GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT038-3-2-OODJ

OBJECT ORIENTED PROGRAMMING WITH JAVA

APD2F2305CS(DA), APU2F2305CS(DA), APD2F2305CS(CYB), APU2F2305CS(CYB)

HAND OUT DATE: 1 JULY 2023

HAND IN DATE: 10 SEPTEMBER 2023

WEIGHTAGE: 50%

INSTRUCTIONS TO CANDIDATES:

1 **Submit your assignment at the administrative counter.**

2 **Students are advised to underpin their answers with the use of references (cited using the Harvard Name System of Referencing)**

3 **Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.**

4 **Cases of plagiarism will be penalized.**

5 **The assignment should be bound in an appropriate style (comb bound or stapled).**

6 **Where the assignment should be submitted in both hardcopy and softcopy, the softcopy of the written assignment and source code (where appropriate) should be on a CD in an envelope / CD cover and attached to the hardcopy.**

7 **You must obtain 50% overall to pass this module.**

NAME	STUDENT ID
Muhammad Nabil Hakim Bin Yusaidi	TP066763
Gassar Haithm Saleh Ba Hashwan	TP068741
Wan Ahmad Fahim Munir Bin Wan Zaki	TP075144

Table of Contents

INTRODUCTION 3
USE CASE DIAGRAM 4
  Use Case Diagram Description 4
CLASS DIAGRAM 15
I/O OF THE SYSTEM 17
  Sales Manager 17
  Purchase Manager 29
  Administrator 36
OBJECT ORIENTED CONCEPT 43
  Class 43
  Object 48
  Inheritance 49
  Polymorphism 52
    Overriding 52
    Overloading 53
  Encapsulation 55
  Abstraction 56
SYSTEM LIMITATIONS 57
CONCLUSION 58
REFERENCES 59

INTRODUCTION

In the realm of business, operational efficiency is paramount to success. SIGMA SDN BHD (SSB), a wholesaler in Johor Bharu, Malaysia has recognized the pressing need for streamlining their daily operations. Specifically, their objective is to introduce automation to their Purchase Order (PO) process to bolster their purchasing efficiency. To meet this objective, this project aims to implement a Java-based system rooted in Object-Oriented Programming principles. The system's primary goal is to streamline and optimize the Purchase Order workflow within SSB. Leveraging key OOP concepts such as classes, objects, inheritance, polymorphism, encapsulation, and abstraction, the system will effectively model various aspects of the Purchase Order process. To fulfil the objectives, this application will encapsulate a range of essential functionalities with each aligned with specific user requirements.

Three roles are identified with each playing a pivotal part in the Purchase Order process: Sales Manager, Purchase Manager and Administrator. Sales Manager are responsible for initiating Purchase Requisitions (PR). These requisitions specify the required items, quantities, and the required dates. Subsequently, the Purchase Manager then takes over the responsibility for processing the received Purchase Requisitions. Their role culminates in the generation of authorized Purchase Orders (PO). The administrator has the highest authority with exclusive rights and responsibilities such as full access to all application functionalities and data. The administrator has the authority to create and manager all the three types of users mentioned in the Purchase Order Management System (POM).

The expectation for this project is to enhance operational efficiency which can be achieved by the automation of the Purchase Order process. This can reduce manual errors and increase precision in catering to the demands of retailers. Moreover, the project aspires to elevate customer satisfaction level thus making it possible for SSB to deliver orders more promptly and accurately. Lastly, the OOP-based system will be designed with scalability in mind, ensuring it can readily adapt to SSB's future growth and evolving business requirements.

In summary, this ambitious project seeks to harness Object-Oriented Programming concepts in Java to redefine the Purchase Order process within SSB. By optimizing operations, improving efficiency, and enhancing customer satisfaction, it aspires to position SSB at the forefront of the wholesale distribution market while laying the foundation for sustainable growth and success.

USE CASE DIAGRAM

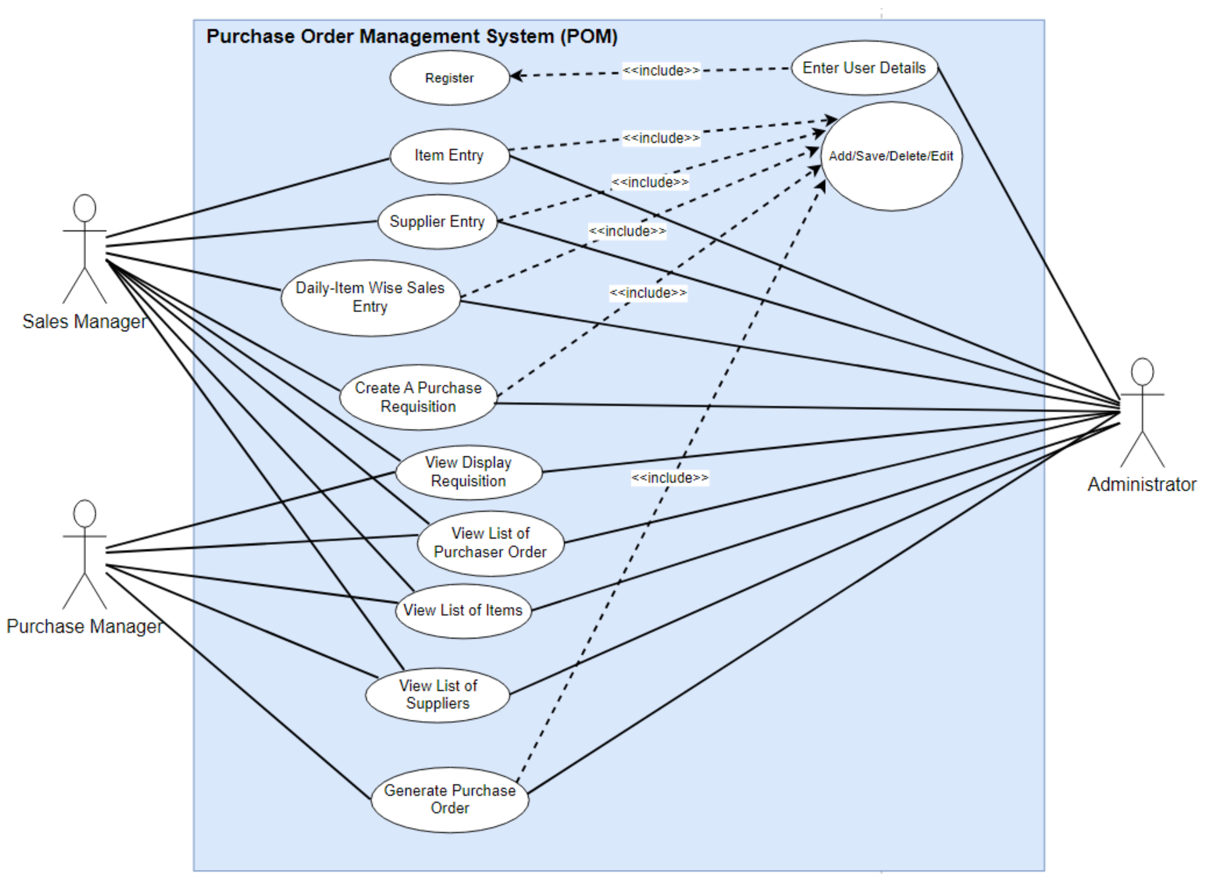


Figure 1.0 Use Case Diagram

Use Case Diagram Description

Use Case	Register
Brief Description	This use case is used by administrator to register new users into the system by entering necessary credentials.
Actors	Administrator
Precondition	Administrator must enter user's credentials
Main Flow	Administrator will first enter user's role, name, contact number, address and password.
Alternative Flow	-

Use Case	Add Item Entry
Brief Description	Sales Manager and Administrator can add new items
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to enter their User ID ii) Enter password after entering User ID iii) User will be redirected to Main Menu iv) For Sales Manager select "Item entries" then "Add Item". For Administrator, select "Sales Manager Menu" first before proceeding.
Alternative Flow	If user ID and password are incorrect, users shall re-enter their valid user ID and password.

Use Case	Save Item Entry
Brief Description	The newly added data will be saved into a text file.
Actors	Sales Manager and Administrator
Precondition	Sales Manager or Administrator should add new items.
Main Flow	i) Once all item detail is filled, then the process of adding new item is complete and saved in text file.
Alternative Flow	-

Use Case	Delete Item Entry
Brief Description	Sales Manager and Administrator can delete existing item from the text file.
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to select the line number that they wish to delete. ii) Item will be deleted from text file.
Alternative Flow	-

Use Case	Edit Item Entry
Brief Description	Sales Manager and Administrator can edit the existing item from the text file.
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to select the line number that they wish to edit. ii) Select which column they wish to edit. iii) Enter new data based on selected column and new data is updated into text file.
Alternative Flow	-

Use Case	Add Supplier Entry
Brief Description	Sales Manager and Administrator can add new supplier.
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login
Main Flow	v) User is required to enter their User ID vi) Enter password after entering User ID vii) User will be redirected to Main Menu

	i) For Sales Manager select “Suppliers entries” then “Add Supplier”. For Administrator, select “Sales Manager Menu” first before proceeding.
Alternative Flow	-

Use Case	Save Supplier Entry
Brief Description	The newly added data will be saved into a text file.
Actors	Sales Manager and Administrator
Precondition	Sales Manager or Administrator should add new supplier.
Main Flow	Once all supplier detail is filled, then the process of adding new item is complete and saved into text file.
Alternative Flow	-

Use Case	Delete Supplier Entry
Brief Description	Sales Manager and Administrator can delete existing supplier from the text file.
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to select the line number that they wish to delete. ii) Supplier will be deleted from text file.
Alternative Flow	-

Use Case	Edit Supplier Entry
Brief Description	Sales Manager and Administrator can edit the existing supplier from the text file.
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login

Main Flow	i) User is required to select the line number that they wish to edit. ii) Select which column they wish to edit. iii) Enter new data based on selected column and new data is updated into text file.
Alternative Flow	-

Use Case	Add Daily-Item Wise Sales Entry
Brief Description	Sales Manager and Administrator can add new daily-item wise sales.
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to enter their User ID ii) Enter password after entering User ID iii) User will be redirected to Main Menu iv) For Sales Manager select “Daily Item-wise Sales Entries” then “Add Selling”. For Administrator, select “Sales Manager Menu” first before proceeding.
Alternative Flow	-

Use Case	Save Daily-Item Wise Sales Entry
Brief Description	The details of daily-item wise sales will be saved into the text file.
Actors	Sales Manager and Administrator
Precondition	Sales Manager or Administrator should add new supplier.

Main Flow	Once all daily-item wise sales detail is filled, then the process of adding new item sales is completed and saved into text file.
Alternative Flow	-

Use Case	Delete Daily-Item Wise Sales Entry
Brief Description	Sales Manager and Administrator can delete existing daily-item wise sales from the text file.
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to select the line number that they wish to delete. ii) Daily-item wise sales will be deleted from text file.
Alternative Flow	-

Use Case	Edit Daily-Item Wise Sales Entry
Brief Description	Sales Manager and Administrator can edit the existing daily-item wise sales from the text file.
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to select the line number that they wish to edit. ii) Select the column they wish to edit. iii) Enter new data based on selected column and new data is updated into text file.
Alternative Flow	-

Use Case	Add Purchase Requisition
Brief Description	Sales Manager and Administrator can add purchase requisition.
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to enter their User ID ii) Enter password after entering User ID iii) User will be redirected to Main Menu iv) For Sales Manager select "Purchase Requisition" then "Add purchase requisition". For Administrator, select "Sales Manager Menu" first before proceeding.
Alternative Flow	-

Use Case	Save Purchase Requisition
Brief Description	The details of purchase requisition will be saved into the text file.
Actors	Sales Manager and Administrator
Precondition	Sales Manager or Administrator should add new purchase requisition.
Main Flow	Once all purchase requisition sales details are filled, then the process of adding purchase requisition is completed and saved into text file.
Alternative Flow	-

Use Case	Delete Purchase Requisition
Brief Description	Sales Manager and Administrator can delete existing purchase requisition from the text file.
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login

Main Flow	i) User is required to select the line number that they wish to delete. ii) Selected line number will be deleted from text file.
Alternative Flow	-

Use Case	Edit Purchase Requisition
Brief Description	Sales Manager and Administrator can edit the existing purchase requisition from the text file.
Actors	Sales Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to select the line number that they wish to edit. ii) Select the column they wish to edit. iii) Enter new data based on selected column and new data is updated into text file.
Alternative Flow	-

Use Case	View Display Requisition
Brief Description	Sales Manager, Purchase Manager and Administrator view the requisition
Actors	Sales Manager, Purchase Manager and Administrator
Precondition	All actors are required to login into the system
Main Flow	i) User is required to enter their User ID ii) Enter password after entering User ID iii) User will be redirected to Main Menu and select "Purchase Requisition" then select "Display Requisition". iv) All requisitions will be displayed.
Alternative Flow	-

Use Case	View List of Purchaser Order
Brief Description	Sales Manager, Purchase Manager and Administrator can view the list of purchaser order from the text file.
Actors	Sales Manager, Purchase Manager and Administrator
Precondition	Sales Manager, Purchase Manager and Administrator should login into the system.
Main Flow	i) User is required to enter their User ID ii) Enter password after entering User ID iii) User will be redirected to Main Menu and select “Display Purchase Order” then select “Display Purchase Orders”.
Alternative Flow	-

Use Case	View List of Items
Brief Description	Sales Manager, Purchase Manager and Administrator can view the list of purchaser order from the text file.
Actors	Sales Manager, Purchase Manager and Administrator
Precondition	All actors are required to login
Main Flow	i) User is required to enter their User ID ii) Enter password after entering User ID iii) User will be redirected to Main Menu and select “Item Entries” then select “Display All Items”.
Alternative Flow	-

Use Case	View List of Suppliers
Brief Description	Sales Manager, Purchase Manager and Administrator can view the list of suppliers of the text file.
Actors	Sales Manager, Purchase Manager and Administrator
Precondition	All actors are required to login
Main Flow	i) User is required to enter their User ID ii) Enter password after entering User ID iii) User will be redirected to Main Menu and select "Suppliers Entries" then select "Display All Suppliers".
Alternative Flow	-

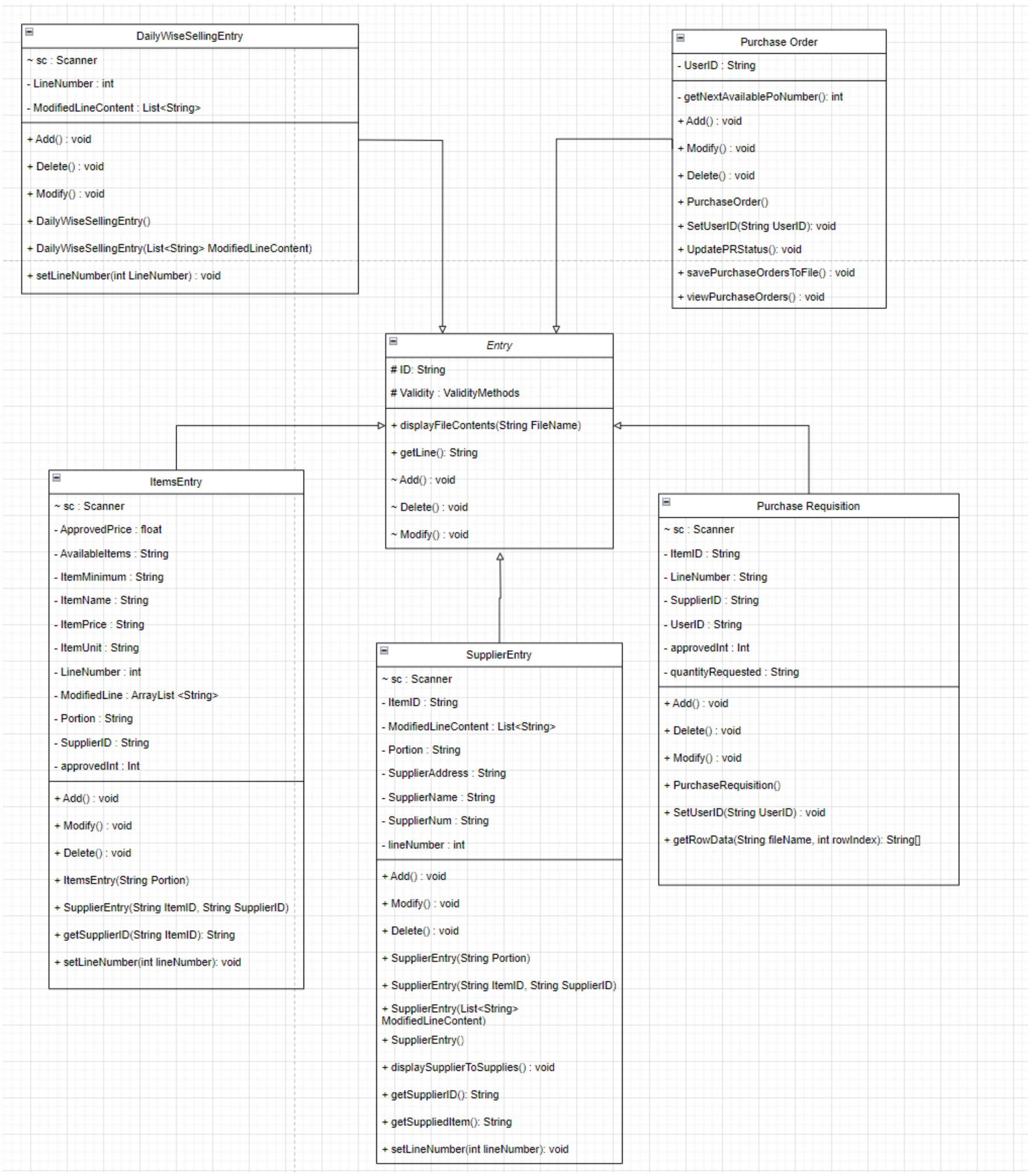
Use Case	Add Purchase Order
Brief Description	Purchase Manager and Administrator can add new purchase order
Actors	Purchase Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to enter their User ID ii) Enter password after entering User ID iii) User will be redirected to Main Menu and select "Generate Purchase Order" then select "Add Purchase Order". iv) Enter PR number to search for v) Enter all necessary information.
Alternative Flow	-

Use Case	Save Purchase Order
Brief Description	Purchase Manager and Administrator can save newly added purchase order into the text file.
Actors	Purchase Manager and Administrator
Precondition	Actors need to create a purchase order
Main Flow	Once all purchase orders details are filled, then the process of adding purchase order is completed and saved into text file.
Alternative Flow	-

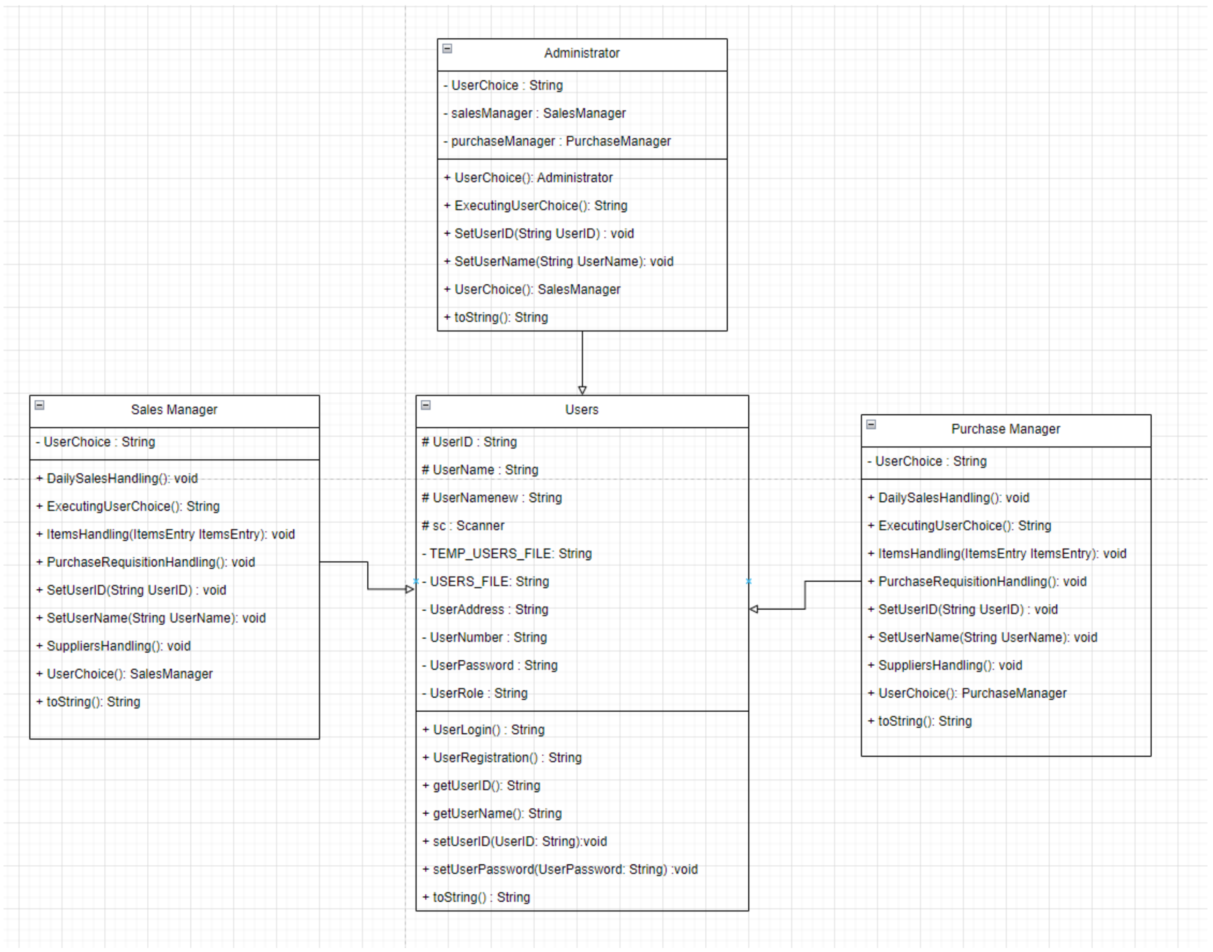
Use Case	Delete Purchase Order
Brief Description	Purchase Manager and Administrator can delete existing purchase order from the text file.
Actors	Purchase Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to enter the purchase order number that they wish to delete. ii) Selected purchase order number will be deleted from text file.
Alternative Flow	-

Use Case	Edit Purchase Order
Brief Description	Purchase Manager and Administrator can edit the existing purchase order from the text file.
Actors	Purchase Manager and Administrator
Precondition	Both actors are required to login
Main Flow	i) User is required to enter the purchase order number that they wish to edit. ii) Select the column they wish to edit. iii) Enter new data based on the prompt produced by the system and new data is updated into text file.
Alternative Flow	-

CLASS DIAGRAM


DailyWiseSellingEntry
~ sc : Scanner
- LineNumber : int
- ModifiedLineContent : List<String>
+ Add() : void
+ Delete() : void
+ Modify() : void
+ DailyWiseSellingEntry()
+ DailyWiseSellingEntry(List<String> ModifiedLineContent)
+ setLineNumber(int LineNumber) : void
Purchase Order
- UserID : String
- getNextAvailablePoNumber(): int
+ Add() : void
+ Modify() : void
+ Delete() : void
+ PurchaseOrder()
+ SetUserID(String UserID): void
+ UpdatePRStatus(): void
+ savePurchaseOrdersToFile() : void
+ viewPurchaseOrders() : void
Entry
# ID: String
# Validity : ValidityMethods
+ displayFileContents(String FileName)
+ getLine(): String
~ Add() : void
~ Delete() : void
~ Modify() : void
ItemsEntry
~ sc : Scanner
- ApprovedPrice : float
- AvailableItems : String
- ItemMinimum : String
- ItemName : String
- ItemPrice : String
- ItemUnit : String
- LineNumber : int
- ModifiedLine : ArrayList <String>
- Portion : String
- SupplierID : String
- approvedInt : Int
+ Add() : void
+ Modify() : void
+ Delete() : void
+ ItemsEntry(String Portion)
+ SupplierEntry(String ItemID, String SupplierID)
+ getSupplierID(String ItemID): String
+ setLineNumber(int lineNumber): void
SupplierEntry
~ sc : Scanner
- ItemID : String
- ModifiedLineContent : List<String>
- Portion : String
- SupplierAddress : String
- SupplierName : String
- SupplierNum : String
- lineNumber : int
+ Add() : void
+ Modify() : void
+ Delete() : void
+ SupplierEntry(String Portion)
+ SupplierEntry(String ItemID, String SupplierID)
+ SupplierEntry(List<String> ModifiedLineContent)
+ SupplierEntry()
+ displaySupplierToSupplies() : void
+ getSupplierID(): String
+ getSuppliedItem(): String
+ setLineNumber(int lineNumber): void
Purchase Requisition
~ sc : Scanner
- ItemID : String
- LineNumber : String
- SupplierID : String
- UserID : String
- approvedInt : Int
- quantityRequested : String
+ Add() : void
+ Delete() : void
+ Modify() : void
+ PurchaseRequisition()
+ SetUserID(String UserID) : void
+ getRowData(String fileName, int rowIndex): String[]

Class Diagram for Entry class and its subclasses

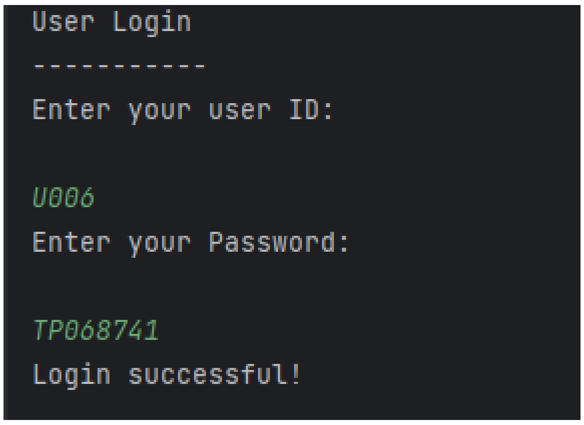

Administrator
- UserChoice : String
- salesManager : SalesManager
- purchaseManager : PurchaseManager
+ UserChoice(): Administrator
+ ExecutingUserChoice(): String
+ SetUserID(String UserID) : void
+ SetUserName(String UserName): void
+ UserChoice(): SalesManager
+ toString(): String
Sales Manager
- UserChoice : String
+ DailySalesHandling(): void
+ ExecutingUserChoice(): String
+ ItemsHandling(ItemsEntry ItemsEntry): void
+ PurchaseRequisitionHandling(): void
+ SetUserID(String UserID) : void
+ SetUserName(String UserName): void
+ SuppliersHandling(): void
+ UserChoice(): SalesManager
+ toString(): String
Users
# UserID : String
# UserName : String
# UserNamenew : String
# sc : Scanner
- TEMP_USERS_FILE: String
- USERS_FILE: String
- UserAddress : String
- UserNumber : String
- UserPassword : String
- UserRole : String
+ UserLogin() : String
+ UserRegistration() : String
+ getUserID(): String
+ getUserName(): String
+ setUserID(UserID: String):void
+ setUserPassword(UserPassword: String) :void
+ toString() : String
Purchase Manager
- UserChoice : String
+ DailySalesHandling(): void
+ ExecutingUserChoice(): String
+ ItemsHandling(ItemsEntry ItemsEntry): void
+ PurchaseRequisitionHandling(): void
+ SetUserID(String UserID) : void
+ SetUserName(String UserName): void
+ SuppliersHandling(): void
+ UserChoice(): PurchaseManager
+ toString(): String

Class diagram for Users class and its subclasses

I/O OF THE SYSTEM

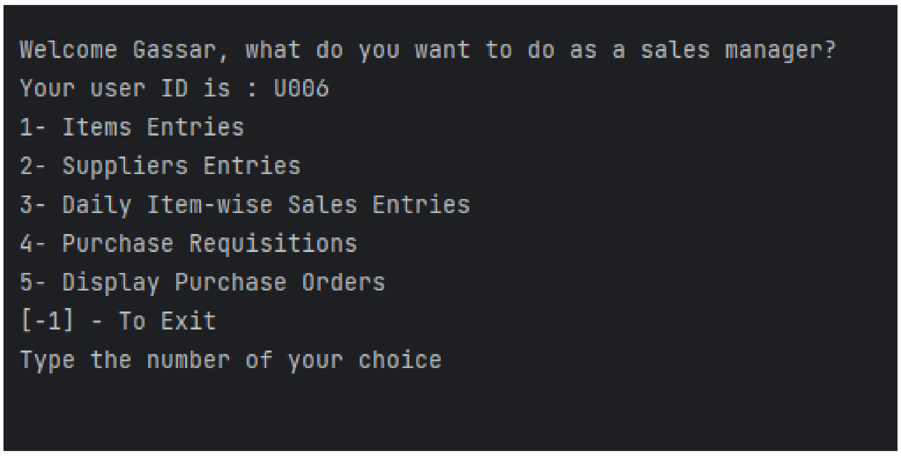
Sales Manager

After the user log in the system will dedicate his role, if the user is Sales Manager the next Menu will be displayed.



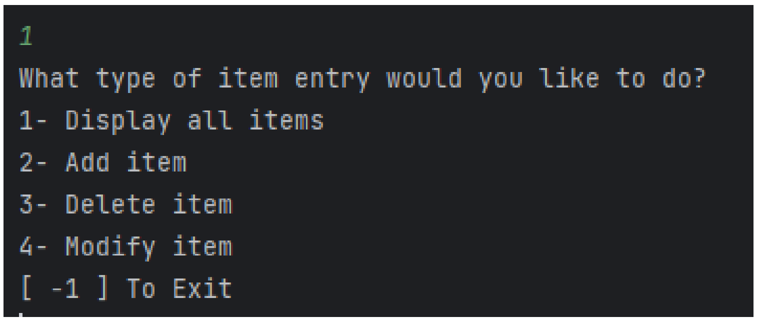
```
User Login
-----------
Enter your user ID:

U006
Enter your Password:

TP068741
Login successful!
```

Here is the main menu for all the functionalities of the Sales Manager user, the user ID and name will be shown in the top.



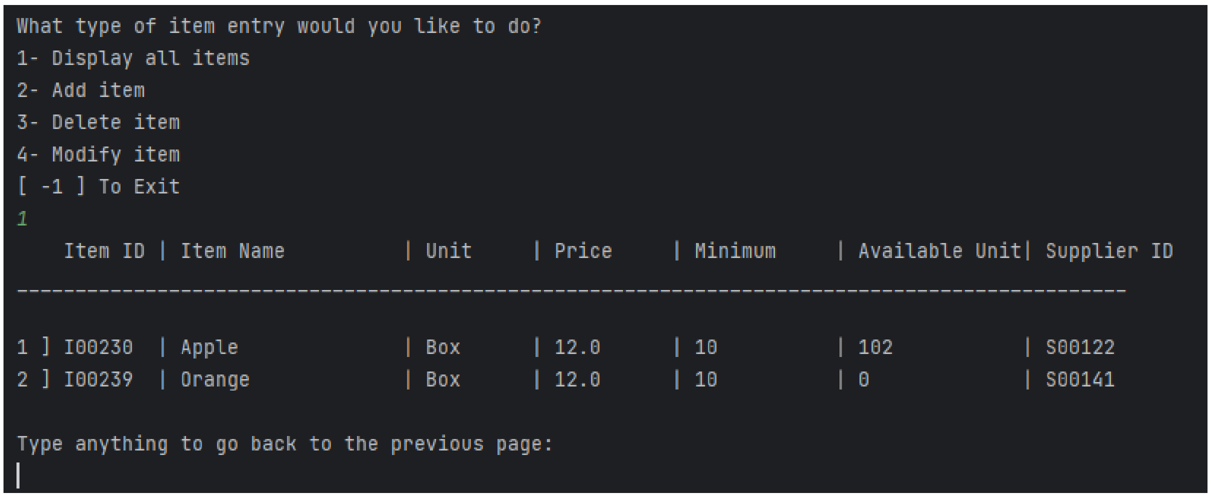
```
Welcome Gassar, what do you want to do as a sales manager?
Your user ID is : U006
1- Items Entries
2- Suppliers Entries
3- Daily Item-wise Sales Entries
4- Purchase Requisitions
5- Display Purchase Orders
[-1] - To Exit
Type the number of your choice
```

If the user chooses 1, the Item Entry page will be displayed, and the user can apply all the functionalities that is shown in the below figure:



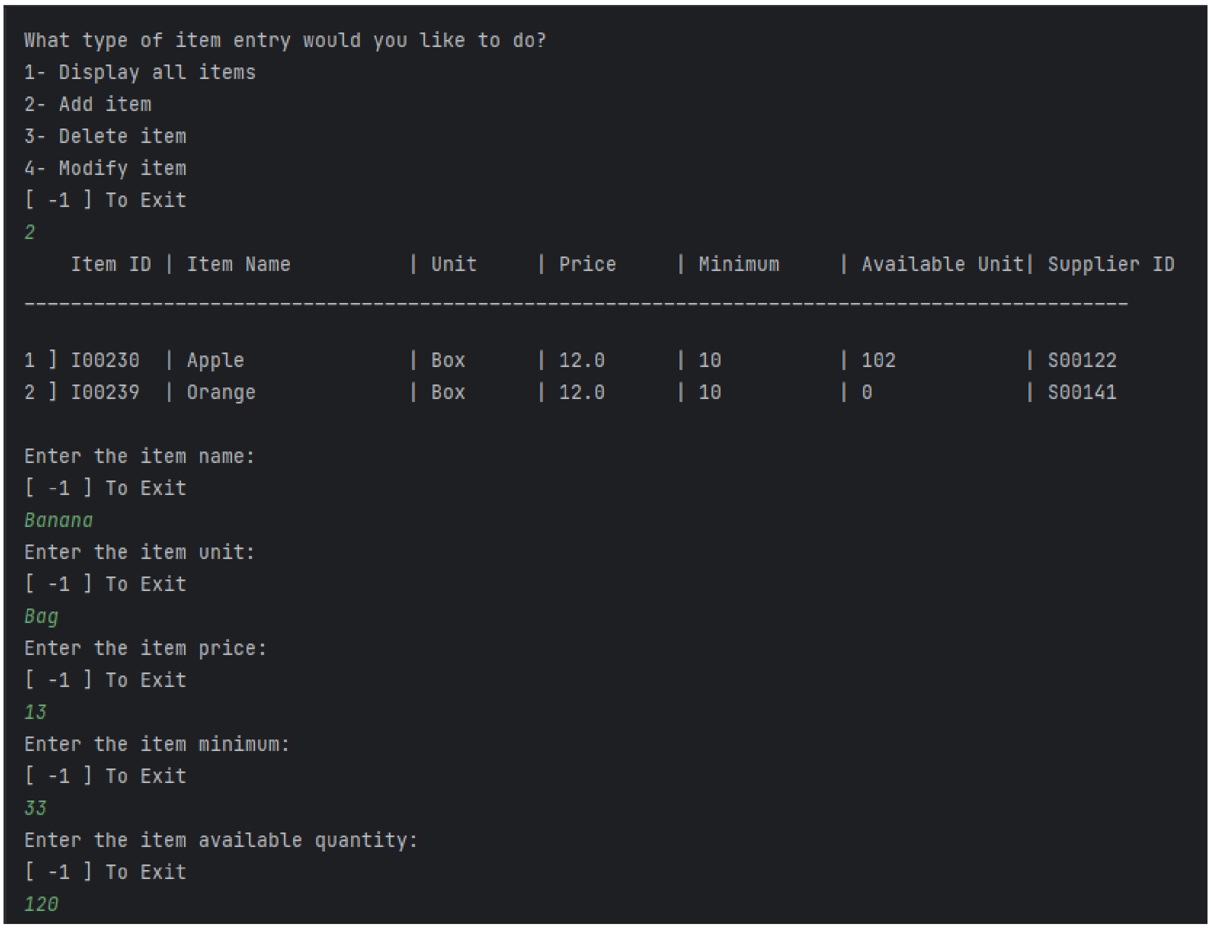
```
1
What type of item entry would you like to do?
1- Display all items
2- Add item
3- Delete item
4- Modify item
[ -1 ] To Exit
```

When the user enters 1 the Items list will be displayed, the user can go back by pressing enter.



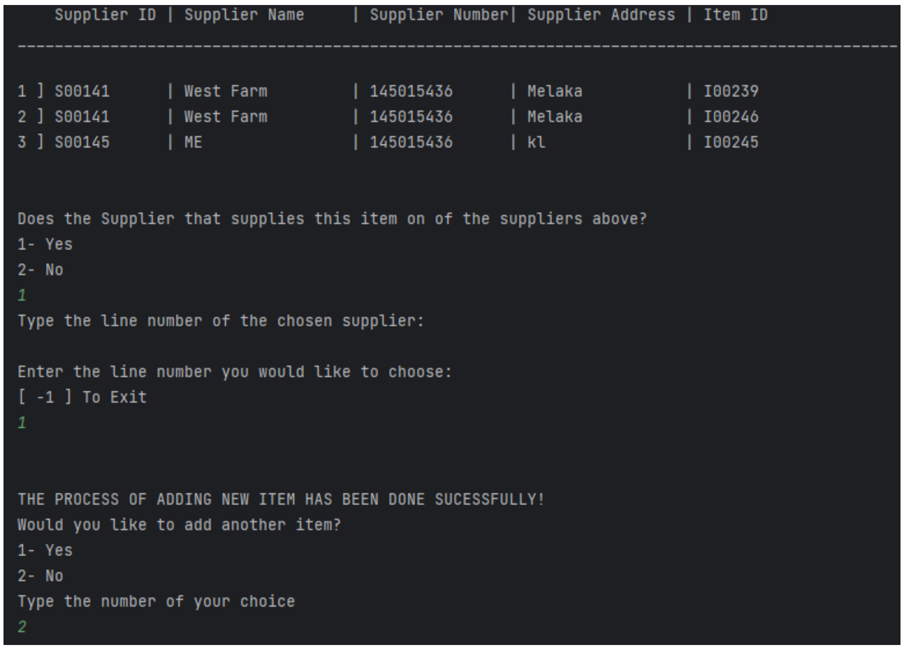
```
What type of item entry would you like to do?
1- Display all items
2- Add item
3- Delete item
4- Modify item
[ -1 ] To Exit
1
    Item ID | Item Name          | Unit     | Price      | Minimum     | Available Unit| Supplier ID
-----------------------------------------------------------------------------------------------

1 ] I00230  | Apple              | Box      | 12.0       | 10          | 102           | S00122
2 ] I00239  | Orange             | Box      | 12.0       | 10          | 0             | S00141

Type anything to go back to the previous page:
```

The second choice in the Items entry menu is to add an item, couple of questions will come to fill the item details as shown below:



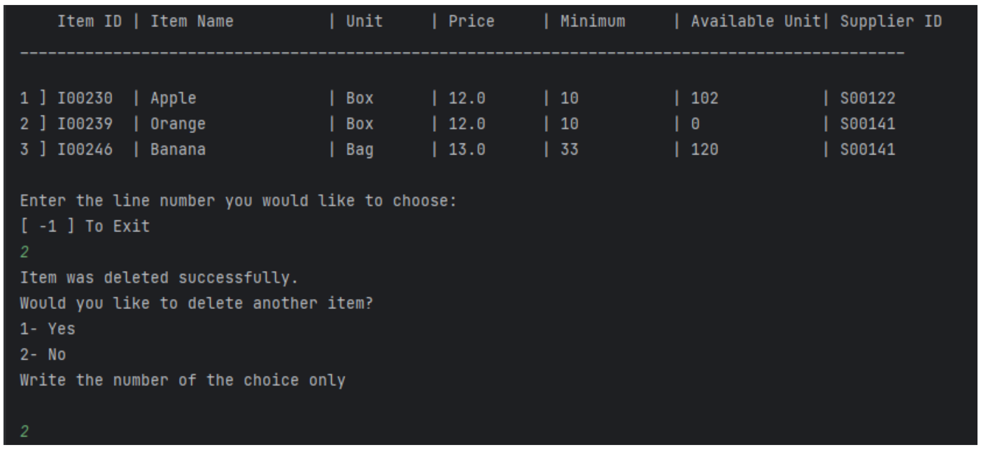
```
What type of item entry would you like to do?
1- Display all items
2- Add item
3- Delete item
4- Modify item
[ -1 ] To Exit
2
    Item ID | Item Name          | Unit     | Price      | Minimum     | Available Unit| Supplier ID
-----------------------------------------------------------------------------------------------

1 ] I00230  | Apple              | Box      | 12.0       | 10          | 102           | S00122
2 ] I00239  | Orange             | Box      | 12.0       | 10          | 0             | S00141

Enter the item name:
[ -1 ] To Exit
Banana
Enter the item unit:
[ -1 ] To Exit
Bag
Enter the item price:
[ -1 ] To Exit
13
Enter the item minimum:
[ -1 ] To Exit
33
Enter the item available quantity:
[ -1 ] To Exit
120
```

The system will ask after all the details are filled if this item is supplied already by an existing supplier or not, the suppliers list will be displayed and the user can choose either to say yes and choose the supplier line number, or to say no by writing 2 and make a new supplier (It's a must to assign a supplier for every item). Explanation for adding the Supplier will be later.



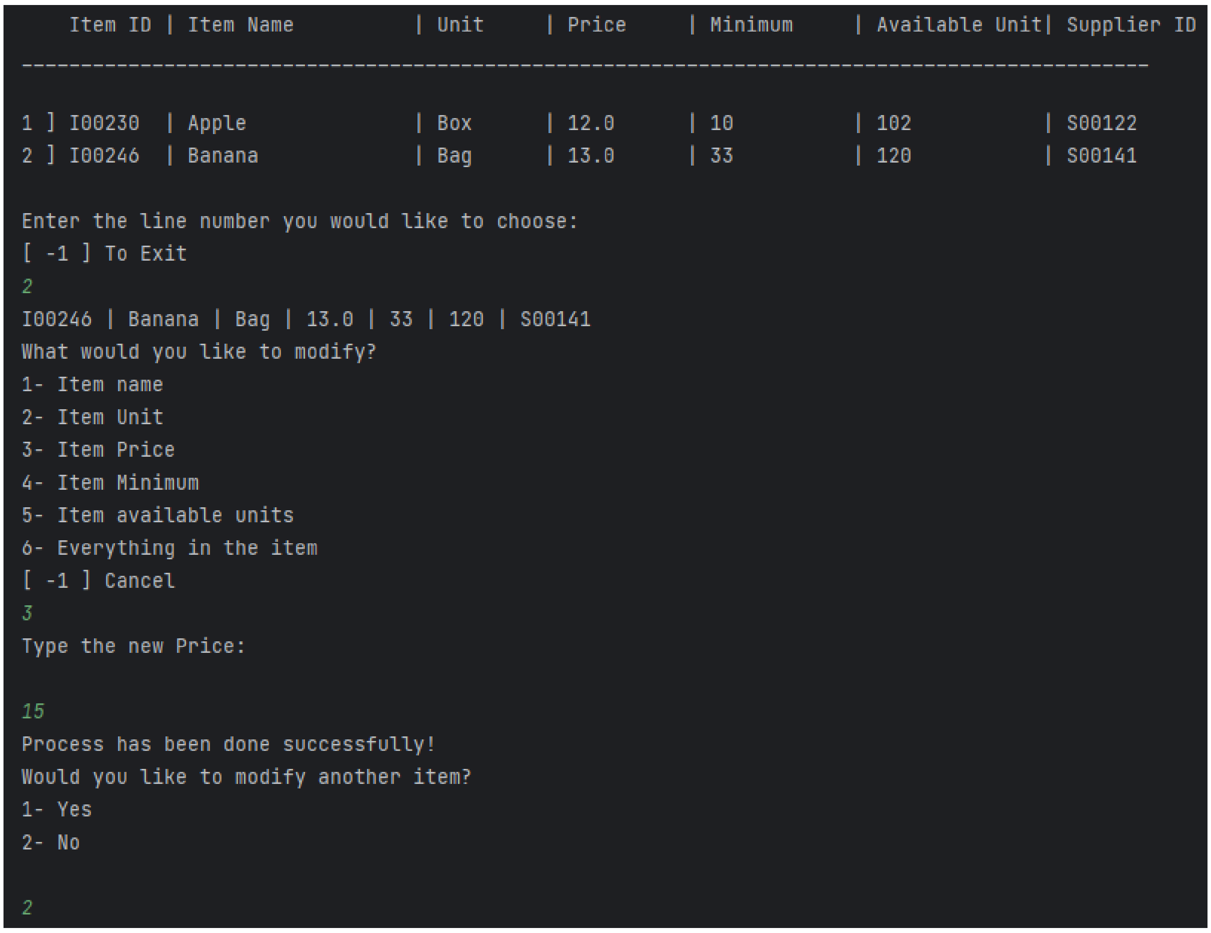
```
    Supplier ID | Supplier Name     | Supplier Number| Supplier Address | Item ID
-----------------------------------------------------------------------------------------------

1 ] S00141      | West Farm         | 145015436      | Melaka           | I00239
2 ] S00141      | West Farm         | 145015436      | Melaka           | I00246
3 ] S00145      | ME                | 145015436      | kl               | I00245


Does the Supplier that supplies this item on of the suppliers above?
1- Yes
2- No
1
Type the line number of the chosen supplier:

Enter the line number you would like to choose:
[ -1 ] To Exit
1


THE PROCESS OF ADDING NEW ITEM HAS BEEN DONE SUCESSFULLY!
Would you like to add another item?
1- Yes
2- No
Type the number of your choice
2
```

The next function in the item entry is delete, by entring 4 in the item entry page the items list will be displayed for the user to write the line of the item the user wants to delete, after that the system will ask if the user would like to delete another item.



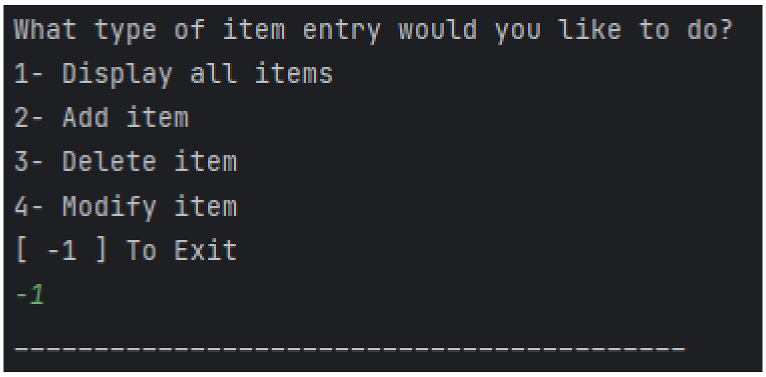
```
    Item ID | Item Name          | Unit      | Price     | Minimum      | Available Unit| Supplier ID
-----------------------------------------------------------------------------------------------

1 ] I00230  | Apple              | Box       | 12.0      | 10           | 102           | S00122
2 ] I00239  | Orange             | Box       | 12.0      | 10           | 0             | S00141
3 ] I00246  | Banana             | Bag       | 13.0      | 33           | 120           | S00141

Enter the line number you would like to choose:
[ -1 ] To Exit
2
Item was deleted successfully.
Would you like to delete another item?
1- Yes
2- No
Write the number of the choice only

2
```

Last function in the item entry is Modify, the list of items will be displayed and the user will write the line of the item he wants to modify, after that a list of options will be shown, the user can choose what option he need and write its number. For example here we want to modify the Price so we write 3 and assign the new price. The system will ask if the user want to modify another item or not.



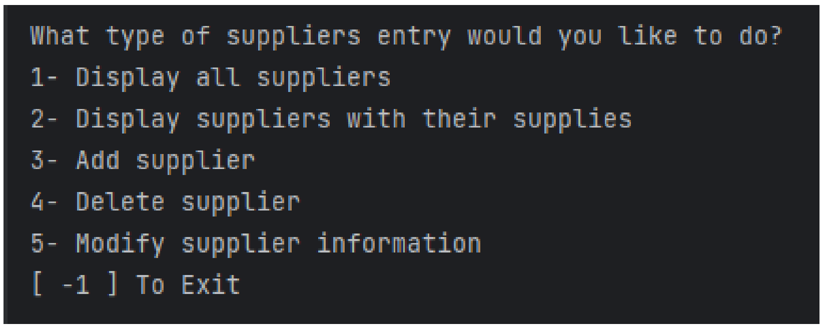
```
    Item ID | Item Name          | Unit     | Price      | Minimum      | Available Unit| Supplier ID
----------------------------------------------------------------------------------------------

1 ] I00230  | Apple              | Box      | 12.0       | 10           | 102            | S00122
2 ] I00246  | Banana             | Bag      | 13.0       | 33           | 120            | S00141

Enter the line number you would like to choose:
[ -1 ] To Exit
2
I00246 | Banana | Bag | 13.0 | 33 | 120 | S00141
What would you like to modify?
1- Item name
2- Item Unit
3- Item Price
4- Item Minimum
5- Item available units
6- Everything in the item
[ -1 ] Cancel
3
Type the new Price:

15
Process has been done successfully!
Would you like to modify another item?
1- Yes
2- No

2
```

If the user wants to go back to the main menu of the sales manager, the user must write -1.



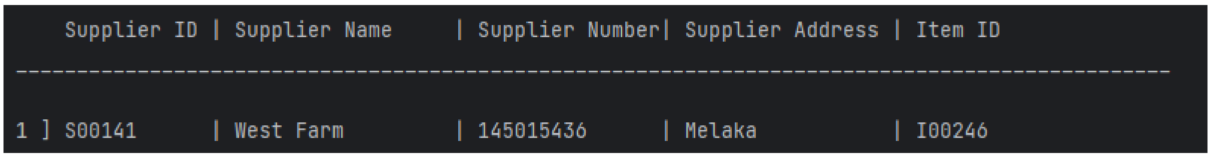
```
What type of item entry would you like to do?
1- Display all items
2- Add item
3- Delete item
4- Modify item
[ -1 ] To Exit
-1
---------------------------------------------
```

The socend page in the main menu is for Supplier Entry, this page the user will be able to manage everything about the Suppliers. To start with the first function is to display the suppliers in the system by writing 1.



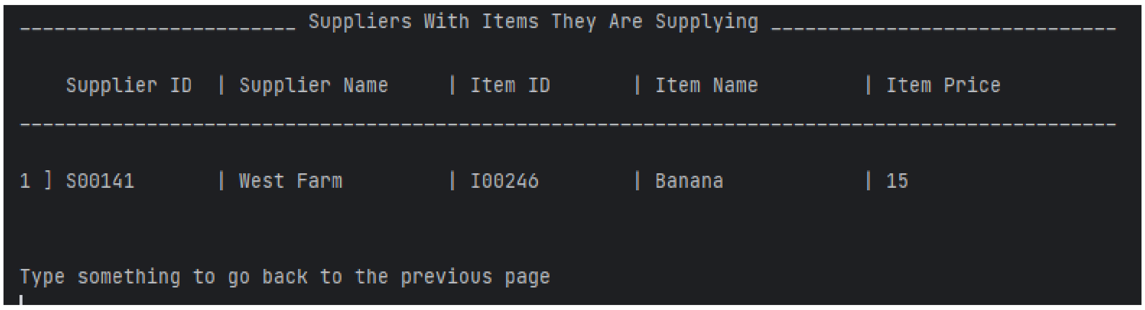
```
What type of suppliers entry would you like to do?
1- Display all suppliers
2- Display suppliers with their supplies
3- Add supplier
4- Delete supplier
5- Modify supplier information
[ -1 ] To Exit
```

This is how it will look like when the user displays the suppliers.



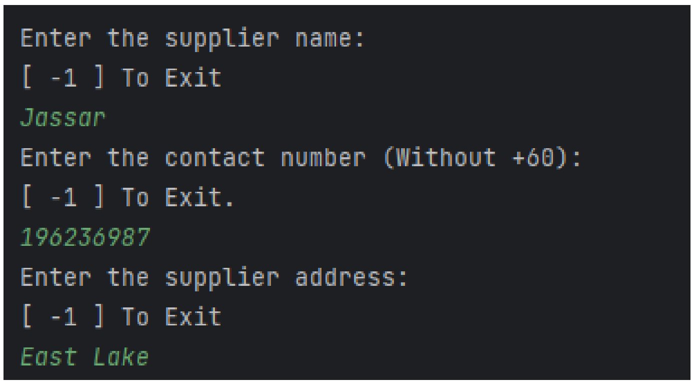
```
    Supplier ID | Supplier Name     | Supplier Number| Supplier Address | Item ID
-----------------------------------------------------------------------------------------------

1 ] S00141      | West Farm         | 145015436      | Melaka           | I00246
```

If the user chooses 2, a supplier list with the suppliers item will be display, here you can see all the suppliers with their items.



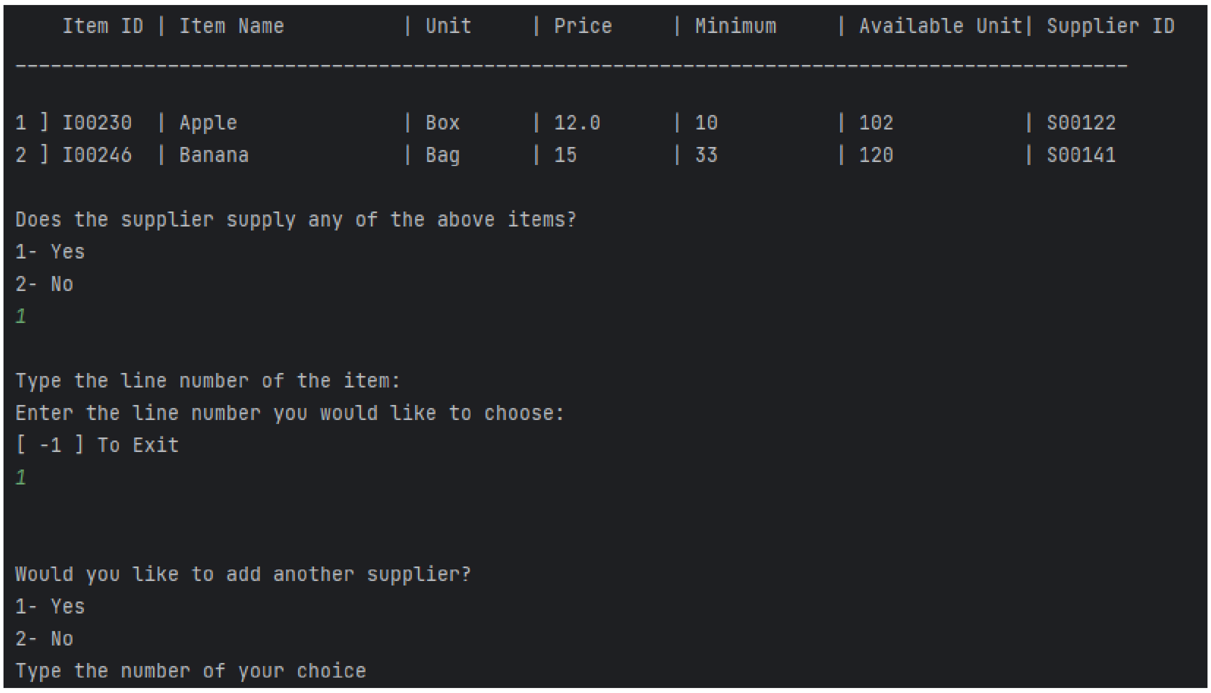
```
________________________ Suppliers With Items They Are Supplying ______________________________

    Supplier ID  | Supplier Name     | Item ID       | Item Name         | Item Price
-----------------------------------------------------------------------------------------------

1 ] S00141       | West Farm         | I00246        | Banana            | 15


Type something to go back to the previous page
```

The third option in the supplier entry page is to add a supplier, you need to fill some information to add a supplier first.



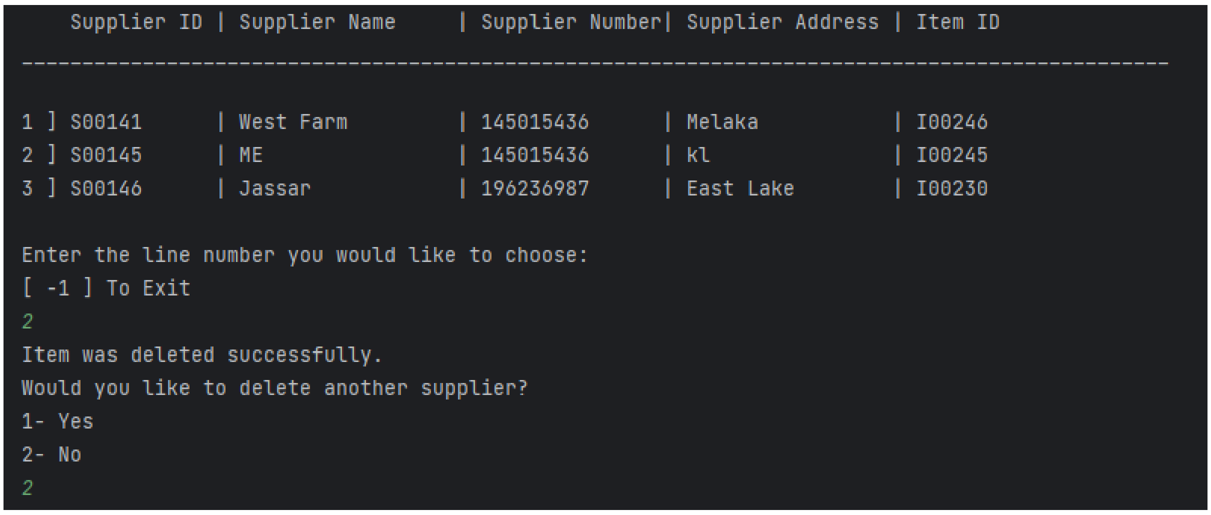
```
Enter the supplier name:
[ -1 ] To Exit
Jassar
Enter the contact number (Without +60):
[ -1 ] To Exit.
196236987
Enter the supplier address:
[ -1 ] To Exit
East Lake
```

Than the system will ask if any of the items already in the system are supplied by the new supplier, if yes write 1 and write the item line, if no write 2 and add a new item this supplier is suppling (the process of adding new item was explained before).



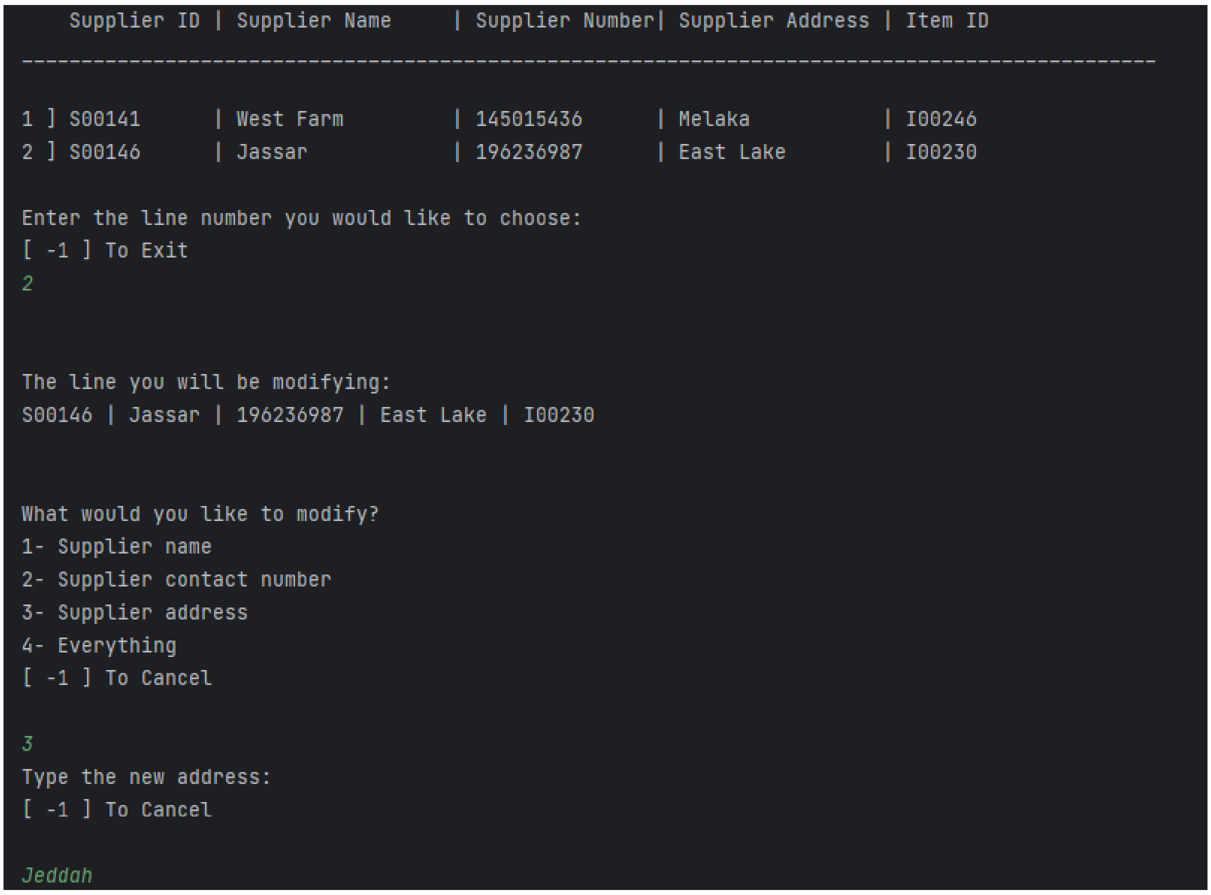
```
    Item ID | Item Name          | Unit     | Price      | Minimum      | Available Unit| Supplier ID
-----------------------------------------------------------------------------------------------

1 ] I00230  | Apple              | Box      | 12.0       | 10           | 102            | S00122
2 ] I00246  | Banana             | Bag      | 15         | 33           | 120            | S00141

Does the supplier supply any of the above items?
1- Yes
2- No
1

Type the line number of the item:
Enter the line number you would like to choose:
[ -1 ] To Exit
1


Would you like to add another supplier?
1- Yes
2- No
Type the number of your choice
```

The user can delete a supplier by writing 4, the list of suppliers will display and the user must choose the line number of the supplier to delete, after that he can either to delete another one by writing 1 or end the process by writing 2.



```
    Supplier ID | Supplier Name     | Supplier Number| Supplier Address | Item ID
-----------------------------------------------------------------------------------------------

1 ] S00141      | West Farm         | 145015436      | Melaka           | I00246
2 ] S00145      | ME                | 145015436      | kl               | I00245
3 ] S00146      | Jassar            | 196236987      | East Lake        | I00230

Enter the line number you would like to choose:
[ -1 ] To Exit
2
Item was deleted successfully.
Would you like to delete another supplier?
1- Yes
2- No
2
```

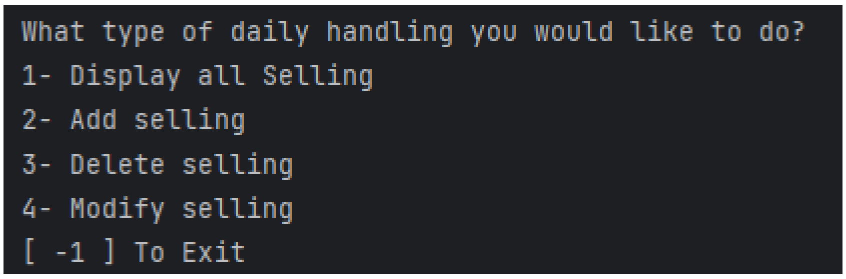
The last function in the supplier entry is to modify by writing 5, the user will write the line of the supplier he wants to modify and then choose what to modify exactly or to modify everything. In this example I am changing the address.



```
    Supplier ID | Supplier Name     | Supplier Number| Supplier Address | Item ID
-----------------------------------------------------------------------------------------------

1 ] S00141      | West Farm         | 145015436      | Melaka           | I00246
2 ] S00146      | Jassar            | 196236987      | East Lake        | I00230

Enter the line number you would like to choose:
[ -1 ] To Exit
2


The line you will be modifying:
S00146 | Jassar | 196236987 | East Lake | I00230


What would you like to modify?
1- Supplier name
2- Supplier contact number
3- Supplier address
4- Everything
[ -1 ] To Cancel

3
Type the new address:
[ -1 ] To Cancel

Jeddah
```

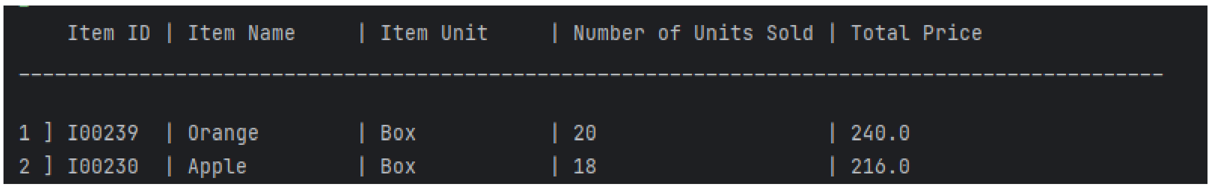
Lastly to get back to the main page write -1.

The third option in the main menu for the sale manager is the daily wise sales entry by clicking on 3, in this page the user can control everything about the daily item-wise sales.



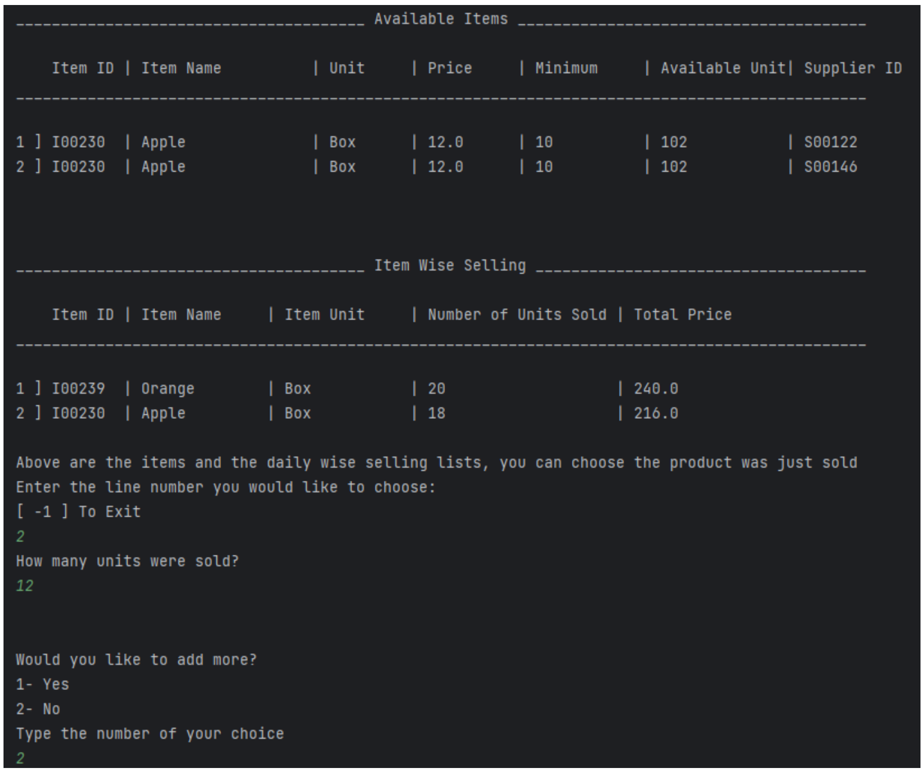
```
What type of daily handling you would like to do?
1- Display all Selling
2- Add selling
3- Delete selling
4- Modify selling
[ -1 ] To Exit
```

The first choice is to display the daily item-wise sales list by writing 1.



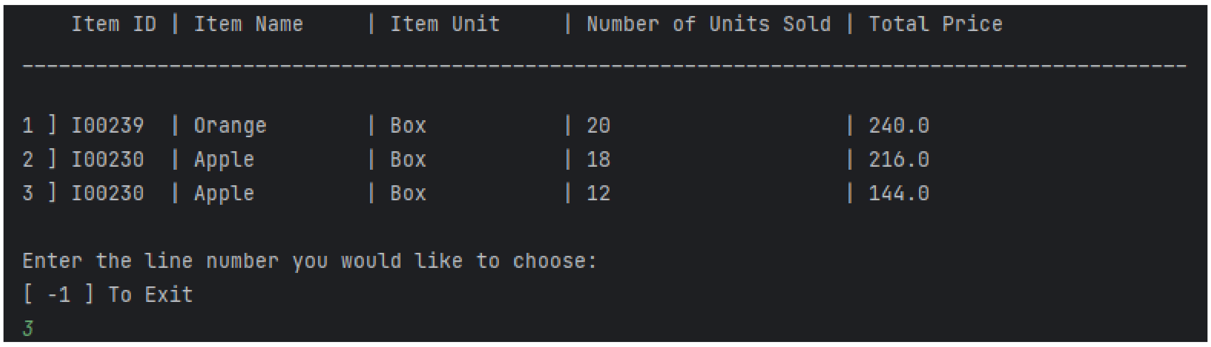
```
    Item ID | Item Name     | Item Unit     | Number of Units Sold | Total Price
----------------------------------------------------------------------------------------------

1 ] I00239  | Orange        | Box           | 20                   | 240.0
2 ] I00230  | Apple         | Box           | 18                   | 216.0
```

The second is to add a new selling, two lists will be display after writing 2, the first one is the available items and the second one is item wise list, the user will choose the line number from the items list to assign a new selling, after that the user must assign how many units were sold, this is how to add a new selling, the system will ask if the user want to repeat the process.



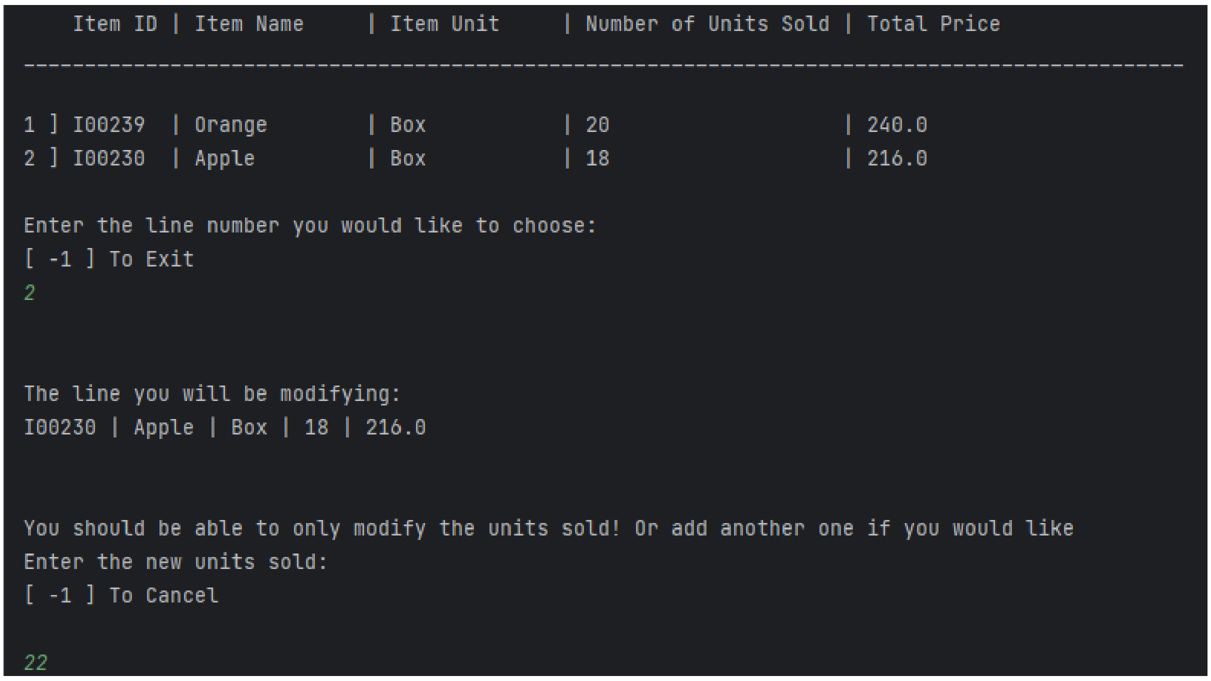
```
_______________________________________ Available Items _______________________________________

    Item ID | Item Name          | Unit     | Price     | Minimum     | Available Unit| Supplier ID
-----------------------------------------------------------------------------------------------

1 ] I00230  | Apple              | Box      | 12.0      | 10          | 102           | S00122
2 ] I00230  | Apple              | Box      | 12.0      | 10          | 102           | S00146



_______________________________________ Item Wise Selling _____________________________________

    Item ID | Item Name     | Item Unit     | Number of Units Sold | Total Price
-----------------------------------------------------------------------------------------------

1 ] I00239  | Orange        | Box           | 20                   | 240.0
2 ] I00230  | Apple         | Box           | 18                   | 216.0

Above are the items and the daily wise selling lists, you can choose the product was just sold
Enter the line number you would like to choose:
[ -1 ] To Exit
2
How many units were sold?
12


Would you like to add more?
1- Yes
2- No
Type the number of your choice
2
```

Next feature is to delete a daily item-wise by clicking 3, the list of daily item-wise will be displayed and the user should choose the line number he wants to delete.



```
    Item ID | Item Name      | Item Unit      | Number of Units Sold | Total Price
-----------------------------------------------------------------------------------------------

1 ] I00239  | Orange         | Box            | 20                   | 240.0
2 ] I00230  | Apple          | Box            | 18                   | 216.0
3 ] I00230  | Apple          | Box            | 12                   | 144.0

Enter the line number you would like to choose:
[ -1 ] To Exit
3
```

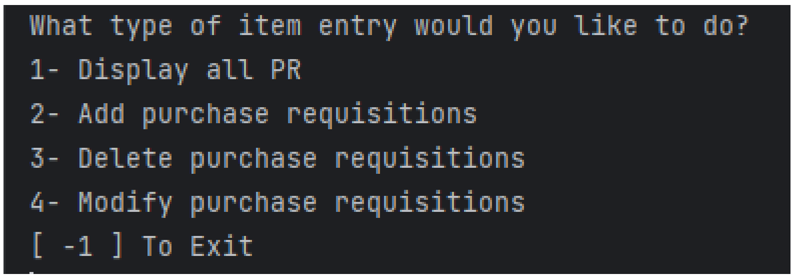
To end the daily item-wise sales with modify function which can be used by writing 4. After that the user can choose the line to modify, and the only thing that is allowed to modify is units sold. This is all for the daily item-wise sales.



```
    Item ID | Item Name      | Item Unit      | Number of Units Sold | Total Price
-----------------------------------------------------------------------------------------------

1 ] I00239  | Orange         | Box            | 20                   | 240.0
2 ] I00230  | Apple          | Box            | 18                   | 216.0

Enter the line number you would like to choose:
[ -1 ] To Exit
2



The line you will be modifying:
I00230 | Apple | Box | 18 | 216.0



You should be able to only modify the units sold! Or add another one if you would like
Enter the new units sold:
[ -1 ] To Cancel

22
```

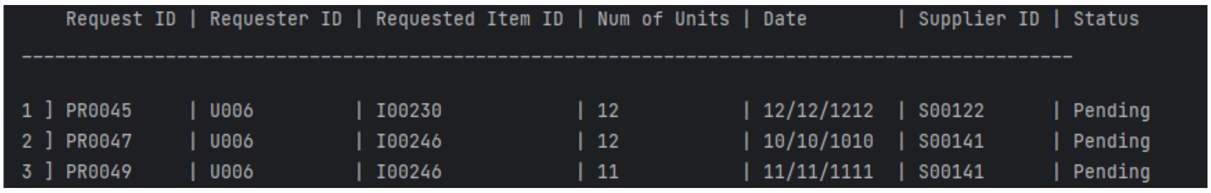
To get back to the main menu write -1.

The forth option in the main menu for the sales manager is for purchase requisition.



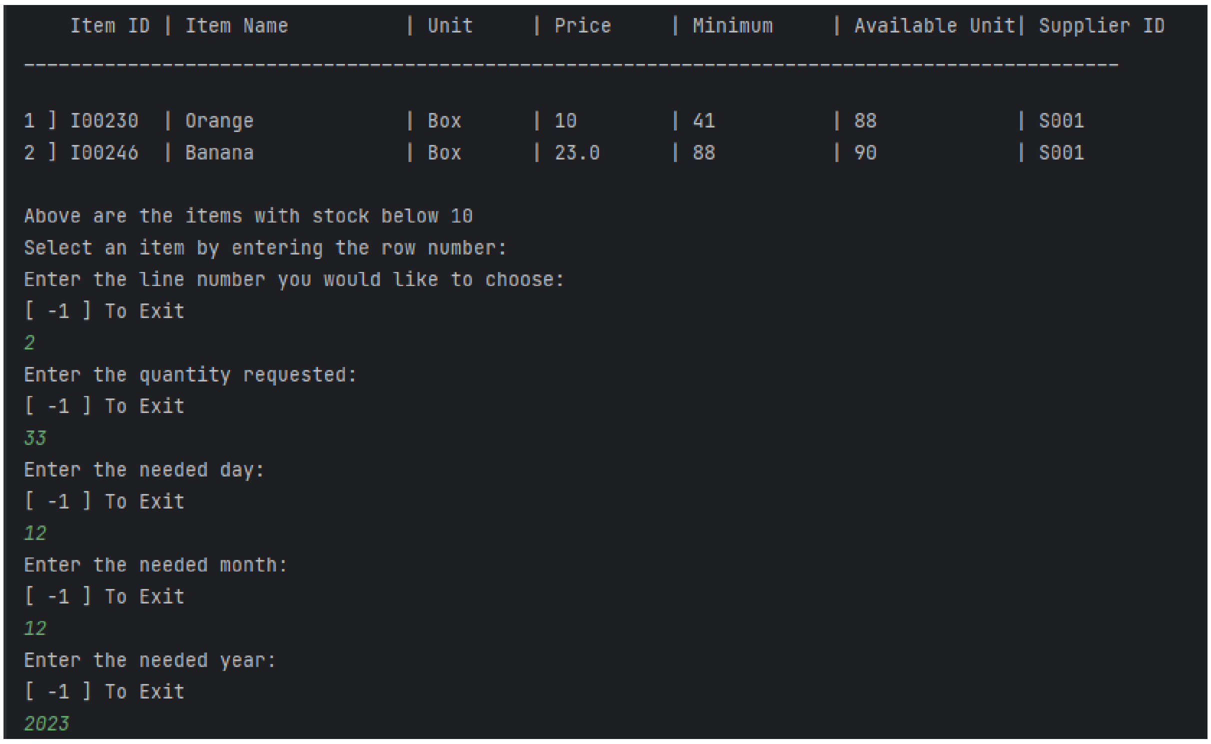
```
What type of item entry would you like to do?
1- Display all PR
2- Add purchase requisitions
3- Delete purchase requisitions
4- Modify purchase requisitions
[ -1 ] To Exit
```

If the user wants to display all the PR list can enter 1 and the list will be displayed.



```
    Request ID | Requester ID | Requested Item ID | Num of Units | Date       | Supplier ID | Status
--------------------------------------------------------------------------------------------------

1 ] PR0045     | U006         | I00230            | 12           | 12/12/1212  | S00122      | Pending
2 ] PR0047     | U006         | I00246            | 12           | 10/10/1010  | S00141      | Pending
3 ] PR0049     | U006         | I00246            | 11           | 11/11/1111  | S00141      | Pending
```

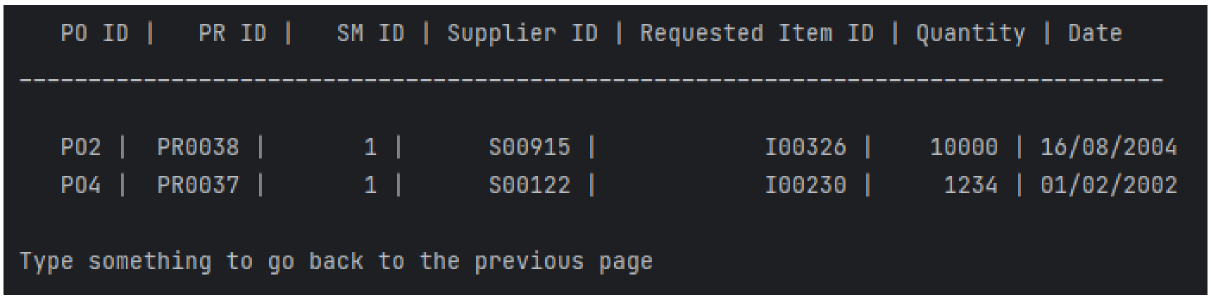
The second choice is to add a new PR, enter 2 to add a new PR, the user will be asked the following data to be fill for the new PR. The system will ask as usual if the user want to add another PR or to get back to the menu.



```
    Item ID | Item Name          | Unit      | Price      | Minimum      | Available Unit| Supplier ID
------------------------------------------------------------------------------------------------------

1 ] I00230  | Orange             | Box       | 10         | 41           | 88             | S001
2 ] I00246  | Banana             | Box       | 23.0       | 88           | 90             | S001

Above are the items with stock below 10
Select an item by entering the row number:
Enter the line number you would like to choose:
[ -1 ] To Exit
2
Enter the quantity requested:
[ -1 ] To Exit
33
Enter the needed day:
[ -1 ] To Exit
12
Enter the needed month:
[ -1 ] To Exit
12
Enter the needed year:
[ -1 ] To Exit
2023
```

After that the user can go back to the main menu by writing -1.

The list functionality for the sales manager user is to display the Purchase Order (PO) list, in the main menu for the sales manager write 5 to display the PO list.

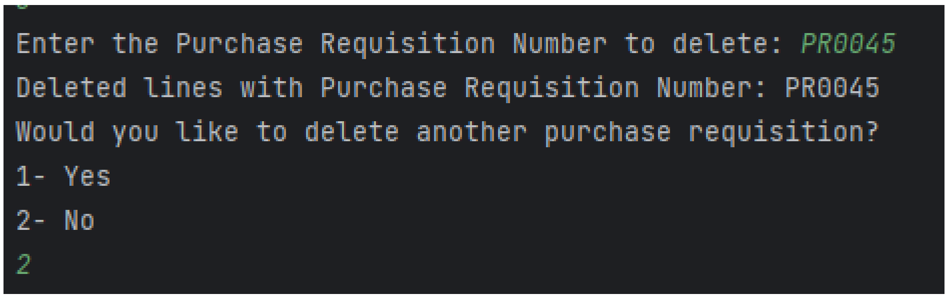


```
   PO ID |   PR ID |   SM ID | Supplier ID | Requested Item ID | Quantity | Date
------------------------------------------------------------------------------------

   PO2 |  PR0038 |       1 |      S00915 |            I00326 |    10000 | 16/08/2004
   PO4 |  PR0037 |       1 |      S00122 |            I00230 |     1234 | 01/02/2002

Type something to go back to the previous page
```

That was all for the Sales Manager capabilities in the system.

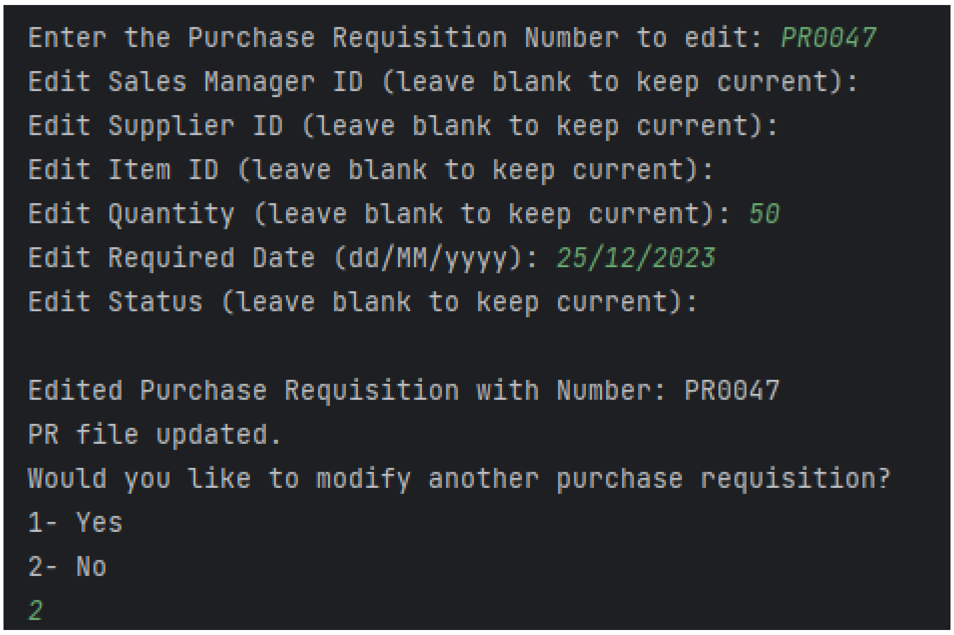
*Please take note that all the data was shown only examples to demonstrate the system, in addition to all the system is covered with validations to avoid any entry/input errors.

The third function for the PR is to delete, after writing 3 the user will be asked to enter the PR ID to be deleted, we can refer to the PR list to choose which PR the user want to delete.



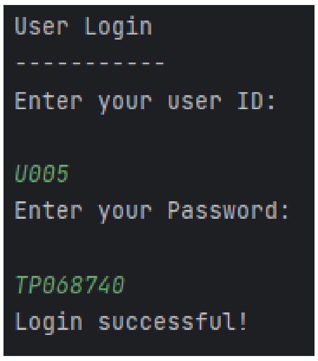
```
Enter the Purchase Requisition Number to delete: PR0045
Deleted lines with Purchase Requisition Number: PR0045
Would you like to delete another purchase requisition?
1- Yes
2- No
2
```

Last function in PR is modify, the user will be asked to enter the PR ID should be modified and change the things the user needs, the user can leave the one he doesn't want to change empty and change only the things required.

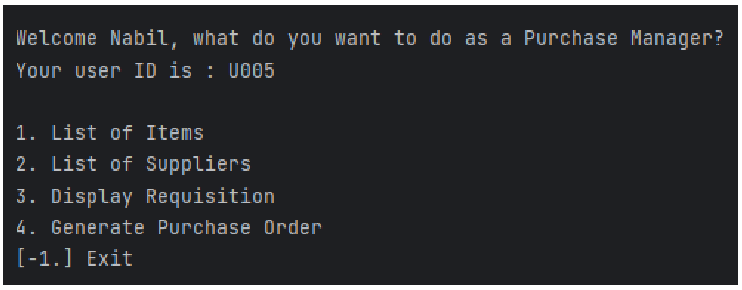


```
Enter the Purchase Requisition Number to edit: PR0047
Edit Sales Manager ID (leave blank to keep current):
Edit Supplier ID (leave blank to keep current):
Edit Item ID (leave blank to keep current):
Edit Quantity (leave blank to keep current): 50
Edit Required Date (dd/MM/yyyy): 25/12/2023
Edit Status (leave blank to keep current):

Edited Purchase Requisition with Number: PR0047
PR file updated.
Would you like to modify another purchase requisition?
1- Yes
2- No
2
```

Purchase Manager



```
User Login
-----------
Enter your user ID:

U005
Enter your Password:

TP068740
Login successful!
```

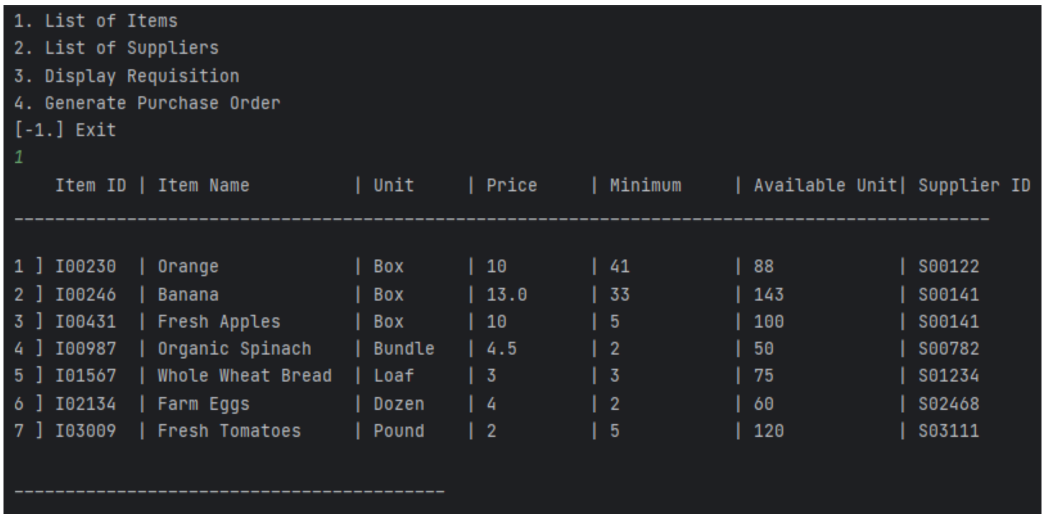


```
Welcome Nabil, what do you want to do as a Purchase Manager?
Your user ID is : U005

1. List of Items
2. List of Suppliers
3. Display Requisition
4. Generate Purchase Order
[-1.] Exit
```

Login *Purchase Manager menu*

Login successful justifies the successful user login by demonstrating the correct input of a valid User ID and Password. User ID 'U005' serves as the unique identifier for the user while the password acts as the confidential key to the Purchase Manager account. The login successful message confirm that the provided credentials matched the records in the system. Hence, validating the user's identity and granting them access to the system.

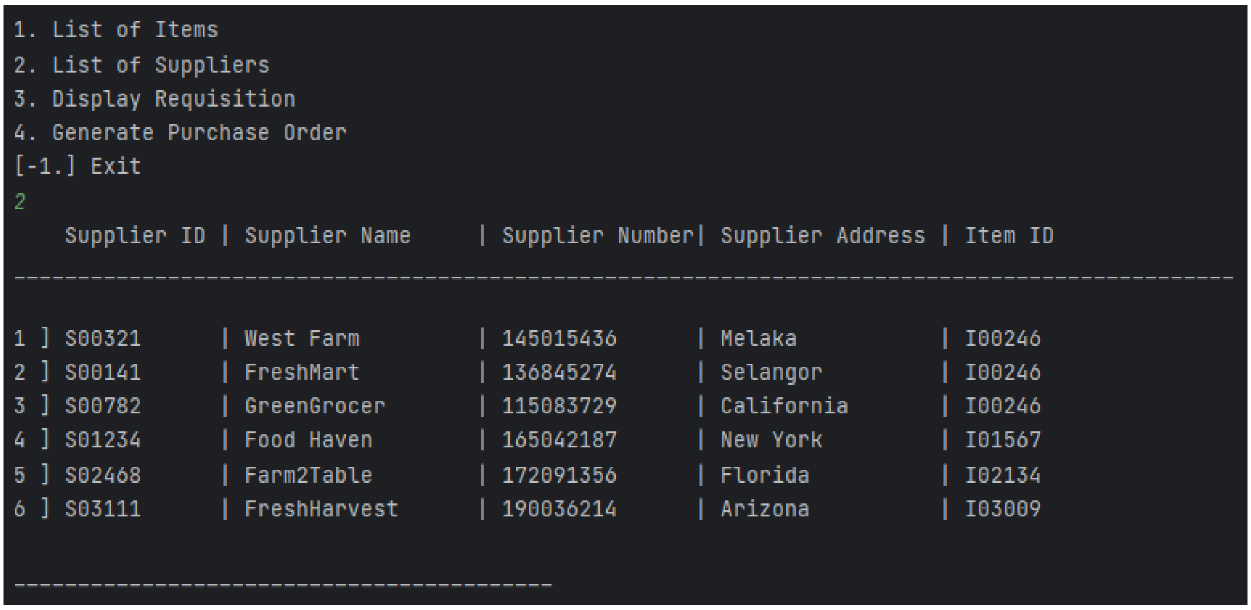
Once logged in, the system presents a menu with five options. The user then selects option 1 which leads to the display of all items inside Items.txt.



```
1. List of Items
2. List of Suppliers
3. Display Requisition
4. Generate Purchase Order
[-1.] Exit
1
    Item ID | Item Name          | Unit     | Price     | Minimum      | Available Unit| Supplier ID
--------------------------------------------------------------------------------------------

1 ] I00230  | Orange             | Box      | 10        | 41           | 88            | S00122
2 ] I00246  | Banana             | Box      | 13.0      | 33           | 143           | S00141
3 ] I00431  | Fresh Apples       | Box      | 10        | 5            | 100           | S00141
4 ] I00987  | Organic Spinach    | Bundle   | 4.5       | 2            | 50            | S00782
5 ] I01567  | Whole Wheat Bread  | Loaf     | 3         | 3            | 75            | S01234
6 ] I02134  | Farm Eggs          | Dozen    | 4         | 2            | 60            | S02468
7 ] I03009  | Fresh Tomatoes     | Pound    | 2         | 5            | 120           | S03111

------------------------------------------
```

List of Items

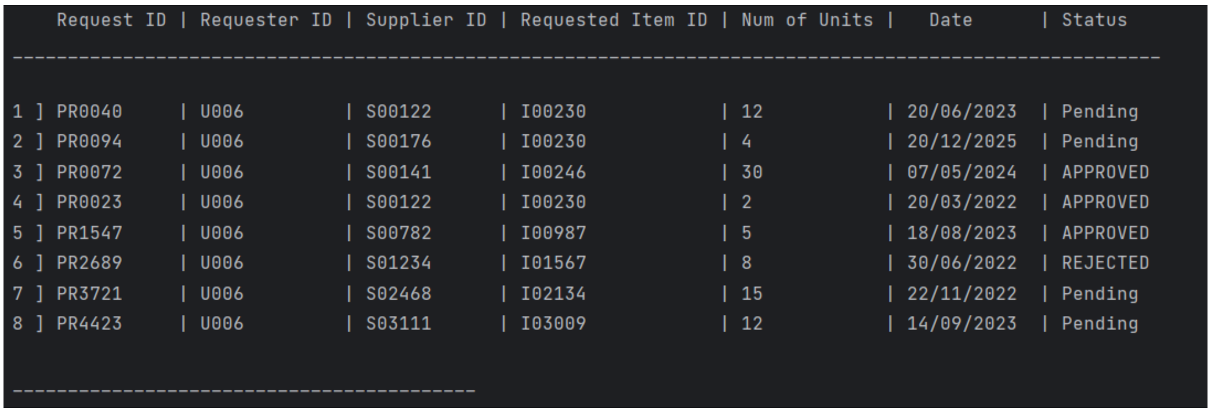
The presented output lists items and their associated details. This presentation offers a clear and organized view of essential information, enhancing ease of navigation and data retrieval. By displaying this information in a tabular format, it aligns with object-oriented programming principles in Java, as it could be implemented using classes and objects. Item class will encapsulate properties as shown above as instance variables.



```
1. List of Items
2. List of Suppliers
3. Display Requisition
4. Generate Purchase Order
[-1.] Exit
2
    Supplier ID | Supplier Name     | Supplier Number| Supplier Address | Item ID
-----------------------------------------------------------------------------------------------

1 ] S00321      | West Farm         | 145015436      | Melaka           | I00246
2 ] S00141      | FreshMart         | 136845274      | Selangor         | I00246
3 ] S00782      | GreenGrocer       | 115083729      | California       | I00246
4 ] S01234      | Food Haven        | 165042187      | New York         | I01567
5 ] S02468      | Farm2Table        | 172091356      | Florida          | I02134
6 ] S03111      | FreshHarvest      | 190036214      | Arizona          | I03009


------------------------------------------
```

List of Suppliers

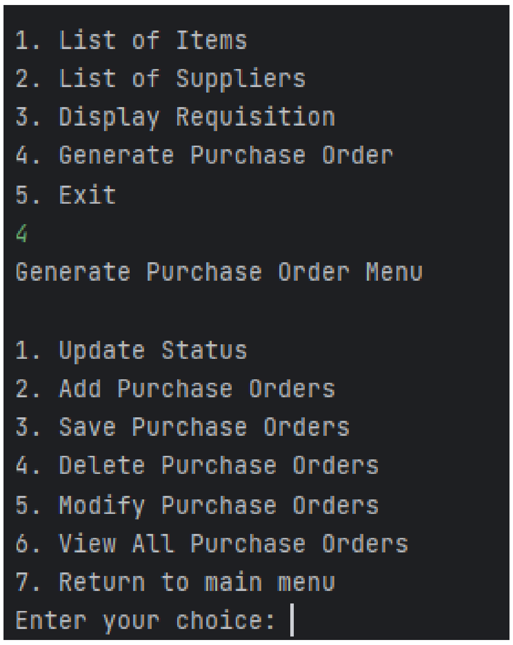
The provided output presents a menu for list of all suppliers when option 2 is selected. Similar to items list, this option also displays a well-organized table of supplier information. Purchase Manager can use this option to find out all the supplier information before making an informed decision of approving and rejecting. The items are assigned to a number in which it is easier to grab and use,



```
    Request ID | Requester ID | Supplier ID | Requested Item ID | Num of Units |    Date      | Status
--------------------------------------------------------------------------------------------------------

1 ] PR0040     | U006         | S00122      | I00230            | 12           | 20/06/2023  | Pending
2 ] PR0094     | U006         | S00176      | I00230            | 4            | 20/12/2025  | Pending
3 ] PR0072     | U006         | S00141      | I00246            | 30           | 07/05/2024  | APPROVED
4 ] PR0023     | U006         | S00122      | I00230            | 2            | 20/03/2022  | APPROVED
5 ] PR1547     | U006         | S00782      | I00987            | 5            | 18/08/2023  | APPROVED
6 ] PR2689     | U006         | S01234      | I01567            | 8            | 30/06/2022  | REJECTED
7 ] PR3721     | U006         | S02468      | I02134            | 15           | 22/11/2022  | Pending
8 ] PR4423     | U006         | S03111      | I03009            | 12           | 14/09/2023  | Pending


------------------------------------------
```

Display Requisition

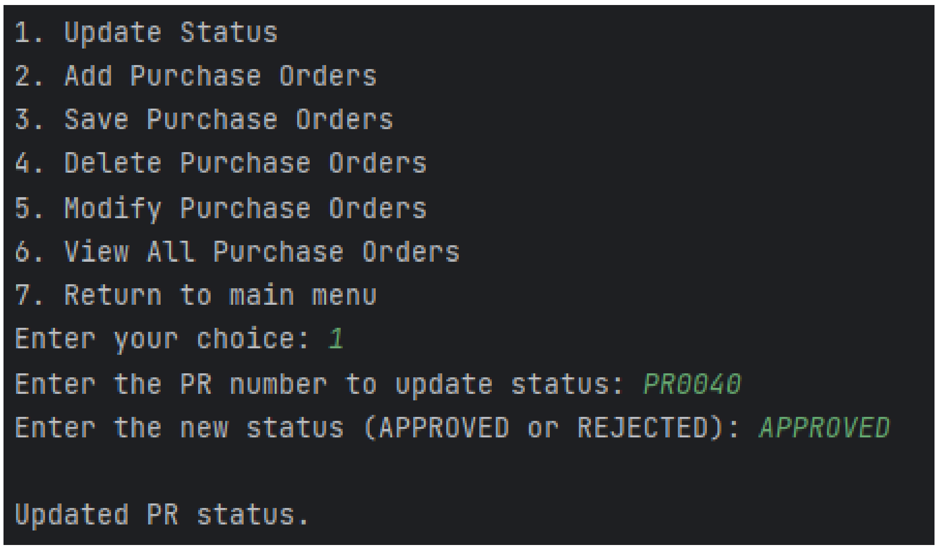
The displayed output presents a menu with numbered system where option 3 has been selected. This choice will lead to a tabular structure representing requisition information. Each row in the table corresponds to a unique identifier which is Request ID. Meanwhile Requester ID is Manager ID where the manager who requests the items to be purchased.



```
1. List of Items
2. List of Suppliers
3. Display Requisition
4. Generate Purchase Order
5. Exit
4
Generate Purchase Order Menu

1. Update Status
2. Add Purchase Orders
3. Save Purchase Orders
4. Delete Purchase Orders
5. Modify Purchase Orders
6. View All Purchase Orders
7. Return to main menu
Enter your choice: 
```

Generate Purchase Order Menu

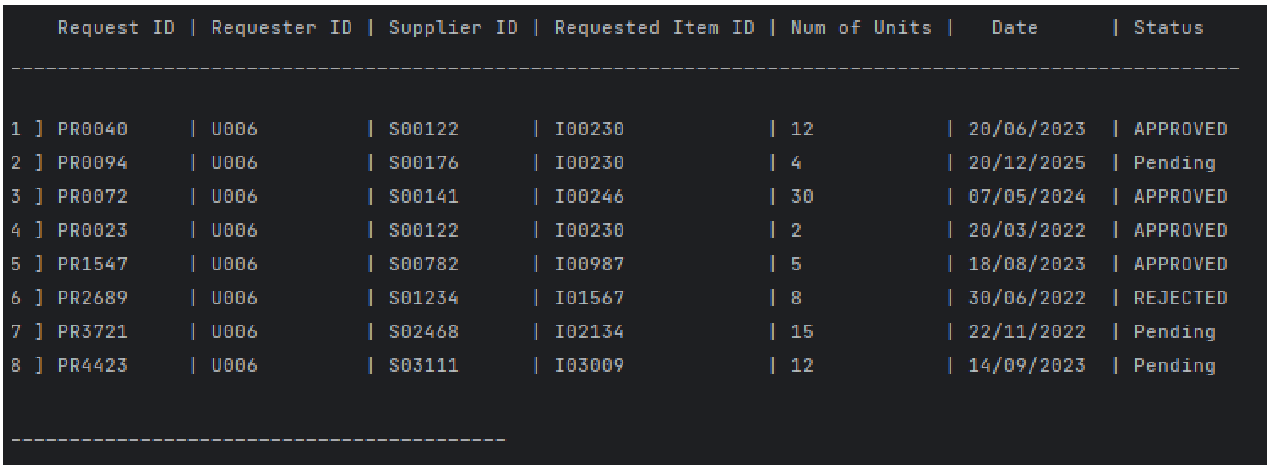
The provided output displays a menu with numbered options under generate purchase order section when option 4 is selected. These new numbered options allow Purchase Manager to perform authorised actions related to the purchase order generation process. Each option corresponds to a specific operation.



```
1. Update Status
2. Add Purchase Orders
3. Save Purchase Orders
4. Delete Purchase Orders
5. Modify Purchase Orders
6. View All Purchase Orders
7. Return to main menu
Enter your choice: 1
Enter the PR number to update status: PR0040
Enter the new status (APPROVED or REJECTED): APPROVED

Updated PR status.
```

Update Status

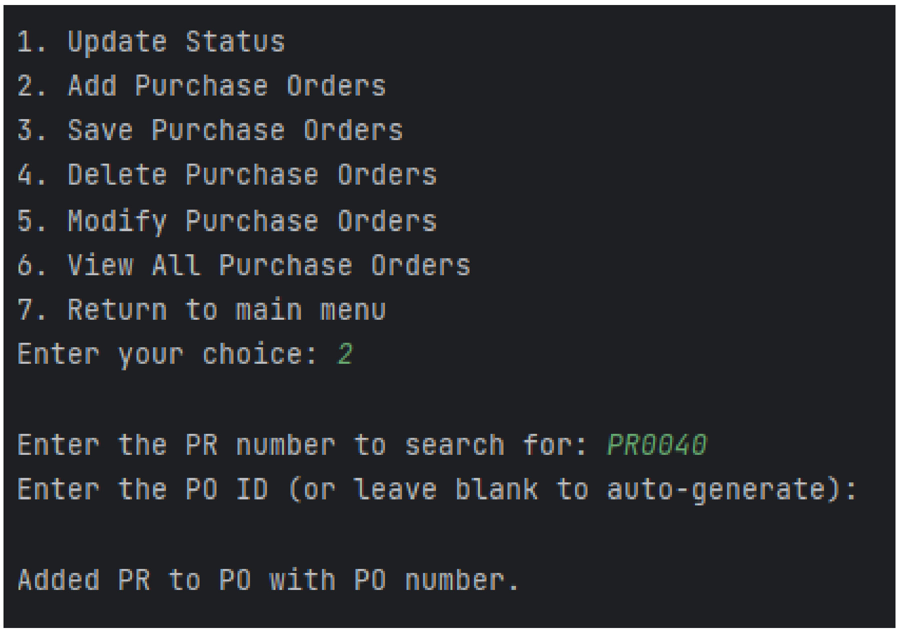
The provided output gives a prompt to search for the Purchase Requisition ID for the system to update. Then it prompts for Purchase Managers to choose whether to approve or reject the PR. Once chosen, the updated status will be updated in the Purchase Requisition text file.



```
    Request ID | Requester ID | Supplier ID | Requested Item ID | Num of Units |   Date      | Status
------------------------------------------------------------------------------------------------------

1 ] PR0040     | U006         | S00122      | I00230            | 12           | 20/06/2023  | APPROVED
2 ] PR0094     | U006         | S00176      | I00230            | 4            | 20/12/2025  | Pending
3 ] PR0072     | U006         | S00141      | I00246            | 30           | 07/05/2024  | APPROVED
4 ] PR0023     | U006         | S00122      | I00230            | 2            | 20/03/2022  | APPROVED
5 ] PR1547     | U006         | S00782      | I00987            | 5            | 18/08/2023  | APPROVED
6 ] PR2689     | U006         | S01234      | I01567            | 8            | 30/06/2022  | REJECTED
7 ] PR3721     | U006         | S02468      | I02134            | 15           | 22/11/2022  | Pending
8 ] PR4423     | U006         | S03111      | I03009            | 12           | 14/09/2023  | Pending


-----------------------------------------
```

After updating status (Output)

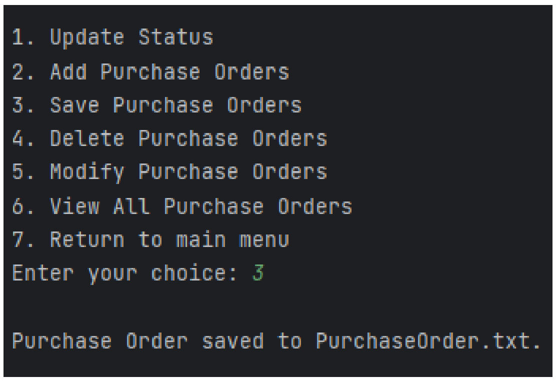
Figure above shows the updated status inside the text file.



```
1. Update Status
2. Add Purchase Orders
3. Save Purchase Orders
4. Delete Purchase Orders
5. Modify Purchase Orders
6. View All Purchase Orders
7. Return to main menu
Enter your choice: 2

Enter the PR number to search for: PR0040
Enter the PO ID (or leave blank to auto-generate):

Added PR to PO with PO number.
```

Add Purchase Order

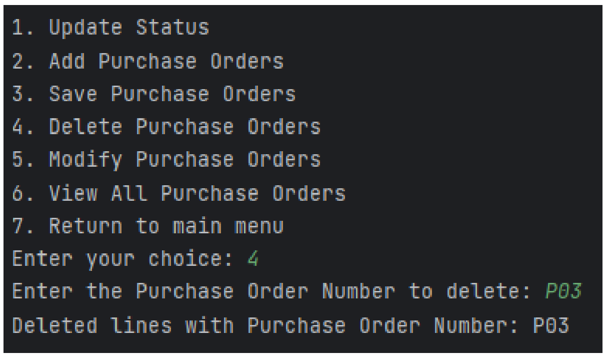
The provided output gives a prompt to search for the Purchase Requisition ID for the system to update. Afterwards, the system will ask Purchase Manager to input the Purchase Order ID, or it will generate automatically by sequence. The updated status from Purchase Requisition (PurchaseRequisition.txt) then will be assigned to Purchase Order text file (PurchaseOrder.txt)



```
1. Update Status
2. Add Purchase Orders
3. Save Purchase Orders
4. Delete Purchase Orders
5. Modify Purchase Orders
6. View All Purchase Orders
7. Return to main menu
Enter your choice: 3

Purchase Order saved to PurchaseOrder.txt.
```

Save Purchase Order

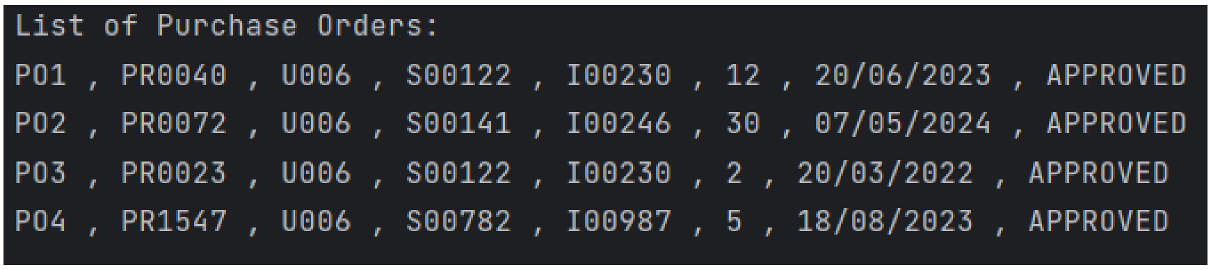
The option 3 will save the purchase orders added inside PurchaseOrder.txt. A confirmation message will pop up to indicate that the purchase orders have been saved inside the text file.



```
1. Update Status
2. Add Purchase Orders
3. Save Purchase Orders
4. Delete Purchase Orders
5. Modify Purchase Orders
6. View All Purchase Orders
7. Return to main menu
Enter your choice: 4
Enter the Purchase Order Number to delete: P03
Deleted lines with Purchase Order Number: P03
```

Delete PO

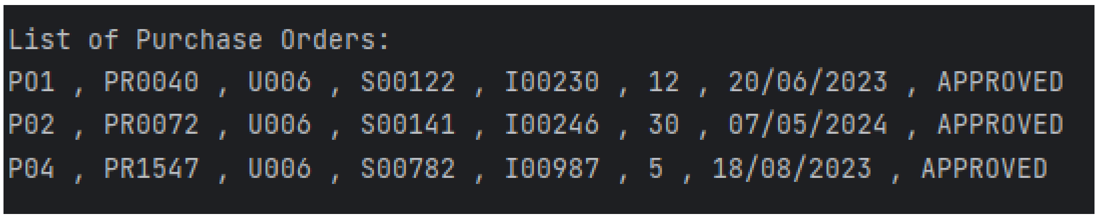
Based on Figure above, option 4 is selected indicating Purchase Manager wants to delete a Purchase Order. The system will then prompt the Purchase Order number that they want to delete. And in this case, PO3 was selected. A successful message with the purchase order number selected will clarify whether PO3 is deleted or not. If not, an error message will display.



```
List of Purchase Orders:
P01 , PR0040 , U006 , S00122 , I00230 , 12 , 20/06/2023 , APPROVED
P02 , PR0072 , U006 , S00141 , I00246 , 30 , 07/05/2024 , APPROVED
P03 , PR0023 , U006 , S00122 , I00230 , 2 , 20/03/2022 , APPROVED
P04 , PR1547 , U006 , S00782 , I00987 , 5 , 18/08/2023 , APPROVED
```

Before delete (Output)

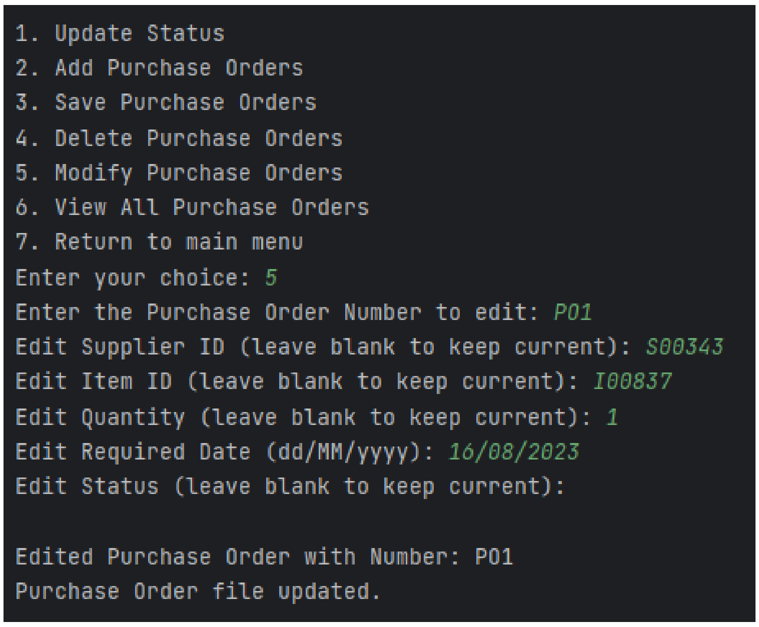
Figure above shows the list of Purchase Order before Purchase Manager prompts to delete PO3.



```
List of Purchase Orders:
P01 , PR0040 , U006 , S00122 , I00230 , 12 , 20/06/2023 , APPROVED
P02 , PR0072 , U006 , S00141 , I00246 , 30 , 07/05/2024 , APPROVED
P04 , PR1547 , U006 , S00782 , I00987 , 5 , 18/08/2023 , APPROVED
```

After delete (Output)

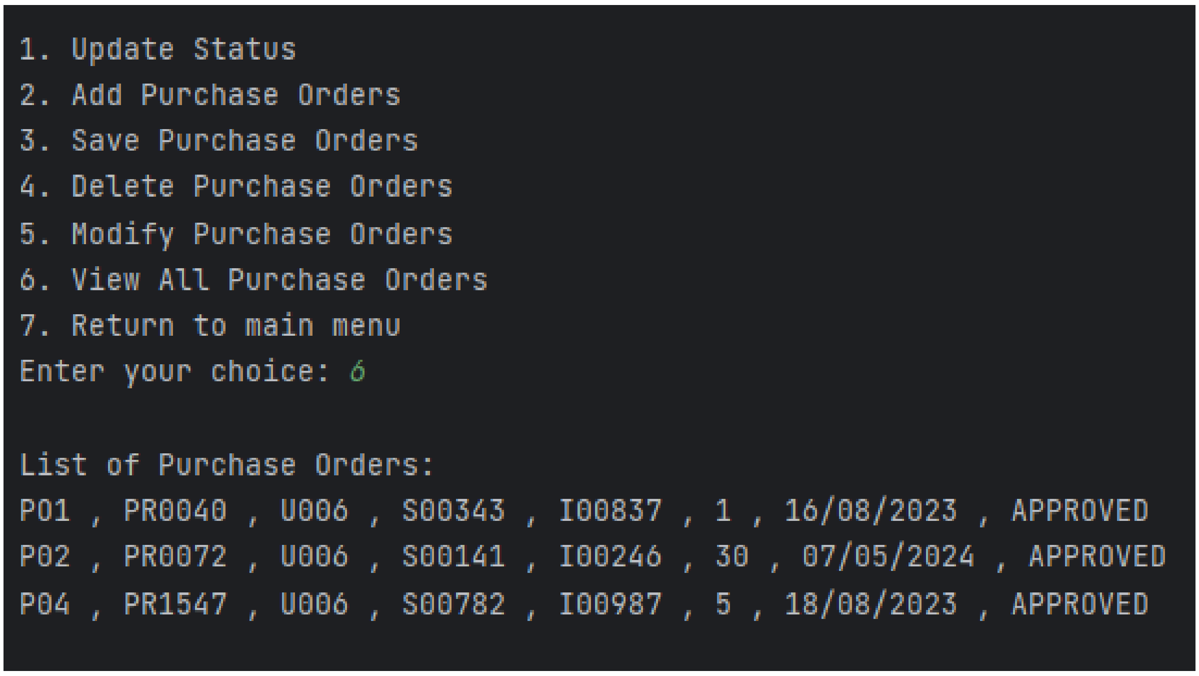
After Purchase Manager deletes the PO3, it will update the Purchase Order text file (PurchaseOrder.txt) and it will automatically save it inside the text file.



```
1. Update Status
2. Add Purchase Orders
3. Save Purchase Orders
4. Delete Purchase Orders
5. Modify Purchase Orders
6. View All Purchase Orders
7. Return to main menu
Enter your choice: 5
Enter the Purchase Order Number to edit: P01
Edit Supplier ID (leave blank to keep current): S00343
Edit Item ID (leave blank to keep current): I00837
Edit Quantity (leave blank to keep current): 1
Edit Required Date (dd/MM/yyyy): 16/08/2023
Edit Status (leave blank to keep current):

Edited Purchase Order with Number: P01
Purchase Order file updated.
```

Modify Purchase Order

Based on Figure above, option 5 is selected indicating Purchase Manager wants to Modify a Purchase Order. The system will then prompt the Purchase Order number that they want to modify. And in this case, PO1 was selected. The system then will prompt the Purchase Manager to enter all the modifications to each attributes such as Supplier ID, Item ID, Quantity, Required Date and Status. Purchase Manager can modify the status if there is an overturn on the decision on the status. The modifications then will be updated and saved in the Purchase Order text file. A successful message will be prompted indicating that the modifies have been updated and saved.

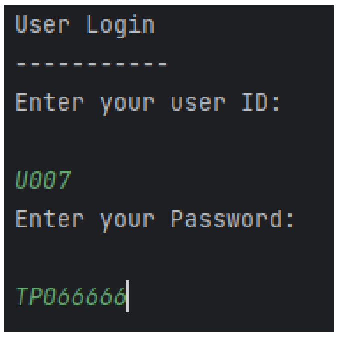


```
1. Update Status
2. Add Purchase Orders
3. Save Purchase Orders
4. Delete Purchase Orders
5. Modify Purchase Orders
6. View All Purchase Orders
7. Return to main menu
Enter your choice: 6

List of Purchase Orders:
P01 , PR0040 , U006 , S00343 , I00837 , 1 , 16/08/2023 , APPROVED
P02 , PR0072 , U006 , S00141 , I00246 , 30 , 07/05/2024 , APPROVED
P04 , PR1547 , U006 , S00782 , I00987 , 5 , 18/08/2023 , APPROVED
```

View All Purchase Orders (Output)

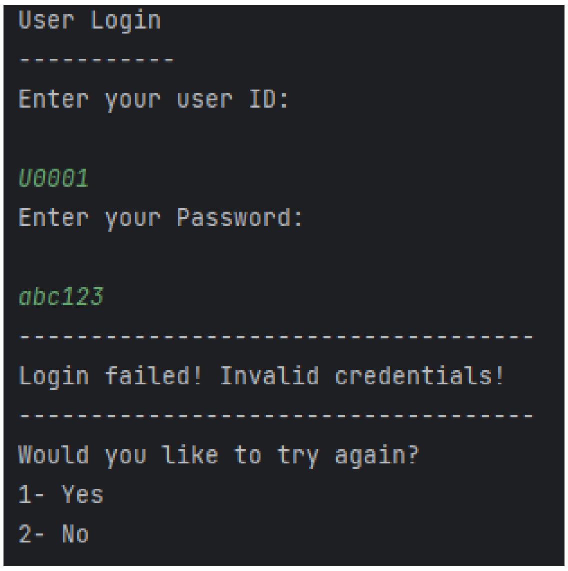
In the figure above, option 6 was selected by Purchase Manager to view all purchase orders. And based on the figure above, all modifications have updated proving that the system did not have any problems on updating and saving the modifications.

Administrator



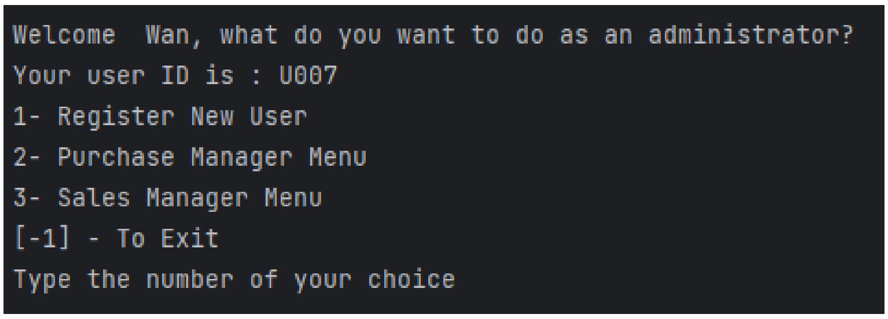
```
User Login
-----------
Enter your user ID:

U007
Enter your Password:

TP066666
```

Once the user has entered their user ID and password for login, the system will determine what is the role of the user based on the credentials that have been inserted. In this case, it is administrator.



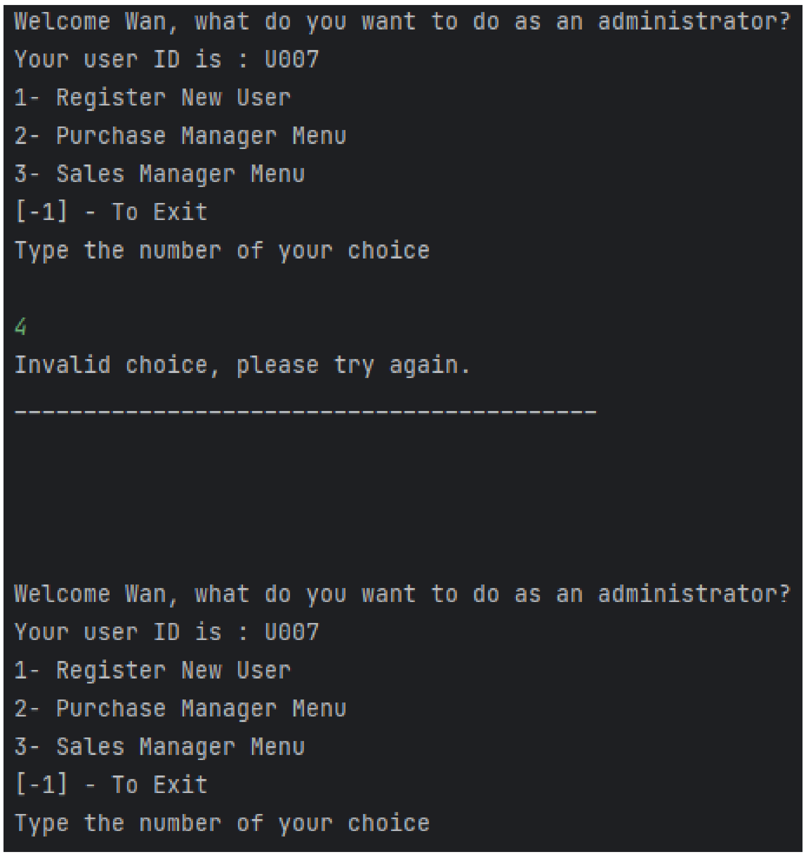
```
User Login
-----------
Enter your user ID:

U0001
Enter your Password:

abc123
-----------------------------------
Login failed! Invalid credentials!
-----------------------------------
Would you like to try again?
1- Yes
2- No
```

If the user ID and password does not exist in User.txt text file, the system will display that login has failed due to invalid credentials and user will be prompted if they want to try again by inputting "1" for "Yes" or "2" for "No".

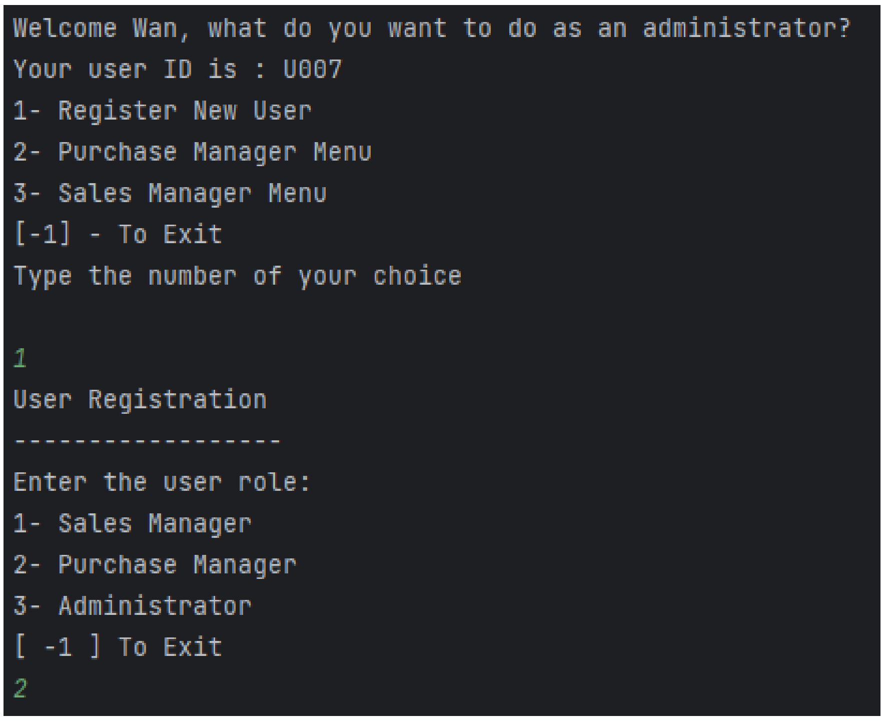


```
Welcome  Wan, what do you want to do as an administrator?
Your user ID is : U007
1- Register New User
2- Purchase Manager Menu
3- Sales Manager Menu
[-1] - To Exit
Type the number of your choice
```

Then, once the system has validated the user ID and password as administrator, a main menu will be prompted which allows the user to choose what kind of functionalities they want to conduct as an administrator by inserting either “1” or “2” or “3” as inputs. In addition, on top of the screen shows the username, role, and user ID of the current user.

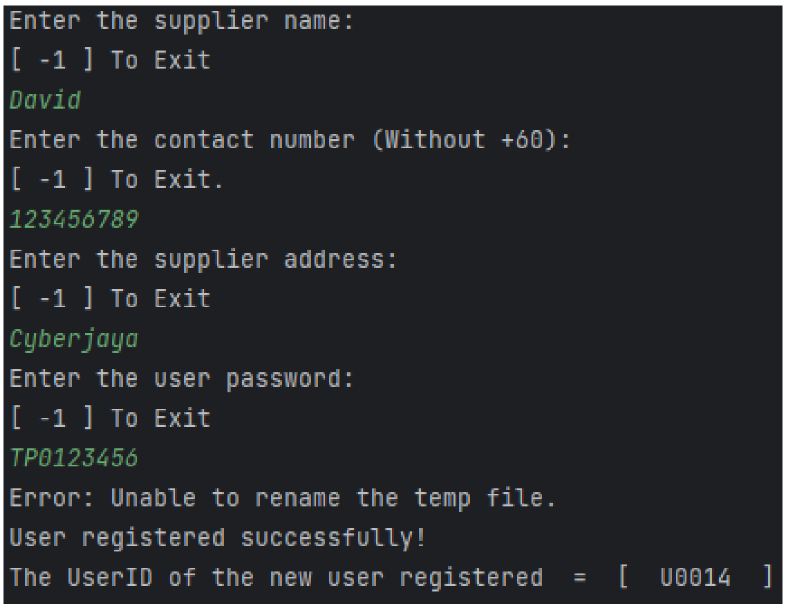

```
Welcome Wan, what do you want to do as an administrator?
Your user ID is : U007
1- Register New User
2- Purchase Manager Menu
3- Sales Manager Menu
[-1] - To Exit
Type the number of your choice

4
Invalid choice, please try again.
------------------------------------------




Welcome Wan, what do you want to do as an administrator?
Your user ID is : U007
1- Register New User
2- Purchase Manager Menu
3- Sales Manager Menu
[-1] - To Exit
Type the number of your choice
```

If a different input is inserted instead of “1” or “2” or” 3” or ”-1”, the system will display a message of invalid input has been inserted and the system will prompt the user to try again. In the figure above, number “4” is inserted, and the user is required to try again by entering a valid choice.



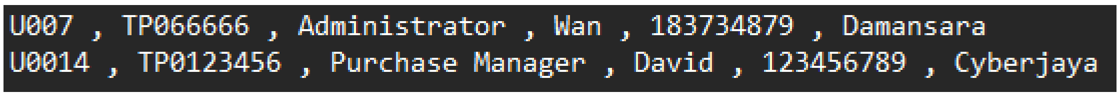
```
Welcome Wan, what do you want to do as an administrator?
Your user ID is : U007
1- Register New User
2- Purchase Manager Menu
3- Sales Manager Menu
[-1] - To Exit
Type the number of your choice

1
User Registration
------------------
Enter the user role:
1- Sales Manager
2- Purchase Manager
3- Administrator
[ -1 ] To Exit
2
```

If the administrator selects number "1" from the main menu, the administrator will have the function to register a new user for the system. To register a new user, the administrator is required to enter the role of the new user that could be Sales Manager, Purchase Manager or Administrator. In the figure above, number "2" is selected which is Purchase Manager.



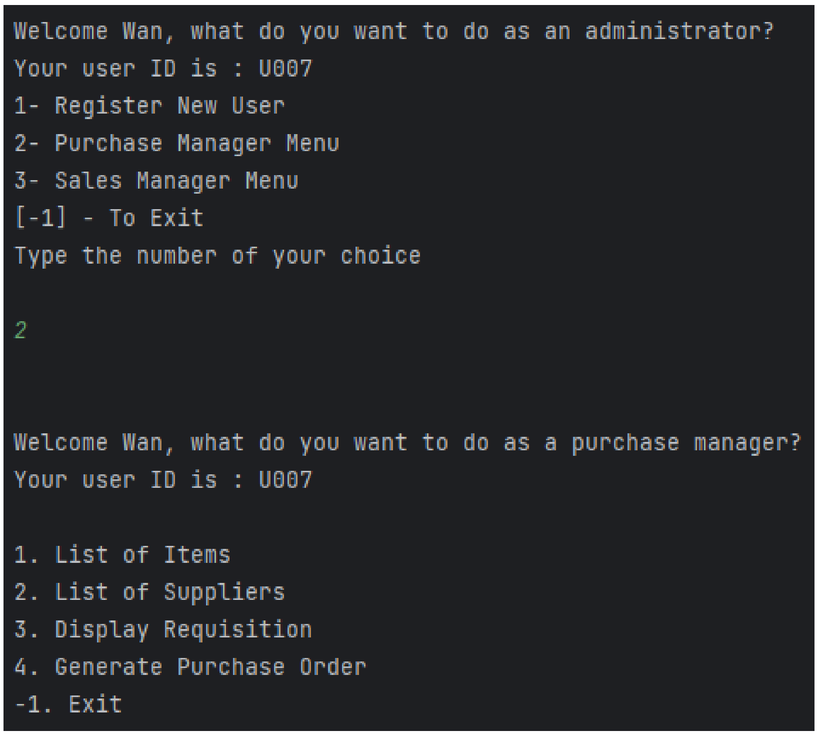
```
Enter the supplier name:
[ -1 ] To Exit
David
Enter the contact number (Without +60):
[ -1 ] To Exit.
123456789
Enter the supplier address:
[ -1 ] To Exit
Cyberjaya
Enter the user password:
[ -1 ] To Exit
TP0123456
Error: Unable to rename the temp file.
User registered successfully!
The UserID of the new user registered  =  [  U0014  ]
```

Then, the administrator is required to enter the supplier's name, contact number, address and password. "David", "123456789", "Cyberjaya" and "TP0123456" would be the data that has been inserted respectively. The registration is complete after all necessary information are filled.



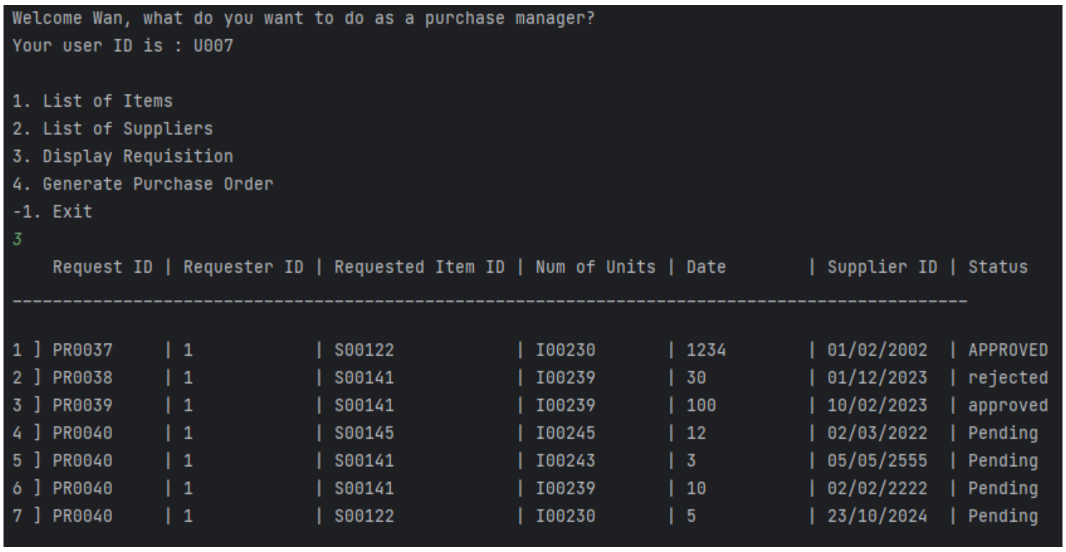
```
U007 , TP066666 , Administrator , Wan , 183734879 , Damansara
U0014 , TP0123456 , Purchase Manager , David , 123456789 , Cyberjaya
```

An updated Users.txt file will be produced by the system after registration is done. Besides, the system also generates a user ID automatically as shown in the figure above which is U0014.



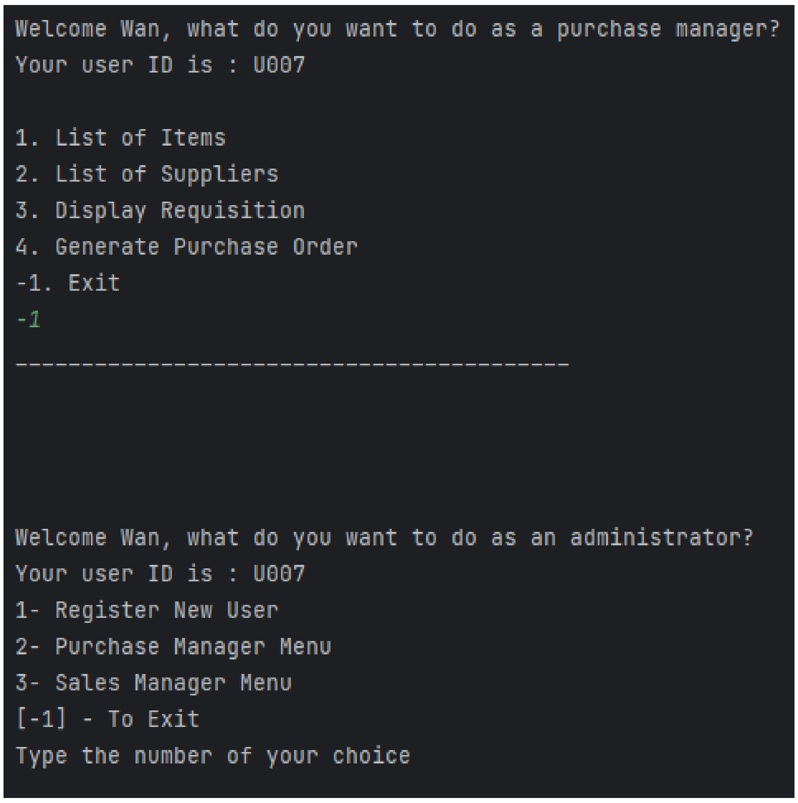
```
Welcome Wan, what do you want to do as an administrator?
Your user ID is : U007
1- Register New User
2- Purchase Manager Menu
3- Sales Manager Menu
[-1] - To Exit
Type the number of your choice

2



Welcome Wan, what do you want to do as a purchase manager?
Your user ID is : U007

1. List of Items
2. List of Suppliers
3. Display Requisition
4. Generate Purchase Order
-1. Exit
```

The second function of administrator is having the access of Purchase Manager. To do that, the administrator is required to enter number "2" when prompted as shown above and the main menu for Purchase Manager will be displayed and the administrator has full access to all the functionalities of Purchase Manager.



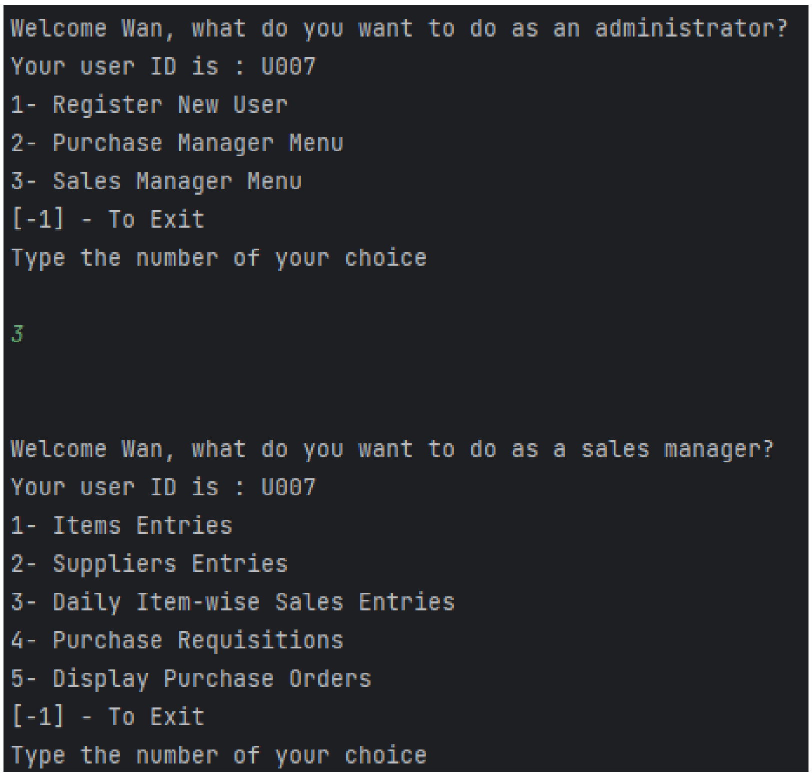
```
Welcome Wan, what do you want to do as a purchase manager?
Your user ID is : U007

1. List of Items
2. List of Suppliers
3. Display Requisition
4. Generate Purchase Order
-1. Exit
3
    Request ID | Requester ID | Requested Item ID | Num of Units | Date         | Supplier ID | Status
-----------------------------------------------------------------------------------------------

1 ] PR0037     | 1            | S00122            | I00230       | 1234         | 01/02/2002  | APPROVED
2 ] PR0038     | 1            | S00141            | I00239       | 30           | 01/12/2023  | rejected
3 ] PR0039     | 1            | S00141            | I00239       | 100          | 10/02/2023  | approved
4 ] PR0040     | 1            | S00145            | I00245       | 12           | 02/03/2022  | Pending
5 ] PR0040     | 1            | S00141            | I00243       | 3            | 05/05/2555  | Pending
6 ] PR0040     | 1            | S00141            | I00239       | 10           | 02/02/2222  | Pending
7 ] PR0040     | 1            | S00122            | I00230       | 5            | 23/10/2024  | Pending
```

If the administrator selects number “3”, the system will display the list of requisition just like how a Purchase Manager would see.



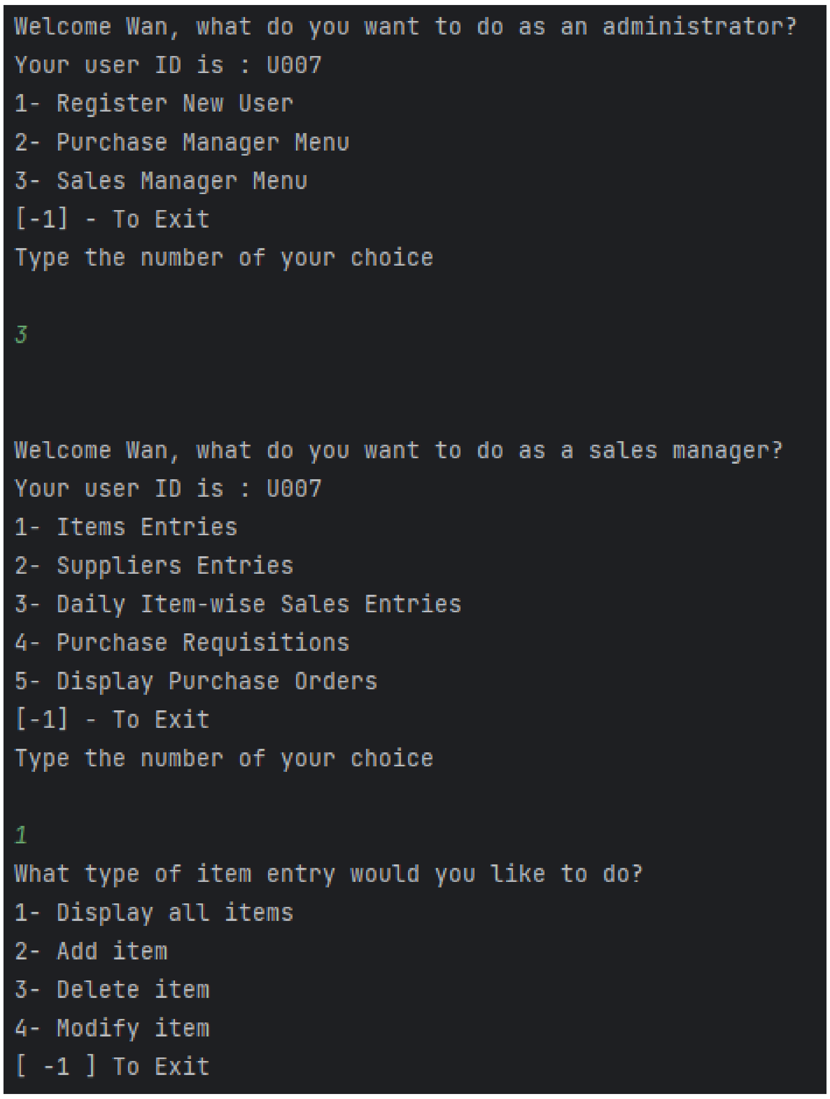
```
Welcome Wan, what do you want to do as a purchase manager?
Your user ID is : U007

1. List of Items
2. List of Suppliers
3. Display Requisition
4. Generate Purchase Order
-1. Exit
-1
------------------------------------------




Welcome Wan, what do you want to do as an administrator?
Your user ID is : U007
1- Register New User
2- Purchase Manager Menu
3- Sales Manager Menu
[-1] - To Exit
Type the number of your choice
```

Then the administrator is able to go back to the main menu of administrator by entering the number “-1”



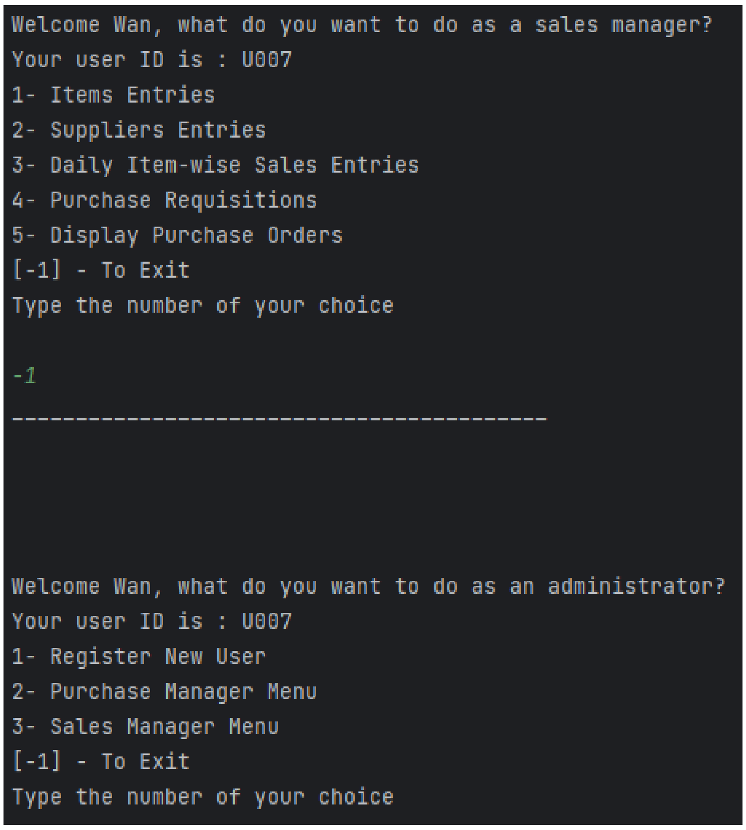
```
Welcome Wan, what do you want to do as an administrator?
Your user ID is : U007
1- Register New User
2- Purchase Manager Menu
3- Sales Manager Menu
[-1] - To Exit
Type the number of your choice

3


Welcome Wan, what do you want to do as a sales manager?
Your user ID is : U007
1- Items Entries
2- Suppliers Entries
3- Daily Item-wise Sales Entries
4- Purchase Requisitions
5- Display Purchase Orders
[-1] - To Exit
Type the number of your choice
```

The third function of administrator is having the access of Sales Manager. To do that, the administrator must input number “3” when prompted and the main menu of sales manager will be displayed, and full access of Sales Manager will be given to the administrator.



```
Welcome Wan, what do you want to do as an administrator?
Your user ID is : U007
1- Register New User
2- Purchase Manager Menu
3- Sales Manager Menu
[-1] - To Exit
Type the number of your choice

3



Welcome Wan, what do you want to do as a sales manager?
Your user ID is : U007
1- Items Entries
2- Suppliers Entries
3- Daily Item-wise Sales Entries
4- Purchase Requisitions
5- Display Purchase Orders
[-1] - To Exit
Type the number of your choice

1
What type of item entry would you like to do?
1- Display all items
2- Add item
3- Delete item
4- Modify item
[ -1 ] To Exit
```

As shown in the figure above, once the administrator (as Sales Manager) selected number “1” which is Items Entries, the administrator is able to do the functionalities within Items Entries which are Display all items, Add item, Delete item and Modify item.



```
Welcome Wan, what do you want to do as a sales manager?
Your user ID is : U007
1- Items Entries
2- Suppliers Entries
3- Daily Item-wise Sales Entries
4- Purchase Requisitions
5- Display Purchase Orders
[-1] - To Exit
Type the number of your choice

-1
------------------------------------------




Welcome Wan, what do you want to do as an administrator?
Your user ID is : U007
1- Register New User
2- Purchase Manager Menu
3- Sales Manager Menu
[-1] - To Exit
Type the number of your choice
```

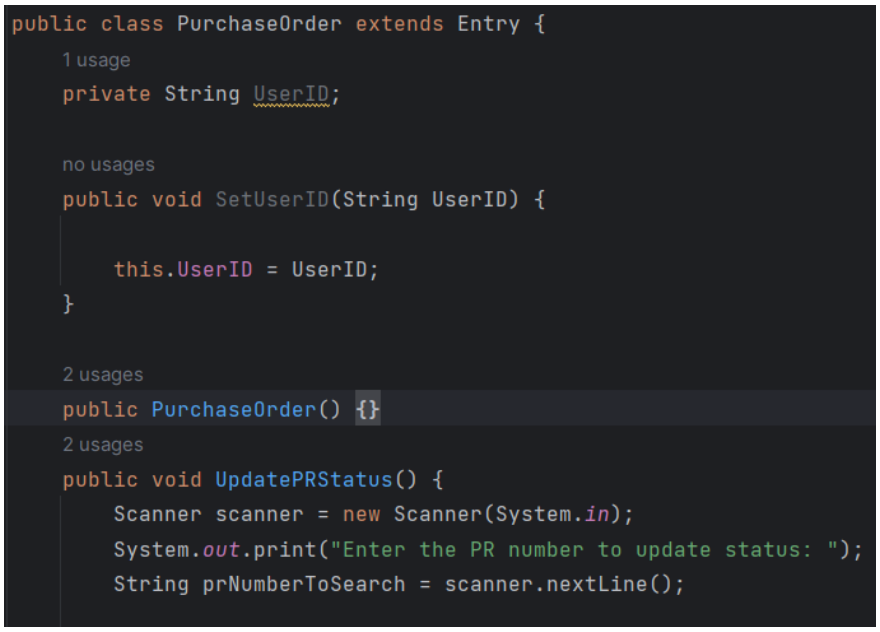
Administrator can return to main menu once they are done using their functionalities by inserting “-1” as inputs as shown in the figure above.

OBJECT ORIENTED CONCEPT

Class

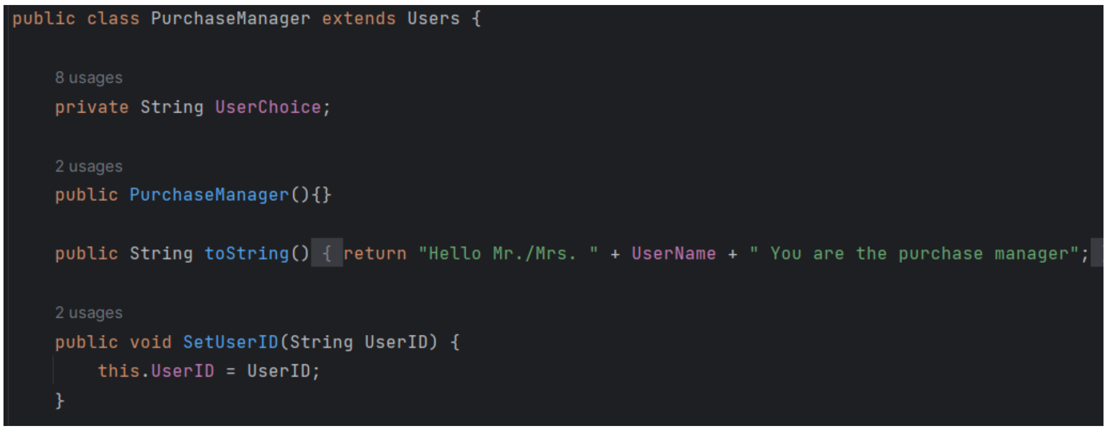
A class is one of the most fundamental elements of Object-Oriented programming. Without a class, an object could not be created and the Object-Oriented concept cannot be applied. Class is the bread and butter of coding. It acts as a template for creating objects. A class consists of the name of the class, the attribute which is the variable and method which is the function.

When developing the Purchase Order Management System (POM), creating a class is a must for the code to fully function.



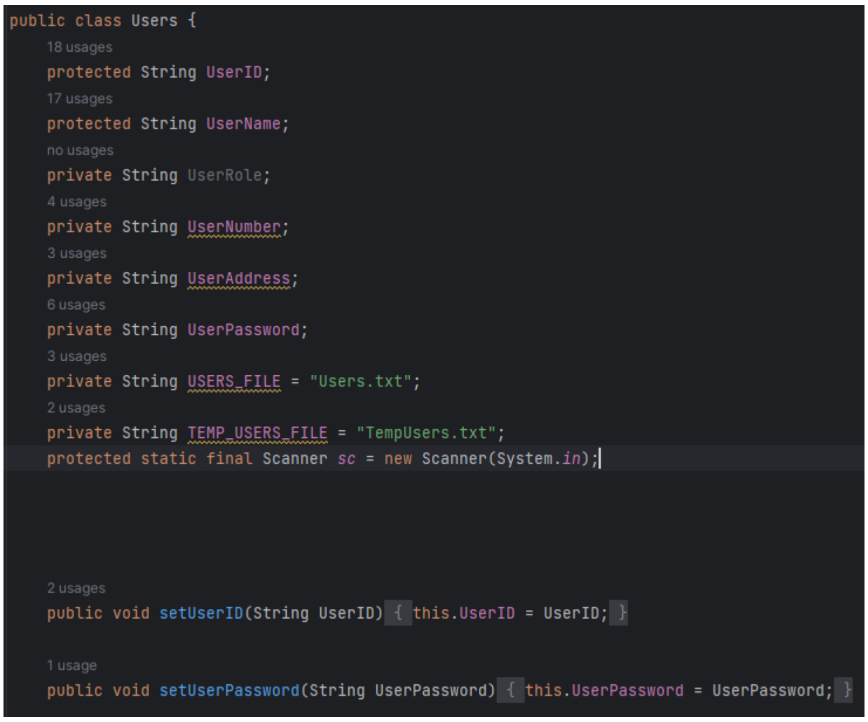
```
public class PurchaseOrder extends Entry {
    1 usage
    private String UserID;

    no usages
    public void SetUserID(String UserID) {

        this.UserID = UserID;
    }

    2 usages
    public PurchaseOrder() {}
    2 usages
    public void UpdatePRStatus() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the PR number to update status: ");
        String prNumberToSearch = scanner.nextLine();
```

Based on the screenshot above, we are using "PurchaseOrder" class to do all functions that is under Purchase Order. As said earlier, in a class there will be 3 main components which are class name, attributes and method. The name of the class is "PurchaseOrder" and there is an attribute called UserID which uses a String variable and private access modifier. Since it is private, it is encapsulated within the class and exclusively for "PurchaseOrder" class only. Finally, one of the methods that can be seen from the screenshot is the function to update purchase requisition status called "UpdatePRStatus" and within the method will have the respective code to make the method work.



```
public class PurchaseManager extends Users {

    8 usages
    private String UserChoice;

    2 usages
    public PurchaseManager(){}

    public String toString() { return "Hello Mr./Mrs. " + UserName + " You are the purchase manager";

    2 usages
    public void SetUserID(String UserID) {
        this.UserID = UserID;
    }
```

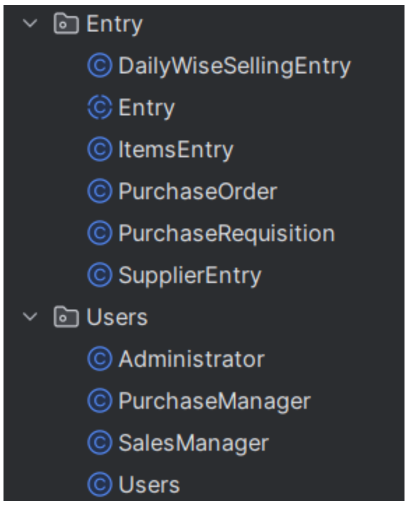
The screenshot above is another example of the usage of class while developing the system. In this case, we are referring to "PurchaseManager" class and there are three main components of a class which are; name of the class, the attributes, and methods. The name of the class is PurchaseManager, while UserChoice is the attribute using a String variable and an access modifier of private. The final component of the class is the method called "toString" to obtain a String representation of an object.



```
public class Users {
    18 usages
    protected String UserID;
    17 usages
    protected String UserName;
    no usages
    private String UserRole;
    4 usages
    private String UserNumber;
    3 usages
    private String UserAddress;
    6 usages
    private String UserPassword;
    3 usages
    private String USERS_FILE = "Users.txt";
    2 usages
    private String TEMP_USERS_FILE = "TempUsers.txt";
    protected static final Scanner sc = new Scanner(System.in);



    2 usages
    public void setUserID(String UserID) { this.UserID = UserID; }

    1 usage
    public void setUserPassword(String UserPassword) { this.UserPassword = UserPassword; }
```

This is another example of a class that have been used in development of the system. The name of the class would be "Users" and there are eight attributes which uses the variable "String" and consists of private and protected access modifier. One of the methods that is distinguishable is the setUserID method with a method parameter of "String UserID".

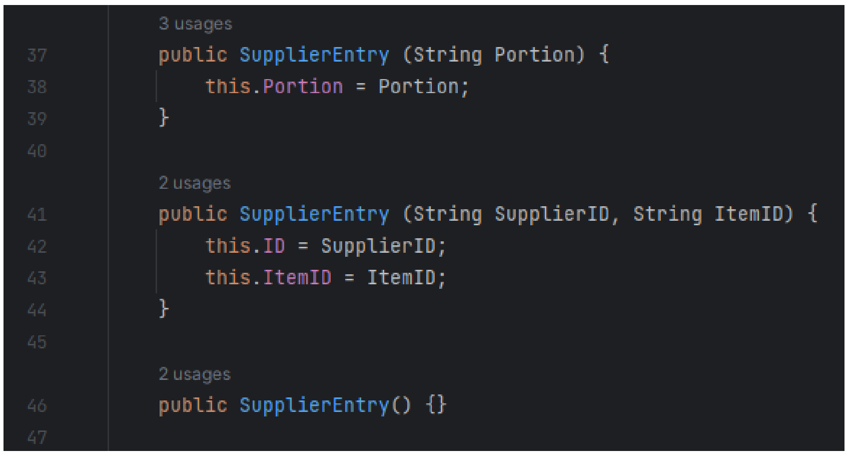
Throughout the whole development of the system, many classes were created fully utilized the Object Oriented concepts. The concept for every class will always be the same as it is the foundation for developing a fully functioning and workable code. The screenshot below shows the number of classes that we implemented while developing the system. Every class that had been created would have the same components as all normal classes do. It will consist of name of the class, attributes, and method and the content of the components differ from each class to another.



```
Entry
    DailyWiseSellingEntry
    Entry
    ItemsEntry
    PurchaseOrder
    PurchaseRequisition
    SupplierEntry
Users
    Administrator
    PurchaseManager
    SalesManager
    Users
```

Constructors

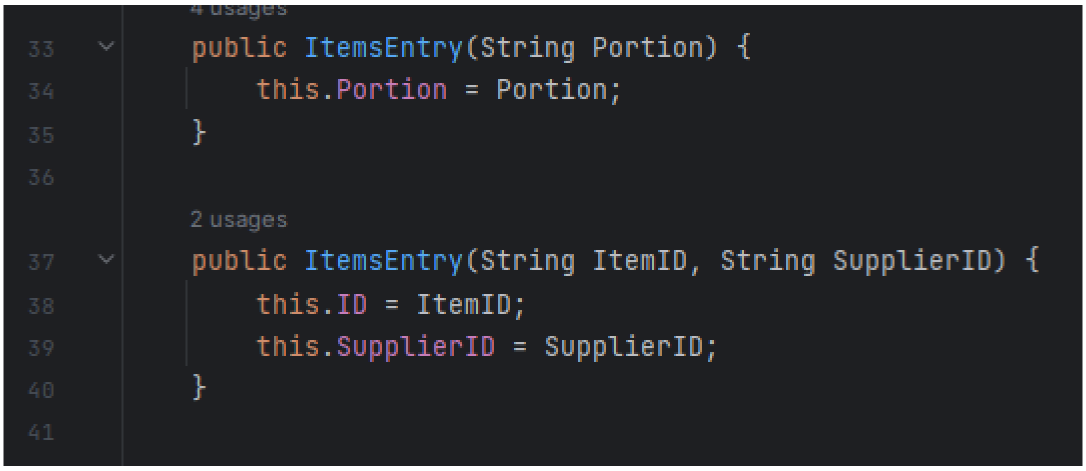
Constructors play a pivotal role in object-oriented programming within Java. This specialized method serves as the blueprints for initializing object, allowing for their instantiation and setup. Their significance stems from their ability to ensure that objects are in a valid state upon creation, setting up initial values for object attributes. In each class we can have more than one constructer, however each constructer must have different signature, or a compilation error will appear if two constructors have the same signature.



```java
    3 usages
    public SupplierEntry (String Portion) {
        this.Portion = Portion;
    }

    2 usages
    public SupplierEntry (String SupplierID, String ItemID) {
        this.ID = SupplierID;
        this.ItemID = ItemID;
    }

    2 usages
    public SupplierEntry() {}

```

SupplierEntry Constructors

In the SupplierEntry class, we created 3 constructors that returns different parameters that serves the purpose of providing flexibility and convenience when creating instances such as Portion, ID and ItemID. The first constructor takes a single String parameter, “Portion” and initializes this instance variable with the value passed as an argument. This way, it allows us to create instances of the SupplierEntry class while specifying the instance value during object creation. Second constructor takes two String parameters, “SupplierID” and “ItemID” and initializes the instance variables, “ID” and “ItemID” with the values passed as arguments. It enables the creation of instances of the SupplierEntry class while specifying both the SupplierID and itemID during object creation. The third constructor is a default constructor where it does not take any parameters and initializes the SupplierEntry objects without any specific initial values. It is useful when we want to create instances of the SupplierEntry class with default values for all instance variables. In many cases, the default constructor suffices because it allows us to access and set the properties of an object using encapsulation. These methods provide a means to interact with the object’s attributes without requiring additional constructors. Therefore, in scenarios where we do not need to provide values during object creation, we use the default constructor to rely on (Pankaj, 2022).



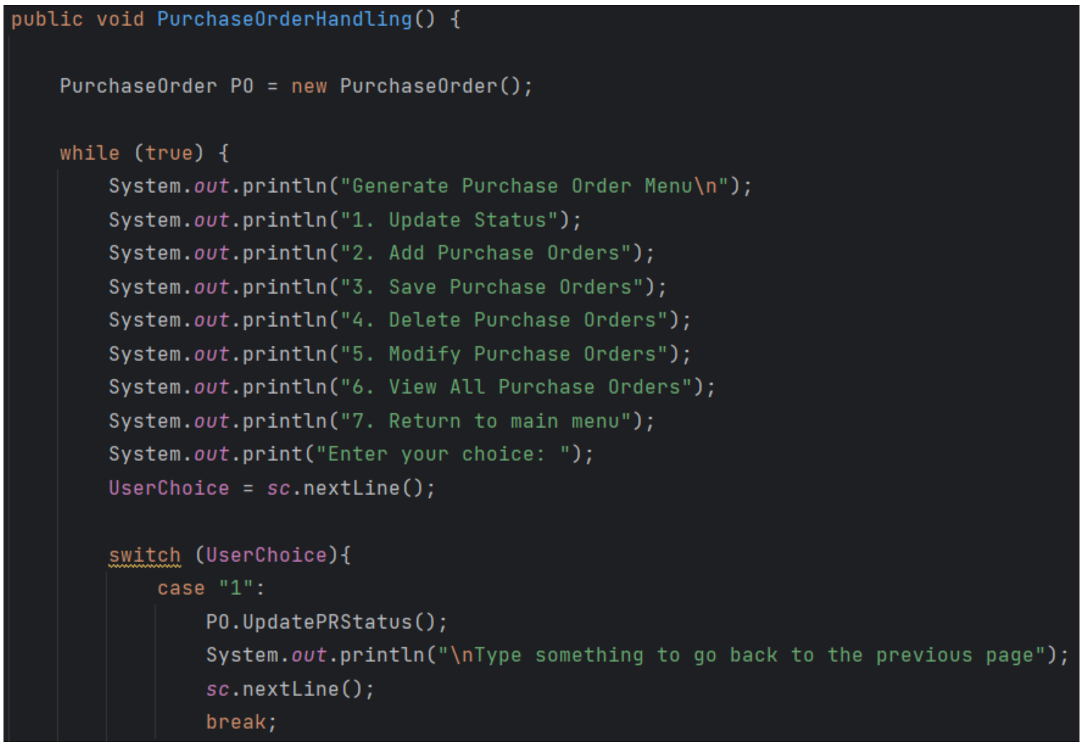
```
    public ItemsEntry(String Portion) {
        this.Portion = Portion;
    }

    2 usages
    public ItemsEntry(String ItemID, String SupplierID) {
        this.ID = ItemID;
        this.SupplierID = SupplierID;
    }
```

ItemsEntry Constructors

In the figure above shows two constructors in the ItemEntry class. The first constructor accepts a single parameter, “Portion” and initializes the “Portion” instance variable with the value passed as an argument. This allows for the creation of ItemEntry objects with a specified “Portion” value. Hence, providing customization during object instantiation. Meanwhile the second constructor takes two parameters, “ItemID” and “SupplierID”, and assigns these values to the “ID” and “SupplierID” instance variables respectively. These constructors showcase object-oriented programming’s emphasis on reusability, enabling the class to be employed in scenarios such as in SupplierEntry class, PurchaseManager class, SalesManager class, PurchaseOrder class, PurchaseRequisition class. These classes use the values that instance variables to be used to showcase code reusability and applying object-oriented programming principles. By offering different parameter combinations, it supports flexibility in object creation, allowing customization or predefined values. As a result, the constructors in the ItemsEntry class adhere to object-oriented programming principles, fostering encapsulation and reusability while enabling robust object initialization in Java.

Object

Object is the most essential part of Object-Oriented programming as an object is created based on a class. Class and object complement each other because the attributes and methods of a class is makes and object works. An object represents an entity, either physical, conceptual or software that is unique even if the attributes and methods are identical to another object. To create an object, we must instantiate using the word “new” then followed by the class’s constructor.



```java
public void PurchaseOrderHandling() {

    PurchaseOrder PO = new PurchaseOrder();

    while (true) {
        System.out.println("Generate Purchase Order Menu\n");
        System.out.println("1. Update Status");
        System.out.println("2. Add Purchase Orders");
        System.out.println("3. Save Purchase Orders");
        System.out.println("4. Delete Purchase Orders");
        System.out.println("5. Modify Purchase Orders");
        System.out.println("6. View All Purchase Orders");
        System.out.println("7. Return to main menu");
        System.out.print("Enter your choice: ");
        UserChoice = sc.nextLine();

        switch (UserChoice){
            case "1":
                PO.UpdatePRStatus();
                System.out.println("\nType something to go back to the previous page");
                sc.nextLine();
                break;
```

Figure 1

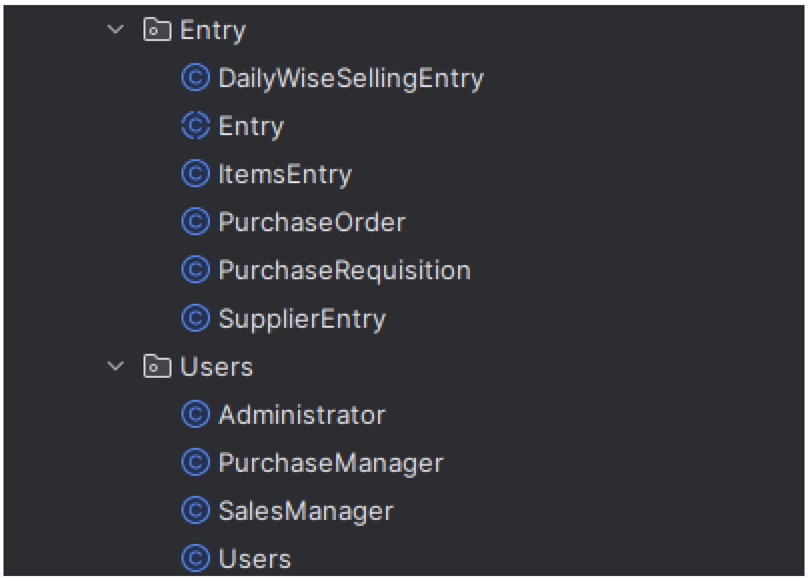
The screenshot above will provide a better understanding on how to create an object (or instance). To create an object for “PurchaseOrder()”, the first word should be the name of the class, followed by object name, new, and finally the class name again. Object “PO” is created and we can use the attributes and methods from “PurchaseOrder()” class and apply it to different classes as well without having to write the code from scratch which saves time and encourage code reusability. When an object has been created, we can implement the method from “PurchaseOrder()” class to “PurchaseOrderHandling()” class. The method that was implemented in the code is “UpdatePRStatus()” method, and in order to use the function, the syntax shall be “PO.UpdatePRStatus()” (name_of_object.name_of_method). This is called invocation of method. Instantiate an object is used throughout the whole development process as it enables us to use a variety of methods from multiple classes that helps us to maintain the readability and understanding the structure of how the classes correlates with other classes.

Inheritance

In the Java programming language, the concept of inheritance is used as a mechanism that enables an object to acquire and possess all the attributes and behaviours of its parent object. The inclusion of object-oriented programming (OOP) is a fundamental aspect (Java T Point).

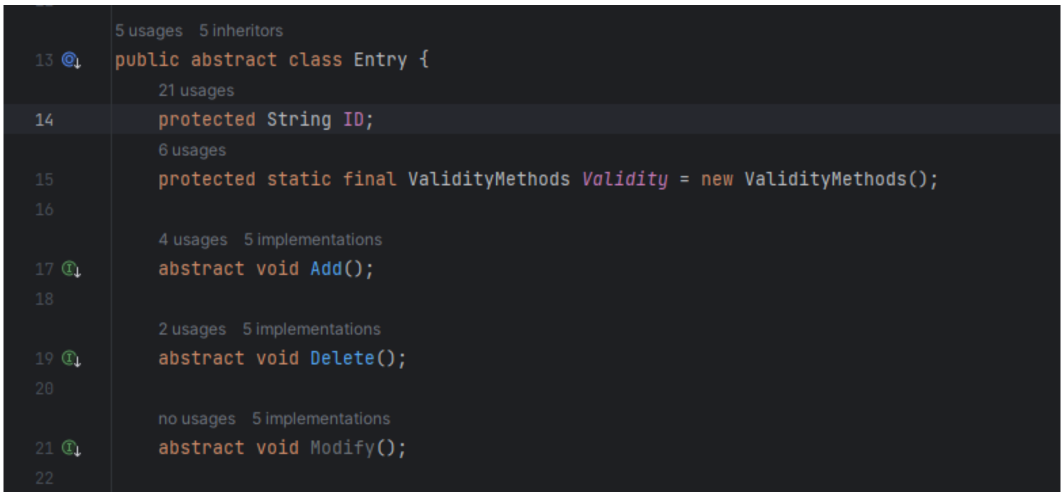
The concept of inheritance refers to the ability to create new classes that inherit their characteristics and behaviours from pre-existing classes. When extending a class, it is possible to use the methods and attributes of the parent class. Incorporation of additional methods and attributes into the pre-existing class is likewise permissible. (Java T Point, n.d.).

We have applied the inheritance in our project, the super classes are Entry and Users, you can see all the child classes in each package in the next figure:



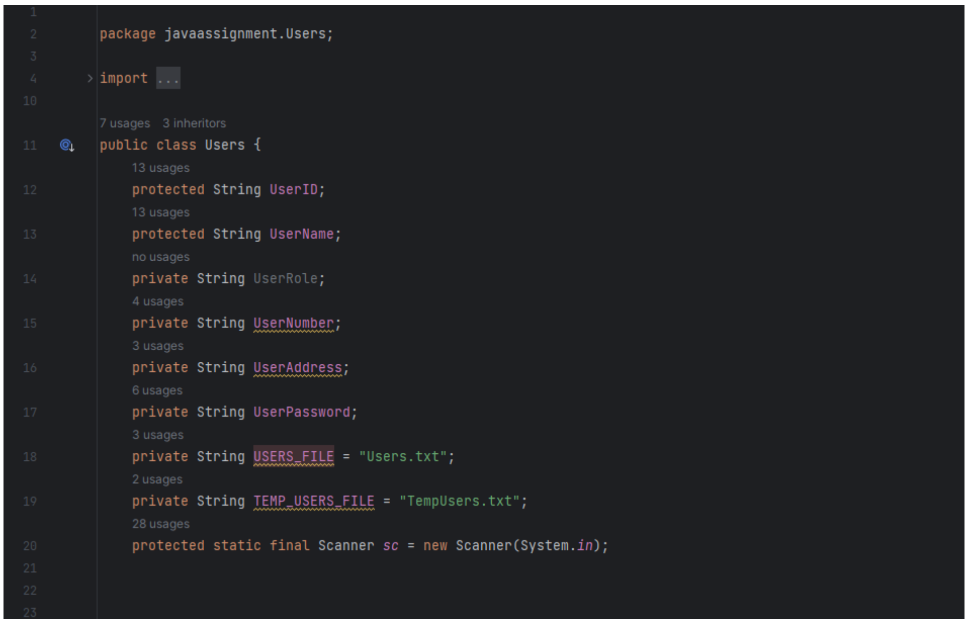
```
Entry
    DailyWiseSellingEntry
    Entry
    ItemsEntry
    PurchaseOrder
    PurchaseRequisition
    SupplierEntry
Users
    Administrator
    PurchaseManager
    SalesManager
    Users
```

The Entry class acts as an abstract foundation class for its subclasses, defining common methods and attributes. It offers abstract methods Add(), Delete(), and Modify() that child classes must implement. It also has methods for showing file contents in a certain format and getting input validation, such as displayFileContents(). Subclasses may inherit from this class to reuse these methods and attributes while tailoring their behaviour to certain sorts of entries such as Items, Suppliers, DailyWiseSales, PurchaseRequisition, and PurchaseOrder.



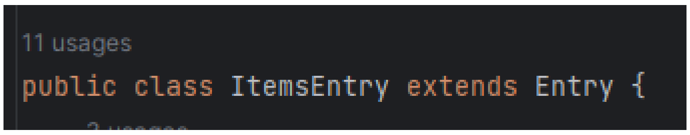
```
5 usages  5 inheritors
public abstract class Entry {
    21 usages
    protected String ID;
    6 usages
    protected static final ValidityMethods Validity = new ValidityMethods();

    4 usages  5 implementations
    abstract void Add();

    2 usages  5 implementations
    abstract void Delete();

    no usages  5 implementations
    abstract void Modify();
```

The Users class is a fundamental class for managing users in a Java program and provides a simple user management system. Other classes may inherit from this class to utilise its user management capabilities, enabling them to customize and extend user-related features while utilizing the common user data management logic. This usage of inheritance aids in the maintenance of code consistency and reusability across various user types or roles inside the program.



```

package javaassignment.Users;

import ...

7 usages  3 inheritors
public class Users {
    13 usages
    protected String UserID;
    13 usages
    protected String UserName;
    no usages
    private String UserRole;
    4 usages
    private String UserNumber;
    3 usages
    private String UserAddress;
    6 usages
    private String UserPassword;
    3 usages
    private String USERS_FILE = "Users.txt";
    2 usages
    private String TEMP_USERS_FILE = "TempUsers.txt";
    28 usages
    protected static final Scanner sc = new Scanner(System.in);
```

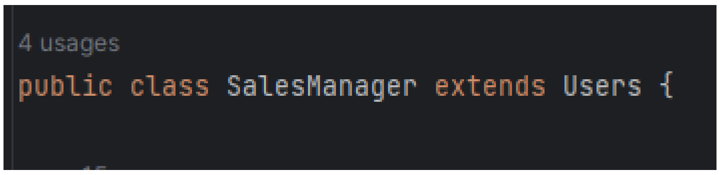
For example, of how to extend the Entry class to one of the child classes:



```
11 usages
public class ItemsEntry extends Entry {
```

Using the keyword extends with the class name is how to inherit the classes, extend is valid to use once in each class, the attempt of extending two classes into one class will result to a compilation error.

And here is an example of extending the User class using extends keyword in one of the child classes.



```
4 usages
public class SalesManager extends Users {
```

The same thing is applied to all the classes related to Entry and Users classes.

Finally, inheritance is important in Java programming because it allows new classes to inherit both the attributes and behaviours of parent classes. In our project, we used inheritance efficiently to create a class hierarchy, with the Entry class acting as a foundation for different entry-related subclasses and the Users class providing user management features that may be modified by child classes. This inheritance-based structure increases consistency and facilitates the administration of similar functionality across multiple software components. In general, inheritance is a key notion in object-oriented programming that helps with code structure and the development of more maintainable and extensible software systems.

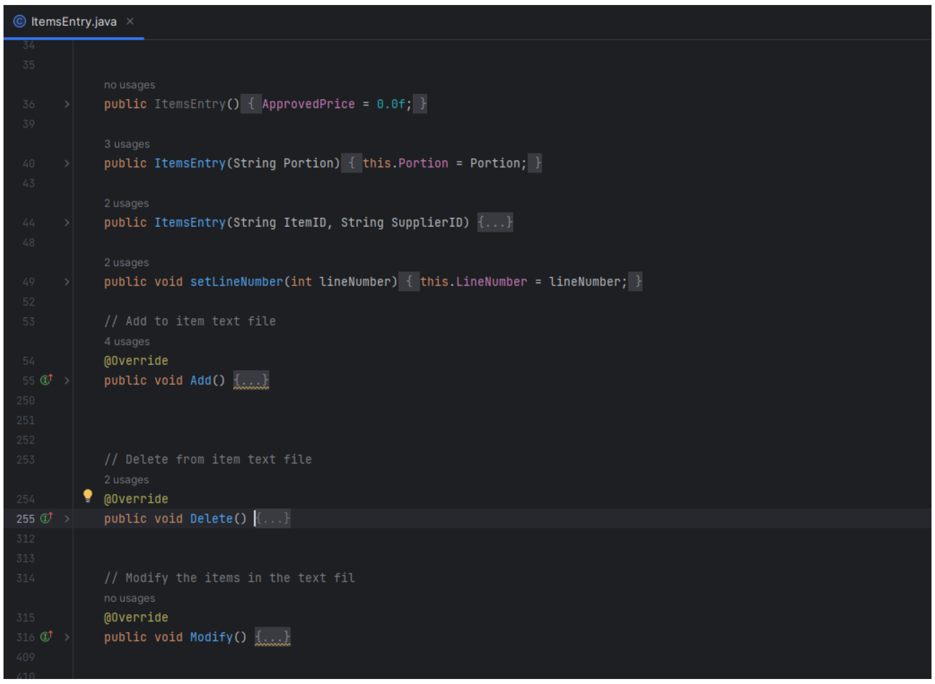
Polymorphism

When something has polymorphism, it may take on several forms. In Object-Oriented Programming, polymorphism is most often used when referring to an object of a child class by referencing its parent class (Tutorialspoint).

Understanding that a variable is required to obtain an object is crucial. A reference variable may only be one type. Once a reference variable is defined, its type cannot be altered (Tutorialspoint).

Overriding

A class may inherit similar behaviour from a superclass and then tailor that behaviour to its own needs thanks to the ability of a subclass to override a method. When a method is overridden, it must have the same name, number, and types of arguments, and return type as the method it is replacing (Oracle).


ItemsEntry.java

```
    no usages
    public ItemsEntry() { ApprovedPrice = 0.0f; }

    3 usages
    public ItemsEntry(String Portion) { this.Portion = Portion; }

    2 usages
    public ItemsEntry(String ItemID, String SupplierID) {...}

    2 usages
    public void setLineNumber(int lineNumber) { this.LineNumber = lineNumber; }

    // Add to item text file
    4 usages
    @Override
    public void Add() {...}



    // Delete from item text file
    2 usages
    @Override
    public void Delete() {...}



    // Modify the items in the text fil
    no usages
    @Override
    public void Modify() {...}
```

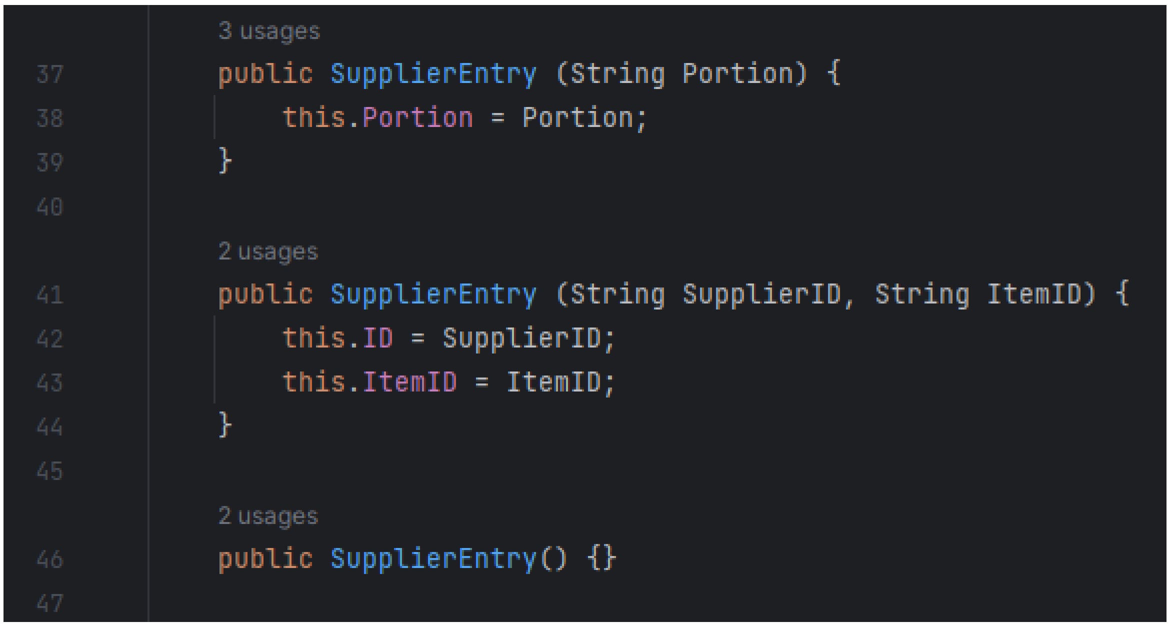
Overriding was applied in this project most of the time, Method overriding is used in the ItemsEntry, SupplierEntry, Daily item-wise sales, PR and PO class to provide individual implementations of the Add(), Delete(), and Modify() methods, which are inherited from the parent abstract class Entry. These overridden methods modify the behavior of item entry operations (addition, deletion, and modification) to meet the needs of the application's item management. For example, the Add() method modifies the parent function to handle both new item addition and supplier addition in the ItemsEntry class to existing items. Similarly, the

Delete() method overrides the parent method to delete item records and, if required, related supplier entries. Finally, the Modify() method overrides the parent method to allow for the changing of item attributes while engaging with the user to choose which parts of the item should be changed.

Overloading

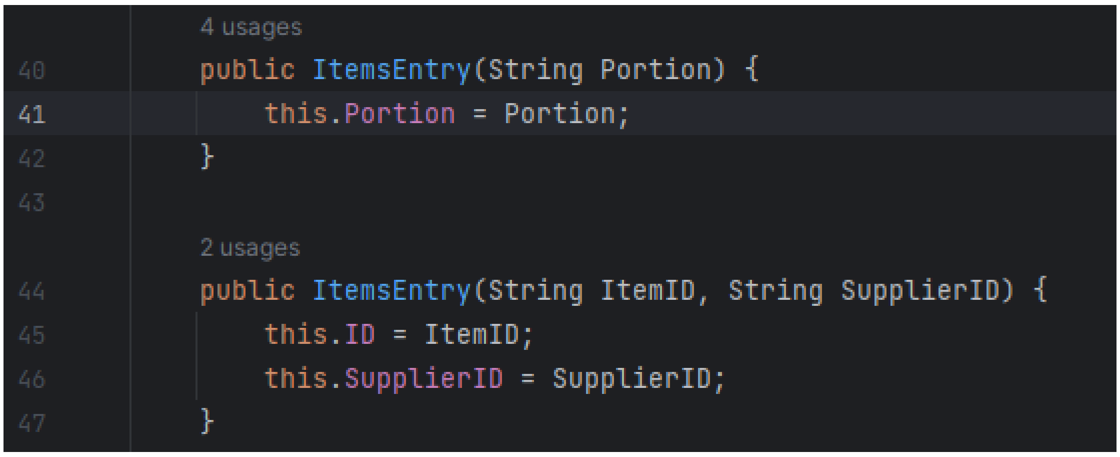
The feature of method overloading in Java allows a class to define numerous Methods with the same name but distinct argument types. The application of this notion was implemented in the project.

In one of the cases, constructors for SupplierEntry initialize instance variables. Public SupplierEntry(String Portion) is the first of this class's three constructors. It initialises the Portion instance variable to denote whether the entry is for a supplier or an item. The second constructor, public SupplierEntry(String SupplierID, String ItemID), initialises the instance variables required to represent a supplier-item relationship. The third constructor, SupplierEntry(), creates a SupplierEntry object without instance variable values **by default**.



```
3 usages
public SupplierEntry (String Portion) {
    this.Portion = Portion;
}

2 usages
public SupplierEntry (String SupplierID, String ItemID) {
    this.ID = SupplierID;
    this.ItemID = ItemID;
}

2 usages
public SupplierEntry() {}
```

Another place we can have a look to is the ItemsEntry class, it uses constructors to initialise instance variables and provide alternative methods for creating instances of the class. The public ItemsEntry(String Portion) constructor is used to initialise the Portion instance variable when adding a new item or a new supplier to an item. The second instance constructor, public ItemsEntry(String ItemID, String SupplierID), is used to initialise the ID and SupplierID instance variables with an item's ID and its associated supplier's ID. These constructors make interacting with ItemsEntry objects more flexible by enabling them to be generated with varied initializations depending on the use case.

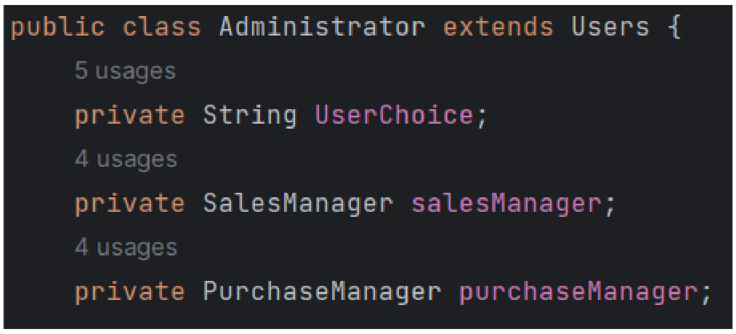


```
    4 usages
    public ItemsEntry(String Portion) {
        this.Portion = Portion;
    }

    2 usages
    public ItemsEntry(String ItemID, String SupplierID) {
        this.ID = ItemID;
        this.SupplierID = SupplierID;
    }
```

Method overloading was used efficiently in our project to build numerous constructors with the same class name but different arguments. In the SupplierEntry class, for example, we have three constructors that allow for varied initializations depending on the environment. Constructors were also utilised in the ItemsEntry class to specify instance variables and enable flexibility in object building, in addition to the other classes in the project. By limiting constructor use to particular contexts, this method simplifies object instantiation and improves code readability, making the project more flexible and adaptive.

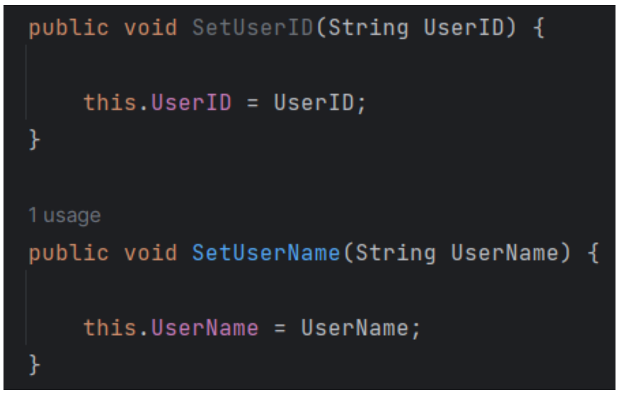
Encapsulation

Encapsulation is one of the fundamentals in Object Oriented programming. Encapsulation allows the data that is being used to be hidden. Typically, the data that is declared in a class is either protected or private, preventing data loss from happening. Most of the time, data that is worth encapsulating is sensitive and private data such as ID number and password. In addition, encapsulation will hide the functionality of our class along with the data structure and algorithm.

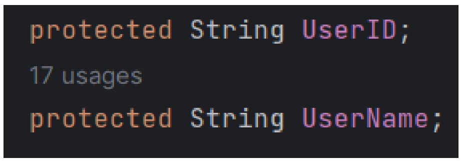


```
public class Administrator extends Users {
    5 usages
    private String UserChoice;
    4 usages
    private SalesManager salesManager;
    4 usages
    private PurchaseManager purchaseManager;
```

Based on the code snippet above, the instances variables are created using private access modifier, which means that they are only accessible within the class itself only. By making those variables private, their internal state is hidden and be used or modified through Administrator class only.



```
public void SetUserID(String UserID) {

    this.UserID = UserID;
}

1 usage
public void SetUserName(String UserName) {

    this.UserName = UserName;
}
```

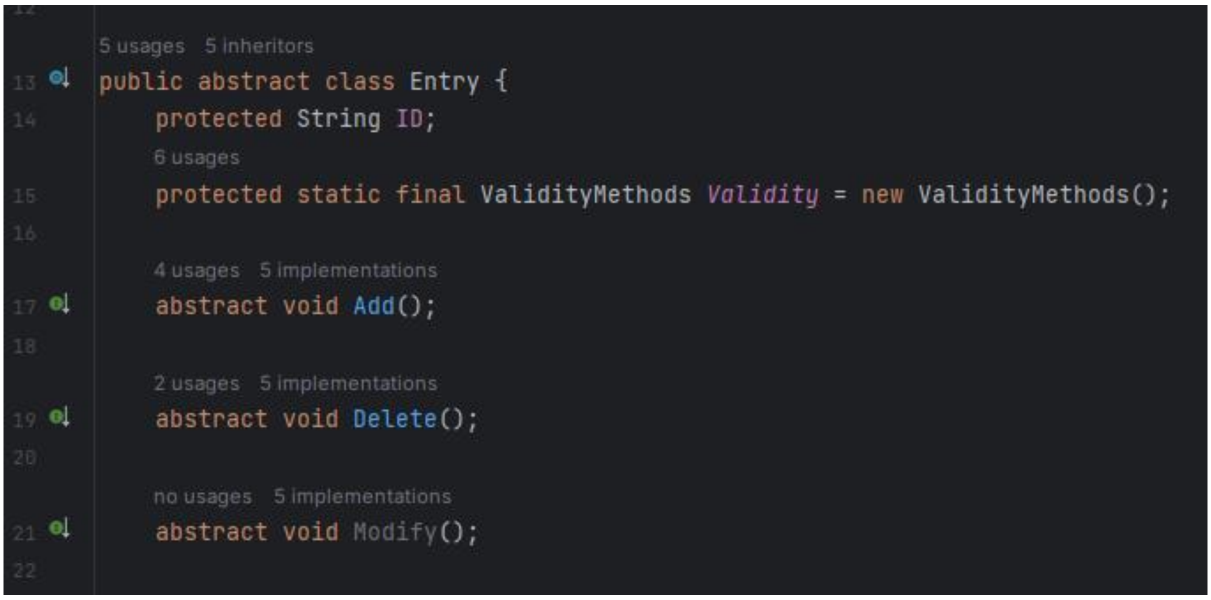


```
protected String UserID;
17 usages
protected String UserName;
```

Figure 1 Figure 2

This is another example of applying encapsulation to a class. “SetUserID” and “SetUserName” (refer Figure 1) are used to set the values of protected instance variable “UserID” and “UserName” as shown in Figure 2. By using the setter methods, the class can impose regulations or validations, preserving the data’s integrity.

Abstraction



```
5 usages  5 inheritors
public abstract class Entry {
    protected String ID;
    6 usages
    protected static final ValidityMethods Validity = new ValidityMethods();

    4 usages  5 implementations
    abstract void Add();

    2 usages  5 implementations
    abstract void Delete();

    no usages  5 implementations
    abstract void Modify();

```

Abstract Class Entry

Abstraction involves focusing on ideas and concepts rather than the specific details of events or processes. The abstraction shields users from the intricate mechanics, allowing them to perform tasks without delving into technical complexities (Tutorialspoint, n.d.). The Entry class is declared as abstract using the abstract keyword indicating that it cannot be instantiated on its own. This class will then encapsulate common functionalities like displayFileContents that has String Filename as the parameter, ValidityMethods where it validates user input throughout the whole package, and dynamic formatting where the class will format the data for display, allowing the presentation of data specific to different entry types to be flexible.

The Entry class also defines a contract for concrete subclasses to implement such as ItemsEntry and SuppliersEntry, DailyWiseSellingEntry, PurchaseRequisition and PurchaseOrder. These subclasses must override these methods and provide specific implementations based on their requirements. The Entry class has abstract methods as well which are Add, Delete and Modify. These abstract methods outline common operations for data entry and management. The abstract methods ensure that certain behaviours are consistent across all subclasses while allowing flexibility in their actual implementations.

SYSTEM LIMITATIONS

User Interface: Instead of implementing a Graphical User Interface (GUI), a Command Line Interface (CLI) was chosen for this undertaking. This option represents a system limitation. However, it is worth noting that enhancing the system to integrate a GUI would provide users with a more flexible and user-friendly experience, making it more comfortable to interact with the system.

Scalability: Issues The system's scalability may become problematic since the system is applying text files as its data store, as it may encounter limitations when administering larger datasets or accommodating ever-increasing operational demands. This limitation may be caused by factors such as processing capacity, memory constraints, or inefficient algorithms, all of which hinder the system's ability to scale up efficiently to meet increasing user demand.

Security: The provided System lacks high level of input validation, access control for file operations, and encryption for sensitive data, making it subject to potential security threats such as injection attacks and unauthorised file access. Additionally, hardcoded credentials if present, can pose a security concern. In the near future, however, these security issues could be resolved through the implementation of secure coding practises, routine code audits, and the incorporation of security libraries or frameworks. By proactively addressing these issues, the system can improve its overall security posture and guarantee the protection of vital data.

CONCLUSION

The successful completion of this assignment is a turning point in our journey towards understanding the fundamentals and mastering Object-Oriented Programming and fully utilize what we have learned in developing and improving programs with Java. The main goal of this assignment was to plan, design, implement and develop a fully workable application of Purchase Order Management (POM) system in Java while applying the 4 main concepts of Object-Oriented Programming which are inheritance, polymorphism, abstraction, and encapsulation. Throughout the whole process of completing this assignment, we have not only gained a thorough grasp of the basic principles of Object-Oriented Programming, but we have also improved our ability to properly plan, organize and implement complex Object-Oriented Programming frameworks collectively.

One of the important accomplishments that we are proud of is to develop an application that is dynamic and user-friendly menu driven Purchase Order Management (POM) system. To simplify and improve the efficiency for SIGMA SDN BHD (SSB) to automate the management process of Purchase Order, Purchase Requisition, and Supplier management. This system provides a variety of functions, and a role-based authentication makes this system prominent because every role has a different access to the system. The three user roles are: Administrator, Purchase Manager and Sales Manager.

Sales Manager was given a unique access privilege that were tailored to their role. They are allowed to carry out necessary functions like creating a purchase requisition, item entry, supplier entry, daily-item wise sales entry, and view requisition. As for Purchase Manager, they are given the privilege to view list of items, suppliers, and requisitions. They also have to ability to generate purchase order and view the list of purchase order. Lastly, Administrator role have the greatest level of privilege as they have the complete control of all functionalities that are present in the system which includes the privilege of Sales and Purchase Manager.

In conclusion, with hands-on experience in developing this system, we have managed to enhance our skills and knowledge to conceptualize complex system through Object Oriented Programming's four main principles of inheritance (code reusability), polymorphism (dynamism and extensibility), encapsulation (ensuring data integrity), and abstraction. The skills we have gained would not only help us in our assignment but also applying it in real-world software development.

REFERENCES

Java T Point. (n.d.). *Java T Point*. Retrieved from Inheritance in Java: https://www.javatpoint.com/inheritance-in-java

Oracle. (n.d.). *Overriding and Hiding Methods*. Retrieved from Oracle Java documentation : https://docs.oracle.com/javase/tutorial/java/IandI/override.html#:~:text=The%20ability%20of%20a%20subclass,the%20method%20that%20it%20overrides.

Pankaj. (4 August, 2022). *DigitalOcean*. Retrieved from DigitalOcean: https://www.digitalocean.com/community/tutorials/constructor-in-java

Tutorialspoint. (n.d.). Retrieved from Tutorialspoint: https://www.tutorialspoint.com/java/java_abstraction.htm

Tutorialspoint. (n.d.). *Java - Polymorphism*. Retrieved from Tutorialspoint: https://www.tutorialspoint.com/java/java_polymorphism.htm#:~:text=Polymorphism%20is%20the%20ability%20of,is%20considered%20to%20be%20polymorphic.