
Make a Choise
[1] Sign In
[2] Sign Up
[E] Exit
> 1
```

```
Make a Choise
[1] Sign In
[2] Sign Up
[E] Exit
> 1
2
Role :
> 1
Customer
Username :
> 2
SeydaNur
Password :
> 1
123
Username must be unique.
```

```
Make a Choise
[1] Sign In
[2] Sign Up
[E] Exit
> 1
1
Role :
> 1
Customer
Username :
> 2
SeydaNurx
Password :
> 1
xyz
Username could not be found.
```

```
Make a Choise
[1] Sign In
[2] Sign Up
[E] Exit
> 1
1
Role :
> 1
Customer
Username :
> 2
SeydaNur
Password :
> 1
xyzz
Password does not match with username.
```

List Products by Searchin, Filtering, Sorting Test

1. Customer can list products by searching
2. Customer can list products by filtering
3. Customer can list products by sorting

Test Results are Here

```
User signed in.
Make a Choise
[1] List Products (Search, Filter, Sort)
[E] Exit
> 1
130
List Products (Search, Filter, Sort)
(Note : You can enter "none" to choose default settings.)
Search : You can search products by entering text
dotted
Products searching by search text "dotted"
Filter : You can filter products by entering trader name
Twin
Products filtering by trader name "Twin"
Filter : You can filter products by entering category
Dress
Products filtering by category "Dress"
Filter : You can filter products by entering min price
100
Products filtering by min price "100"
Filter : You can filter products by entering max price
250
Products filtering by max price "250"
Sort : You can sort products by entering order
Ascending Price
Products sorting by order "Ascending Price"
Displayed All Products As You Entered
Make a Choise
```

Order Product Test

1. Customer can order products by entering product id

Test Results are Here

```
Make a Choise          break;
[1] List Products (Search, Filter, Sort)
[2] Create Order       customerObject.creat
[E] Exit               break;
> 37                   case 'e': {
238                   customerObject.sign0
Created order ...      System.out.println(")
```

Add, Edit, Remove Product Test

1. Trader can add new product by entering product details
2. Trader can edit any product details by entering product id
3. Trader can remove any product by entering product id

Test Results are Here

```
User signed in. break;
Make a Choise case "E": {
[1] List Products (Search, Filter, Sort)
[2] Create Order      System.out.println("
[3] Add Product      System.out.println("
[4] Edit Product
[5] Remove Product break;
[6] List Order default:
[7] Meet Order /* Left Blank */
[8] Cancel Order
[E] Exit
> >>
```

END OF THE REPORT

LAST UPDATE

May 28, 2021 Friday 05:40

STUDENT

**Şeyda Nur DEMİR
12 10 44 042**

LECTURER

Prof. Dr. Fatih Erdoğan SEVİLGİN

TEACHING ASSISTANT

**Başak KARAKAŞ
Mehmet Burak KOCA**

KOCAELİ, 2021