GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

NAME :	Soong Gim Hoy	TP053242
	Chan Jia Le	TP049952

INTAKE CODE :	UC2F1908CS (DA)
LECTURE NAME :	LEE KIM KEONG
SUBJECT :	OBJECT-ORIENTED DEVELOPMENT
MODULE CODE :	CT038-3-2-OODJ

Contents

1.0 | Introduction

This documentation includes the development of a product catalogue management system using object-oriented programming concepts. The coding language used is Java and the Integrated Development Environment used is NetBeans IDE 8.2. The system has been designed and developed using object-oriented approach covering object orientation concepts and principles. The object-oriented approach of the system will be explained and defined in this documentation. The product catalogue management system is developed in order to allow for easier visualization and modification of user details, product catalogues, items and categories. There will be 3 types of users utilizing this system which is Administrator, Product Manager and Supplier. Each user account type or position has certain limitation and functionality, and those functionalities and limitations will be covered in the assumptions. Each user will have a main page that displays a dashboard that contains details of accounts or product details based on their authority and position. Use cases and Class Diagram are developed and provided in the documentation to allow a clearer understanding of each users' functionality.

1.1 | Assumptions

The system is expected to be utilize by users with computer literacy. Thus, it is expected that most users will provide appropriate inputs while performing task using the system. It is assumed that there will be 3 users containing Administrator, Product Manager and Supplier. All users can modify and edit their personal details except for their User ID. All accounts are created and registered by the Administrator; each user contains a unique User ID that will be generated by the system. Admin has the authority to modify all details of supplier and product manager except for user ID and password. Accounts with no activity will be modified to Inactive by the Admin but it is not allowed to be deleted for managing purposes. Both administrator and product manager can manage (add/update/delete) product category and item, but only product manager can manage product catalogue. Supplier is only allowed to add product items. In order to achieve the functionalities, user must be active, if the user is inactive, all manage functionality will be closed for the user.

The system can only contain 999 Admin, Product Manager and Supplier. Upon the start of the system, system will auto generate a master account for all users, in order to have a more user-friendly system. Login history is only accessible to Admin and it can be exported to PDF. Special character is not encouraged to be input by users but can still be accepted. The system will be able to operate flawlessly if it is used with proper input by users.

2.0 | Requirement Analysis

2.1 | Use-Case Diagram

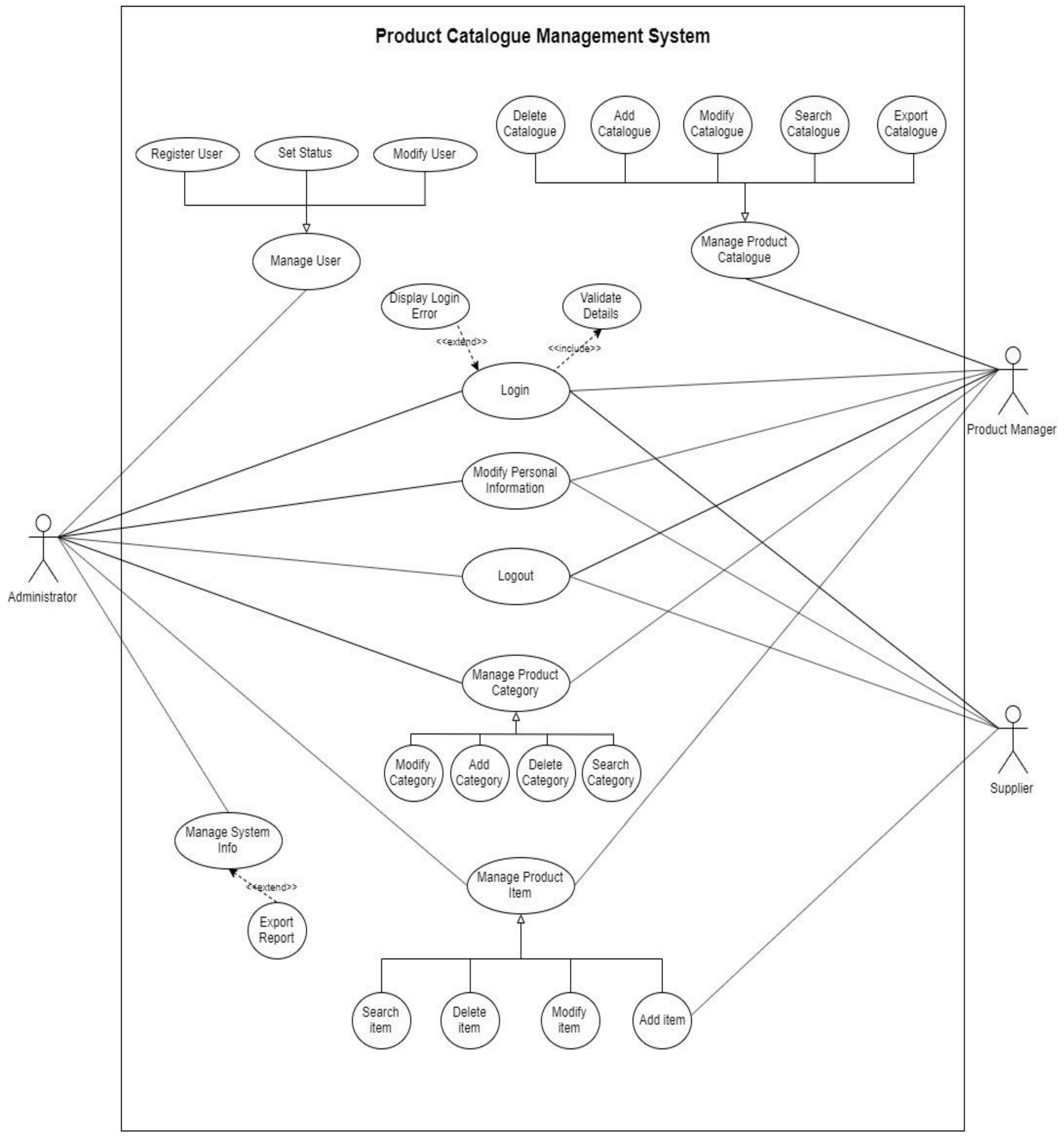


Figure 2.1 Use Case Diagram

2.2 | Use Case Specification

2.2.1 | Login

Name	Login
Description	The intention of this use case is to allow user to access the system
Actors	Administrator, Product Manager and Supplier
Pre-condition	User's account was registered as Administrator, Product Manager or Supplier
Main Flow	1. This use case begins when user enters the system 2. Actors gives their user id and password as input 3. System validates the user id and password from data files.
Alternative Flow	User entered user id or password incorrectly and will proceed to "Display Login Error" use case

2.2.2 | Validate Details

Name	Validate Details
Description	The intention of this use case is to validate user id and password
Actors	Administrator, Product Manager and Supplier
Pre-condition	User entered user id and password
Main Flow	1. Actors gives their user id and password as input 2. System checks data file to determine if user id and password is valid 3. System proceeds to main page 4. System records user id, login time, validation to a data file
Alternative Flow	User entered user id or password incorrectly and will proceed to "Display Login Error" use case

2.2.3 | Login Error

Name	Validate Details
Description	The intention of this use case is to notify actors that the user id and password input is invalid.
Actors	Administrator, Product Manager and Supplier
Pre-condition	User entered user id and password incorrectly
Main Flow	1. Use case starts when “Login” or “Validation” use case gets into Alternative Flow 2. System shows a notice box to notify the unsuccessful login reason 3. Return to the “Login” use case
Alternative Flow	-

2.2.4 | Modify Personal Information

Name	Modify Personal Information
Description	The intention of this use case is to allow users to amend personal profile
Actors	Administrator, Product Manager and Supplier
Pre-condition	Upon successful login to the system
Main Flow	1. This use case begins when actor request to amend personal profile 2. Actors enters information that is desired to be updated 3. System updates actor information to the data file. 4. Profile updated successfully
Alternative Flow	1. The actor information that is empty will be requested to be filled by the actor.

2.2.5 | Logout

Name	Logout
Description	The intention of this use case is to allow users to logout from the system
Actors	Administrator, Product Manager and Supplier
Pre-condition	Upon successful login to the system
Main Flow	1. This use case begins when actor request to logout from the system 2. System will store the user id, logout time to the data file. 3. User logout successfully
Alternative Flow	-

2.2.6 | Manage Product Category

Name	Manage Product Category
Description	The intention of this use case is to manage product category information for instance Add, Modify, Remove and Search
Actors	Administrator and Product Manager
Pre-condition	Actors request to manage product category
Main Flow	1. This use case begins when actor request to manage product category 2. Actor request to Add product category - Proceed to “Add Category” use case. 3. Actor request to Modify product category - Proceed to “Modify Category” use case. 4. Actor request to Remove product category - Proceed to “Remove Category” use case. 5. Actor request to Search product category - Proceed to “Add Category” use case
Alternative Flow	-

2.2.7 | Modify Product Category

Name	Modify Category
Description	The intention of this use case is to allow users to modify a product category
Actors	Administrator and Product Manager
Pre-condition	Actors request to edit product category and selects the product category that actor desires to modify
Main Flow	1. This use case begins when actor request to modify the product category 2. Actors select a product category (ref. Alternative Flow 1) 3. Actors modify the product category 4. System updates the product category information to the data file 5. Product category updated successfully.
Alternative Flow	1. The actor did not select product category and it will be requested to choose a product category

2.2.8 | Add Product Category

Name	Add Category
Description	The intention of this use case is to allow users to add a product category
Actors	Administrator and Product Manager
Pre-condition	Actors request to add product category
Main Flow	1. This use case begins when actor request to add a new product category. 2. Actor enters a new product category (ref. Alternative Flow 1) 3. Actor fills in the details of product category (ref. Alternative Flow 2) 4. System store all product category details to data file 5. New product category added
Alternative Flow	1. Product category Id that existed and will be requested to enter another unique product category id. 2. Product category information that is empty will be requested to fill in all inputs.

2.2.9 | Delete Product Category

Name	Delete Category
Description	The intention of this use case is to allow users to delete product category
Actors	Administrator and Product Manager
Pre-condition	Actors request to remove product category and selects a product category that is desired to be removed
Main Flow	1. This use case begins when actor request to remove a category 2. Actor must pick a category (ref. Alternative Flow) 3. System removes the category details from the data file 4. Product category successfully removed
Alternative Flow	1. Actor didn’t choose a category, system will request actor to choose a category

2.2.10 | Search Product Category

Name	Search Product Category
Description	The intention of this use case is to allow users to search for a product category
Actors	Administrator and Product Manager
Pre-condition	Actors request to manage product list and search for a product category
Main Flow	1. This use case begins when actor request to search for a product category 2. Actor enters the product category 3. System shows the product category matched with the input from the actor.
Alternative Flow	1. Actor didn’t enter any category to search, system will request actor to choose a category

2.2.11 | Manage Product Item

Name	Manage Product Item
Description	The intention of this use case is to allow users to manage product item such as Search, Delete, Modify and Add.
Actors	Administrator and Product Manager
Pre-condition	Actor request to manage product
Main Flow	1. This use case begins when actor request to manage product item 2. Actor request to Remove product - Proceed to “Remove Product” use case 3. Actor request to Modify product - Proceed to “Modify Product” use case 4. Actor request to Search product - Proceed to “Search Product” use case 5. Actor request to Add Product - Proceed to “Add Product” use case
Alternative Flow	-

2.2.12 | Search Product Item

Name	Search Item
Description	The intention of this use case is to allow users to search for a product item
Actors	Administrator and Product Manager
Pre-condition	Actors request to manage product list and search for a product
Main Flow	1. This use case begins when actor request to search for a product. 2. Actor enter the product name. 3. System show the product matched with the input of the actor.
Alternative Flow	-

2.2.13 | Delete Product Item

Name	Delete Item
Description	The intention of this use case is to allow users to delete a product item
Actors	Administrator and Product Manager
Pre-condition	Actors request to remove selected products
Main Flow	1. This use case begins when actor request to remove a product 2. Actors select a product item (ref. Alternative Flow) 3. System remove the product details from the data file 4. The product items are removed
Alternative Flow	1. The actor did not choose a product and will be requested to choose a product

2.2.14 | Modify Product Item

Name	Modify Item
Description	The intention of this use case is to allow users to modify a product item
Actors	Administrator and Product Manager
Pre-condition	Actors request to modify selected product item
Main Flow	1. This use case begins when actor request to remove product 2. Actor select a product (ref. Alternative Flow 1) 3. Actor modify the product information (ref. Alternative Flow 2) 4. System updates the product information to the data file 5. Product item updated successfully
Alternative Flow	1. The actor did not select a product item and system will request an input from the actor. 2. The product information is empty, and input will be requested by the system.

2.2.15 | Add Product Item

Name	Add item
Description	The intention of this use case is to allow users to add a new product
Actors	Administrator, Product Manager and Supplier
Pre-condition	Actor request to add new product.
Main Flow	1. This use case begins when the actor request to add a product 2. Actors enters a new product id (ref. Alternative Flow 1) 3. Actors fills in the product details (ref. Alternative Flow 2) 4. System stores all product details to the data file 5. New Product item is added
Alternative Flow	1. Product id will be scanned, and if exists, system will request user to input another product id. 2. The product information is empty, and input will be requested by the system.

2.2.16 | Manage System Info

Name	Manage System Info
Description	The intention of this use case is to allow users to manage system info
Actors	Administrator
Pre-condition	Actors request to manage system info
Main Flow	1. This use case begins when the actor request to manage system info
Alternative Flow	1. User request to “Export Report”

2.2.17 | Export System Info Report

Name	Export Report
Description	The intention of this use case is to allow users to export system info
Actors	Administrator
Pre-condition	Upon successful login to the system
Main Flow	1. This use case begins when actor request to export report 2. System will take all the data from data file and generate a PDF 3. Report generated

Alternative Flow	-

2.2.18 | Manage User

Name	Manage User
Description	The intention of this use case is to allow users to manager user information such as modify, search, add and set status
Actors	Administrator
Pre-condition	Upon successful login to the system
Main Flow	1. This use case begins when actor request to manage user. 2. Actor selects from product manager or supplier to edit 3. Actor requests to Modify a user - Perform to “Modify User” use case Actor request to Search for a user - Perform to “Search User” use case Actor request to Add a user - Perform to “Add User” user case
Alternative Flow	-

2.2.19 | Register User

Name	Register User
Description	The intention of this use case is to allow users to register user
Actors	Administrator
Pre-condition	Actor request to register new user
Main Flow	1. This use case begins when the actor request to add a user 2. Actors enters a new user id (ref. Alternative Flow 1) 3. Actors fills in the user details (ref. Alternative Flow 2) 4. System stores all user details to the data file 5. New User registered
Alternative Flow	1. User id will be scanned, and if exists, system will request user to input another user id. 2. The user information is empty, and input will be requested by the system.

2.2.20 | Set User Status

Name	Set Status
Description	The intention of this use case is to allow users to set user status
Actors	Administrator
Pre-condition	Upon successful login to the system
Main Flow	1. This use case begins when actor request to set user status 2. Actor selects users to set status to Active or Inactive
Alternative Flow	-

2.2.21 | Modify User

Name	Modify User
Description	The intention of this use case is to allow users to modify user details
Actors	Administrator
Pre-condition	Actors request to modify selected user details
Main Flow	1. This use case begins when actor request to modify details 2. Actor select a user (ref. Alternative Flow 1) 3. Actor modify the user information (ref. Alternative Flow 2) 4. System updates the user information to the data file 5. User Details updated successfully
Alternative Flow	1. The actor did not select a user details and system will request an input from the actor. 2. The user details are empty, and input will be requested by the system.

2.2.22 | Manage Product Catalogue

Name	Manage Product Catalogue
Description	The intention of this use case is to allow users to manage product catalogue such as modify, search, add and set status
Actors	Product Manager
Pre-condition	Actor request to manage product catalogue
Main Flow	1. This use case begins when actor request to manage product catalogue. 2. Actor selects from product manager or supplier to edit 3. Actor requests to Modify a user - Perform to "Modify User" use case 4. Actor request to Search for a user - Perform to "Search User" use case 5. Actor request to Add a user - Perform to "Add User" user case
Alternative Flow	-

2.2.23 | Delete Product Catalogue

Name	Delete Catalogue
Description	The intention of this use case is to allow users to delete a product catalogue
Actors	Product Manager
Pre-condition	Actors request to remove selected product catalogue
Main Flow	1. This use case begins when actor request to remove a product catalogue 2. Actors select a product catalogue (ref. Alternative Flow) 3. System remove the product catalogue from the data file 6. The product items are removed
Alternative Flow	The actor did not choose a product catalogue and will be requested to choose a product

<u>2.2.24 | Add Product Catalogue</u>

Name	Add Catalogue
Description	The intention of this use case is to allow users to add a new product catalogue
Actors	Product Manager
Pre-condition	Actor request to add new product catalogue.
Main Flow	1. This use case begins when the actor request to add a product catalogue 2. Actors enters a new product catalogue id (ref. Alternative Flow 1) 3. Actors fills in the product catalogue details (ref. Alternative Flow 2) 4. System stores all product catalogue details to the data file 5. New Product catalogue item is added
Alternative Flow	1. Product catalogue id will be scanned, and if exists, system will request user to input another product id. 2. The product catalogue information is empty, and input will be requested by the system.

<u>2.2.25 | Modify Product Catalogue</u>

Name	Modify Catalogue
Description	The intention of this use case is to allow users to modify a product catalogue
Actors	Product Manager
Pre-condition	Actors request to modify selected product catalogue
Main Flow	1. This use case begins when actor request to remove product catalogue 2. Actor select a product catalogue (ref. Alternative Flow 1) 3. Actor modify the product catalogue information (ref. Alternative Flow 2) 4. System updates the product catalogue information to the data file 5. Product catalogue updated successfully
Alternative Flow	1. The actor did not select a product catalogue and system will request an input from the actor.

	2. The product catalogue information is empty, and input will be requested by the system.

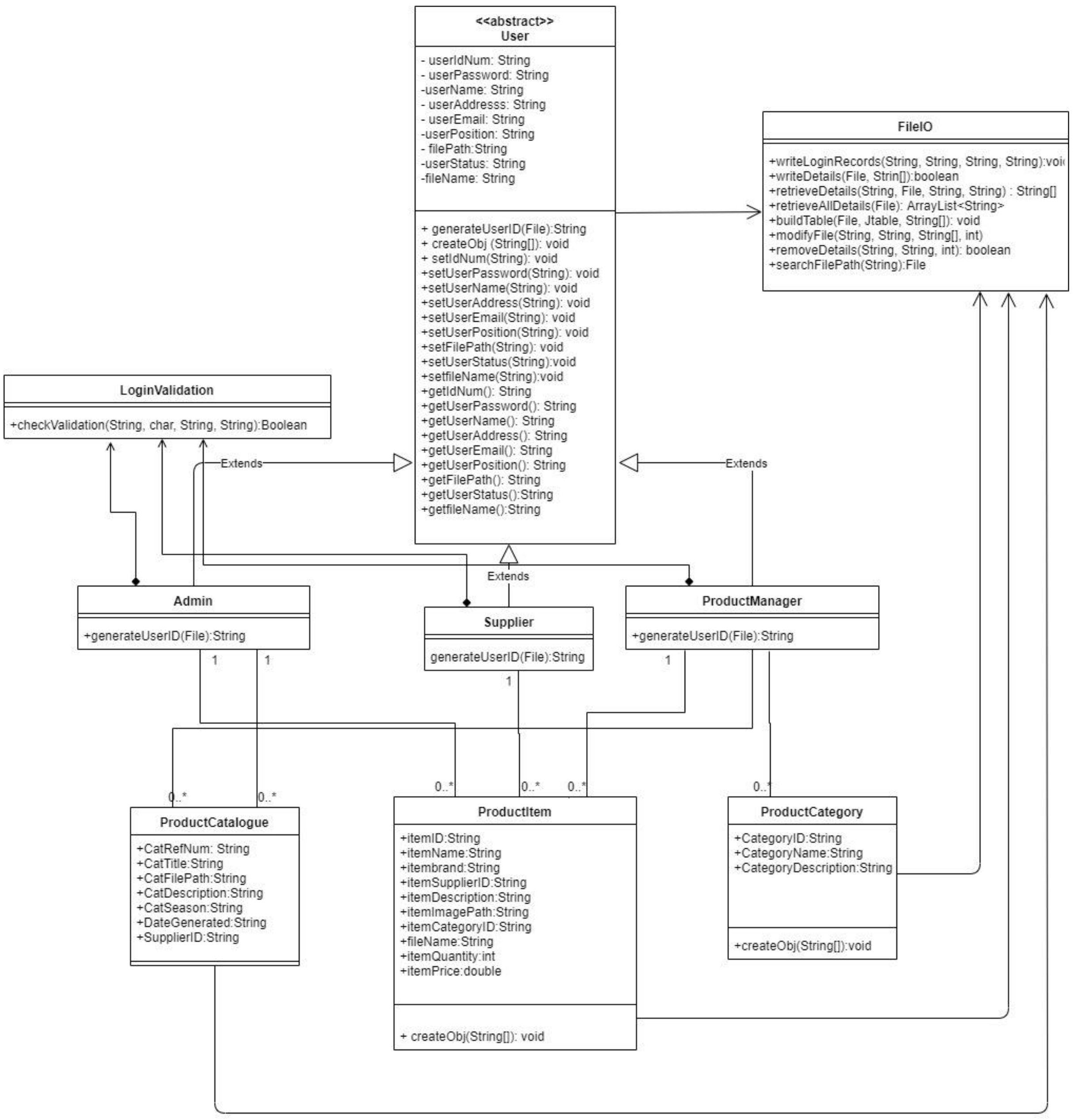
2.2.26 | Search Product Catalogue

Name	Search Catalogue
Description	The intention of this use case is to allow users to search for a product catalogue
Actors	Product Manager
Pre-condition	Actors request to manage product list and search for a product catalogue
Main Flow	1. This use case begins when actor request to search for a product catalogue. 2. Actor enter the product catalogue name. 3. System show the product catalogue matched with the input of the actor.
Alternative Flow	-

2.2.27 | Export Product Catalogue

Name	Export Catalogue
Description	The intention of this use case is to allow users to export product catalogue
Actors	Product Manager
Pre-condition	Upon successful login to the system
Main Flow	1. This use case begins when actor request to export product catalogue 2. System will take all the data from data file and generate a PDF 3. Product catalogue PDF generated
Alternative Flow	-

2.3 | Class Diagram


<<abstract>>
User
- userIdNum: String
- userPassword: String
-userName: String
- userAddresss: String
- userEmail: String
-userPosition: String
- filePath:String
-userStatus: String
-fileName: String
+ generateUserID(File):String
+ createObj (String[]): void
+ setIdNum(String): void
+setUserPassword(String): void
+setUserName(String): void
+setUserAddress(String): void
+setUserEmail(String): void
+setUserPosition(String): void
+setFilePath(String): void
+setUserStatus(String):void
+setfileName(String):void
+getIdNum(): String
+getUserPassword(): String
+getUserName(): String
+getUserAddress(): String
+getUserEmail(): String
+getUserPosition(): String
+getFilePath(): String
+getUserStatus():String
+getfileName():String
FileIO
+writeLoginRecords(String, String, String, String):voi
+writeDetails(File, Strin[]):boolean
+retrieveDetails(String, File, String, String) : String[]
+retrieveAllDetails(File): ArrayList<String>
+buildTable(File, Jtable, String[]): void
+modifyFile(String, String, String[], int)
+removeDetails(String, String, int): boolean
+searchFilePath(String):File
LoginValidation
+checkValidation(String, char, String, String):Boolean
Extends
Extends
Extends
Admin
+generateUserID(File):String
Supplier
generateUserID(File):String
ProductManager
+generateUserID(File):String
1
1
1
1
0..*
0..*
0..*
0..*
0..*
0..*
ProductCatalogue
+CatRefNum: String
+CatTitle:String
+CatFilePath:String
+CatDescription:String
+CatSeason:String
+DateGenerated:String
+SupplierID:String
ProductItem
+itemID:String
+itemName:String
+itembrand:String
+itemSupplierID:String
+itemDescription:String
+itemImagePath:String
+itemCategoryID:String
+fileName:String
+itemQuantity:int
+itemPrice:double
+ createObj(String[]): void
ProductCategory
+CategoryID:String
+CategoryName:String
+CategoryDescription:String
+createObj(String[]):void

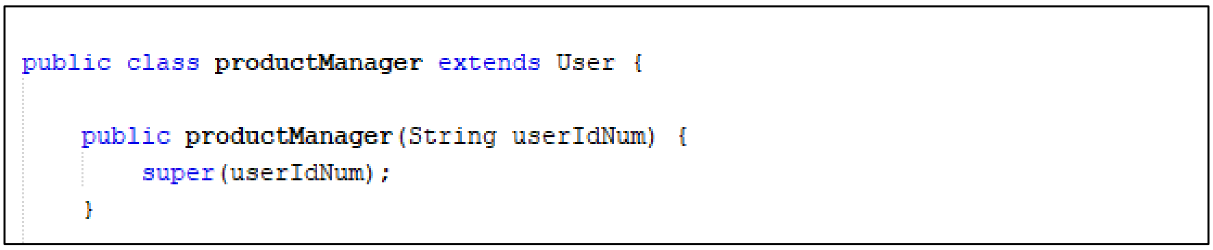
3.0 | Object-oriented programming concept

Object-oriented programming refers to languages that uses object in programming. It is aimed to mimic a real-word entity with the use of the object-oriented principles. Objects can be defined as data filed that has unique attributes and behaviours (Rouse, 2019). For instance, a dog (object) has names, breeds and gender (attribute), thus it walks, runs and eats (methods). This concept focuses more on objects and the relationship between them. An object is an instance of a class, a class is a blueprint or template of constructing an object. In addition to that, developers can develop based on objects, so it is very scalable, reusable hence the reason why it is very suitable for large, complex and actively updated or maintained project. (GeeksforGeeks, 2019)

There are four main principles that surrounds object-oriented programming, out of which is inheritance, polymorphism, abstraction and encapsulation (Petkov, 2018). These principles allow for better reusability without compromising security (Stackify, 2017).

3.1 | Inheritance

Inheritance is a special feature in object-oriented programming where it can derive a class from another class for a hierarchy class to share the same set of attributes and methods. For instance, in this project, the administrator, project manager and supplier are derived from the user class, meaning each of them share the same attribute as the user class. The derived class will be called as a subclass or a child class, whereas the other will be named as the superclass or parent class (Janssen, 2017). In Java, in order to inherit from a class, the keyword "extend" is used. The reason behind using inheritance is to provide the reusability of code so that a class just has to write unique feature, whereas the common ones are can be taken by the parent class. It allows for a lesser amount of duplicated code, as the common attribute and method doesn't require additional code which in turn had reduced the redundancy of the code. As per mentioned, this can be a major benefit for large projects as it could reduce a lot of redundant code making it easier to understand and at the same time make the code more flexible and interchangeable (Scientech Easy, 2019).



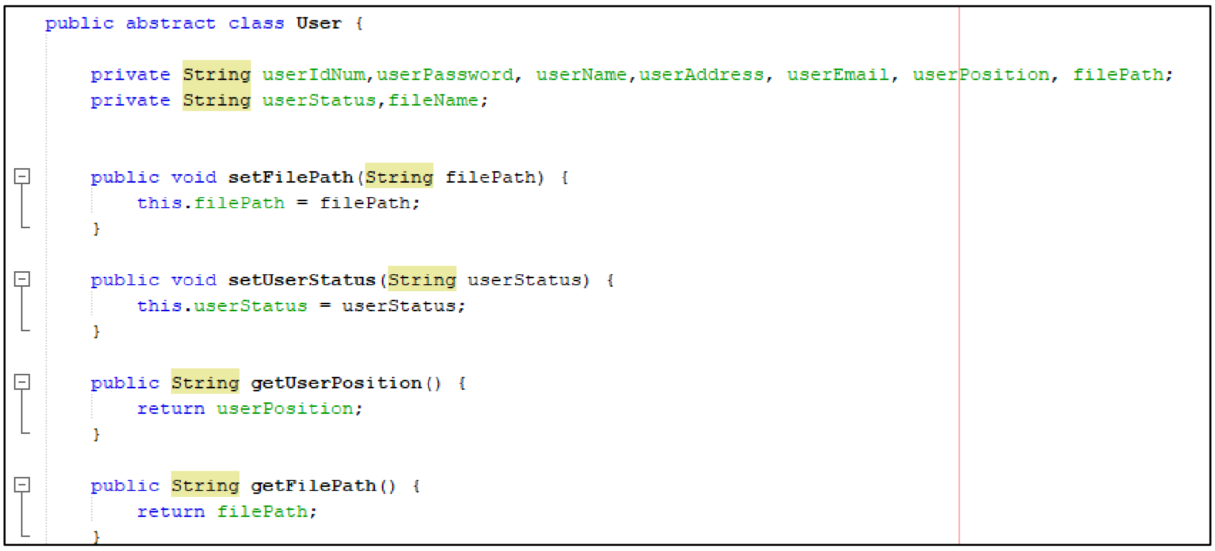
```
public class productManager extends User {

    public productManager(String userIdNum) {
        super(userIdNum);
    }
```

Figure 3.1.1 Inheritance in Product Catalogue Management System

Figure 3.1.1 shows the inheritance sample that is used in the Product Catalogue Management System. The product manager class is a derived child class which extends to the User class. In addition to that, there is a “super” keyword that refers to the constructor of the super or parent class.

3.2 | Encapsulation

Encapsulation is one the four main pillars of object-oriented programming. It refers to wrapping up of data under a single unit. It is a mechanism that binds code and the data it manipulates, it also known as data hiding (Vaidya, 2019). During encapsulation, the variables inside a class are hidden from the other classes, and only can be accessed by going through the methods of their current class. The purpose of encapsulation is to provide a layer of protection for sensitive data, making the data secured. Encapsulation is achieved by setting all the variables in a class as private and provide a getter and setter methods to modify and view the variable values. Programmers can define whether the data is only readable or writable or both. By implementing encapsulation, if for example the implementation only allows for read only, the user data will be protected from unwanted modification or corruption, making data safer for users and providing an extra layer of security.



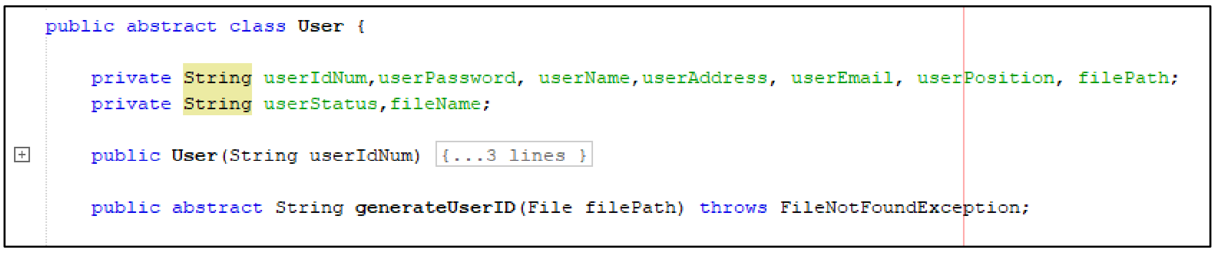
```java
public abstract class User {

    private String userIdNum,userPassword, userName,userAddress, userEmail, userPosition, filePath;
    private String userStatus,fileName;


    public void setFilePath(String filePath) {
        this.filePath = filePath;
    }

    public void setUserStatus(String userStatus) {
        this.userStatus = userStatus;
    }

    public String getUserPosition() {
        return userPosition;
    }

    public String getFilePath() {
        return filePath;
    }
```

Figure 3.2.1 Encapsulation in Product Catalogue Management System

Figure 3.2.1 shows the encapsulation principle applied inside the product catalogue management system. All variables are set to private, and setter and getter function are implemented in order to modify and retrieve the data, which means only the specific class can be able to access the data through the getter and setter methods.

3.3 | Abstraction

Abstraction is the other principle or pillar of object-oriented programming. Abstraction is the process of hiding the implementation details from the user. Only the functionality will be provided to the user (Filip, 2017). For example, a smart phone is viewed as a phone instead of viewing it as an individual component. It is an act of ignoring the irrelevant details. The main goal of implementing abstraction is to hide the complexity of code or data, as the project scales higher and higher, the complexity and number of codes rises as well, it will cause the code to be difficult to understand or solve. Hence, implementing abstraction could help ease the issue as each object only exposes to a high-level mechanism for using it. This allows developers implement more complex logic on top of abstraction without fully understanding about the hidden complexity. To simplify, one class doesn't need to understand the inner attributes in order to use it. Implementing abstraction is crucial as it avoids repeating the same work or code multiple times (Janssen, 2017).



```
public abstract class User {

    private String userIdNum,userPassword, userName,userAddress, userEmail, userPosition, filePath;
    private String userStatus,fileName;

    public User(String userIdNum) {...3 lines }

    public abstract String generateUserID(File filePath) throws FileNotFoundException;
```

Figure 3.3.1 Abstraction in Product Catalogue Management System

Figure 3.3.1 exhibits an abstract class called User. All the classes that inherit User class will be needed to implement all the abstract method within this User class. The abstract method that is in user class has no body, as it would be further defined by the child class.

3.4 | Polymorphism

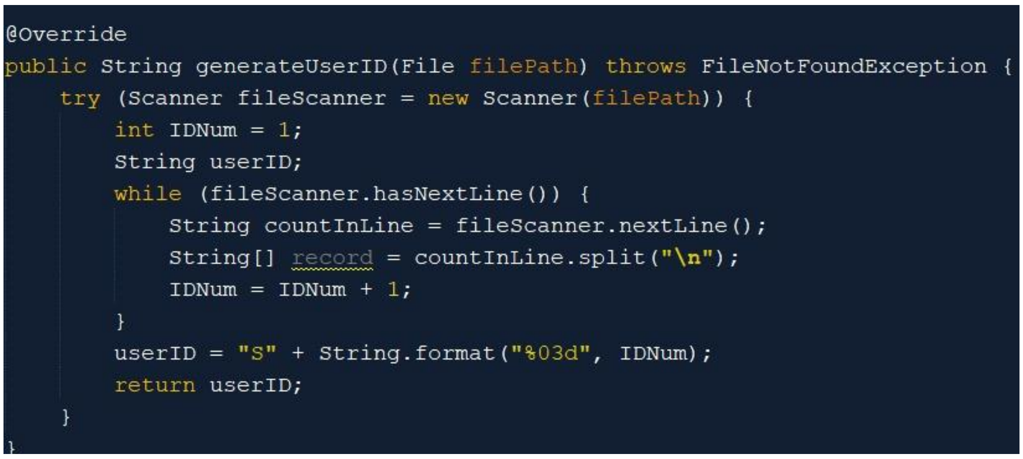
Polymorphism in java is an object-oriented programming concept by which a single action can be performed in different ways. (javaTpoint, 2019) A single method for example, can have different parameters or different return values but the actions are kept the same.

There are 2 types of polymorphism in Java which can be implemented in Java programs for a more object-oriented experience. The built-in methods of Java, which are overriding and overloading, are prominent for object-oriented coding and ease the use of polymorphism in Java programs. Runtime polymorphism, also known as Dynamix Method Dispatch is a process where overridden method is resolved at runtime rather than compile-time. (javaTpoint, 2019)Runtime is the instance where the executables or the methods are being executed and running. Compile time however is the instance where the code entered is converted to executable. These terms are also commonly used to define and detect the possible errors occurring in the program. (net_informations, 2019)

<u>Override</u>

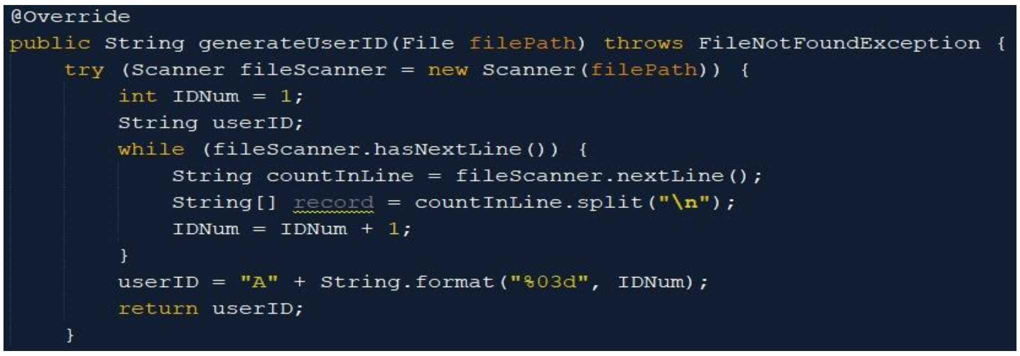
With that being said, compile time polymorphism occurs during the instant the java code entered is turned into machine language and runtime polymorphism occurs while the java program is already turned into machine language and being under execution. Object-oriented programming encourages a clear modular structure for programs and wishes for programs to act more human-like. Polymorphism can be known in real life as a person who takes up many roles at different times producing different results or requiring different inputs. A person can be a mother towards their child, producing love and care for them but can also be a wife who is doted by their husband. These real-life scenarios are meant to be met in programming through object-oriented programming.

The methods used in the product management catalogue system to implement polymorphism can be found in the classes of product Manager, Admin and Supplier. These classes are the subclasses of the User class and there happen to be a public abstract class known as generateUserID() which is used by all.



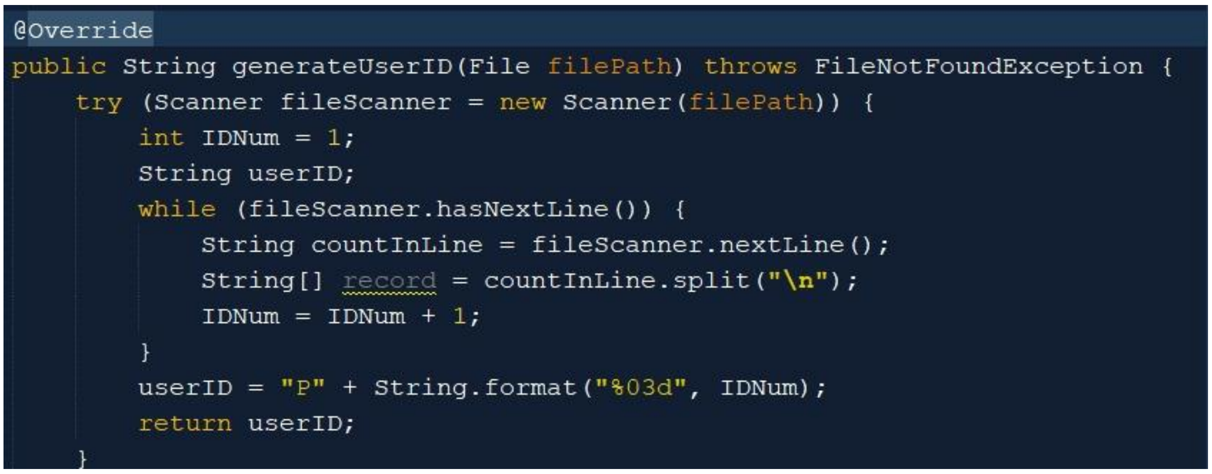
```java
@Override
public String generateUserID(File filePath) throws FileNotFoundException {
    try (Scanner fileScanner = new Scanner(filePath)) {
        int IDNum = 1;
        String userID;
        while (fileScanner.hasNextLine()) {
            String countInLine = fileScanner.nextLine();
            String[] record = countInLine.split("\n");
            IDNum = IDNum + 1;
        }
        userID = "S" + String.format("%03d", IDNum);
        return userID;
    }
}
```

Figure 3.4.1: Polymorphism in Supplier class



```java
@Override
public String generateUserID(File filePath) throws FileNotFoundException {
    try (Scanner fileScanner = new Scanner(filePath)) {
        int IDNum = 1;
        String userID;
        while (fileScanner.hasNextLine()) {
            String countInLine = fileScanner.nextLine();
            String[] record = countInLine.split("\n");
            IDNum = IDNum + 1;
        }
        userID = "A" + String.format("%03d", IDNum);
        return userID;
    }
```

Figure 3.4.2: Polymorphism in Admin class

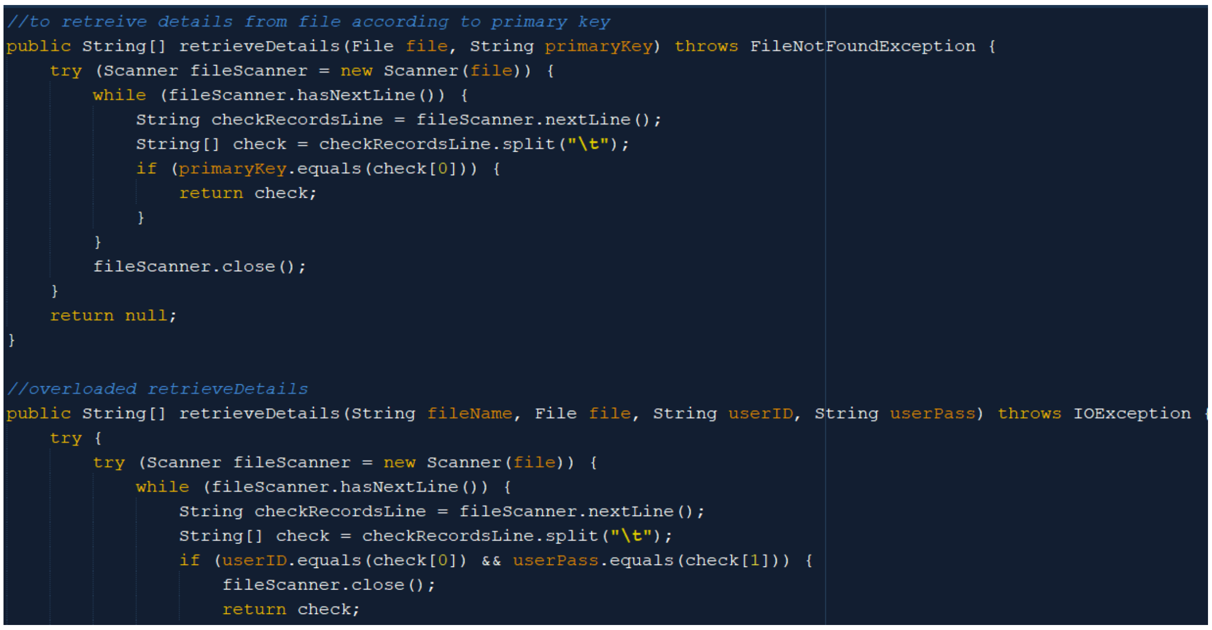


```java
@Override
public String generateUserID(File filePath) throws FileNotFoundException {
    try (Scanner fileScanner = new Scanner(filePath)) {
        int IDNum = 1;
        String userID;
        while (fileScanner.hasNextLine()) {
            String countInLine = fileScanner.nextLine();
            String[] record = countInLine.split("\n");
            IDNum = IDNum + 1;
        }
        userID = "P" + String.format("%03d", IDNum);
        return userID;
    }
```

Figure 3.4.3: Polymorphism in Product Manager class

Figure 3.4.1 shows polymorphism in Supplier class of the Product Catalogue Management System. Figure 3.4.2 shows polymorphism in Admin class of the Product Catalogue Management System. Figure 3.4.3 shows polymorphism in Product Manager class of the Product Catalogue Management System. The @Override annotation can be seen clearly above the classes which shows that even Netbeans has approved of the systems overriding. If noticed clearly, at the 11th line of code, the character after the equal sign is different for all three classes with it being A for admin, P for Product Manager and S for Supplier. These small changes are sufficient to show polymorphism in a system and are useful with it being able to be called from anywhere where product manager, supplier and admin classes are present.

<u>Overload</u>



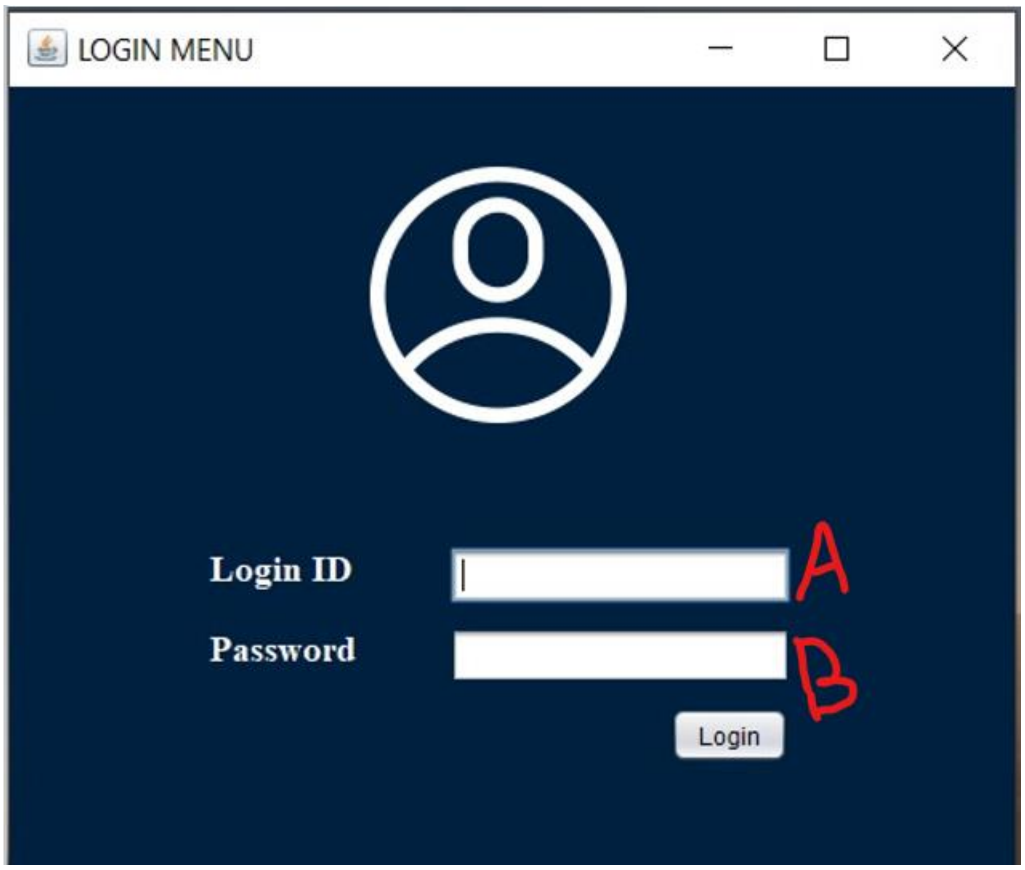
```java
//to retreive details from file according to primary key
public String[] retrieveDetails(File file, String primaryKey) throws FileNotFoundException {
    try (Scanner fileScanner = new Scanner(file)) {
        while (fileScanner.hasNextLine()) {
            String checkRecordsLine = fileScanner.nextLine();
            String[] check = checkRecordsLine.split("\t");
            if (primaryKey.equals(check[0])) {
                return check;
            }
        }
        fileScanner.close();
    }
    return null;
}

//overloaded retrieveDetails
public String[] retrieveDetails(String fileName, File file, String userID, String userPass) throws IOException
    try {
        try (Scanner fileScanner = new Scanner(file)) {
            while (fileScanner.hasNextLine()) {
                String checkRecordsLine = fileScanner.nextLine();
                String[] check = checkRecordsLine.split("\t");
                if (userID.equals(check[0]) && userPass.equals(check[1])) {
                    fileScanner.close();
                    return check;
```

Figure 3.4.4: Code snippet containing overloading methods

Figure 3.4.4 shows the code snippet containing overloading methods. It is seen clearly in the first method of retrieveDetails that there are only 2 parameters passed into the method to be processed. The second method of retrieveDetails has however, 4 parameters passed into the method for further processing. Both methods have the same name, but different amount of parameters passed inside. It is concluded that the method with 4 parameters passed in is overloaded.

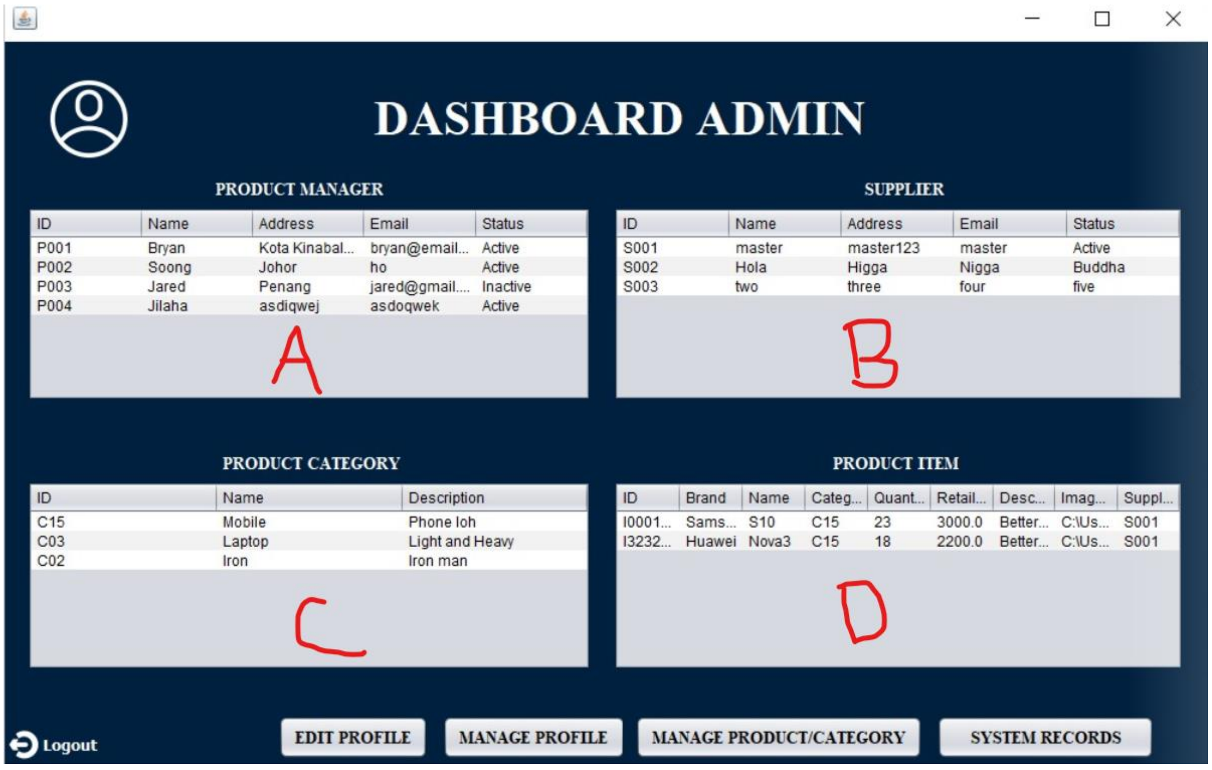
4.0 | User Manual

4.1 Login Menu



Label	Functions	Conditions
A	Text Field for users to fill their login ID	-
B	Password Field for users to fill their passwords	-
Login Button	Button to proceed to dashboard summary after filling A and B	User ID and user password exits in the text file and are identical

4.2 | Dashboard Summary



PRODUCT MANAGER

ID	Name	Address	Email	Status
P001	Bryan	Kota Kinabal...	bryan@email...	Active
P002	Soong	Johor	ho	Active
P003	Jared	Penang	jared@gmail....	Inactive
P004	Jilaha	asdiqwej	asdoqwek	Active

SUPPLIER

ID	Name	Address	Email	Status
S001	master	master123	master	Active
S002	Hola	Higga	Nigga	Buddha
S003	two	three	four	five

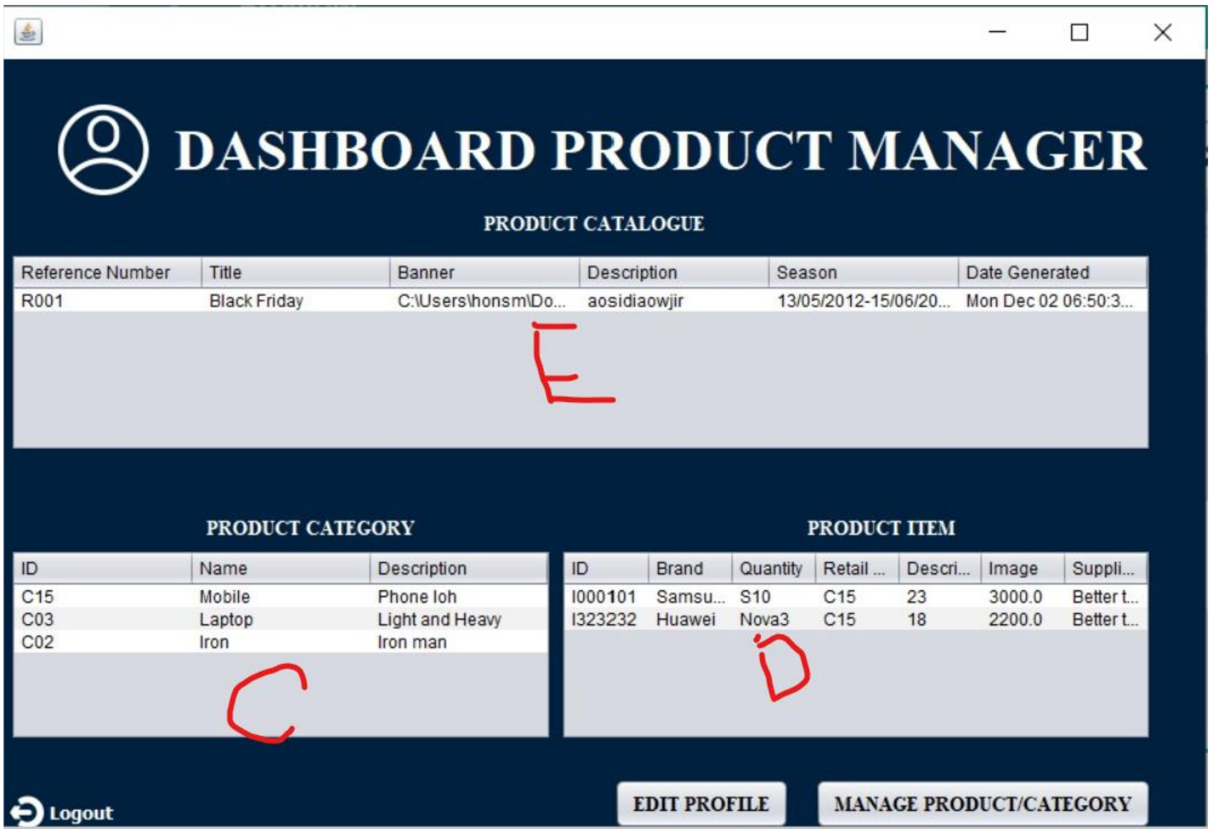
PRODUCT CATEGORY

ID	Name	Description
C15	Mobile	Phone loh
C03	Laptop	Light and Heavy
C02	Iron	Iron man

PRODUCT ITEM

ID	Brand	Name	Categ...	Quant...	Retail...	Desc...	Imag...	Suppl...
I0001...	Sams...	S10	C15	23	3000.0	Better...	C:\Us...	S001
I3232...	Huawei	Nova3	C15	18	2200.0	Better...	C:\Us...	S001

Label	Functions
A	Table showing summary of Product Manager table
B	Table showing summary of Supplier Table
C	Table showing summary of Product Category Table
D	Table showing summary of Product Item Table

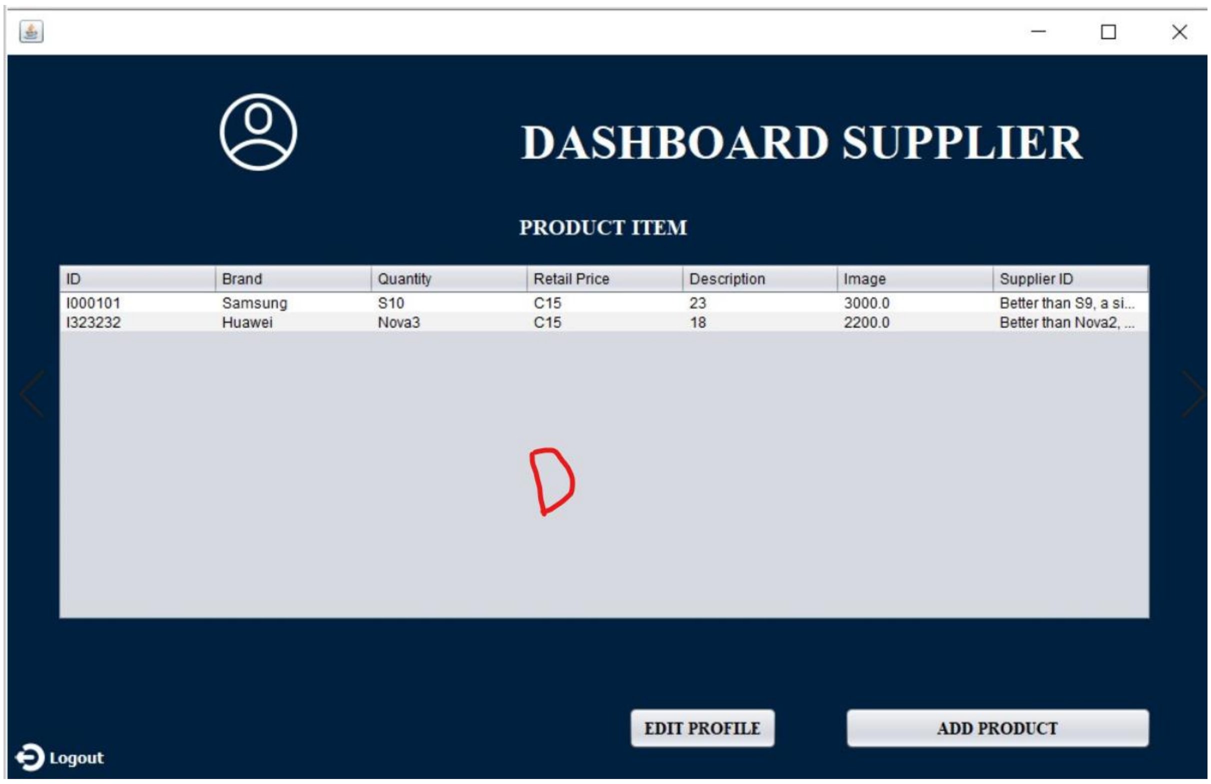


Reference Number	Title	Banner	Description	Season	Date Generated
R001	Black Friday	C:\Users\honsm\Do...	aosidiaowjir	13/05/2012-15/06/20...	Mon Dec 02 06:50:3...

ID	Name	Description
C15	Mobile	Phone loh
C03	Laptop	Light and Heavy
C02	Iron	Iron man

ID	Brand	Quantity	Retail ...	Descri...	Image	Suppli...
I000101	Samsu...	S10	C15	23	3000.0	Better t...
I323232	Huawei	Nova3	C15	18	2200.0	Better t...

Label	Function
C	Table showing Product Category Table
D	Table showing Product Item Table
E	Table showing Product Catalogue Table

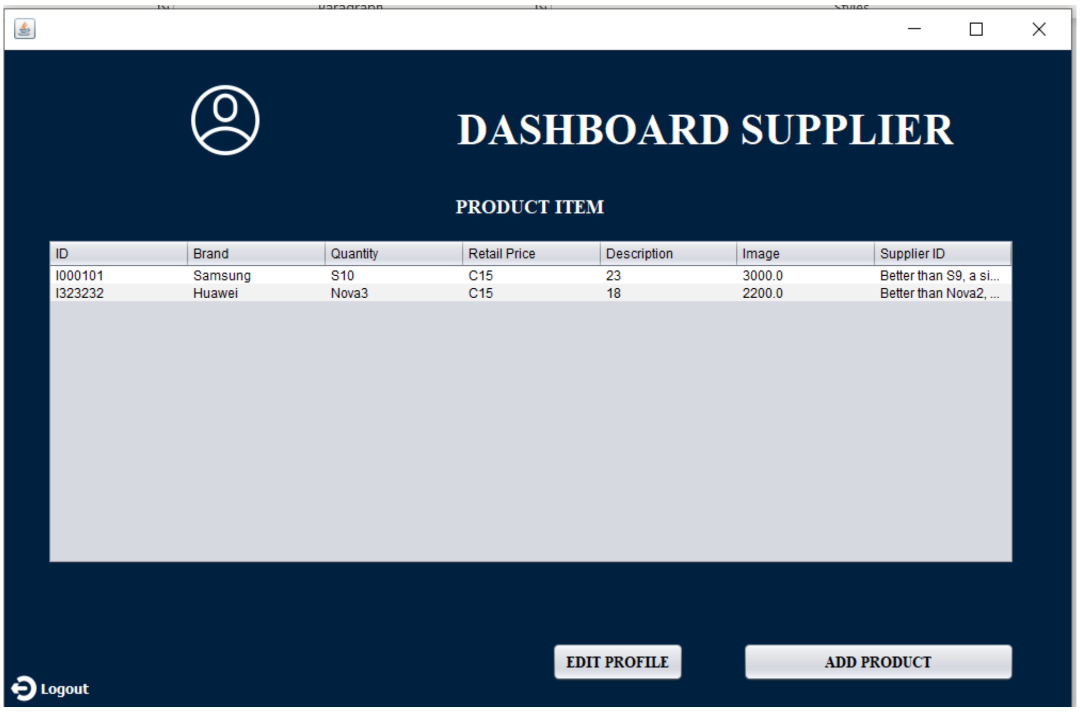


ID	Brand	Quantity	Retail Price	Description	Image	Supplier ID
I000101	Samsung	S10	C15	23	3000.0	Better than S9, a si...
I323232	Huawei	Nova3	C15	18	2200.0	Better than Nova2, ...

Label	Function
D	Table showing the summary of Product Item Table

5.0 | Extra Features

5.1 | Supplier Account



DASHBOARD SUPPLIER

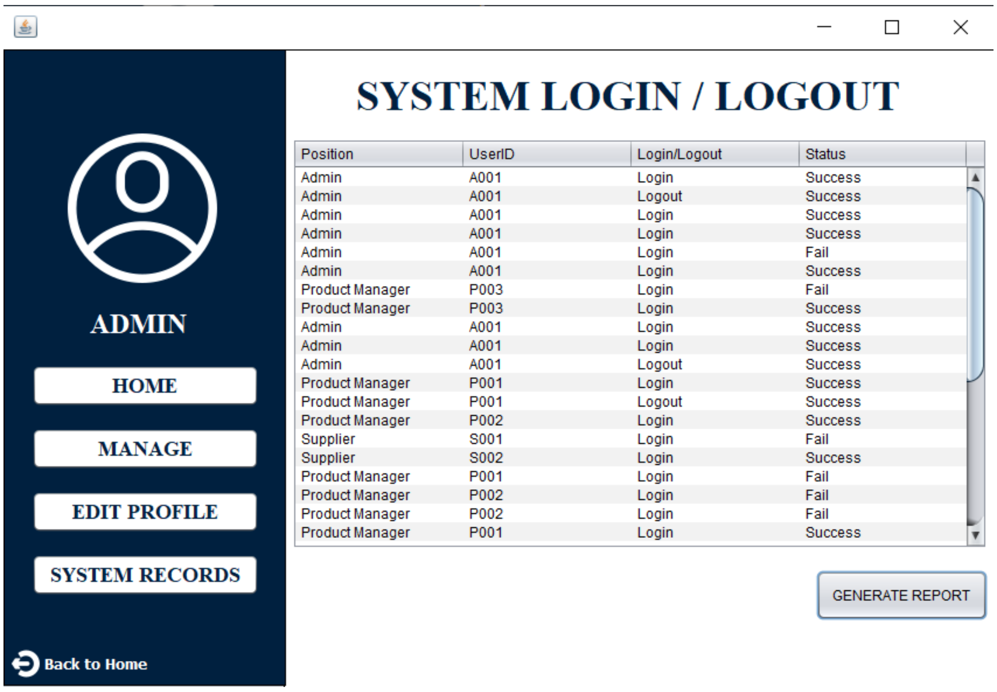
PRODUCT ITEM

ID	Brand	Quantity	Retail Price	Description	Image	Supplier ID
I000101	Samsung	S10	C15	23	3000.0	Better than S9, a si...
I323232	Huawei	Nova3	C15	18	2200.0	Better than Nova2, ...

Figure 5.1.1: Dashboard for Supplier

Figure 5.1.1 shows the dashboard for supplier. Supplier are managed as users of the system and are provided with unique userIDs and user-defined passwords. Suppliers can login to the system, edit their personal profile and add product items. However, if their status is inactive, they will not be able to add product items. Suppliers cannot change their own status and only admins are able to change their status.

5.2 | Export System Login/Logout Record



Position	UserID	Login/Logout	Status
Admin	A001	Login	Success
Admin	A001	Logout	Success
Admin	A001	Login	Success
Admin	A001	Login	Success
Admin	A001	Login	Fail
Admin	A001	Login	Success
Product Manager	P003	Login	Fail
Product Manager	P003	Login	Success
Admin	A001	Login	Success
Admin	A001	Login	Success
Admin	A001	Logout	Success
Product Manager	P001	Login	Success
Product Manager	P001	Logout	Success
Product Manager	P002	Login	Success
Supplier	S001	Login	Fail
Supplier	S002	Login	Success
Product Manager	P001	Login	Fail
Product Manager	P002	Login	Fail
Product Manager	P002	Login	Fail
Product Manager	P001	Login	Success

Figure 5.2.1: System records for login and logout

Figure 5.2.1 shows the systems records for login and logout. It is only available for admin users. Once clicked, the Generate Report button will instruct the system to create a pdf file containing the system records. Admin Users can choose the path and the name of the pdf file generated.

Figure 5.2.2: Generated Report

Figure 5.2.2 shows the example of a generated report. The report is generated after the admin user has chosen the path to generate the report and the name of the report.

6.0 | Conclusion

All in all, object-oriented programming concepts are crucial in a more complex and larger project as it could reduce code redundancy, have better code reusability and flexibility, effective problem solving and so much more. Upon the completion of this project, the team had learned the four principles of object-oriented programming concept. Going through many trial and error, the team had acknowledged and understood the benefits behind this concept. In addition, the team had implemented all the principles learn which includes inheritance, polymorphism, abstraction and encapsulation. Hence, the team had also acquired the skills to construct a use case and class diagram. To conclude, it has been a great eye opener and opportunity to learn this concept, as it will be a crucial skill to have in the future of the team.

7.0 Workload Matrix

	Soong Gim Hoy (TP053242)	**Chan Jia Le (TP045592)**
1.0 Introduction	50%	50%
1.1 Assumptions	50%	50%
2.1 Use Case Diagram	50%	50%
2.2 Class Diagram	50%	50%
3.0 Object Oriented Programming Principles	50%	50%
6.0 Conclusion	50%	50%
Documentation Report	50%	50%
Implementation	50%	50%
Signature		

References

Filip, 2017. *Java Abstraction Example.* [Online]
Available at: https://javatutorial.net/java-abstraction-example
[Accessed 1 December 2019].

GeeksforGeeks, 2019. *Object Oriented Programming (OOPs) Concept in Java.* [Online]
Available at: https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/
[Accessed 30 November 2019].

Janssen, T., 2017. *OOP Concept for Beginners: What is Abstraction?.* [Online]
Available at: https://stackify.com/oop-concept-abstraction/
[Accessed 25 November 2019].

Janssen, T., 2017. *OOP Concept for Beginners: What is Inheritance?.* [Online]
Available at: https://stackify.com/oop-concept-inheritance/
[Accessed 30 November 2019].

javaTpoint, 2019. *javaTpoint.* [Online]
Available at: https://www.javatpoint.com/runtime-polymorphism-in-java

net_informations, 2019. *net-informations.* [Online]
Available at: http://net-informations.com/python/iq/checking.htm

Petkov, A., 2018. *How to explain object-oriented programming concepts to a 6-year-old.* [Online]
Available at: https://www.freecodecamp.org/news/object-oriented-programming-concepts-21bb035f7260/
[Accessed 25 November 2019].

Rouse, M., 2019. *object-oriented programming (OOP).* [Online]
Available at: https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP
[Accessed 30 November 2019].

Scientech Easy, 2019. *Inheritance in Java | Realtime Example, Use, Advantages.* [Online]
Available at: https://www.scientecheasy.com/2019/01/inheritance-in-java.html
[Accessed 30 November 2019].

Stackify, 2017. *What Are OOP Concepts in Java? The Four Main OOP Concepts in Java, How They Work, Examples, and More.* [Online]
Available at: https://stackify.com/oops-concepts-in-java/
[Accessed 25 November 2019].

Vaidya, N., 2019. *Encapsulation in Java – How to master OOPs with Encapsulation?.* [Online]
Available at: https://www.edureka.co/blog/encapsulation-in-java/
[Accessed 1 December 2019].

Student Name: **Soong Gim Hoy** ID: **TP053242**

TOTAL MARKS

Learning Outcomes	Questions
1. Explain the techniques of Object Oriented Design(C2,PLO1)	
2. Implement a software application that exploits the strength of object-oriented paradigm(C6,PL02)	LO covered in the assignment
3. Demonstrate the use of object oriented concepts and their functionalities in the existing system(A3,PL04)	LO covered in the assignment

REQUIREMENT ANALYSIS [CLO2-PLO2]	**Fail**	**Marginal Fail**	**Pass**	**Credit**	**Distinction**
	0-3	**4**	**5-6**	**7**	**8-10**
USE CASE DIAGRAM WITH DESCRIPTION (10%) [CLO2]	Incorrect overview use case, detail use cases with scenarios (where applicable) and use case descriptions OR incorrect user case notations OR illogical functional design.	Inappropriate overview use case, detail use cases with scenarios (where applicable) and use case descriptions.	The use cases are well presented (generalization) with no major mistake in logic and notation, and described all flows in use case descriptions.	Appropriate labelling and no mistake in logic and notation and clear description for normal flow, subflow and alternative flow in use case descriptions.	Comprehensive provision of the required - overview use case; - detail use case diagrams with scenarios (where applicable: generalization, extends, includes); - use case descriptions for each use case / scenario and no mistake in logic and notation and appropriateness.
CLASS DIAGRAM (10%) [CLO2 and CLO3]	No attributes and no associations are included.	Class diagram with attributes and associations. Both are incomplete and illogical.	Class diagram with attributes and associations. Both are complete with no major errors.	Class diagram with appropriate attributes and associations. With labelling and no mistakes in logic and notation.	Class diagram with appropriate, relevant attributes and associations. With very good and meaningful labelling according to guidelines.
				SUB TOTAL:	

35

IMPLEMENTATION [CLO2-PLO2]	Fail	Marginal Fail	Pass	Credit	Distinction
	0-3	4	5-6	7	8-10
USER LEVEL ACCESS (10%)	No program or work not done. Program incomplete with obvious errors. Not able to compile and run the program.	Not able to compile or run but evidence of the coding is available. Able to compile but not able to run the program. Able to compile and run the program but only able to add less than 50% of the details of access rights module listed in the assignment. Not able to demonstrate the use of object-oriented concepts. Data not stored in file.	Able to compile and run the program. Able to add at least 50% of the details of user access module listed in the assignment. Able to demonstrate the use of at least three of the object-oriented concepts – such as creation of classes, objects and methods. Attempted to write to file but with some errors.	Able to compile and run the program. Able to add at least 80% of the details of access rights module listed in the assignment. Able to demonstrate most of the object-oriented concepts. Able to write to file and read from file with some errors.	Able to compile and run the program. Able to add 90-100% of the details of access rights module listed in the assignment. Able to demonstrate all the object-oriented concepts along with additional features. Able to write to file and read from file with no errors.
PRODUCT AND SUPPLIER MANAGEMENT (30%) **(OR)** **PRODUCT CATALOGUE FOR DIFFERENT FESTIVALS (30%)**	**Fail**	**Marginal Fail**	**Pass**	**Credit**	**Distinction**
	0-11	12-14	15-19	20-22	23-30
	No program or work not done. Program incomplete with obvious errors. Not able to compile and run the program.	Not able to compile or run but evidence of the coding is available. Able to compile but not able to run the program. Able to compile and run the program but only able to add less than 50% of the details of module listed in the assignment. Not able to demonstrate the use of object-oriented concepts. Data not stored in file.	Able to compile and run the program. Able to add at least 50% of the details of module listed in the assignment. Able to demonstrate the use of at least three of the object-oriented concepts – such as creation of classes, objects and methods. Attempted to write to file but with some errors.	Able to compile and run the program. Able to add at least 80% of the details of the module listed in the assignment. Able to demonstrate most of the object-oriented concepts. Able to write to file and read from file with some errors.	Able to compile and run the program. Able to add 90-100% of the details of the module listed in the assignment. Able to demonstrate all the object-oriented concepts along with additional features. Able to write to file and read from file with no errors.
				SUB TOTAL:	

36

REPORT [CLO3-PLO4]	**Fail 0-3**	**Marginal Fail 4**	**Pass 5-6**	**Credit 7**	**Distinction 8-10**
REPORT FORMAT AND REFERENCES (10%)	The reference list is *all inapplicable* OR *irrelevant*. The format is NOT comply proper referencing system. The citation is NOT included at all. The simple document without formatting, header and footer, page number, etc.	The reference list is identified mostly inapplicable OR irrelevant, The format is partially comply proper referencing system. The citation is partially included. The simple document with little formatting.	The reference list is complete but sort of complete citation. The document is complete.	The above +. The citation is clearly specified. The above + with all the formatting criteria.	The above + with rich source of explorations to form a complete reference. The above + precise, clear, complete, all the diagram, chart, picture, pie, symbol, glossary are completely organized.
PROGRAM DOCUMENTATION (20%)	**Fail**	**Marginal Fail**	**Pass**	**Credit**	**Distinction**
	0-7	**8-9**	**10-12**	**13-14**	**15-20**
	Documentation not done. Content of documentation does not adhere to any of the requirements stipulated in the assignment requirements.	At least 1 object-oriented programming concept applied in the solution and briefly described. No implementation code for the object-oriented programming concepts identified. Screen capture of the output of the program does not have any explanation to describe the program.	Description of at least 2 object-oriented programming concepts that are applied in the solution with some evidence of the implementation code is included. Screen capture of the output of the program with minimal explanation to describe the program.	Description of at least 3 object-oriented programming concepts that are applied in the solution are described and evidence of the implementation code being documented. Screen capture of the output of the program with some explanation to describe the program. Description and evidence of at least 1 additional feature which has been incorporated in the solution.	Description of at least 4 object-oriented programming concepts that are applied in the solution and evidence of the implementation code being documented. Screen capture of the output of the program with appropriate explanation to sufficiently describe the program. Description and evidence of at least 2 additional features which have been incorporated in the solution.
INDIVIDUAL PRESENTATION (10%)	**0-3**	**4**	**5-6**	**7**	**8-10**
	Absent OR late OR not prepared for presentation session without valid reasons. Handled questions grossly bad and unable to demonstrate understanding of OO concept.	Reading from presentation material. Presentation material is NOT well prepared. Presentation sequence NOT well organized and not smooth. Handled questions badly and unable to demonstrate understanding of OO concept.	Reading occasionally from presentation material. Presentation material is well prepared. Presentation sequence acceptably organized and smooth. Handled questions well and demonstrated fundamental level of understanding of OO concept.	Good oral presentation. Presentation material is well prepared. Presentation sequence well organized and smooth. Handled questions well and demonstrated good understanding of OO concept.	Give an impactful presentation where the presenter delivers smooth oral presentation aided beautifully by well-prepared presentation material. Presentation sequence excellently planned organized and smooth. Handled questions well and demonstrated good understanding of OO concept.

Student Name: **Chan Jia Le** ID: **TP049952** **TOTAL MARKS**

Learning Outcomes	**Questions**
1. Explain the techniques of Object Oriented Design(C2,PLO1)	
2. Implement a software application that exploits the strength of object-oriented paradigm(C6,PL02)	LO covered in the assignment
3. Demonstrate the use of object oriented concepts and their functionalities in the existing system(A3,PL04)	LO covered in the assignment

REQUIREMENT ANALYSIS [CLO2-PLO2]	**Fail**	**Marginal Fail**	**Pass**	**Credit**	**Distinction**
	0-3	**4**	**5-6**	**7**	**8-10**
USE CASE DIAGRAM WITH DESCRIPTION (10%) [CLO2]	Incorrect overview use case, detail use cases with scenarios (where applicable) and use case descriptions OR incorrect user case notations OR illogical functional design.	Inappropriate overview use case, detail use cases with scenarios (where applicable) and use case descriptions.	The use cases are well presented (generalization) with no major mistake in logic and notation, and described all flows in use case descriptions.	Appropriate labelling and no mistake in logic and notation and clear description for normal flow, subflow and alternative flow in use case descriptions.	Comprehensive provision of the required - overview use case; - detail use case diagrams with scenarios (where applicable: generalization, extends, includes); - use case descriptions for each use case / scenario and no mistake in logic and notation and appropriateness.
CLASS DIAGRAM (10%) [CLO2 and CLO3]	No attributes and no associations are included.	Class diagram with attributes and associations. Both are incomplete and illogical.	Class diagram with attributes and associations. Both are complete with no major errors.	Class diagram with appropriate attributes and associations. With labelling and no mistakes in logic and notation.	Class diagram with appropriate, relevant attributes and associations. With very good and meaningful labelling according to guidelines.
				SUB TOTAL:	

38

IMPLEMENTATION [CLO2-PLO2]	**Fail**	**Marginal Fail**	**Pass**	**Credit**	**Distinction**
	0-3	**4**	**5-6**	**7**	**8-10**
USER LEVEL ACCESS (10%)	No program or work not done. Program incomplete with obvious errors. Not able to compile and run the program.	Not able to compile or run but evidence of the coding is available. Able to compile but not able to run the program. Able to compile and run the program but only able to add less than 50% of the details of access rights module listed in the assignment. Not able to demonstrate the use of object-oriented concepts. Data not stored in file.	Able to compile and run the program. Able to add at least 50% of the details of user access module listed in the assignment. Able to demonstrate the use of at least three of the object-oriented concepts – such as creation of classes, objects and methods. Attempted to write to file but with some errors.	Able to compile and run the program. Able to add at least 80% of the details of access rights module listed in the assignment. Able to demonstrate most of the object-oriented concepts. Able to write to file and read from file with some errors.	Able to compile and run the program. Able to add 90-100% of the details of access rights module listed in the assignment. Able to demonstrate all the object-oriented concepts along with additional features. Able to write to file and read from file with no errors.
PRODUCT AND SUPPLIER MANAGEMENT (30%) **(OR)** **PRODUCT CATALOGUE FOR DIFFERENT FESTIVALS (30%)**	**Fail**	**Marginal Fail**	**Pass**	**Credit**	**Distinction**
	0-11	**12-14**	**15-19**	**20-22**	**23-30**
	No program or work not done. Program incomplete with obvious errors. Not able to compile and run the program.	Not able to compile or run but evidence of the coding is available. Able to compile but not able to run the program. Able to compile and run the program but only able to add less than 50% of the details of module listed in the assignment. Not able to demonstrate the use of object-oriented concepts. Data not stored in file.	Able to compile and run the program. Able to add at least 50% of the details of module listed in the assignment. Able to demonstrate the use of at least three of the object-oriented concepts – such as creation of classes, objects and methods. Attempted to write to file but with some errors.	Able to compile and run the program. Able to add at least 80% of the details of the module listed in the assignment. Able to demonstrate most of the object-oriented concepts. Able to write to file and read from file with some errors.	Able to compile and run the program. Able to add 90-100% of the details of the module listed in the assignment. Able to demonstrate all the object-oriented concepts along with additional features. Able to write to file and read from file with no errors.
				SUB TOTAL:	

39

REPORT [CLO3-PLO4]	**Fail 0-3**	**Marginal Fail 4**	**Pass 5-6**	**Credit 7**	**Distinction 8-10**
REPORT FORMAT AND REFERENCES (10%)	The reference list is *all inapplicable* OR *irrelevant*. The format is NOT comply proper referencing system. The citation is NOT included at all. The simple document without formatting, header and footer, page number, etc.	The reference list is identified mostly inapplicable OR irrelevant, The format is partially comply proper referencing system. The citation is partially included. The simple document with little formatting.	The reference list is complete but sort of complete citation. The document is complete.	The above +. The citation is clearly specified. The above + with all the formatting criteria.	The above + with rich source of explorations to form a complete reference. The above + precise, clear, complete, all the diagram, chart, picture, pie, symbol, glossary are completely organized.
PROGRAM DOCUMENTATION (20%)	**Fail**	**Marginal Fail**	**Pass**	**Credit**	**Distinction**
	0-7	**8-9**	**10-12**	**13-14**	**15-20**
	Documentation not done. Content of documentation does not adhere to any of the requirements stipulated in the assignment requirements.	At least 1 object-oriented programming concept applied in the solution and briefly described. No implementation code for the object-oriented programming concepts identified. Screen capture of the output of the program does not have any explanation to describe the program.	Description of at least 2 object-oriented programming concepts that are applied in the solution with some evidence of the implementation code is included. Screen capture of the output of the program with minimal explanation to describe the program.	Description of at least 3 object-oriented programming concepts that are applied in the solution are described and evidence of the implementation code being documented. Screen capture of the output of the program with some explanation to describe the program. Description and evidence of at least 1 additional feature which has been incorporated in the solution.	Description of at least 4 object-oriented programming concepts that are applied in the solution and evidence of the implementation code being documented. Screen capture of the output of the program with appropriate explanation to sufficiently describe the program. Description and evidence of at least 2 additional features which have been incorporated in the solution.
INDIVIDUAL PRESENTATION (10%)	**0-3**	**4**	**5-6**	**7**	**8-10**
	Absent OR late OR not prepared for presentation session without valid reasons. Handled questions grossly bad and unable to demonstrate understanding of OO concept.	Reading from presentation material. Presentation material is NOT well prepared. Presentation sequence NOT well organized and not smooth. Handled questions badly and unable to demonstrate understanding of OO concept.	Reading occasionally from presentation material. Presentation material is well prepared. Presentation sequence acceptably organized and smooth. Handled questions well and demonstrated fundamental level of understanding of OO concept.	Good oral presentation. Presentation material is well prepared. Presentation sequence well organized and smooth. Handled questions well and demonstrated good understanding of OO concept.	Give an impactful presentation where the presenter delivers smooth oral presentation aided beautifully by well-prepared presentation material. Presentation sequence excellently planned organized and smooth. Handled questions well and demonstrated good understanding of OO concept.