**GIT Department of Computer Engineering CSE 222/505 - Spring 2021 Homework 3 # Report**

**Yeşim Yalçın**

**200104004094**

**1-) System Requirements**

**A-) Functional Requirements**

**-Log In/Registration Requirements**

-The company’s automation system will have users who will use the system. Users can be classified as “Admin”, “Employee” or “Customer”. The system will offer the users different things to do according to their status.

-The system saves all the new users who registers into the system.

-Users can register by putting their name, surname, email and password information.

-Once registered, the user is saved as a customer.

-Branch Employees can be added to the system by Admins, they cannot register by themselves.

-Admins can be added to the system only by editing the AdminsInfo.txt file.

-Every user can be registered with their own unique email address. The system will not accept two users with the same email address.

-Every customer has their own unique customer number, this number will be created by the system when they register for the first time.

**-Menu Requirements**

-Admin Menu

* ADDBranch: Creates a new branch for the company in the system. \*REMOVEBranch: Removes a branch of the company in the system. \*ADDEmployee: Adds a new employee for the company in the system. \*REMOVEEmployee: Removes an employee of the company in the system. \*CHECK: Checks what products should be supplied.

\*QUIT: Logs off.

-Employee Menu

\*STOCK: Shows the stock of a branch/online store or all.

\*ARRANGESales: Arrange the things related to sales.

\*ADDProduct: Adds a product to a branch's stock or to the online store's stock.

\*REMOVEProduct: Removes a product from a branch's stock or from the online store's stock.

\*SUPPLYProducts: Supplies the products that are needed to be supplied.

\*MAKESale: Makes a sale through branch.

\*PASTOrders: Shows the past orders of a customer.

\*QUIT: Logs off.

-Customer Menu

\*SEARCH: Searchs for a product in the stores.

\*LIST: Shows all the products this company produces and sells.

\*SHOP: Starts shopping.

\*PASTOrders: Views past orders.

\*QUIT: Logs off.

**-Store Requirements**

-When the company’s automation system is run for the first time, it creates 4 branches and 1 online store. However more branches can be added or branches can be removed later on by the admins. These changes will be saved.

* If a branch is removed, all the other branch numbers will be changed accordingly. Example: If the Branch3 is deleted, old Branch4 will be the new Branch3. Previous orders from that branch will be changed as if they were done&will be done from the online store not to affect the customers. Also the Supply List will be reseted because of the changes in the branch numbers. Employees can create a new Supply List afterwards.

-There can be only one online store, if more will be tried to created, it will not have any affect.

-The online store cannot be removed.

-All branches and the online stock have their own stock.

-Branch employees can edit the stock of both branches and the online stock.

**-Product Requirements**

-Products are created by determining their type, model and the colour.

-Types can be Office\_Chair, Office\_Desk, Meeting\_Table, Bookcase of Office\_Cabinet.

-All types have different available colours and models.

-When the type is determined, the available colours and the models fort hat product will be set.

-Office\_Chair have 7 models and can be Black, White, Red, Blue, Green

-Office\_Desk have 5 models and can be Black, White, Red, Blue

-Meeting\_Table have 10 models and can be Black, White, Red, Blue

-Bookcase have 12 models and can be Black

-Office\_Cabinet have 12 models and can be Black

-If a non existing product is tried to be created, the system will ask for new information input.

**-Shopping Requirements**

-If a customer wants to buy from a branch and the product exists in their stock, the purchase is done instantly. The product will be added as a past order.

-If a customer wants to buy from a branch and the product does not exist in their stock, the purchase will be added as a new order. When the product is supplied to the stock of the branch, the purchase can be completed.

-If a customer wants to buy from the online store, the purchase will be added as a new order and when a branch employee arrange the sales, the purchase will be done.

**- Arranging Sales Requirements**

-Branch employees can arrange the sales of the customer. They edit a new order of a customer as a past order to complete the purchase. They have to decrease the product amount from the store’s stock. If the product does not exist in the stock, they cannot change the order as a past order. They firstly need to add the product to the Supply List. Supply the necessary product to the store, then arrange that sale again.

**B-) Non-Functional Requirements**

**-**The system is written using Java jdk 15.0.1

-You can use the makefile then use java AutomationSystem command to run the program. -The system comes with the files AdminsInfo.txt, BranchEmployeesInfo.txt, CustomersInfo.txt and BranchCount.txt. These files are important for the system to start properly. If these files are deleted, there will not be any admins, employees or customers that are registered to begin with. Also there will not be any branches. Other files will be created by the system when they are needed.

-To be able to log in as an admin, you can check the AdminsInfo.txt file and log in with one of the existing admin information.

-To be able to log in as a branch employee, you can check the BranchEmployeesInfo.txt file and log in with one of the existing employee information.

-To be able to log in as a customer, you can check the CustomersInfo.txt file and log in with one of the existing customer information or you can register as a new customer.

-The input for the model information for the products is expected to be an integer. If a string is entered, the system will be exited.

-The input for all the string information such as name, surname, email information etc is expected not to have spaces inside. If a string with spaces inside is entered, the information might be saved wrongly and need to be edited later on.

-The status of the users is to determine if they were registered from before or not. If a wrong status is entered, InvalidStatus will be thrown and the system will be exited.

Example: if a user is created with “old” status but the email address is not registered, it is i an invalid status situation.

-If the branch employees try to remove more products than there is, InvalidAmount will be thrown and the system will be exited.

**2-) Class Diagram-Use Case Diagram**

**-**The class diagram is added as ClassDiagram.png to the homework file.

-The use case diagram is added as UseCaseDiagram.ong to the homework file.

**3-) Problem Solution Aproach**

-While starting designing the software, I decided to create the small essential pieces first. These pieces are the Users, Products and the Stores. I have implemented all of their qualities in their own classes.

To be able to create Admin, BranchEmployee and Customer classes, I created an abstract User class. The reason why it is an abstract class is, Admin, BranchEmployee and Customer classes have many common fields and methods.

I decided to create one Product class rather than creating different classes for all different type of products because that would be too much code repetition and too many classes to deal with. I decided to make a Store interface and implemented it on Branch and OnlineStore classes.

The reason why I decided to create an interface was wanting to be able to use both Branches and OnlineStores in the same array with the same functions.

Another thing I paid attention was saving everything to files. The system saves the data to the arrays but I wanted the changes to stay even when I closed the program. Because of that I created a file for everything that might be changed.

After creating every detail about these classes I created the CompanySystem class that gathers all those small pieces all together. This class connects every method of the other classes that need to work together to perform proper tasks like adding a product to a customer’s past orders.

-I wanted to make the general flow of the system to be like this:

\*User logs into the system or gets registered.

\*Their status is determined: Admin/Employee/Customer.

\*They can choose the actions they want to do through the menu of the system.

\*All the changes done to the informations are saved to the files.

\*When the system is opened again, the changes will still be there. Thanks to this, all the users can interact with eachother in a way with the changes they make in the system.

-While designing the containers, I decided to design the KWArrayList without iterators like the implementation in the book and decided to implement the LList with iterators. These containers have the methods of regular ArrayList and LinkedList.

-While designing the HybridList, I decided to determine the max number as 12 because 12 is the maximum amount of a product’s types. Implemented all the necessary get, add and remove functions which will help to add remove and get from any place from the list with different time complexities depending on the needs. For example if the exact position of an item in the list is known, there is no need to iterate over the whole list. You can just iterate over the exact KWArrayList that the element is in. This HybridList will be able to hold all the products in its KWArrayLists.

**4-) Test Cases and Running Results**

-These tests are done in the specified order. The orders are specified with the numbers.

-Note: While doing all the tests, the files were also examined. The changes in the files were done successfuly.

**1-) Driver Code Tests:**

**1-) LList Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **1** | Creating some LList with different methods | Successful creation | The same | Pass |
| **2** | Adding to LLists | Successful addition | The same | Pass |
| **3** | Adding to random indexes | Successful addition | The same | Pass |
| **4** | Removing elements | Successful removal | The same | Pass |
| **5** | Testing exceptions | Exception throwing | The same | Pass |

**B-) KWArraryList Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **6** | Creating some KWArrayList with different methods | Successful creation | The same | Pass |
| **7** | Adding to random indexes | Successful addition | The same | Pass |
| **8** | Removing from random indexes | Successful removal | The same | Pass |
| **9** | Exception Test with 0 | Exception throwing | The same | Pass |
| **10** | Exception Test with 1 | Exception throwing | The same | Pass |
| **11** | Exception Test with 2 | Exception throwing | The same | Pass |

**C-) HybridList Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **12** | Creating a HybridList and adding items to it | Successful creation and addition | The same | Pass |
| **13** | Removing from random indexes | Successful removal | The same | Pass |
| **14** | Exception Test with 0 | Exception throwing | The same | Pass |
| **15** | Exception Test with 1 | Exception throwing | The same | Pass |
| **16** | Exception Test with 2 | Exception throwing | The same | Pass |

**D-) AutomationSystem Driver Code**

**OnlineStore Class Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **17** | Creating an OnlineStore and checking it’s stock | Successful creation and printing all stock | The same | Pass |
| **18** | Removing and adding some items | Successful addition and removal | The same | Pass |
| **19** | Making a clone | Successful cloning | The same | Pass |
| **20** | Exception Test with 0 | Exception throwing | The same | Pass |
| **21** | Exception Test with 1 | Exception throwing | The same | Pass |

**Branch Class Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **22** | Creating 4 Branches and checking their store numbers | Successful creation with right branch numbers | The same | Pass |
| **23** | Removing and adding items | Successful addition and removal | The same | Pass |
| **24** | Changing a branch’s number | Successful change | The same | Pass |
| **25** | Exception Test with 0 | Exception throwing | The same | Pass |
| **26** | Exception Test with 1 | Exception throwing | The same | Pass |

**Product Class Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **27** | Creating some products with different methods | Successful creation | The same | Pass |
| **28** | Setting storeInfo for a product | Successful setting | The same | Pass |
| **29** | Exception Test | Exception throwing | The same | Pass |

**Admin Class Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **30** | Creating an admin with wrong information | Needing a fix then creating successfuly | The same | Pass |
| **31** | Exception Test | Exception throwing | The same | Pass |

**BranchEmployee Class Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **32** | Creating an employee with wrong information | Needing a fix then creating successfuly | The same | Pass |
| **33** | Exception Test | Exception throwing | The same | Pass |

**Customer Class Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **34** | Creating customers with wrong information | Successful creation | The same | Pass |
| **35** | Adding new and past orders | Successful addition | The same | Pass |
| **36** | Removing orders | Successful removal | The same | Pass |
| **37** | Adding information | Successful addition | The same | Pass |
| **38** | Exception Test | Exception throwing | The same | Pass |

**CompanySystem Class Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **39** | Creating a system and printing it’s users | Successful creation | The same | Pass |
| **40** | Adding and removing employees | Successful addition and removal | The same | Pass |
| **41** | Printing all the products | Printing | The same | Pass |
| **42** | Searching for a product | Showing the stocks for the product | The same | Pass |
| **43** | Printing SupplyList, removing items when it is empty | Printing but cannot remove when it is empty | The same | Pass |
| **44** | Adding supplies to SupplyList then removing from it. | Successful addition and removal | The same | Pass |
| **45** | Adding a new branch then removing it | Successful addition and removal | The same | Pass |
| **46** | Exception Test | Exception throwing | The same | Pass |

**A-)Logging In/Registration Tests**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test No** |  | **Scenario** |  | **Expected** | |  | **Real Result** | |  | **Pass/Fail** |
|  |  |  |  | **Result** | |  |  |  |  |  |
| **1** |  | Entering wrong |  | Asking for a | |  | The same | |  | Pass |
|  |  | password as |  | new password. | |  |  |  |  |  |
|  |  | admin then |  | If entered the | |  |  |  |  |  |
|  |  | fixing it. |  | right password, | |  |  |  |  |  |
|  |  |  |  | logs in. | |  |  |  |  |  |
| **2** |  | Entering wrong |  | Asking for a | |  | The same | |  | Pass |
|  |  | password as |  | new password. | |  |  |  |  |  |
|  |  | employee then |  | If entered the | |  |  |  |  |  |
|  |  | fixing it. |  | right password, | |  |  |  |  |  |
|  |  |  |  | logs in. | |  |  |  |  |  |
| **3** |  | Entering wrong |  | Asking for a | |  | The same | |  | Pass |
|  |  | password as |  | new password. | |  |  |  |  |  |
|  |  | customer then |  | If entered the | |  |  |  |  |  |
|  |  | fixing it. |  | right password, | |  |  |  |  |  |
|  |  |  |  | logs in. | |  |  |  |  |  |
| **4** |  | Entering as a |  | Asking for | |  | The same | |  | Pass |
|  |  | new user and |  | information and | |  |  |  |  |  |
|  |  | registering as a |  | then registering | |  |  |  |  |  |
|  |  | new customer. |  | the customer. | |  |  |  |  |  |
| **B-)Customer Menu Tests** | | | |  |  |  |  |  |  |  |
| **Test No** | **Scenario** | | |  | **Expected** | |  | **Real Result** |  | **Pass/Fail** |
|  |  |  |  |  | **Result** | |  |  |  |  |
| **5** | Putting a wrong | | |  | Asking for a | |  | The same |  | Pass |
|  | command | | |  | new command | |  |  |  |  |
| **6** | Searching for a | | |  | Asking for new | |  | The same |  | Pass |
|  | product with wrong | | |  | proper | |  |  |  |  |
|  | type&colour&model | | | | information. If | |  |  |  |  |
|  | information as | | |  | entered | |  |  |  |  |
|  | customer and then | | |  | properly, | |  |  |  |  |
|  | fixing it | | |  | proceeds with | |  |  |  |  |
|  |  |  |  |  | searching for | |  |  |  |  |
|  |  |  |  |  | the product. | |  |  |  |  |
| **7** | Using LIST | | |  | Lists all the | |  | The same |  | Pass |
|  | command | | |  | products | |  |  |  |  |
|  |  |  |  |  | available | |  |  |  |  |
| **8** | Using PASTOrders | | |  | Showing No | |  | The same |  | Pass |
|  | command while | | |  | Past Orders | |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | having no past |  |  |  |
|  | orders |  |  |  |
| **9** | Shopping from | Asking for new | The same | Pass |
|  | Online Store with | proper |  |  |
|  | wrong product | information. If |  |  |
|  | information then | entered |  |  |
|  | fixing it | properly, |  |  |
|  |  | proceeds with |  |  |
|  |  | buying the |  |  |
|  |  | product. |  |  |
| **10** | Shopping from a | Asking for new | The same | Pass |
|  | branch with wrong | proper |  |  |
|  | product information | information. If |  |  |
|  | then fixing it | entered |  |  |
|  |  | properly, |  |  |
|  |  | proceeds with |  |  |
|  |  | buying the |  |  |
|  |  | product. |  |  |
| **11** | Trying to shop from | Asking for new | The same | Pass |
|  | a store that does not | proper |  |  |
|  | exist then fixing it. | information. If |  |  |
|  |  | entered |  |  |
|  |  | properly, |  |  |
|  |  | proceeds with |  |  |
|  |  | shopping. |  |  |
| **12** | Trying to shop from | Asking for a | The same | Pass |
|  | a store that does not | new store to |  |  |
|  | have the specified | choose then |  |  |
|  | product in its stock | proceeds with |  |  |
|  | then changing the | shopping. |  |  |
|  | store. |  |  |  |
| **13** | Trying to shop from | Adding the | The same | Pass |
|  | a store that does not | order as a new |  |  |
|  | have the specified | order and |  |  |
|  | product in its stock | waiting it to be |  |  |
|  |  | supplied. |  |  |
| **14** | Using PASTOrders | Printing out the | The same | Pass |
|  | command while | past orders. |  |  |
|  | having past orders. |  |  |  |
| **15** | Checking if the | Stocks should | The same | Pass |
|  | stocks are updated | have been |  |  |
|  | after making | updated and the |  |  |
|  | purchases with | updated version |  |  |
|  | STOCK command | should be |  |  |
|  |  | printed out. |  |  |
| **C-)Admin Menu Tests** | |  |  |  |
| **Test No** | **Scenario** | **Expected** | **Real Result** | **Pass/Fail** |
|  |  | **Result** |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **16** |  | Using CHECK | |  | Printing out | | The same | | Pass |
|  |  | command when | |  | There is no | |  |  |  |
|  |  | there is no products | |  | Products that | |  |  |  |
|  |  | registered in the | |  | need to be | |  |  |  |
|  |  | SupplyList | |  | supplied | |  |  |  |
| **17** |  | Trying to add an | |  | Asking for | | The same | | Pass |
|  |  | already existing | |  | another email | |  |  |  |
|  |  | employee then | |  | that does not | |  |  |  |
|  |  | fixing it | |  | exist. If entered | |  |  |  |
|  |  |  |  |  | properly, adds | |  |  |  |
|  |  |  |  |  | the employee. | |  |  |  |
| **18** |  | Trying to remove an | |  | Throwing | | The same | | Pass |
|  |  | employee who does | |  | InvalidStatus | |  |  |  |
|  |  | not exist | |  |  |  |  |  |  |
| **19** |  | Using | |  | Removing the | | The same | | Pass |
|  |  | REMOVEEmployee | |  | employee | |  |  |  |
|  |  | command | |  |  |  |  |  |  |
|  |  | successfuly | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| **20** |  | Using ADDBranch | |  | Adding a new | | The same | | Pass |
|  |  | command | |  | branch | |  |  |  |
|  |  | successfuly | |  |  |  |  |  |  |
| **33** |  | Using CHECK | |  | Printing out all | | The same | | Pass |
|  |  | function when there | |  | the products | |  |  |  |
|  |  | are products that | |  | that need to be | |  |  |  |
|  |  | need to be supplied | |  | supplied | |  |  |  |
| **34** |  | Using | |  | Asking for a | | The same | | Pass |
|  |  | REMOVEBranch | |  | proper branch | |  |  |  |
|  |  | function with a | |  | number. If | |  |  |  |
|  |  | wrong branch | |  | entered | |  |  |  |
|  |  | number then fixing | |  | properly, | |  |  |  |
|  |  | it | |  | removes it. | |  |  |  |
| **D-)Employee Menu Test** | | | |  |  |  |  |  |  |
| **Test No** |  | **Scenario** |  | **Expected** | |  | **Real Result** |  | **Pass/Fail** |
|  |  |  |  | **Result** | |  |  |  |  |
| **21** |  | Using STOCK |  | Showing the | |  | The same |  | Pass |
|  |  | command for all |  | stocks of all | |  |  |  |  |
|  |  | stocks |  | stores in total | |  |  |  |  |
| **22** |  | Using STOCK |  | Asking for a | |  | The same |  | Pass |
|  |  | command with a |  | proper input | |  |  |  |  |
|  |  | wrong command |  | then showing | |  |  |  |  |
|  |  | then for ONE |  | the given store’s | |  |  |  |  |
|  |  |  |  | stock only | |  |  |  |  |
| **23** |  | Using |  | Printing out that | |  | The same |  | Pass |
|  |  | PASTOrders |  | there is no | |  |  |  |  |
|  |  | command with a |  | customers with | |  |  |  |  |
|  |  | wrong customer |  | the given | |  |  |  |  |
|  |  | number |  | number | |  |  |  |  |

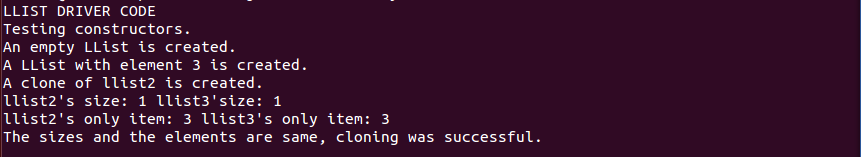
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **24** | Using | Showing the | The same | Pass |
|  | PASTOrders | customer’s past |  |  |
|  | command | orders with the |  |  |
|  | successfuly | given number |  |  |
| **25** | Using | Printing out | The same | Pass |
|  | SUPPLYProducts | there is nothing |  |  |
|  | order when there | that needs to be |  |  |
|  | is nothing to be | supplied |  |  |
|  | supplied. |  |  |  |
| **26** | Using | Asking for | The same | Pass |
|  | ADDProduct with | proper |  |  |
|  | wrong product | information. If |  |  |
|  | informations to a | entered, edits |  |  |
|  | branch then fixing | the stock of the |  |  |
|  | it | branch |  |  |
|  |  | accordingly. |  |  |
| **27** | Using | Asking for | The same | Pass |
|  | ADDProduct with | proper |  |  |
|  | wrong product | information. If |  |  |
|  | informations to | entered, edits |  |  |
|  | Online Store then | the stock of the |  |  |
|  | fixing it | online store |  |  |
|  |  | accordingly. |  |  |
| **28** | Using | Editing the | The same | Pass |
|  | REMOVEProduct | store’s stock |  |  |
|  | command | accordingly. |  |  |
|  | successfuly. |  |  |  |
| **29** | Trying to remove | Throwing | The same | Pass |
|  | a product by | InvalidAmount |  |  |
|  | some amount that |  |  |  |
|  | would make the |  |  |  |
|  | stock of it |  |  |  |
|  | negative |  |  |  |
| **30** | Using | Arranging all | The same | Pass |
|  | ARRANGESales | the sales. |  |  |
|  | command | Editing new |  |  |
|  | successfuly | orders, editing |  |  |
|  |  | the Supply List |  |  |
|  |  | etc |  |  |
| **31** | Using MAKESale | Adding the | The same | Pass |
|  | command for an | order to the |  |  |
|  | existing customer | customer’s new |  |  |
|  | when the product | order’s section |  |  |
|  | is not in the stock | and adding the |  |  |
|  |  | product to the |  |  |
|  |  | Supply List. |  |  |
| **32** | Using MAKESale | Asking for a | The same | Pass |
|  | command for a | new store |  |  |
|  | new customer | number, then |  |  |
|  | firstly from a | making the sale. |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | branch that does | |  |  |  |  |  |
|  |  | not exist, then | |  |  |  |  |  |
|  |  | from a branch | |  |  |  |  |  |
|  |  | which has the | |  |  |  |  |  |
|  |  | product in the | |  |  |  |  |  |
|  |  | stock. | |  |  |  |  |  |
| **35** |  | Using | | Supplying all | | The same | | Pass |
|  |  | SUPPLYProducts | | the products, | |  |  |  |
|  |  | command when | | editing the | |  |  |  |
|  |  | there are product | | stocks. | |  |  |  |
|  |  | that need to be | |  |  |  |  |  |
|  |  | supplied. | |  |  |  |  |  |
| **36** |  | Using | | Arranging all | | The same | | Pass |
|  |  | ARRANGESales | | the sales. | |  |  |  |
|  |  | after supplying | | Editing new | |  |  |  |
|  |  | the products. | | orders | |  |  |  |
| **E-) Extra Tests** |  |  |  |  |  |  |  |  |
| **Test No** |  | **Scenario** |  | **Expected** |  | **Real Result** |  | **Pass/Fail** |
|  |  |  |  | **Result** |  |  |  |  |
| **37** |  | Trying to put a |  | Might cause |  | The same |  | Pass |
|  |  | string input for |  | improper data |  |  |  |  |
|  |  | model instead |  |  |  |  |  |  |
|  |  | of integer. |  |  |  |  |  |  |
| **38** |  | Trying to put |  | Throws |  | The same |  | Pass |
|  |  | spaces in string |  | exception, |  |  |  |  |
|  |  | inputs |  | exists the |  |  |  |  |
|  |  |  |  | prgram |  |  |  |  |
| **39** |  | Using QUIT |  | Exits the |  | The same |  | Pass |
|  |  | command |  | program |  |  |  |  |

**The Results**

**A-) Driver Code Results**

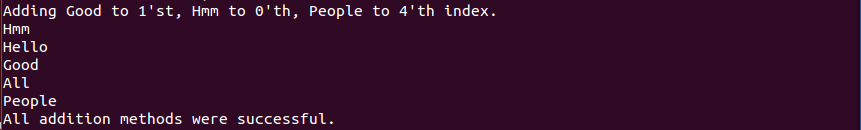
1-) Creating an empty LList (llist1), a LList with 3 elements (llist2) and a clone of llist2 (llist3) also checking if the cloning was succesful by checking sizes and elements inside llist2 and llist3.

****

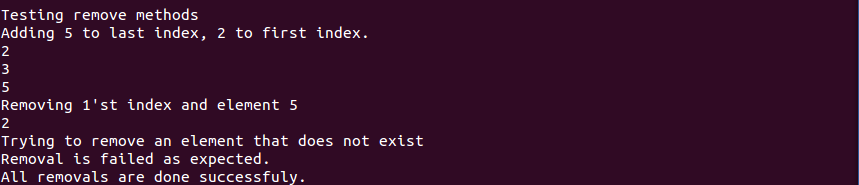
2-) Adding “Hello” then “All” to llist1 then checking it they are added properly.

****

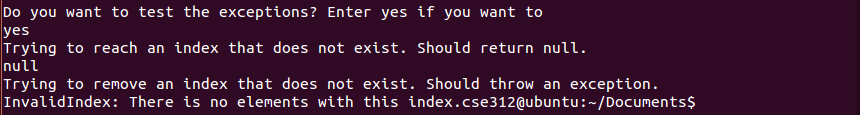
3-) Adding “Good” “Hmm” “People” to random indexes of llist1 then printing llist1 to check.

****

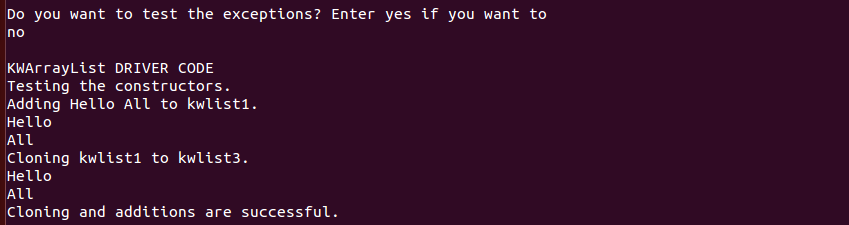
4-) Adding a few more elements afterwards removing some elements then checking.

****

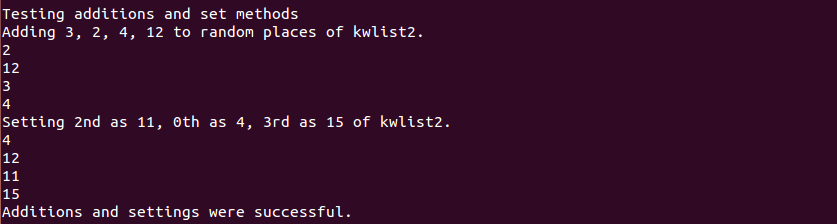
5-) To check exceptions entering yes and trying to reach an idex that does not exist.

****

6-) To continue not typing yes and then creating a KWArrayList with Strings (kwlist1), adding a few wlwmwnts to it and with Integers (kwlist2) also making a clone of kwlist1 (kwlist3).

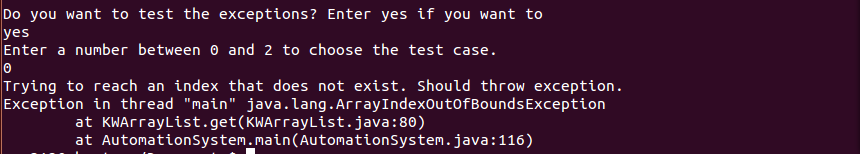
****

7-) Adding elements to random indexes of kwlist2 and then changing the elements of some random indexes.

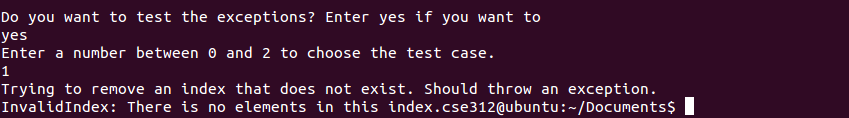
****

****8-) Removing some random indexes from kwlist2

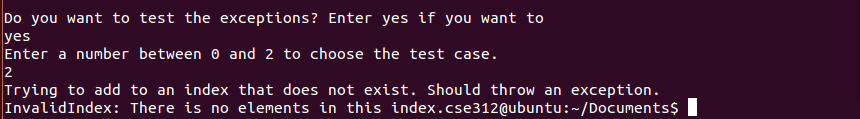
9-) To test exceptions entering yes and then 0. Trying to reach an index that does not exist.



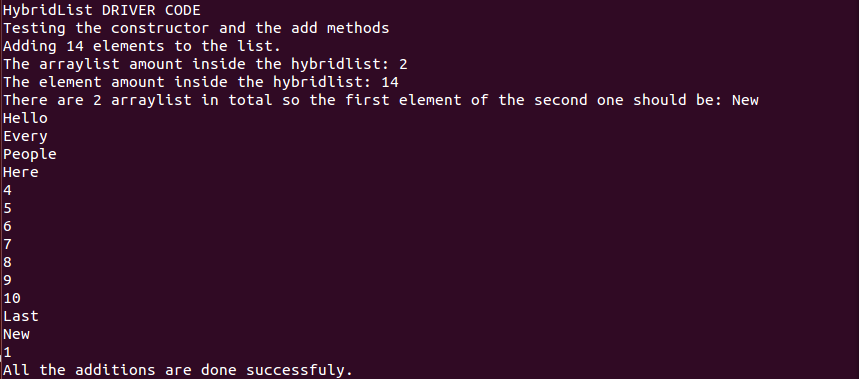
10-) To test exceptions entering yes then 1. Trying to remove an index that does not exist.

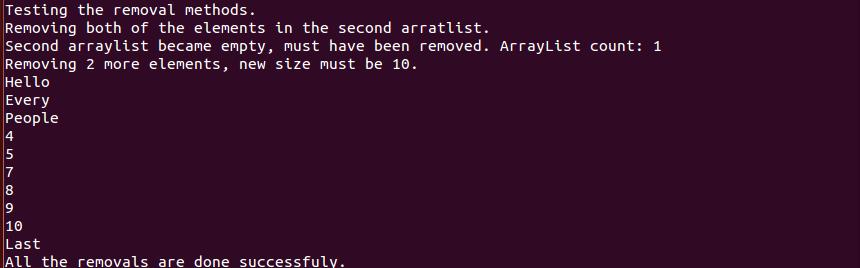


11-) To test exceptions entering yes then 2. Trying to add to an index that does not exist.

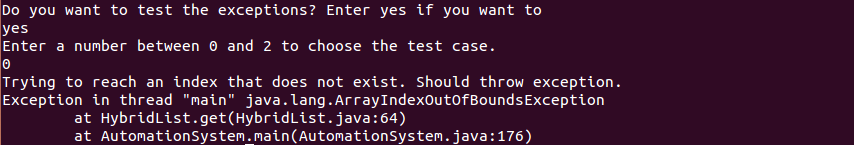


12-) To continue not typing yes and then creating an empty HybridList (hblist1). Afterwards adding 14 elements to it. Checking if the elements are added properly to proper KWArrayLists.

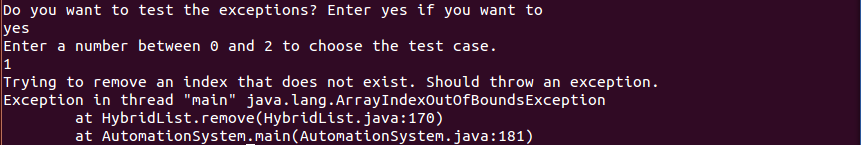


13-) Removing from random indexes of the list. Checking if they are removed properly from proper ArrayLists.

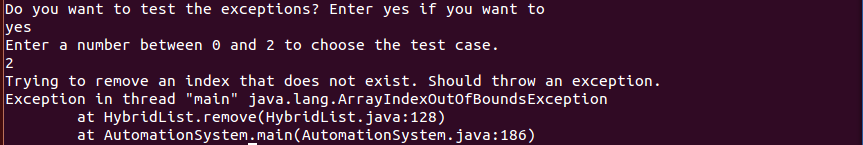
14-) To check exceptions typing yes then 0. Trying to reach an index that does not exist.



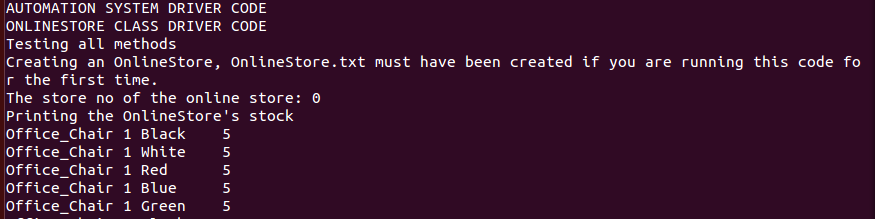
15-) To check exceptions typing yes then 1. Trying to remove an index that does not exist.



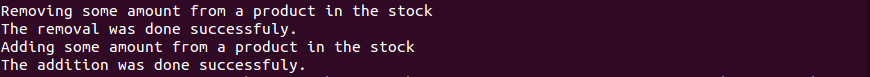
16-) To check exceptions typing yes then 2. Trying to remove an index that does not exist.



17-) To continue not typing yes then creating an OnlineStore and then printing its stock to see if it is created properly.

****

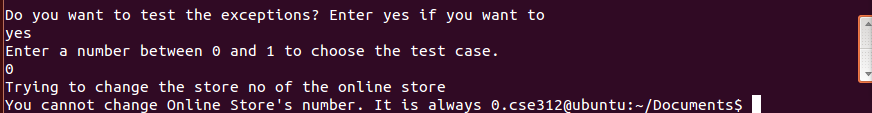
18-) Removing and adding some items to the stock.

****

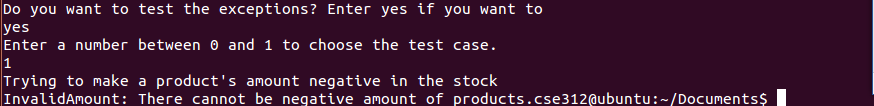
19-) Making a clone of the store.

****

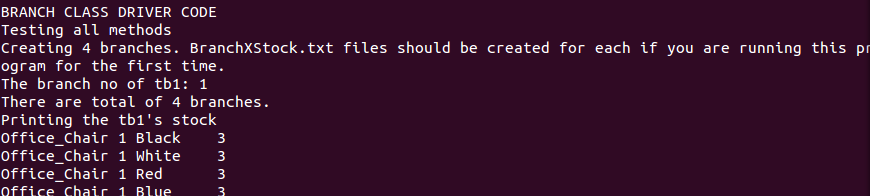
20-) To check exceptions typing yes then 0. Trying to change the store number.

****

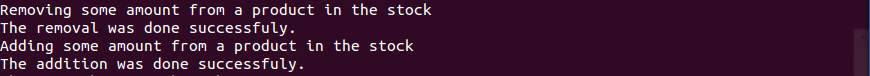
21-) To check exceptions typing yes then 1. Trying to make a product’s amount negative.

****

22-) To continue not typing yes then creating 4 Branches. Checking some branch numbers to see if they are created properly and printing one of the stocks.

****

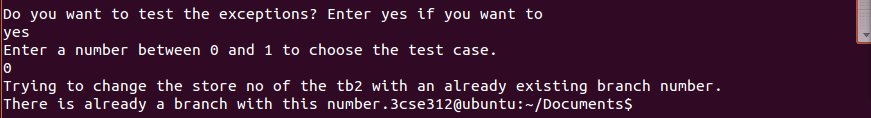
23-) Removing and adding some products to a branch’s stock.

****

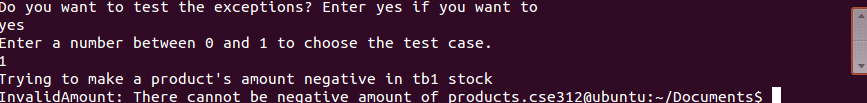
24-) Changing the branch number of branch 4.

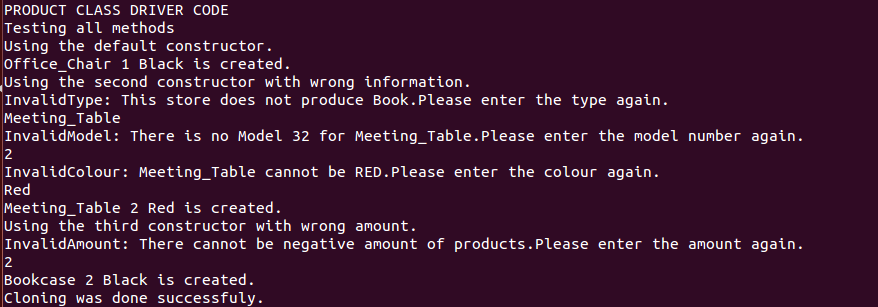
****

25-) To check exceptions typing yes then 0. Trying to change the store number of tb2 with a store

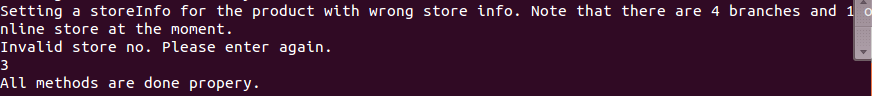
****

26-) To check exceptions typing yes then 1. Trying to make a product’s amount negative.

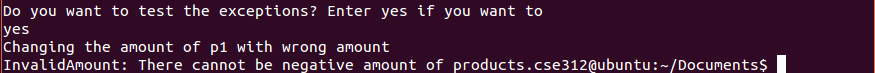
****

****27-) To continue not typing yes then creating some Products with wrong informations. You need to fix the informations. Also creating a clone of a product.

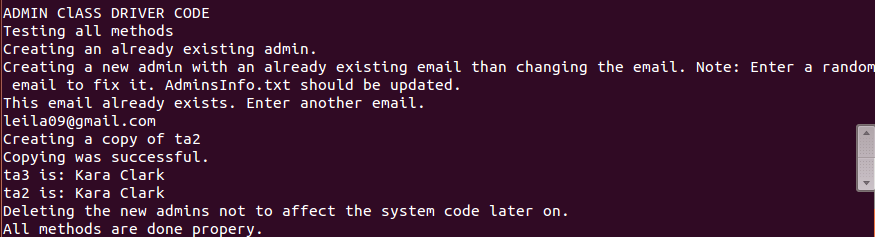
28-) Trying to set a storeInfo for a product with a store number that does not exist. You need to fix the information.



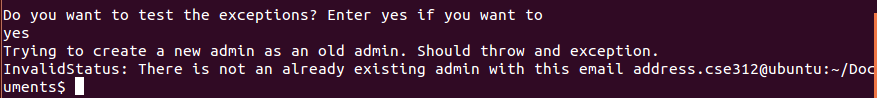
29-) To check exceptions typing yes. Trying to maket he amount of a product negative.



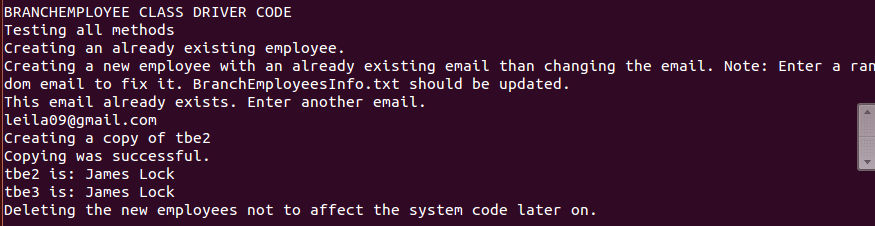
30-) To continue not typing yes then creating admins with proper information and non proper information. You need to fix the information. Also making a clone of an admin.



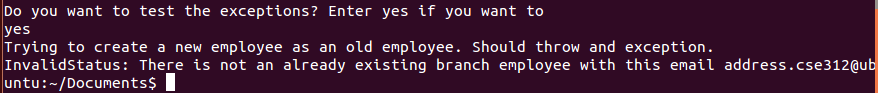
31-) To check exceptions typing yes. Trying to create a new admin as an old admin.



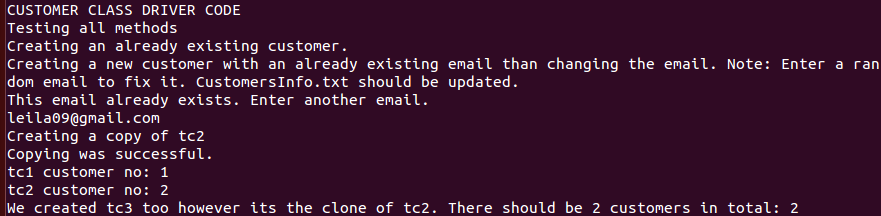
32-) To continue not typing yes and then creating branch employees with proper and non proper information. You need to fix the information. Also making a clone of an employee.

****

33-) To check exceptions typing yes. Trying to create a new employee as an old employee.

****

34-) To continue not typing yes and then creating customers with proper and non proper information. You need to fix the information. Also making a clone of a customer.

****

35-) Adding new orders and past orders to a customer then printing the orders.

****

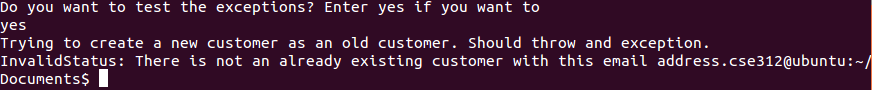
36-) Removing orders from a customer.

****

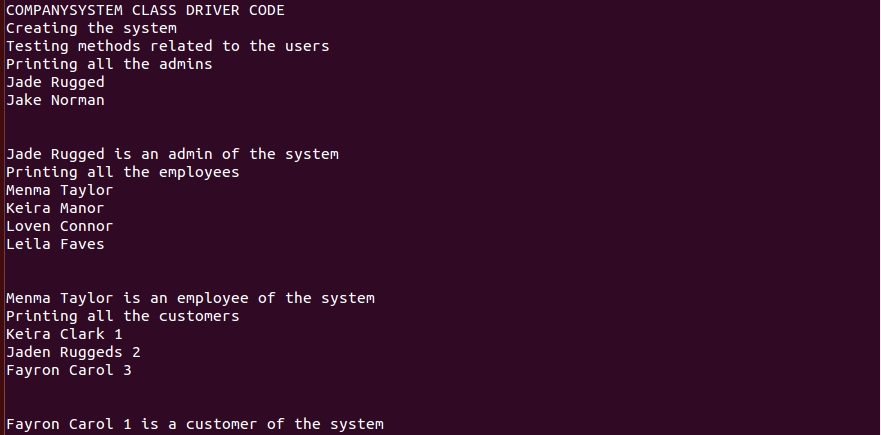
37-) Adding information to a customer.

****

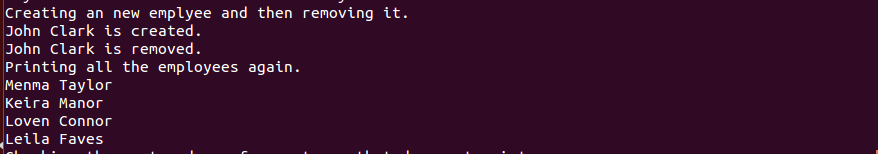
38-) To check exceptions typing yes. Trying to create a customer as an old customer.

****

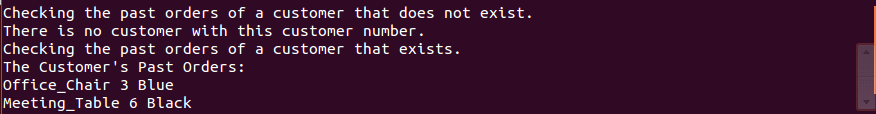
39-) Creating the system then printing admins, employees, customers of it. Checking the validity of some users.

****

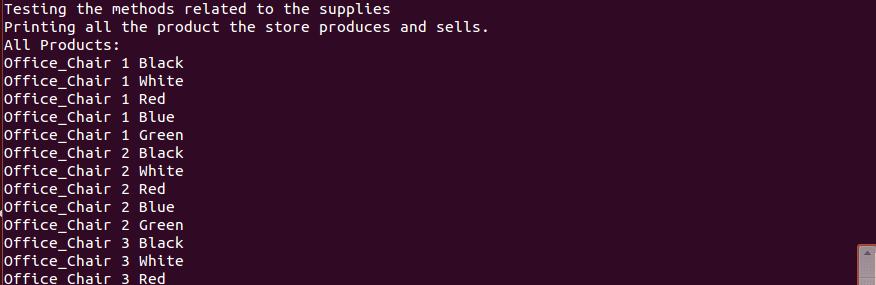
40-) Creating and removing an employee.

****

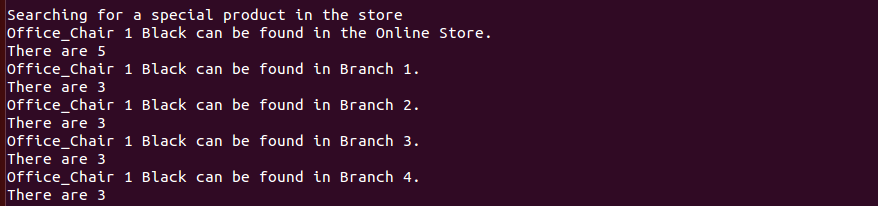
41-) Checking the past orders of a customer that doesnt exist then one that exists.

****

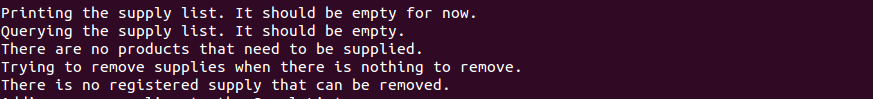
42-) Printing out all the products in the store.

****

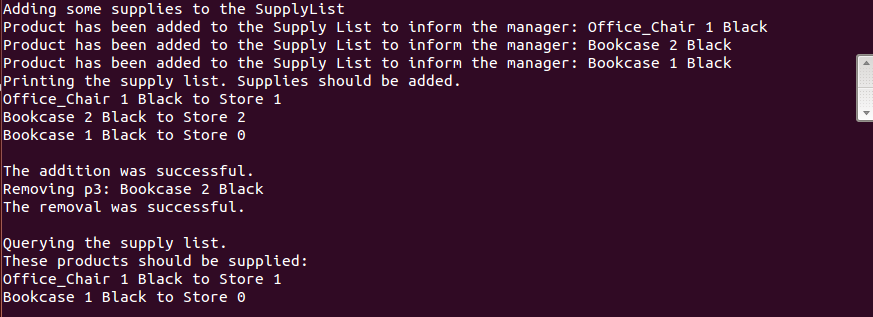
43-) Searching for a product in the store.

****

44-) Printing and querying the supply list when it is empty. Trying to remove when it is empty.

****

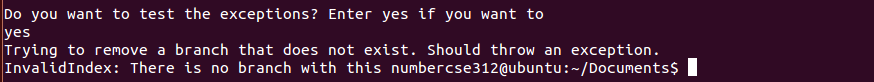
45-) Adding some supplies to the supply lits. Printing the new supply list. Adding to and removing from it.

****

46-) Adding a new branch then removing it.

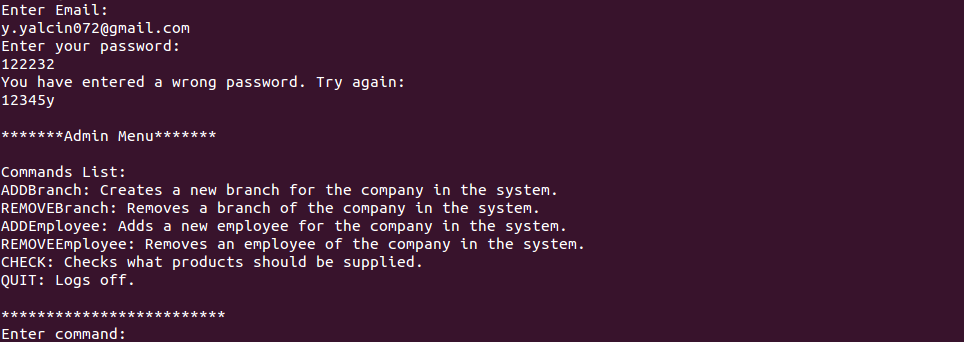
****

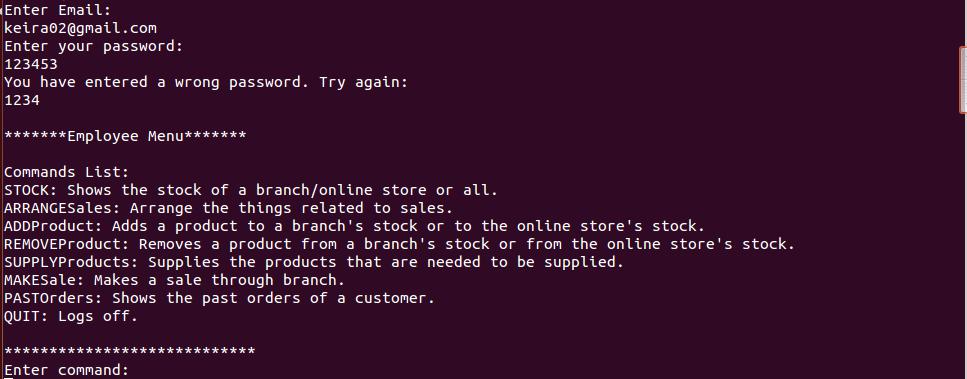
47-) To test the exceptions entering yes. Trying to remove a branch that does not exist.

****

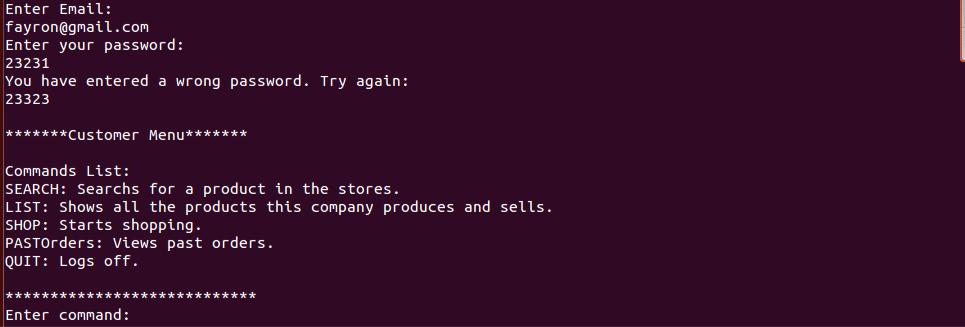
**B-) Menu System Results**

1-) Entering wrong password as admin then fixing it.

2-) Entering wrong password as employee then fixing it.

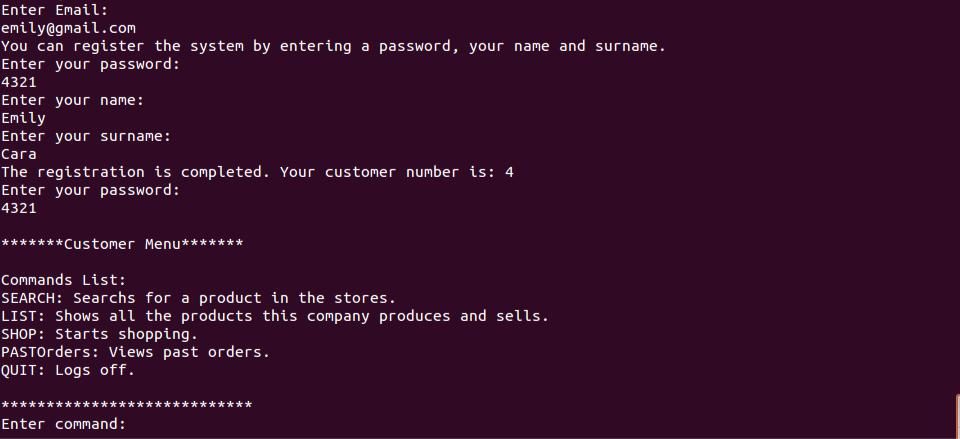


3-) Entering wrong password as customer then fixing it

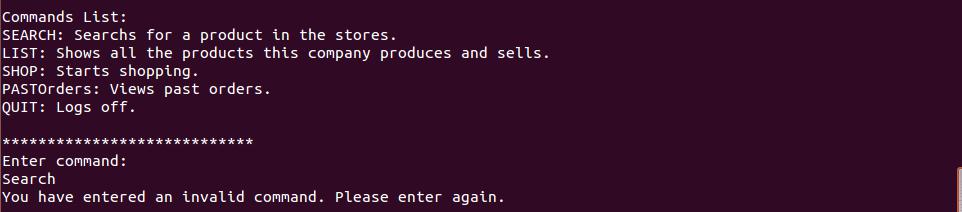


-

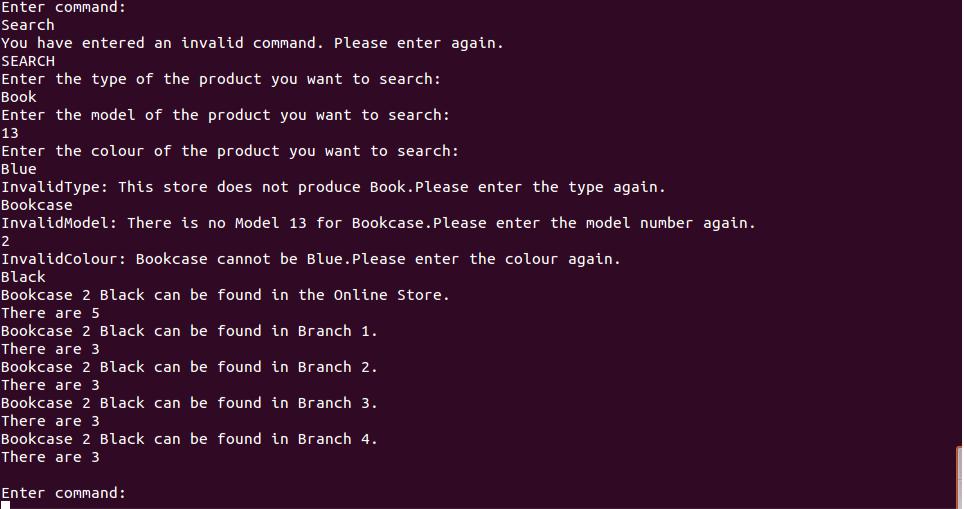
4-) Entering as a new user and registering as a new customer



5-) Putting a wrong command



6-) Searching for a product with wrong type&colour&model information as customer and then fixing it



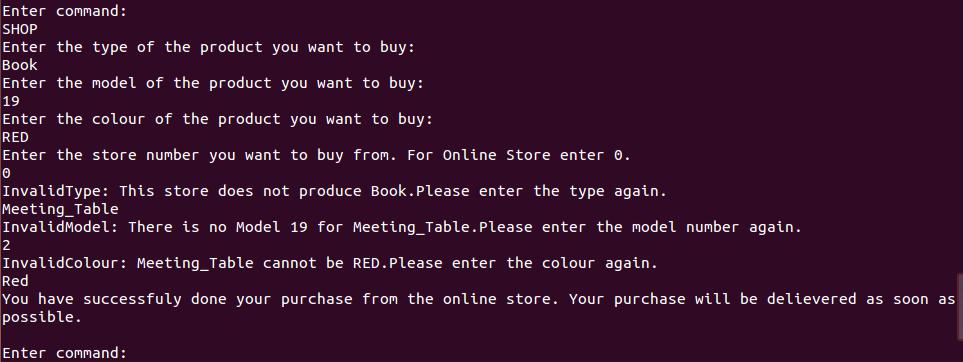
7-) Using LIST command (Not showing all the result because it is long)



8-) Using PASTOrders command while having no past orders



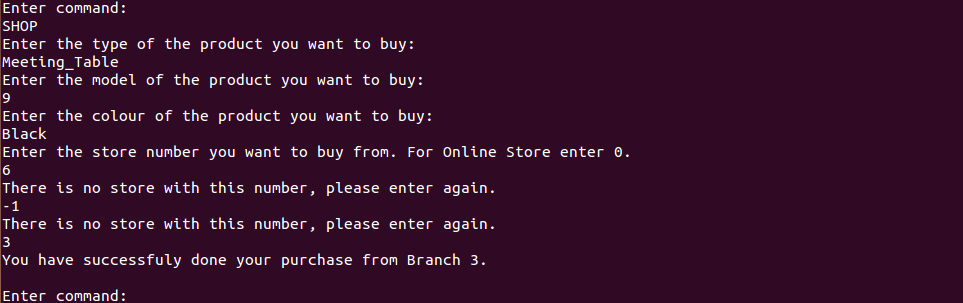
9-) Shopping from Online Store with wrong product information then fixing it



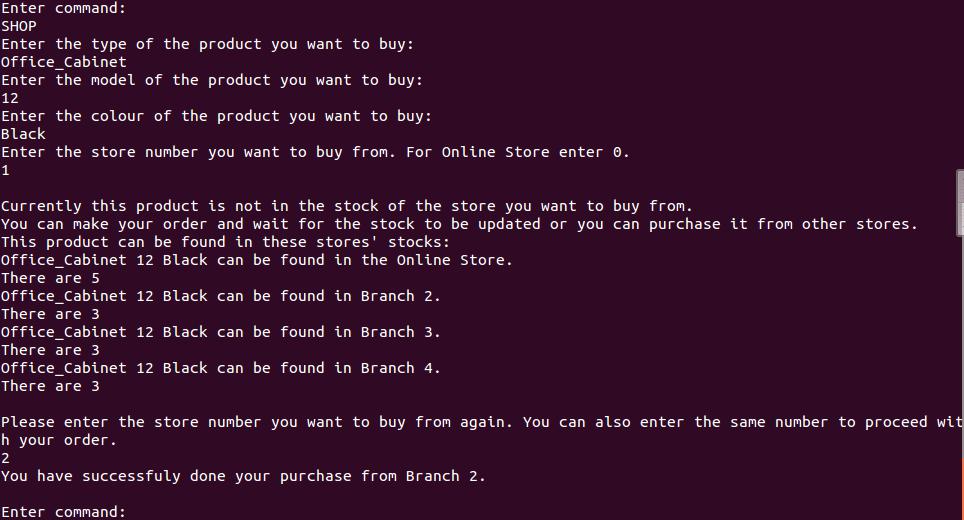
10-) Shopping from a branch with wrong product information then fixing it



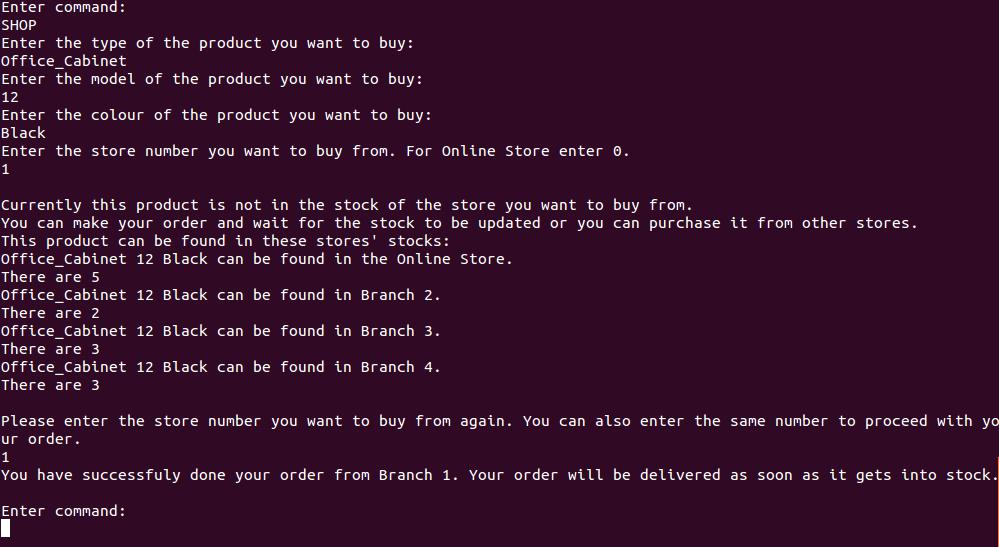
11-) Trying to shop from a store that does not exist then fixing it.



12-) Trying to shop from a store that does not have the specified product in its stock then changing the store.



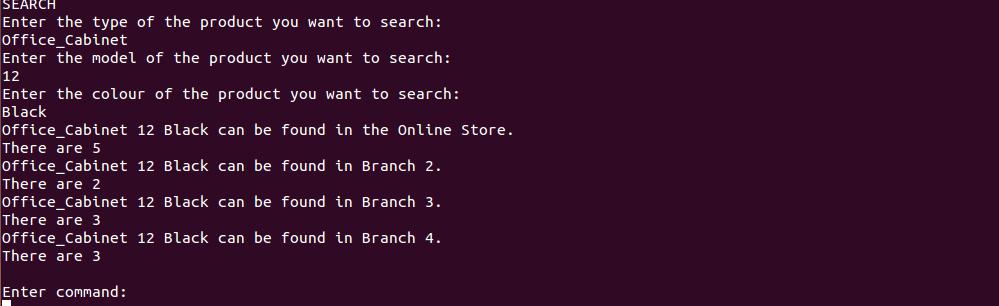
13-) Trying to shop from a store that does not have the specified product in its stock



14-) Using PASTOrders command while having past order



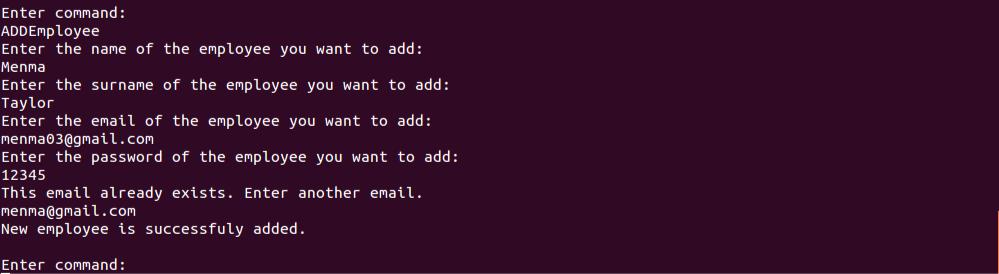
15-) Checking if the stocks are updated after making purchases with STOCK command



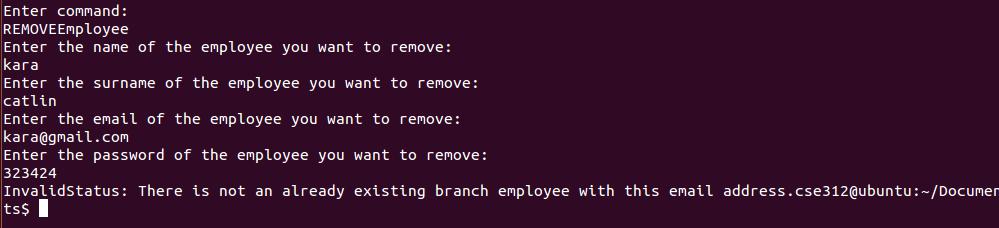
16-) Using CHECK command when there is no products registered in the SupplyList



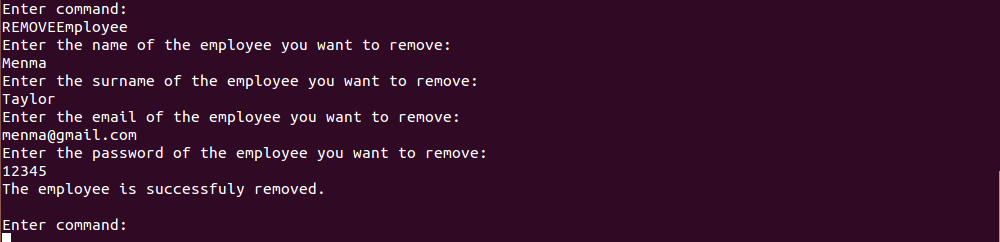
17-) Trying to add an already existing employee then fixing it



18-) Trying to remove an employee who does not exist



19-) Using REMOVEEmployee command successful



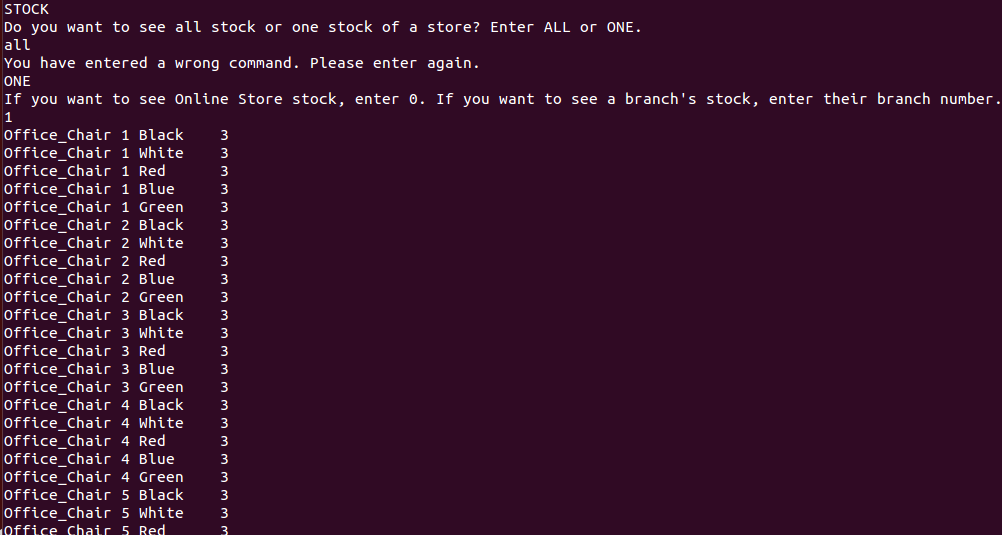
20-) Using ADDBranch command successfuly



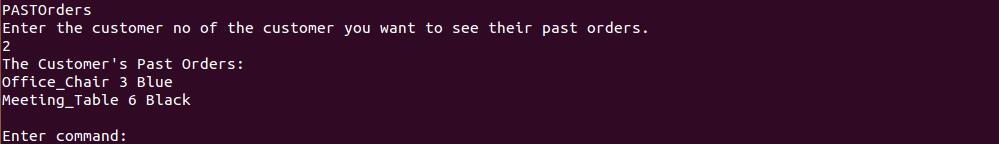
21-) Using STOCK command for all stocks (Not showing all because it is too long)



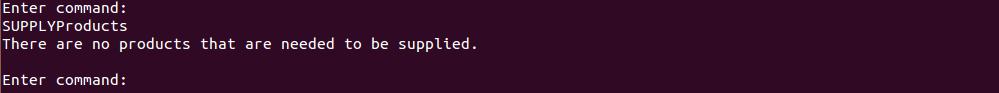
22-) Using STOCK command with a wrong command then for ONE



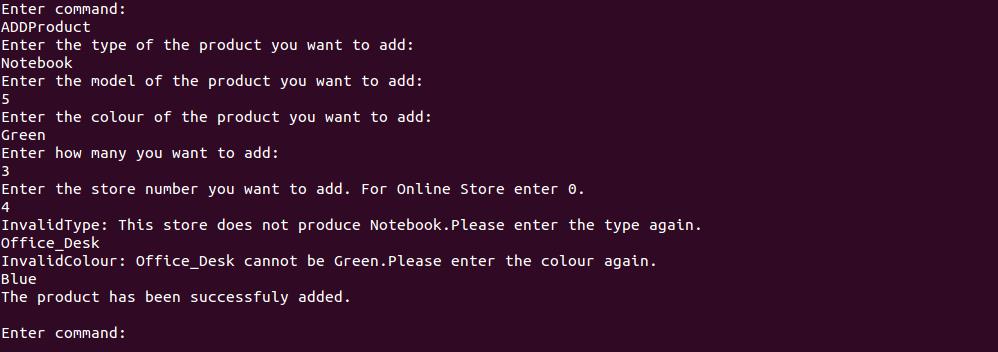
24-) Using PASTOrders command with a customer number



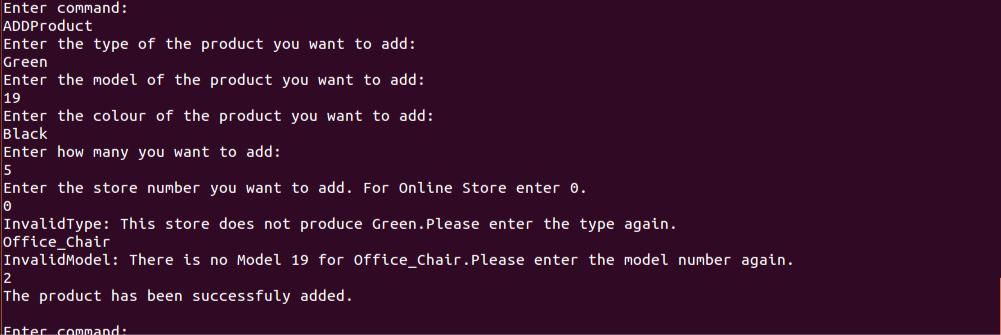
25-) Using SUPPLYProducts order when there is nothing to be supplied.



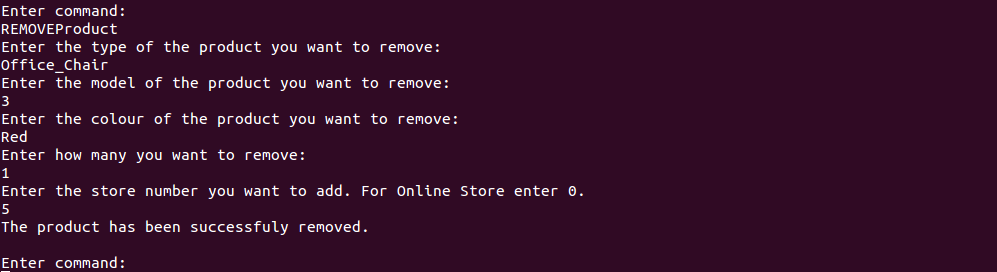
26-) Using ADDProduct with wrong product informations to a branch then fixing it

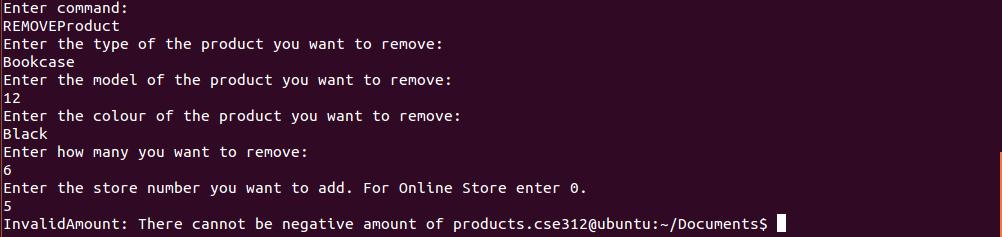


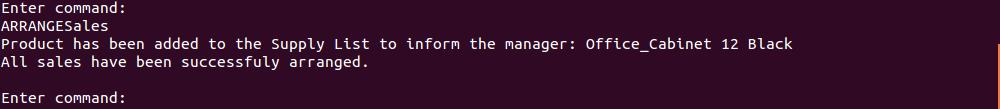
27-) Using ADDProduct with wrong product informations to Online Store then fixing it

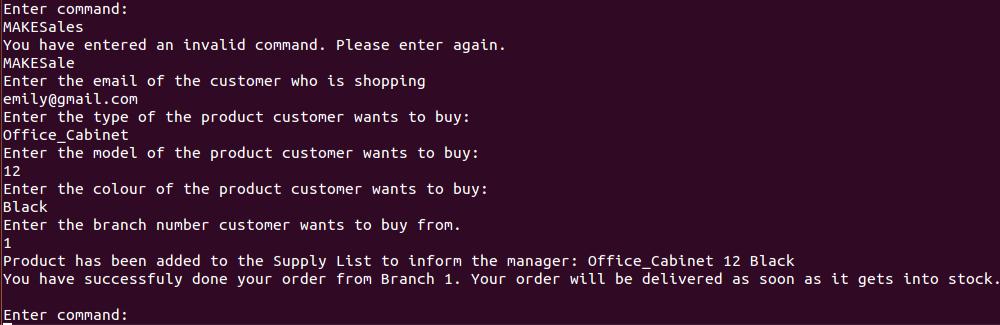


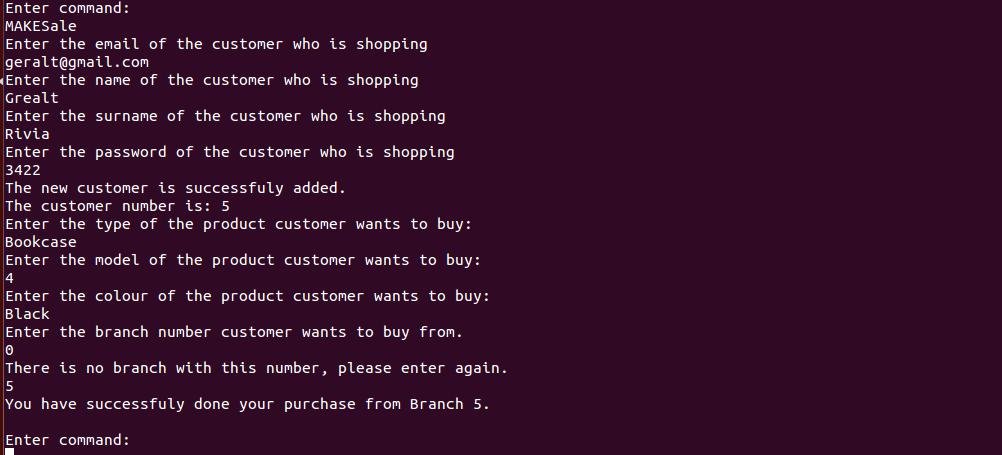
28-) Using REMOVEProduct command successfuly.



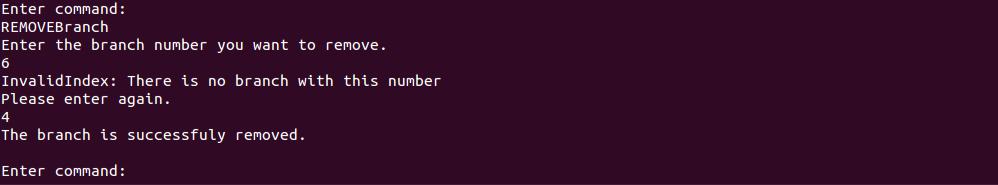
29-) Trying to remove a product by some amount that would make the stock of it negative

30-) Using ARRANGESales command successfuly

31-) Using MAKESale command for an existing customer when the product is not in the stock

32-) Using MAKESale command for a new customer firstly from a branch that does not exist, then from a branch which has the product in the stock

33-) Using CHECK function when there are products that need to be supplied

34-) Using REMOVEBranch function with a wrong branch number then fixing it

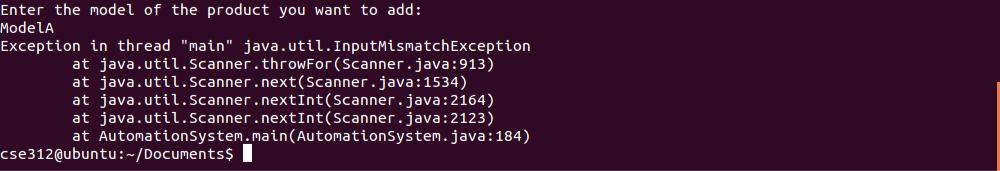
35-) Using SUPPLYProducts command when there are product that need to be supplied.



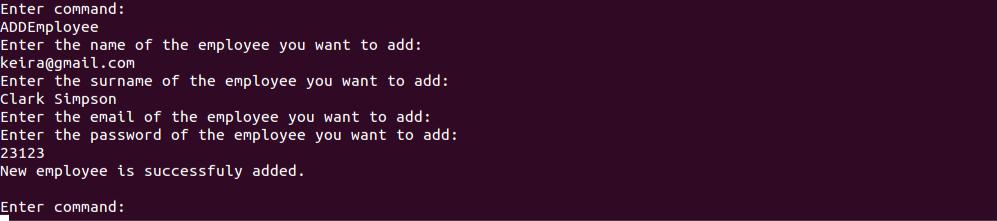
36-) Using ARRANGESales after supplying the products.



37-) Trying to put a string input for model instead of integer.



38-) Trying to put spaces in string inputs



39-) Using QUIT command

