**GIT Department of Computer Engineering CSE 222/505 - Spring 2021 Homework 6 # Report**

**Yeşim Yalçın**

**200104004094**

**1-) System Requirements**

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

-You can use the makefile then use java DriverCode command to run the program.

-DriverCode outputs are really long therefore running the program in an environment which shows all the outputs of the program is recommended.

If you are using terminal on Ubuntu, changing terminal’s scrolling setting is recommended. You can press Edit -> Profile Preferences -> Scrolling Tab -> Unlimited option when you are in terminal.

**2-) Problem Solution Aproach**

-Firstly I decided to create the data files. To create the data file for the Products, I have deleted the wrong formatted lines from the e-commerce-samples. Afterwards I have formated that file in a way that the product names would be in alphabetical order. I have done this by using a PriorityQueue and wrote them on a new Products.csv file. To create the data file for the users, I have gathered all the trader names from the e-commerce-samples and wrote them on a UsersInfo.txt file. I have written each trader only once also gave unique them ID numbers and passwords.

-After arranging the data fields I have created classes to hold the Users and Products. Moreover, I have created a Categories class to hold the categories of the products. This class can convert the string it reads from Products.csv file to a proper String ArrayList that holds the categories separately. I have done this class to have easier time to format the string of the categories also to be able to Access the categories separately.

-I have added methods to satisfy the e-shopping application’s needs like registration, logging in with unique ID’s. No users can share the same ID, they can share the same password and the name. However, traders cannot share the same name with any user. Users can perform all the tasks that are mentioned in the PDF file. Products also cannot share the same ID.

-I have used ArrayList in a few places.

1-) I have used it in Categories. It can store all the categories separately. I have chosen to use ArrayList here because the only method I use here is get() method of ArrayList which has constant time complexity.

2-) I have used it whenever I needed to sort some things in Customer class. I have put the unsorted elements inside the ArrayList and then sorted the ArrayList. I have chosen to use ArrayList here because while sorting no deletion, is needed. Only the add() method and get() method of the ArrayList are used and they both have constant time complexity.

-I have used LinkedList in one place.

1-) I have used LinkedList in Trader class to store the orders. The traders can meet&cancel the orders one by one and they can start meet&cancel from the very first order that was done like a Queue. Because LinkedList has the abilities to be a Queue, I have used LinkedList. I only use addLast, removeFirst methods of the LinkedList which are both constant time.

-I have used Queue as PriorityQueue in a few places.

1-) I have used Queue in Product class to gather the products in a name ordered way. By putting the products in a PriorityQueue I could easily preserve the name order while deleting or adding new products to the stock. Preserving the name order was necessary because I wanted the Products.csv file to be in alphabetical order. I only used poll and offer methods of queue which are both O(logn). Queue can be found in removeFromFile(), saveToFile() methods.

-I have used Tree as TreeSet in one place.

1-) I have used TreeSet in Customer class to gather the Products of a specific trader and then list them. I have used a TreeSet here because I did not want the Products with the same name to be shown more than once and checking if something is present inside a data structure has a good efficiency in a TreeSet which is usually O(logn). Also I wanted the products still have an order.

-I have used Hash Table in one place.

1-) I have used ShoppingSystem class to store the users and reach the users by their ID’s. It is really efficient to get a user with their ID’s from a Hashtable which might be constant time.

-To sort the search results of a customer I have chose InsertionSort, HeapSort and QuickSort. I have firstly implemented these with ArrayLists. I also added comparators for HeaspSort and QuickSort to be able to sort with name, price and discount percentages.

The products are hold in the csv file in alphabetical order. That’s why when I wanted to display them in a name ordered way, I have used InsertionSort. Because when I put the products into the ArrayList they will be already ordered and InsertionSort is the best for already ordered arrays. After one pass the insertion sort stops.

When I wanted to sort the products with their prices, I have used HeapSort because heap sort does not require any extra space and has O(nlogn) time complexity which is better than many sorting algorithms.

When I wanted to sort the products with their discount percentages, I have used QuickSort because like heap sort the time complexity of it is mostly O(nlogn), does not require much extra space unlike some other sorting algorithms.

-I have gathered all the menu and log in&registration activites in the ShoppingSystem class to represent the e shopping application.

**3-) Class Diagram-Use Case Diagram**

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

**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.

**A-) Test Cases**

**1-) Sorts Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **1** | Inserting 11 elements randomly and insertion sorting | Printing the elements in sorted order | The same | Pass |
| **2** | Inserting 11 elements randomly and insertion sorting | Printing the elements in sorted order | The same | Pass |
| **3** | Inserting 11 elements randomly and insertion sorting | Printing the elements in sorted order | The same | Pass |

**2-) Categories Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **4** | Creating a Categories class with a proper string | Successful creation | The same | Pass |
| **5** | Creating a Categories class with a ArrayList | Successful creation | The same | Pass |
| **6** | Trying to create a Categories class with improper string | Throws an exception | The same | Pass |

**3-) Product Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **7** | Creating an already registered product. Then changing the product’s information. | Successful creation and information change | The same | Pass |
| **8** | Creating a new product | Successful creation and stock update | The same | Pass |
| **9** | Removing the created product | Successful creation and stock update | The same | Pass |
| **10** | Trying to create an existing product with non existing ID. | Throws an exception | The same | Pass |

**4-) Customer Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **11** | Creating an already existing customer. | Successful creation no file udptade. | The same | Pass |
| **12** | Searching for products of an existing trader. | Listing the products | The same | Pass |
| **13** | Searching for products of a non existing trader. | No search results are found | The same | Pass |
| **14** | Ordering a product with a proper ID. | Successful order, Orders.csv file update | The same | Pass |
| **15** | Searching for a valid text in the shop. | Listing the search results | The same | Pass |
| **16** | Sorting the results by price | Successfuly sorting and listing | The same | Pass |
| **17** | Filter the results by price | Successfuly filtering and listing | The same | Pass |
| **18** | Searhing for an invalid text in the shop. | No search results are found | The same | Pass |
| **19** | Trying to order a non existing product. | Throws an exception | The same | Pass |

**5-) Trader Driver Code**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **20** | Creating 2 existing traders. | Successful creation | The same | Pass |
| **21** | Listing the trader’s orders. | Listing the orders | The same | Pass |
| **22** | Meeting one and cancelling one order | Successfuly meeting&cancelling also stock&Orders.csv update | The same | Pass |
| **23** | Editing the name, categories, description and prices of a product. | Sucessfuly editing and stock update | The same | Pass |
| **24** | Trying to edit a product that does not belong to the trader. | Throws an exception | The same | Pass |

**6-) Registration-Logging In Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **25** | Register as a new customer with wrong password | Asking for another password then register | The same | Pass |
| **26** | Register as a new trader with already existing trader name. | Register the already existing trader, no new creation | The same | Pass |
| **27** | Register as a new trader | New trader is created and registered, UsersInfo.txt is updated | The same | Pass |
| **28** | Login as a customer with wrong ID. | Throws an exception | The same | Pass |
| **29** | Login as a customer with wrong password | Asking for the password again then log in | The same | Pass |
| **30** | Login as a trader with wrong password | Asking for the password again then log in | The same | Pass |

**7-) Customer Menu Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **31** | List an existing trader’s products | Listing the products | The same | Pass |
| **32** | List a non existing trader’s products | No listing | The same | Pass |
| **33** | Search products & sort it by price and filter with prices | Listing by name order then by price order then only listing the ones in the price range. | The same | Pass |
| **34** | Search products & sort it by discount and filter with categories | Listing by name order then by discount order then only listing the ones in the category. | The same | Pass |
| **35** | Search products with invalid text | No search results. | The same | Pass |
| **36** | Order a product that exists | Successfuly ordering, Orders.csv is updated. | The same | Pass |
| **37** | Order a product that does not exist | Throws an exception. | The same | Pass |

**8-) Trader Menu Tests**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No** | **Scenario** | **Expected Result** | **Real Result** | **Pass/Fail** |
| **38** | List your orders | Listing the orders | The same | Pass |
| **39** | Add a new product | Successfuly adding, Products.cvs is updated. | The same | Pass |
| **40** | Add a new product to the stock with wrong ID. | Asking for an ID change then adding the product. Products.csv is updated. | The same | Pass |
| **41** | Remove an existing product | Successfuly removing. Product.csv is updated. | The same | Pass |
| **42** | Remove a non existing product | Throws an exception. | The same | Pass |
| **43** | Edit name, categories, description and price of a product. | Successfuly editing. Products.csv is updated. | The same | Pass |
| **44** | Trying to Edit name, categories, description and price of a product that does not belong to the trader. | Throws an exception. | The same | Pass |
| **45** | Meet one order. | Successfuly meeting. Orders.csv and Products.csv are updated. | The same | Pass |
| **46** | Cancel one order. | Successfuly canceling. Orders.csv is updated. | The same | Pass |
| **47** | Meet all orders. | Successfuly meeting. Orders.csv and Products.csv are updated. | The same | Pass |
| **48** | Cancel all orders. | Successfuly canceling. Orders.csv is updated. | The same | Pass |

**B-) Results**

1-) Testing Insertion Sort by inserting 11 random elements in random order and sorting it.

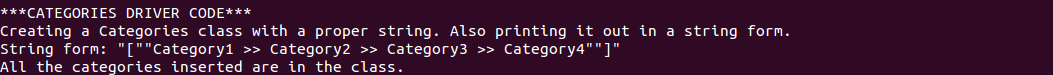
****

2-) Testing Heap Sort by inserting 11 random elements in random order and sorting it.

****

3-) Testing Quick Sort by inserting 11 random elements in random order and sorting it.

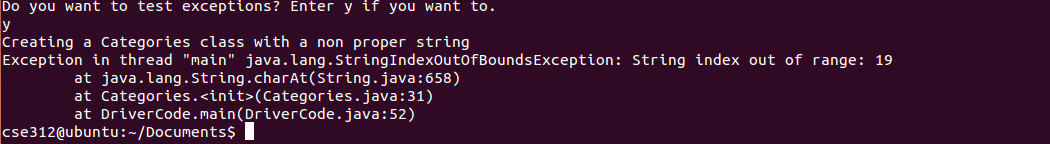
4-) Creating a Categories class with a proper string.



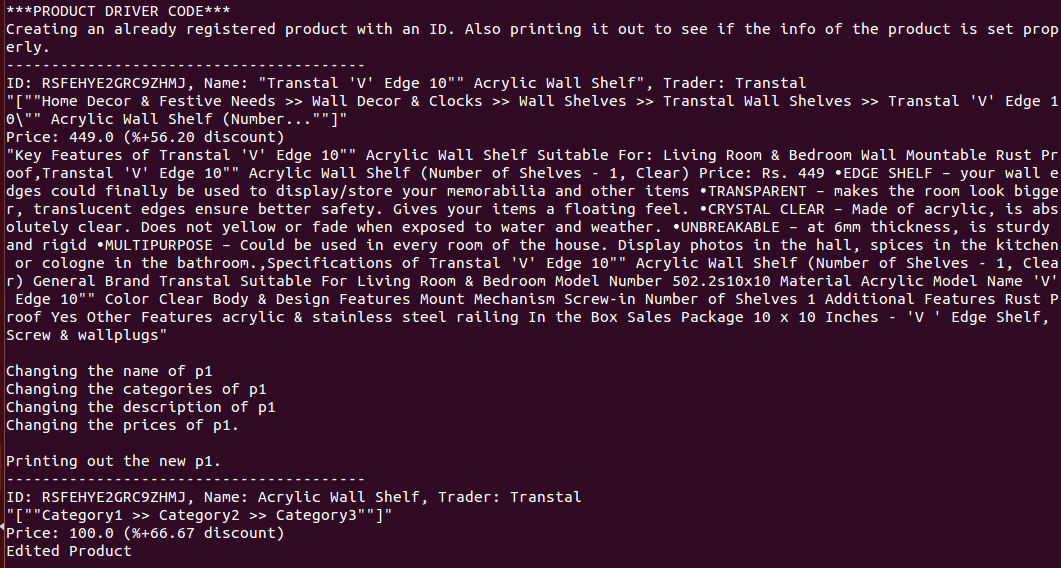
5-) Creating a Categories class with a ArrayList



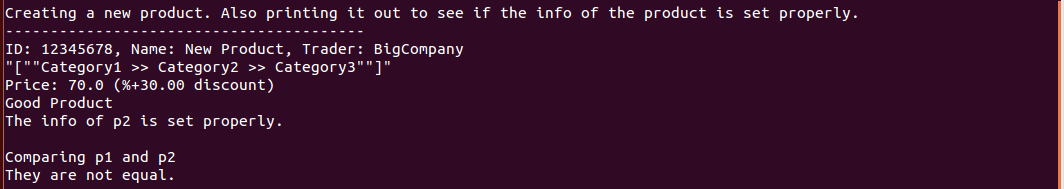
6-) Pressing y to test the exception and trying to create a Categories class with improper string.



7-) Not pressing y to skip the exceptions and creating an already registered product. Then changing the product’s information.



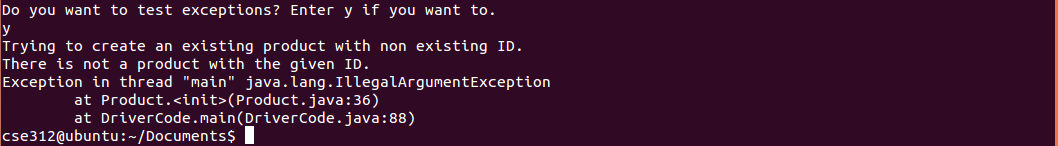
8-) Creating a new product, adding to the stock and comparing the products.



9-) Removing the created product from the stock.



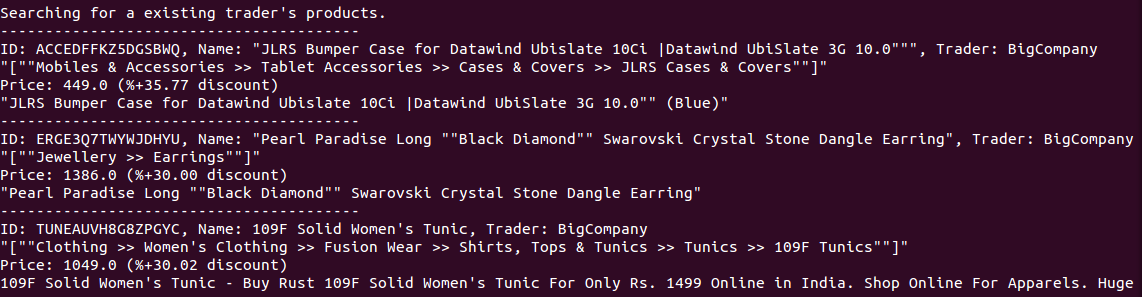
10-) Pressing y to test the exception and trying to create an existing product with non existing ID.



11-) Pressing n to skip the exceptions and Creating an already existing customer.



12-) Searching for products of an existing trader.



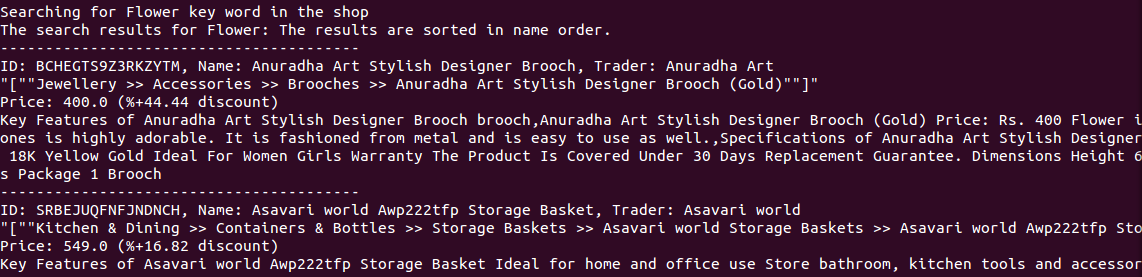
13-) Searching for products of a non existing trader.



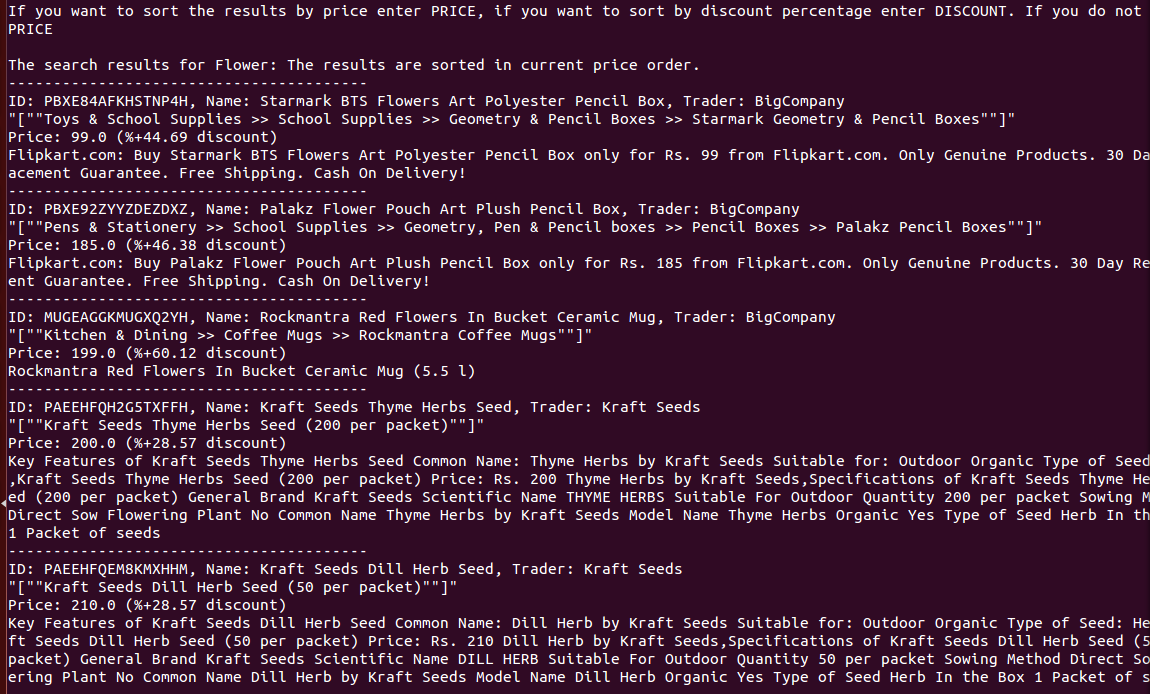
14-) Ordering a product with a proper ID.



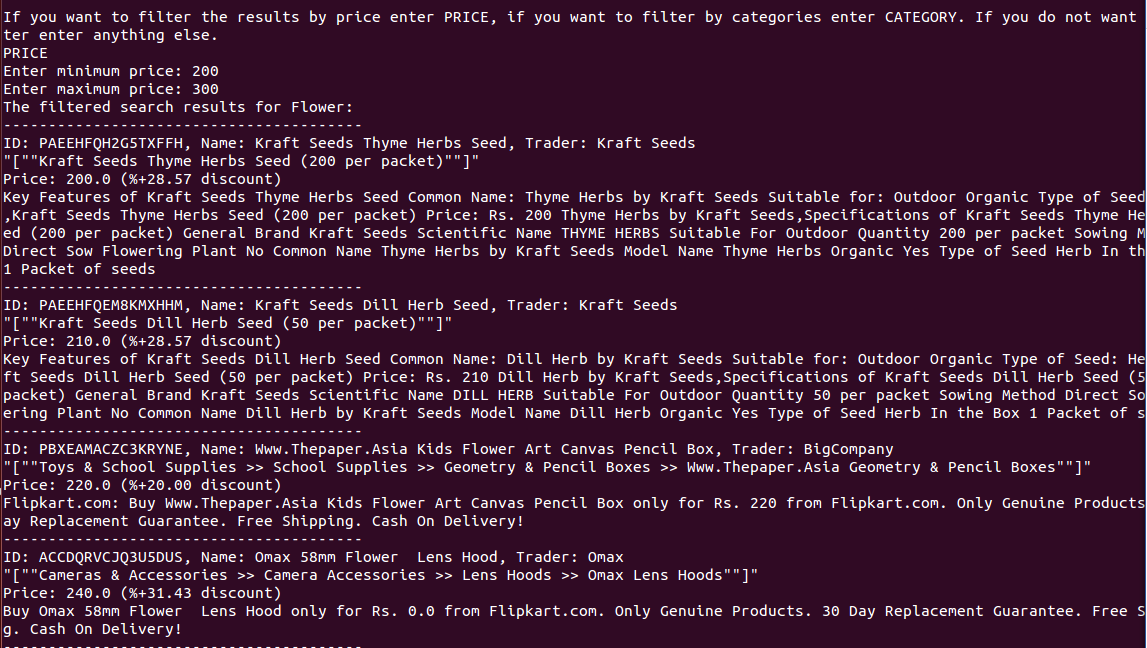
15-) Searching for a valid text in the shop.



16-) Sorting the results by price by entering PRICE.



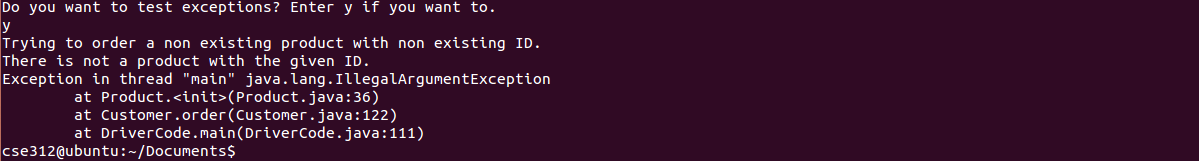
17-) Filter the results by price (200-300) by entering PRICE.



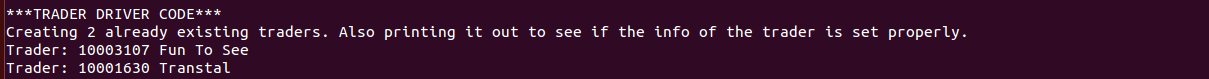
18-) Searhing for an invalid text in the shop.



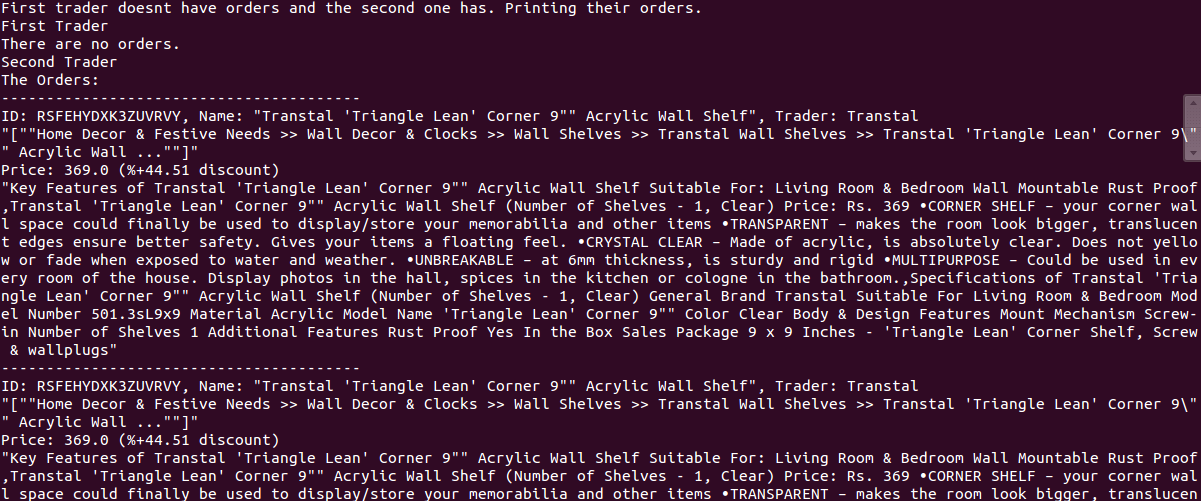
19-) Pressing y to see exception and trying to order a non existing product.



20-) Not pressing y to skip exception and creating 2 existing traders.



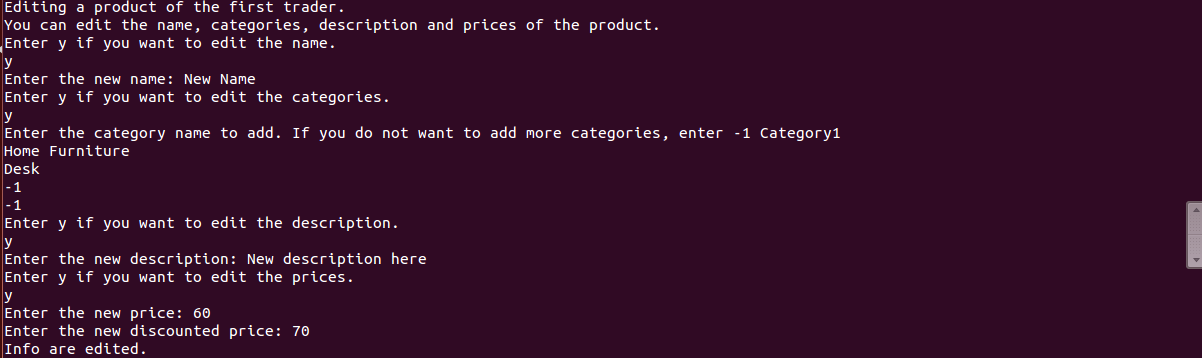
21-) Listing the trader’s orders.



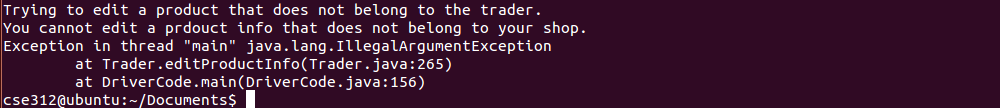
22-) Meeting one and cancelling one order of both traders.



23-) Editing the name, categories, description and prices of a product.

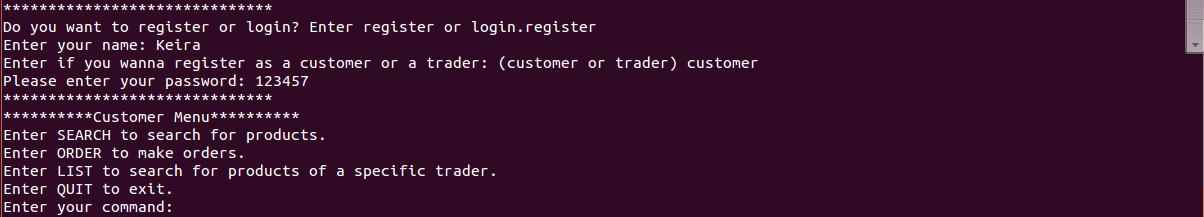


24-) Pressing y to test the exception and trying to edit a product that does not belong to the trader.



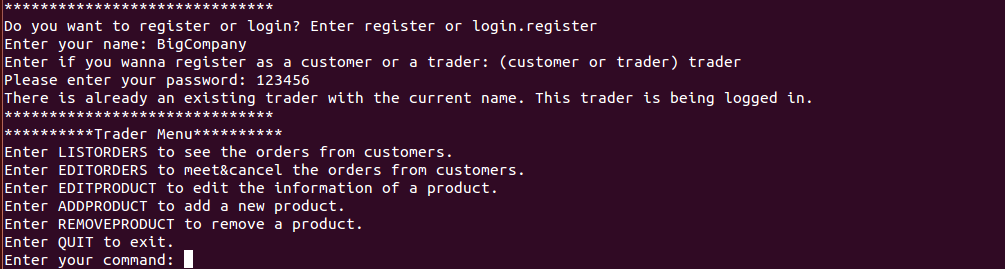
1-) Register as a new customer with wrong password info then fixing it.

Name: Keira Password: 123457



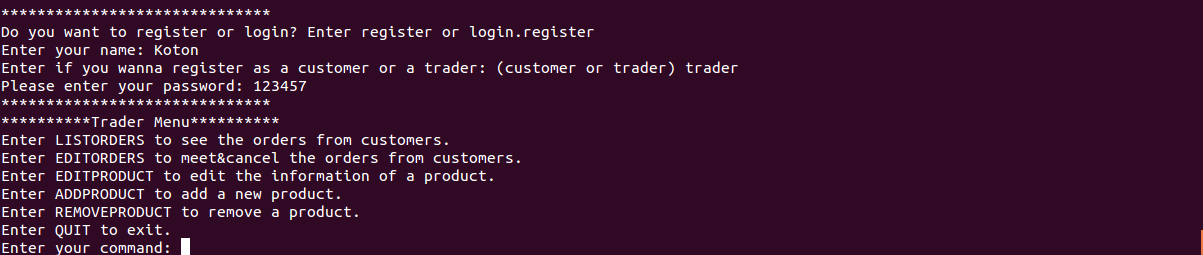
2-) Register as a new trader with already existing trader name.

Name: BigCompany

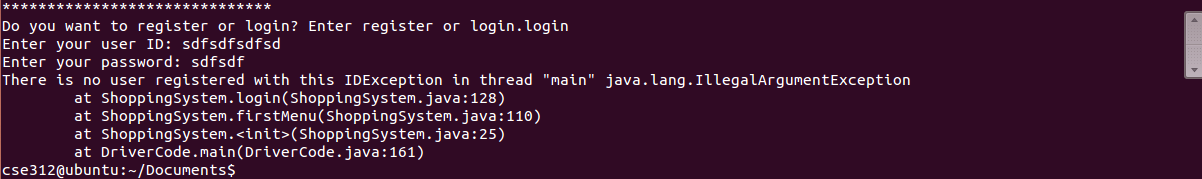


3-) Register as a new trader with a new name.

Name: Koton Password: 123457

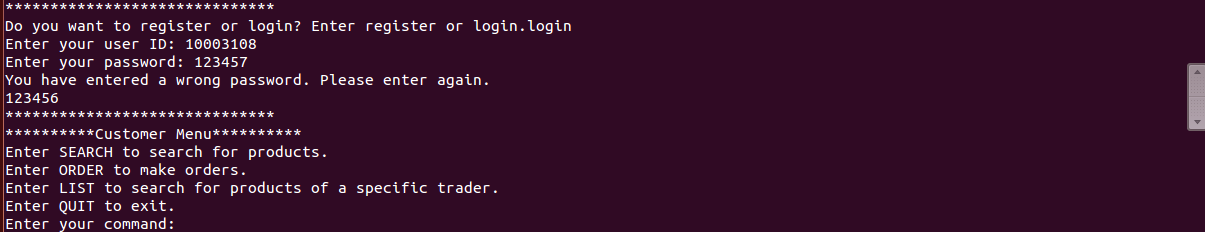


4-) Login as a customer with wrong ID.



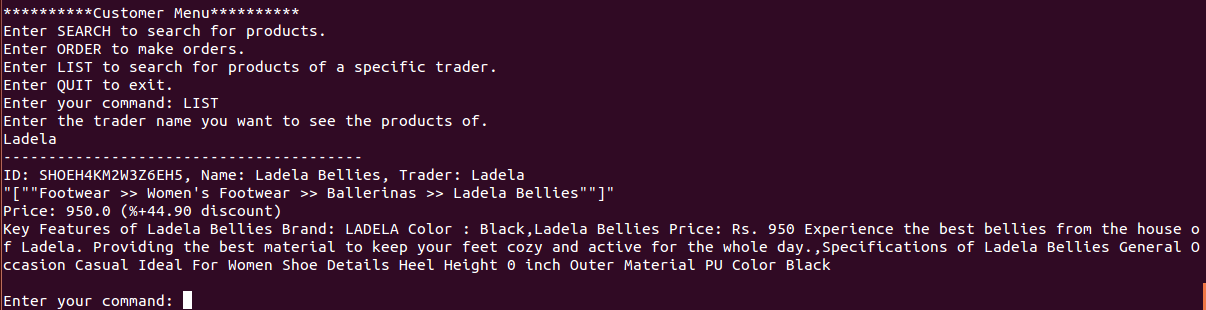
5-) Login as a customer with wrong password then fixing it.

ID: 10003108 Wrong Password:123457 Real Password: 123456



6-) List an existing trader’s products as customer.

Trader: Ladela



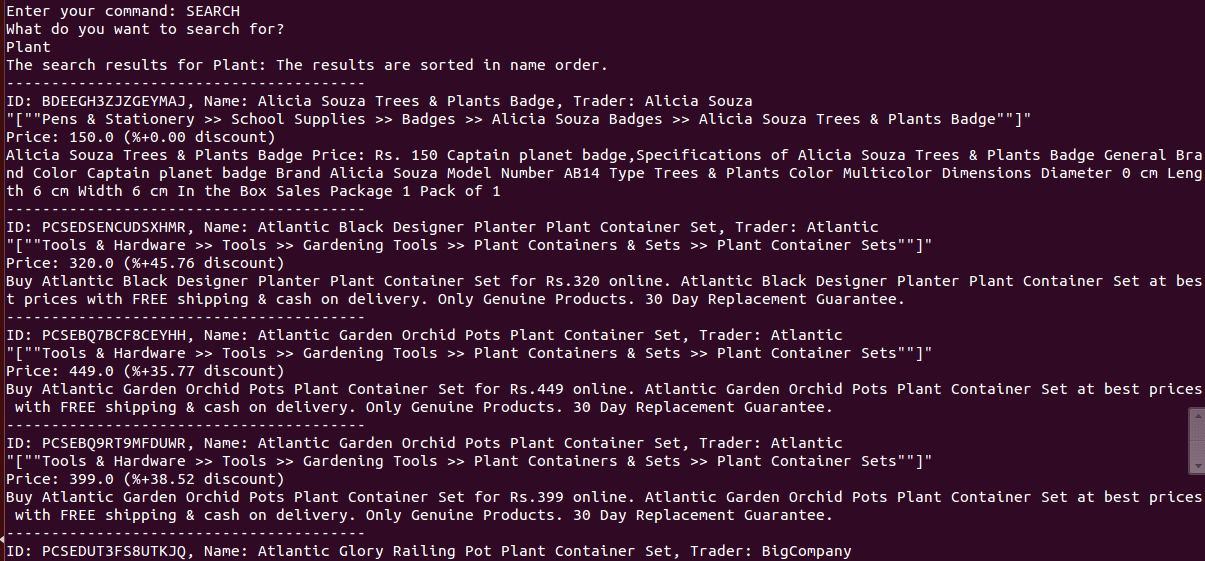
7-) List a non existing trader’s products as customer.

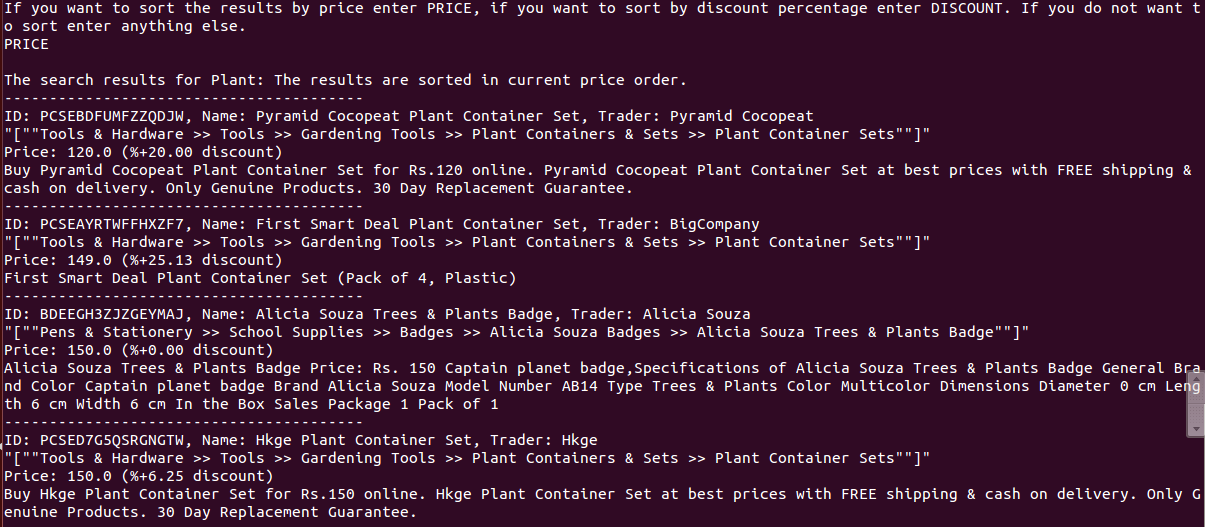
Trader: BigCompany2

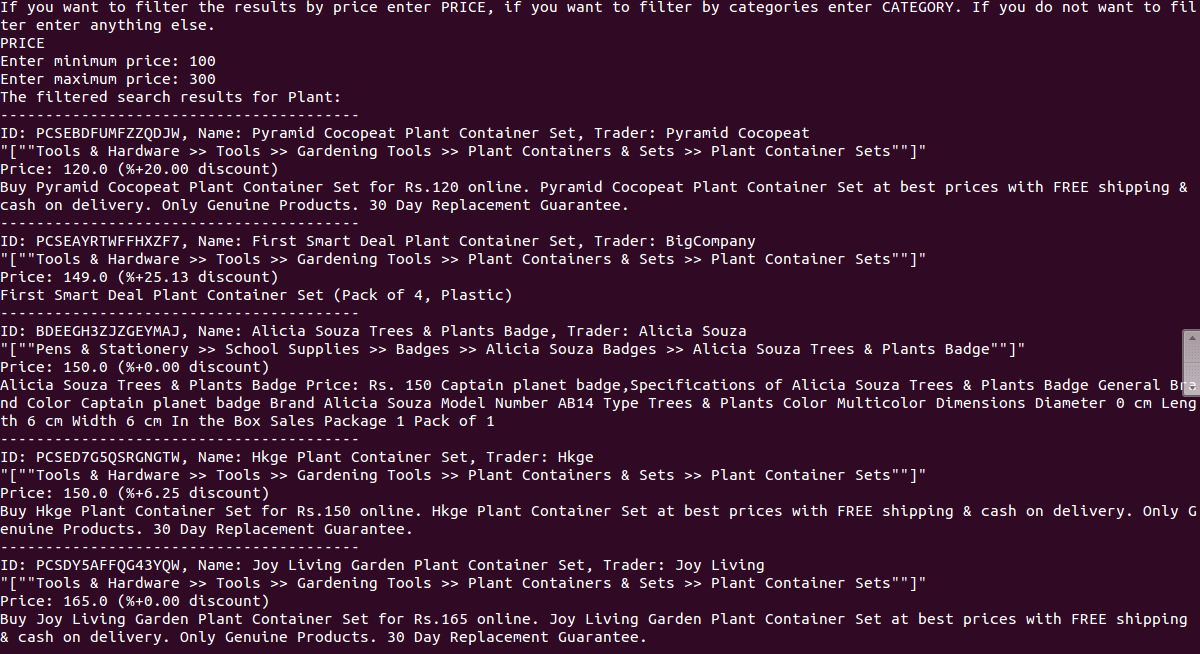


8-) Search products with a valid text as a customer. Then sort it by price and filter with prices.

Text: Plant

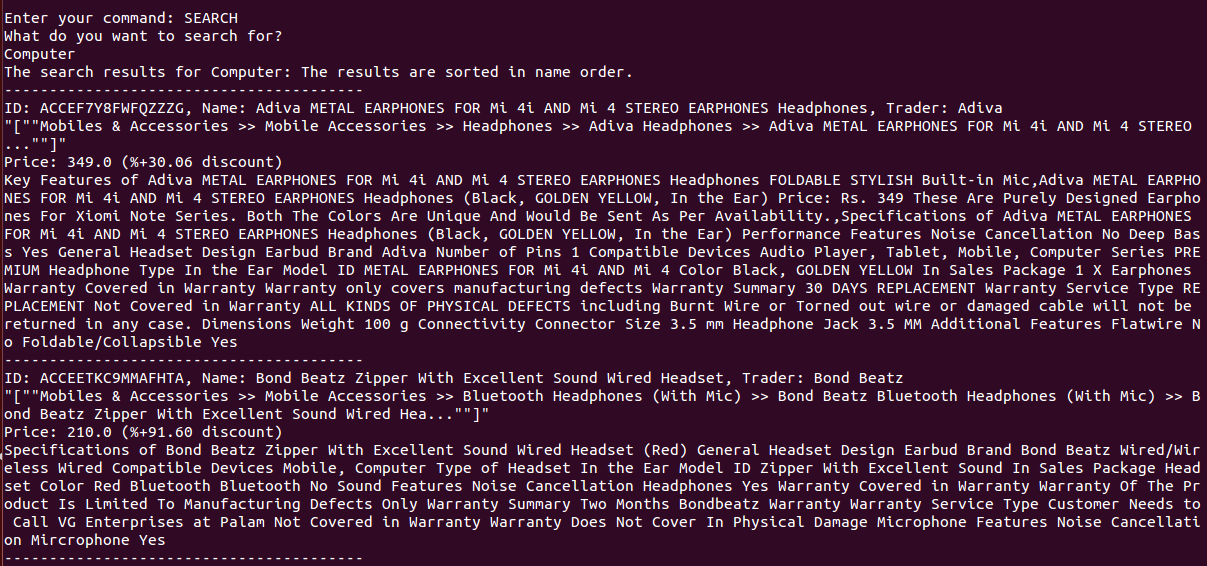


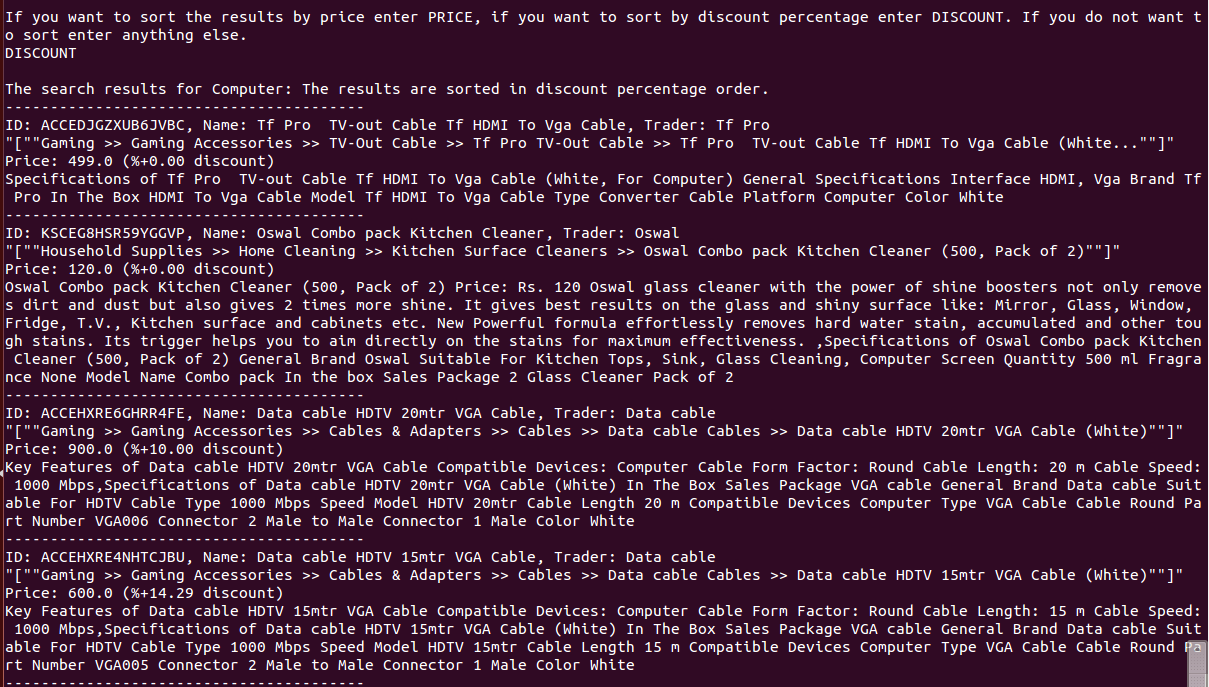


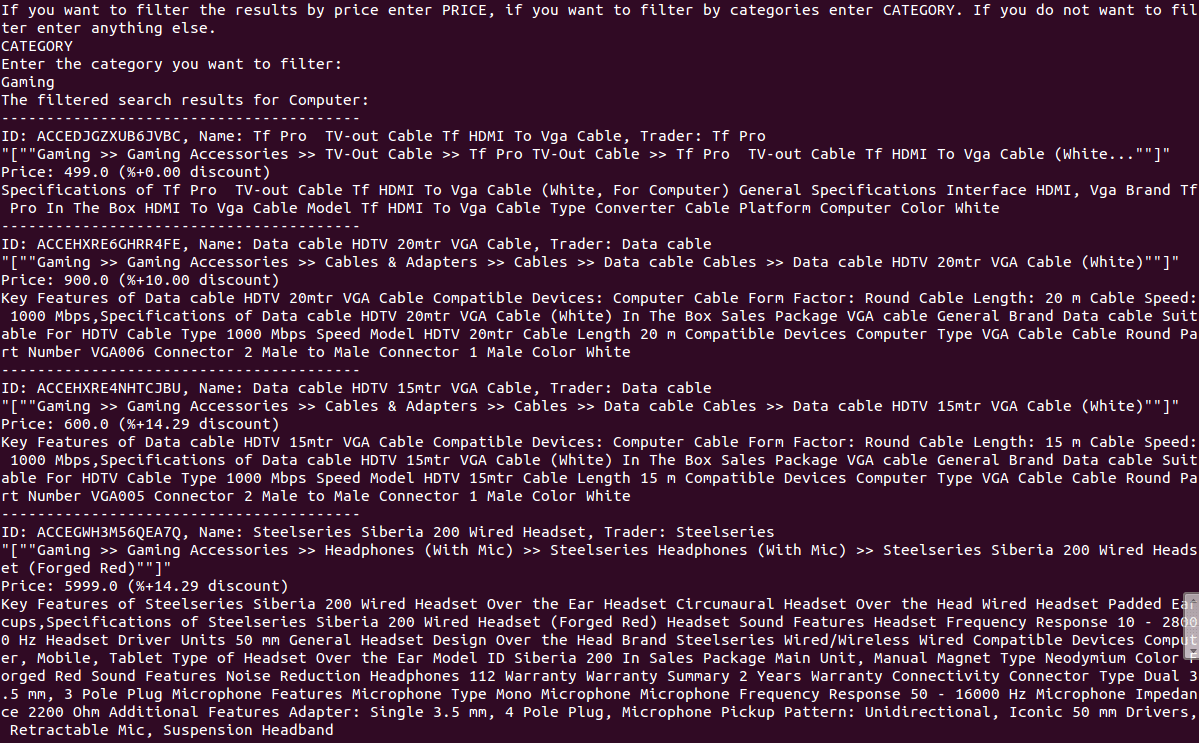


9-) Search products with a valid text as a customer. Then sort it by discount and filter with categories.

Text: Computer Category: Gaming







10-) Search products with invalid text as a customer.

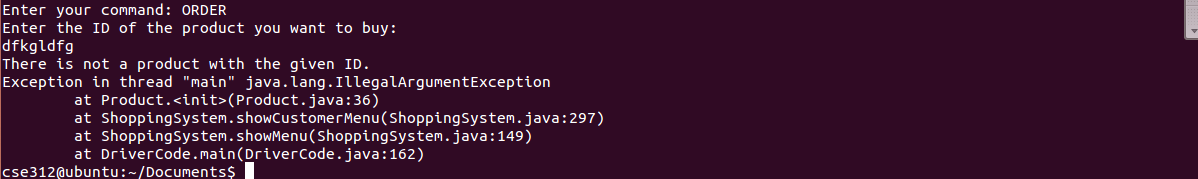


11-) Order a product that exists in the shop.

ID: TSHEJCUYWTCP4QY2

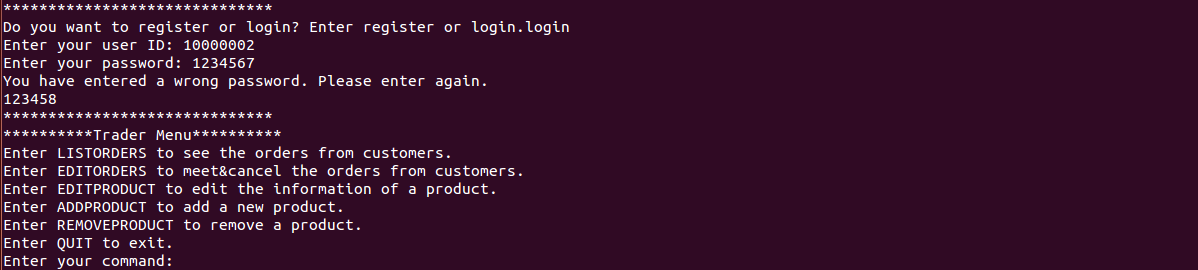


12-) Order a product that does not exist in the shop.

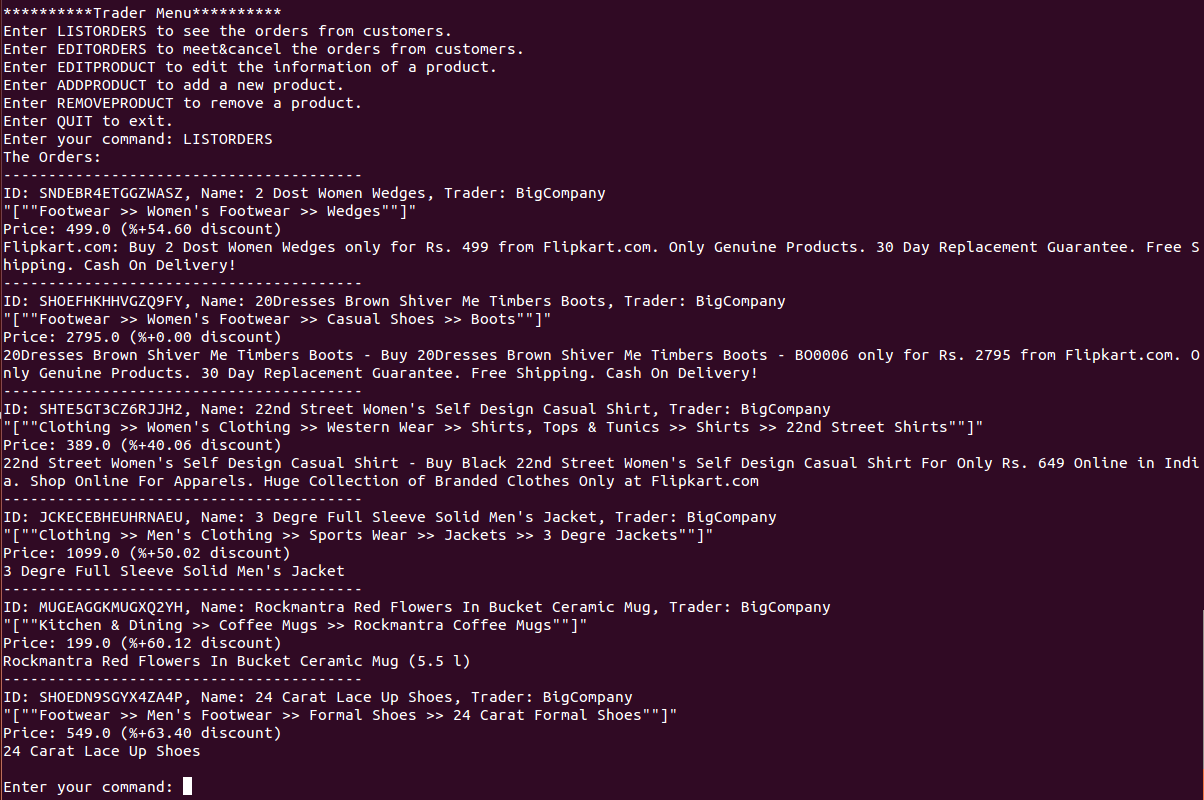


13-) Login as a trader with wrong password then fixing it.

ID: 10000002 Wrong Password: 1234567 Real Password: 123458

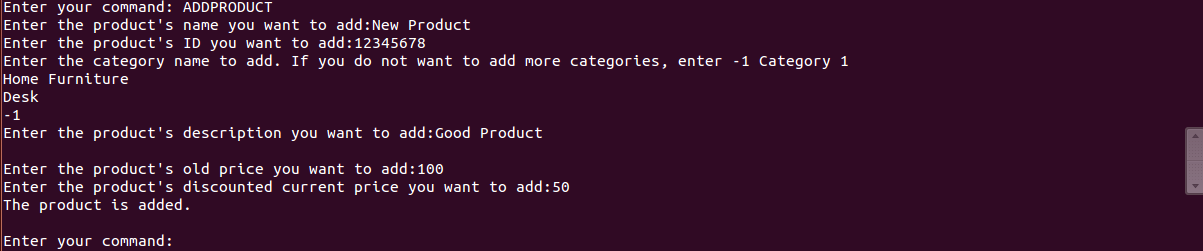


14-) List your orders as a trader.



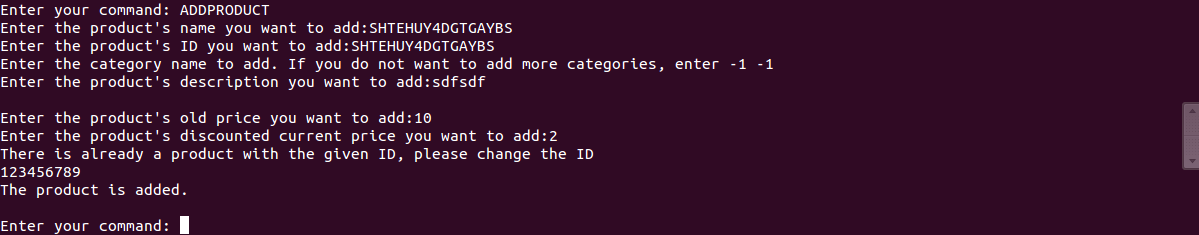
15-) Add a new product to the stock with a proper ID.

ID: 12345678



16-) Add a new product to the stock with wrong ID.

ID: SHTEHUY4DGTGAYBS

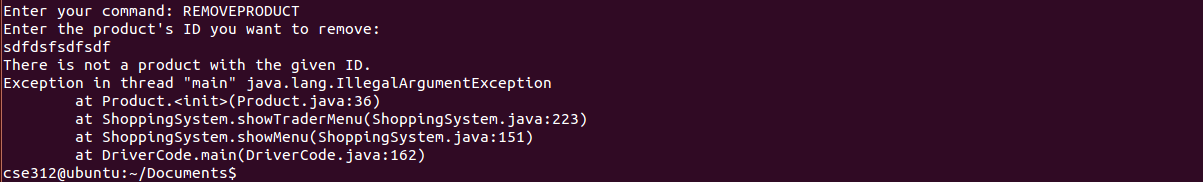


17-) Remove an existing product from stock.

ID: 12345678

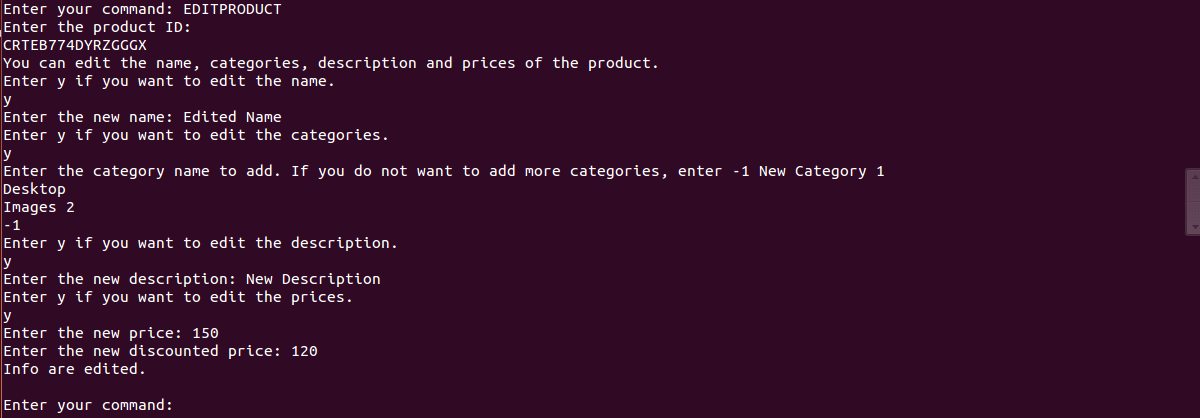


18-) Remove a non existing product from stock.



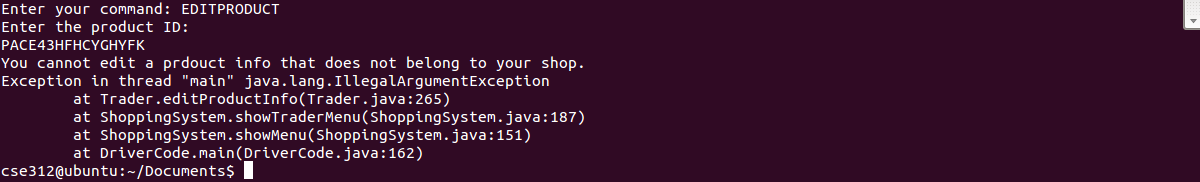
19-) Edit name, categories, description and price of a product.

ID: CRTEB774DYRZGGGX



20-) Trying to Edit name, categories, description and price of a product that does not belong to the trader.

ID: PACE43HFHCYGHYFK



21-) Meet one order.



22-) Cancel one order.



23-) Meet all orders.



24-) Cancel all orders.



25-) List your orders as a trader again.

