

**INDIVIDUAL ASSIGNMENT**

**TECHNOLOGY PARK MALAYSIA**

**CT038-3.5-2-OODJ**

**PROGRAMMING IN CPP**

**NAME: PEDRO FABIAN OWONO ONDO MANGUE (TP063251)**

**HAND-IN DATE: 8 JUNE 2023**

Contents

[INTRODUCTION 3](#_Toc137239448)

[USE CASE DIAGRAM 4](#_Toc137239449)

[CLASS DIAGRAM 5](#_Toc137239450)

[IMPLEMENTATION OF OOP CONCEPTS 7](#_Toc137239451)

[SCREENSHOTS OF THE OUTPUT OF THE PROGRAM 11](#_Toc137239452)

[CONCLUSION 21](#_Toc137239453)

[**REFERENCES** 22](#_Toc137239454)

# INTRODUCTION

Nowadays, since we are getting involved in the digital world. Some companies and industries are getting worried about how they can expand their businesses through digitalisation methods, meaning that, they are thinking to innovate and bringing new ways or mechanisms to significantly increase their capital or economy. So basically, as far as this project is concerned, I am designing an inventory system or a purchase management system where there are three roles or main actors which are:

* Admin
* Purchase Manager
* Sales Manager

They perform different functionalities in the system and their functionalities in the system is as follows:

The first role is the Purchase Manager which provides the following features:

* It can check or view the list of items of the complete system.
* It can check or view the list of the present suppliers of the system.
* It can display or show the Requisition of the complete system.
* Now, the Purchase Manager can manipulate (add, save, drop, and edit) the data for the Purchase Order or generate a purchase order.
* It can finally view the list of Purchase Orders.

The second role is the Sales Manager which provides the following features:

The Sales Manager can manipulate (add, save, delete, and edit) the data for the following.

elements:

* Item Entry
* Supplier
* Daily Item-wise Sales Entry
* Create Purchase Requisition

Finally, it can view the information of the following data:

* Display Requisition
* List of Purchase Orders

Lastly, the final role is the admin who has access to all the functionalities as it can create unpracticed users in the system based on the three roles of the system.

# USE CASE DIAGRAM

Purchase Manager

<<include>>

<<extend>>

<<include>>

<<extend>>

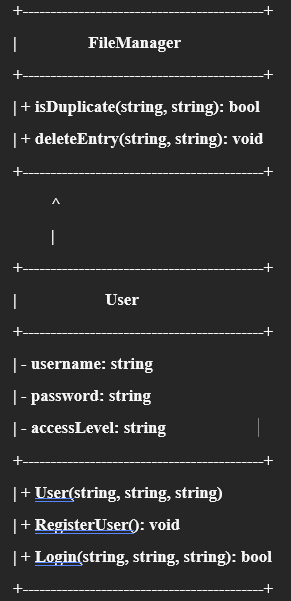
<<extend>>

Sales Manager

Admin

# CLASS DIAGRAM

**A screenshot of a computer program

Description automatically generated with low confidence**

**A screenshot of a computer code

Description automatically generated with low confidence A screen shot of a computer

Description automatically generated with medium confidence**

**A screen shot of a computer program

Description automatically generated with low confidenceA screenshot of a computer program

Description automatically generated with low confidence**

|  |
| --- |
|  |

**A picture containing text, font, screenshot, line

Description automatically generated A screenshot of a computer application

Description automatically generated with low confidence**

# IMPLEMENTATION OF OOP CONCEPTS

There are several object-oriented programming concepts that I have implemented in my program:

* Classes and objects:

Considering or knowing that when we speak about OOP concepts there should be a presence of classes and objects as they are one of the essential characteristics of OOP concepts. As far as my program is concerned, my program is built in a way that I have created classes and objects such as Supplier, Item, SalesEntry, PurchaseRequisition, PurchaseOrder, Application, and some other relevant classes which object I have used to play or manipulate the behaviour of the attributes in some classes used in my program.

A picture containing text, font, screenshot, number

Description automatically generated A screenshot of a computer code

Description automatically generated with low confidence A screenshot of a computer code

Description automatically generated with low confidence

Illustration 1: examples of user, item, and supplier classes

* Inheritance:

Basically, to manipulate my data easily in the program I must implement inheritance for a better organization of my classes based on their information to save them in a text file through the use of vectors.

A picture containing text, screenshot, font, number

Description automatically generated

Illustration 2\_0: Base class File Manager first part

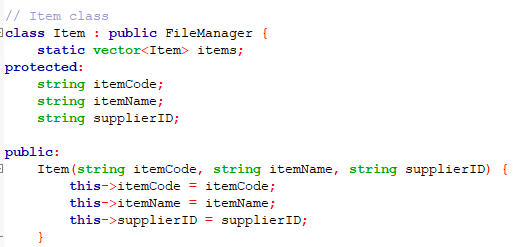


Illustration 2\_1: class Item which inherits File Manager class.

A screen shot of a computer code

Description automatically generated with low confidence

Illustration 2\_2: class Supplier inherits FileManager class.

Some other classes such as SalesEntry, PurchaseRequisition, and PurchaseOrder inherit from the base class which is FileManager. Based on the information discussed early we can say that the OOP concept of inheritance is clearly proved in the program.

* Encapsulation:

In my program, you can appreciate the implementation of an encapsulation. In my code, the data members and methods are encapsulated within classes. For example, in my code basically, the class named User is an example of encapsulation as its data member’s username, password, & accessLevel are set as private and the only way that they can be accessed and changed is through the methods in this class such as RegisterUser. One of the case scenarios of encapsulation in my program is illustrated in the following illustration 3:

A screen shot of a computer code

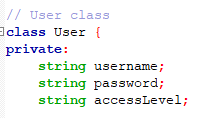
Description automatically generated with low confidence

Illustration 3\_0: User class with its method RegisterUser

* Data Abstraction:

In my program, I used one of the key points for OOP. Firstly, let’s define what Data Abstraction is.

Firstly, if we start from abstraction, we can say that it is a way of showing only the significant or essential information of your program while hiding the details. So basically, I am using abstraction because I defined methods in my classes that perform specific tasks while hiding the implementation details from the user. Now an example of data abstraction in my code would be Item, Supplier, SalesEntry, PurchaseRequisition and User class because they mainly encapsulate the method and provide certain methods to interact with those data.

An example of data abstraction, for example, is the application class because it provides an interface to the user, hiding the implementation details. It could also be a good example of encapsulation.

A screenshot of a computer program

Description automatically generated with low confidence

Illustration 4: Application class

* Polymorphism:

In my program, I have implemented a clear example of polymorphism which is called the HandleUserInput method inside the Application class. The HandleInput method is an example of polymorphism because it is used differently based on the role in this case the accessLevel, meaning if I access as an admin, I will have certain functionalities in the program while if I access for example as the sales manager then I will have different functionalities in the program. The next Illustration will reflect it:

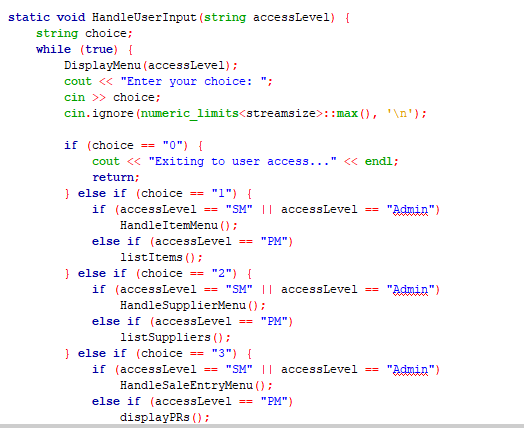


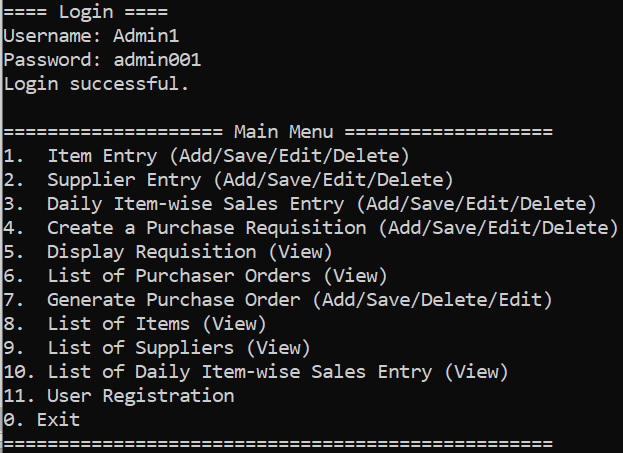
Illustration 5: the HandleInput method

# SCREENSHOTS OF THE OUTPUT OF THE PROGRAM

A screenshot of a computer program

Description automatically generated with low confidenceLOGIN ACCESS:

Illustration 6: login page

Once I run the program the first output that will pop up in the screen is the user access with the three main roles or actor which are admin, sales manager, and purchase manager.

A screenshot of a computer

Description automatically generatedIllustration 7: Login page for admin

So basically, I have logged in as admin and one I press the correct credential it is going to display to me all the functionalities for admin as well as user registration as required for the assignment and at the end, I have the option to exit the program as well.

Now, If I choose option 2 it is going to display the access level as sales manager, then it might ask me for the credentials and then after that, if the password and username are correct then the system will only show me the functionalities of sales manager as you can see in the illustration 8.

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

Illustration 8: login for sales manager

A screenshot of a computer

Description automatically generatedIn the same way, it will happen with the purchase manager access level or user, and you can see it in illustration 9:

A screenshot of a computer

Description automatically generated with medium confidenceIllustration 9: login for purchase manager

USER REGISTRATION

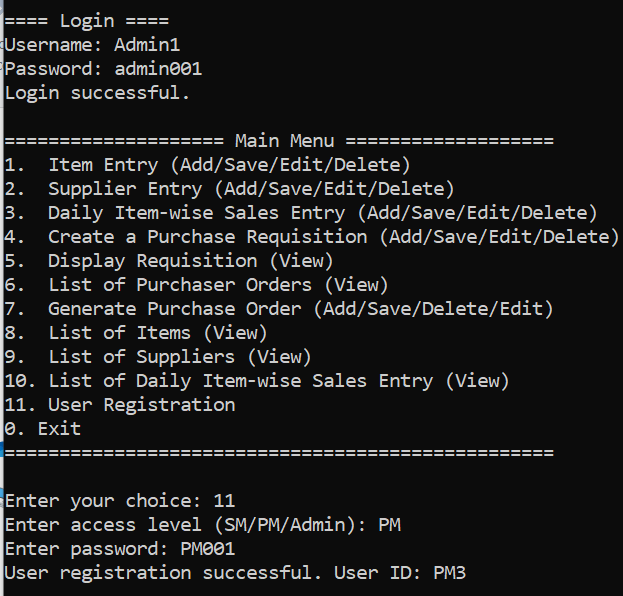
For registration basically the administrator is the only user that have the functionality of creating new users in the system based on their access level, meaning whether the user is PM or SM.

Illustration 10: registration for PM

The next illustration 11 is when the admin creates a new user as Sales Manager.

A screenshot of a computer program

Description automatically generated with medium confidence

Illustration 12: registration for SM

Now, there is something that you must notice in the program. When the admin creates a new user, then it will automatically assign that user a unique ID which is going to be stored in a file called users. Basically, the unique ID for the users in the system is set as the username which is going to go through the first line of the file and then it will generate a unique ID by a sum or increment of 1 which is going to make the users have different username and a unique one. I set the userID feature in my program for username.

ITEM ENTRY AND SUPPLY ENTRY

For this section we must highlight that the users that got access to them are the three main users in the program. Let us try to access into it as the sales manager:

A screen shot of a computer

Description automatically generated with low confidenceA picture containing text, screenshot, software, font

Description automatically generated

Illustration 12: Functionalities of the Sales Manager, Item Entry (Add and Save)

A black screen with white text

Description automatically generated with low confidence

Illustration 13: Functionalities of the Sales Manager, Item Entry (editing Item)

Here is the representation of supply entry for sales manager:

A screenshot of a computer

Description automatically generated

Illustration 14: Functionalities of the Sales Manager, Supply Entry

Now, I am going to show the illustration of the supply entry from Sales Manager

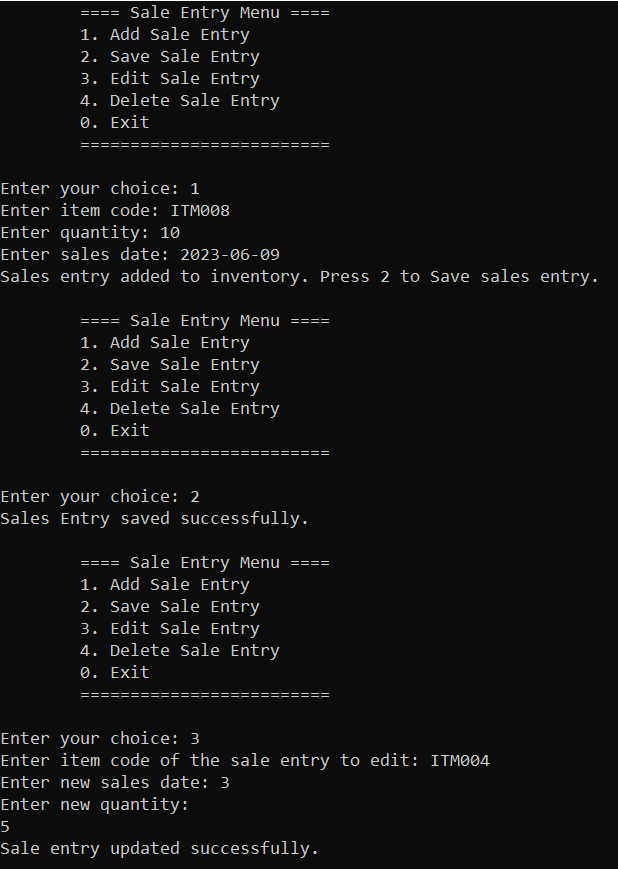


Illustration 14: Operations as Sales Manager for Sale Entry

Now, the next functionality for the Sales Manager is create a purchase requisition which is going to be represented in the next illustration:

A screenshot of a computer program

Description automatically generated with medium confidence

Illustration 15: Sales Manager operations for Purchase Requisition

A screenshot of a computer program

Description automatically generated with medium confidence

Illustration 16: Purchase Requisition for Sales Manager edit and delete.

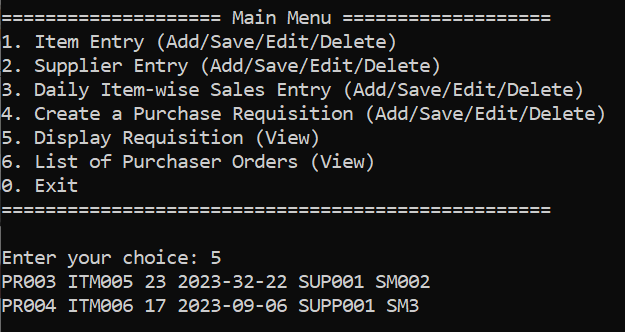
Now I am going to show other functionalities for Sales Managers such as Display Purchase Requisition:  


Illustration 17: Display Requisition for Sales Manager

Finally, the last session for the Sales Manager is to show the list of Purchase Orders available in the system.

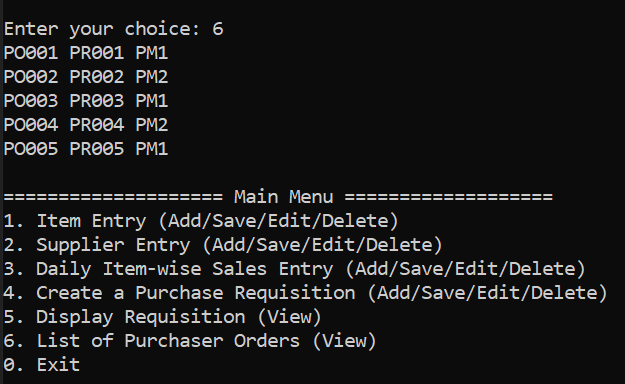


Illustration 18: Display list of Purchase Oder

Now, the functionalities of the Purchase Manager are as follows:

A screenshot of a computer

Description automatically generated

Illustration 19: Purchase Manager Menu

A picture containing text, font, screenshot

Description automatically generatedA screenshot of a computer program

Description automatically generated with medium confidenceA picture containing text, font, screenshot, typography

Description automatically generated

Illustration 20: Purchase Manager for list of items, suppliers, display requisition

A screenshot of a computer

Description automatically generatedNow for generating a purchase order, I have set in my program that the purchase manager is the one who is the one that is going to manipulate the data from it.  
Illustration 21: Purchase Manager functionality for generating a purchase order.

In the end, the purchase manager has the option to display the list of orders.

A screenshot of a computer screen

Description automatically generated with low confidence

Illustration 22: Purchase Manager functionality for displaying the list of orders.

Finally, we have a presence of the admin in the program which got all the functionalities in the system and can do any task as well as create new users and add them to the system.

A screenshot of a computer

Description automatically generated

Illustration: Administrator functionalities

PURCHASE MANAGEMENT FILES:

Five files have been used in this project and they are the following:

* Purchase requisition
* Sales
* Suppliers
* items
* Suppliers
* Users

A screenshot of a computer

Description automatically generated with medium confidence

* Purchase orders

A picture containing text, screenshot, font, line

Description automatically generated

* Purchase requisition

A screenshot of a computer

Description automatically generated

* items

A screenshot of a computer screen

Description automatically generated with medium confidence

* Suppliers

A screenshot of a computer

Description automatically generated

* Users

# CONCLUSION

Finally, at this stage, I have completed the assignment and fulfilled all the criteria. I have implemented object-oriented programming (OOP) concepts such as polymorphism, data abstraction, inheritance, encapsulation and the use of classes and objects.

# **REFERENCES**

* <https://profile.w3schools.com/refresh-session?redirect_url=https%3A%2F%2Fwww.w3schools.com%2Fjava%2Fdefault.asp>

W3Schools C++

* <https://www.geeksforgeeks.org/different-ways-reading-text-file-java/>

Different ways of reading text file C++, 20 December 2021

* <https://youtu.be/A74TOX803D0>

Java Tutorial, freeCodeCamp.org, YouTube