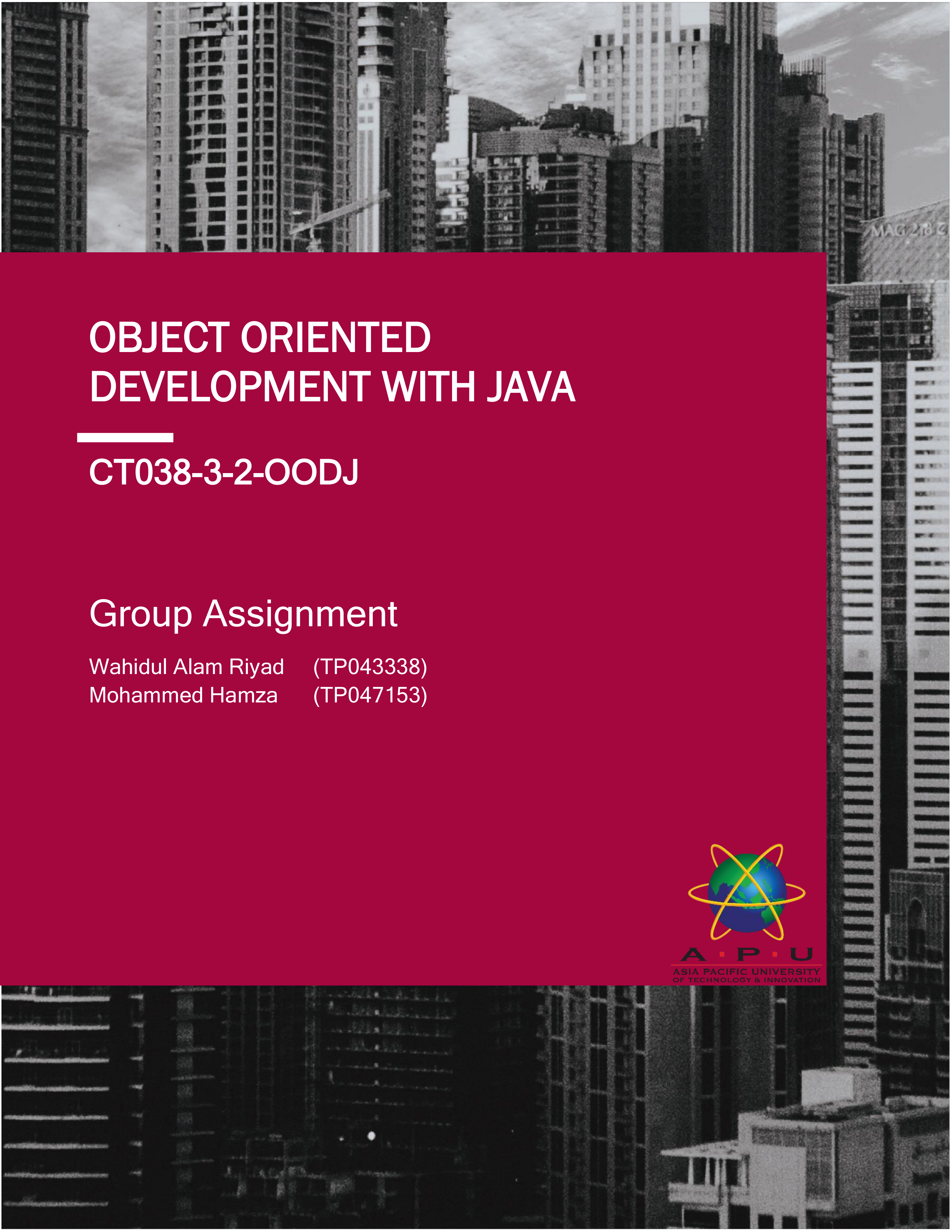


OBJECT ORIENTED DEVELOPMENT WITH JAVA

CT038-3-2-OODJ

Group Assignment

Wahidul Alam Riyad (TP043338)
Mohammed Hamza (TP047153)

ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

1.0 Table of Contents

2.0 Introduction

Object-oriented programming (OOP) is a programming language model organized around objects rather than "actions" and data rather than logic. Historically, a program has been viewed as a logical procedure that takes input data, processes it, and produces output data (Rouse, 2019).

Nowadays, developers implemented object-oriented techniques in variety computing areas, such as maintaining bank transactions, analyzing stored data, designing video games, and developing desktop applications. The main advantage of object-oriented programming is it allows the developers to reuse objects and generates it during the runtime.

In this assignment, students are required to design ***Retail Order Management System*** for small retailers. And there are two different stockholders, ***admin and customer***. Admin has an ability to manage customers, manage products and manage orders. Based on the system requirements, customer can only manage orders.

3.0 UML Diagram

3.1 Use Case Diagram

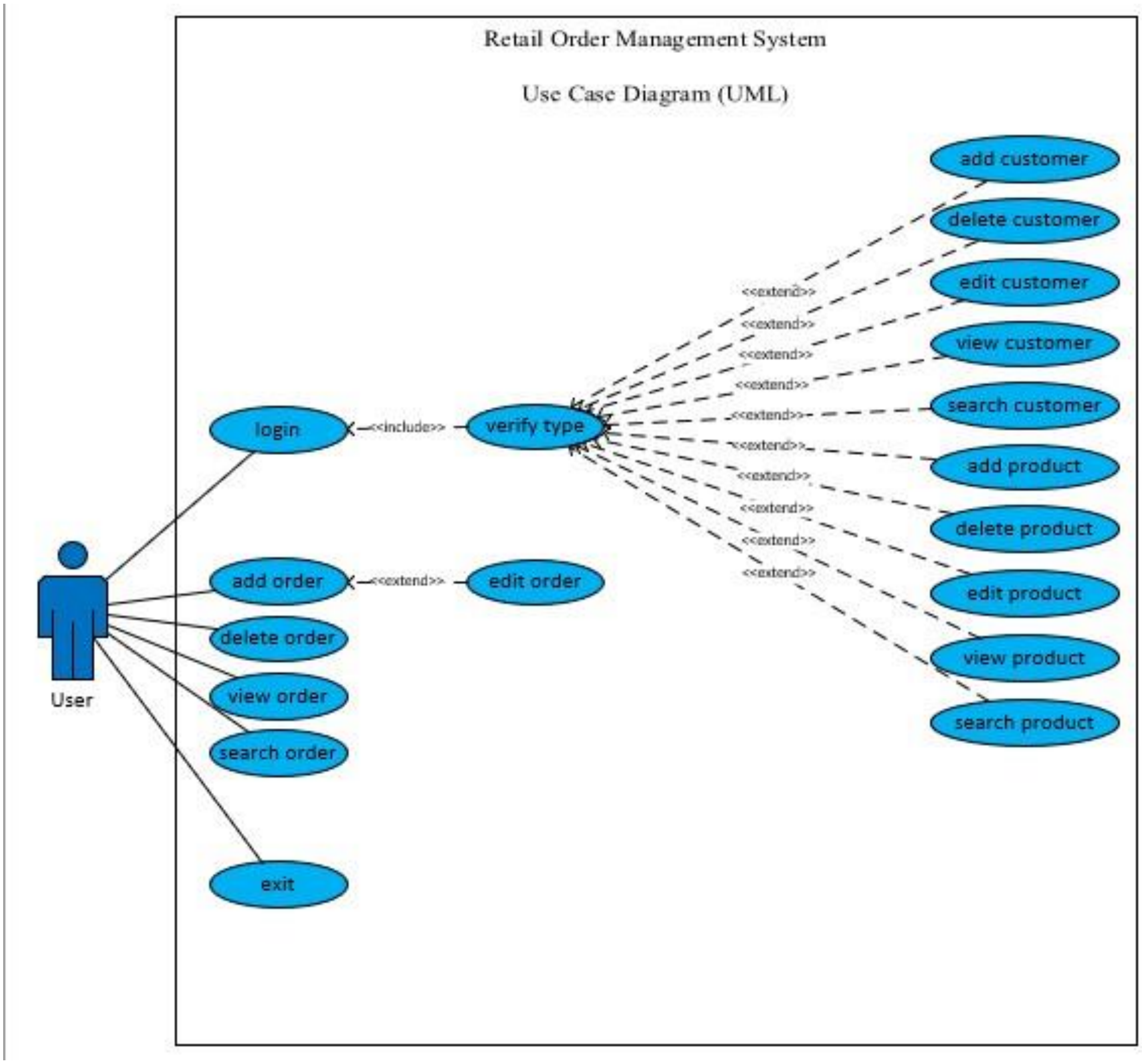


Figure: Use Case Diagram representing the Retail Order Management System

The case scenario represented that there are two different types of users, and both of them have an ability to manage orders, such as add order, delete order, view order and search order. If user is an admin it means there are additional functionalities, namely manage products and manage orders. There is a login function in this use case diagram, and function is responsible to check login credentials and identify user types.

Use Case:	Login
Short Description:	Allows the user to login the system
Pre-Conditions:	The user can verify type.
Post Conditions:	The user can add, delete, edit, view, search customer. And add, delete, edit, view, search product.

Use Case:	Add order
Short Description:	Allows the user to add order to cart.
Pre-Conditions:	The user can place orders.
Post Conditions:	The user can edit order also.

Use Case:	Delete order
Short Description:	The user can delete their order
Pre-Conditions:	The user must have successfully login before being able to delete orders.
Post Conditions:	Order has been Delete.

Use Case:	View order
Short Description:	The user can view their order
Pre-Conditions:	-
Post-Conditions:	The user must have successfully login before being able to view orders.

Use Case:	Search order
Short Description:	The user can search their order
Pre-Conditions:	-
Post Conditions:	The user must have successfully login before being able to view orders.

Use Case:	Exit
Short Description:	The user can exit to the program by exit option.
Pre-Conditions:	-
Post Conditions:	The user must have into system for exiting the Program.

3.2 Class Diagram

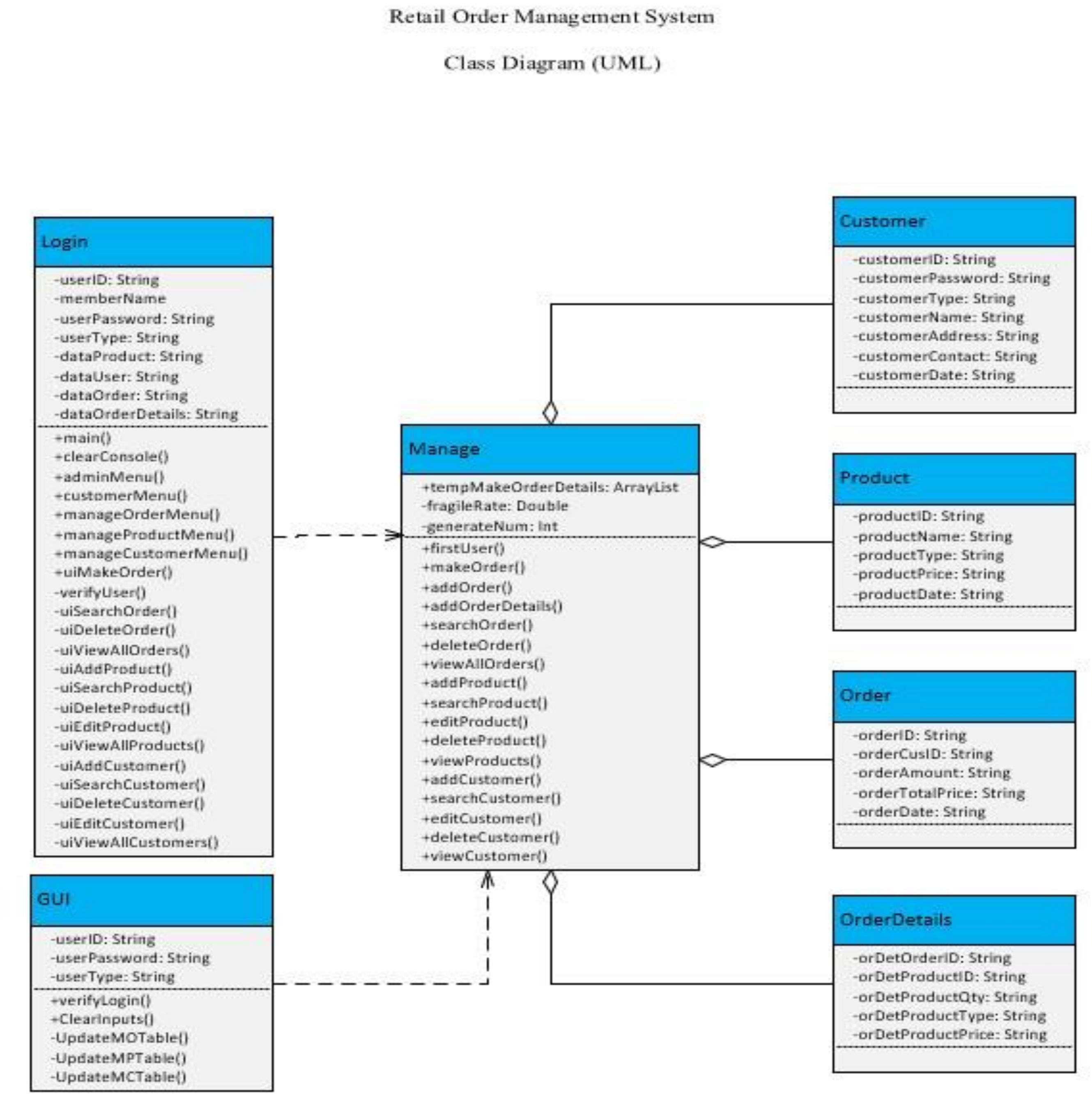


Figure: Class Diagram representing class relationships of the System

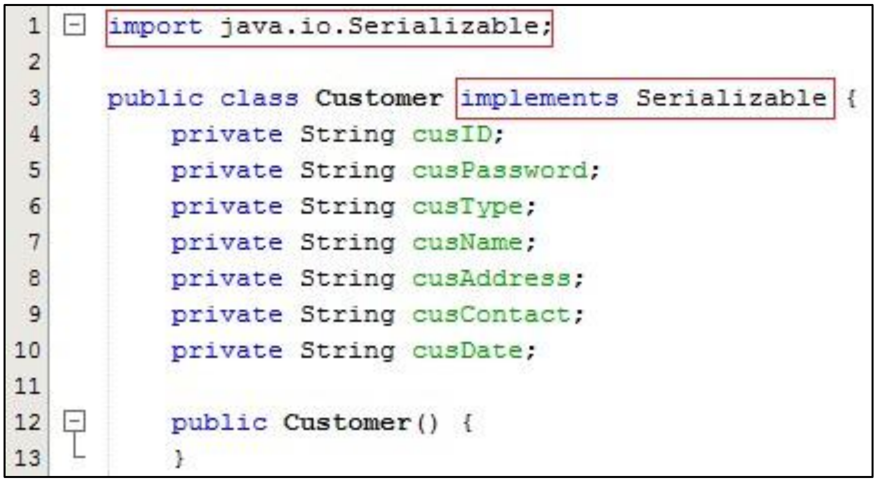
Within the system, there are totally 6 different classes. And 4 classes are used to describe objects, namely customer object, product object, order object, order details object. In addition, all of the object behaviors or object related methods are located in manage class, as well as login class is dependent to manage class and responsible to support system user interactions.

4.0 Implemented Techniques

4.1 Serialization & Deserialization

The process of converting an object into a collection of bytes is called as serialization and converted can be stored to a disk or database or can be sent through streams. The deserialization is a reverse process of creating object from collection of bytes.

In this assignment, student implemented serialization & deserialization techniques to store list of objects into single file (Figure). There are four different types of objects: customer object, product object, order object and order details object. As well as, all stored objects are converted to sequences of bytes and highly protected.

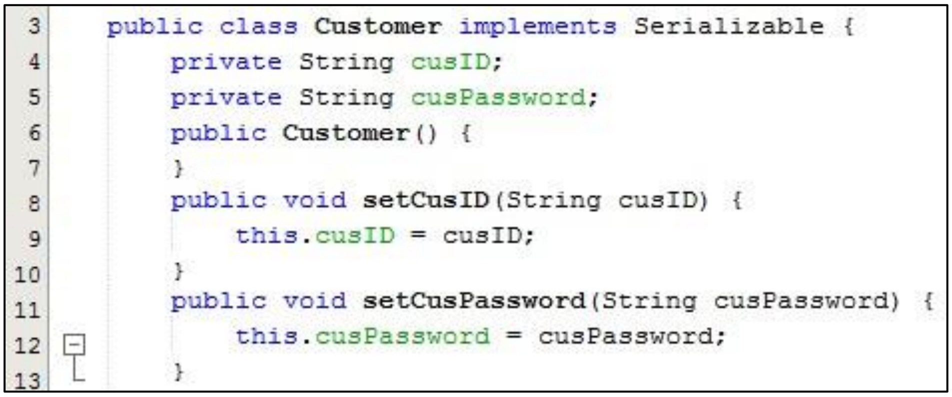


```
import java.io.Serializable;

public class Customer implements Serializable {
    private String cusID;
    private String cusPassword;
    private String cusType;
    private String cusName;
    private String cusAddress;
    private String cusContact;
    private String cusDate;

    public Customer() {
    }
```

Figure: The Serialization techniques implemented to customer class.

4.2 Encapsulation

Encapsulation is one of the vital fundamentals of OOP concepts. Encapsulation in java is a process of wrapping the variables and code acting on the methods together as a single unit. In result, the variables of a class will be invisible from different classes and can be accessible only through the pubic methods of their current class. In this assignment, developer implemented encapsulation techniques to object properties (Figure), as consequence, all original properties are hidden from external classes.

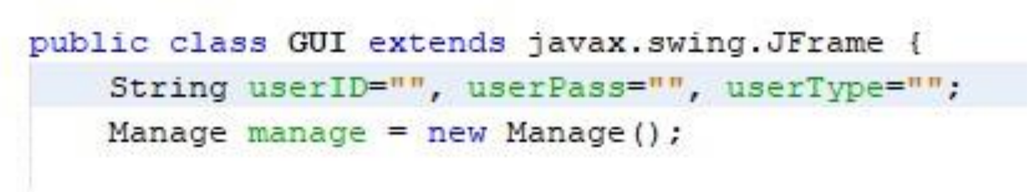


```
public class Customer implements Serializable {
    private String cusID;
    private String cusPassword;
    public Customer() {
    }
    public void setCusID(String cusID) {
        this.cusID = cusID;
    }
    public void setCusPassword(String cusPassword) {
        this.cusPassword = cusPassword;
    }
```

Figure: It represents the implementation of encapsulation techniques.

4.3 Inheritance

An important feature of object-oriented programs is inheritance. The ability to create classes that share the attributes and methods of existing classes, but with more specific features. Inheritance is mainly used for code reusability. So, you are making use of already written the classes and further extending on that. That why we discussed the code reusability the concept. In general, one-line definition, we can tell that deriving a new class from existing class, it’s called as Inheritance.

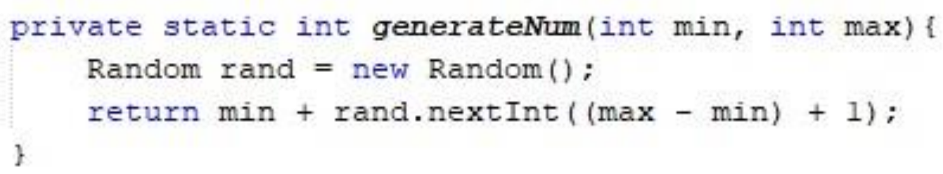


```
public class GUI extends javax.swing.JFrame {
    String userID="", userPass="", userType="";
    Manage manage = new Manage();
```

Figure: It represents the implementation of inheritance technique.

4.4 Polymorphism

When you create a subclass by extending an existing class, the new subclass contains data and methods that were defined in the original superclass. In other words, any child class object has all the attributes of its parent. Sometimes, however, the superclass data fields and methods are not entirely appropriate for the subclass objects; in these cases, you want to override the parent class members.

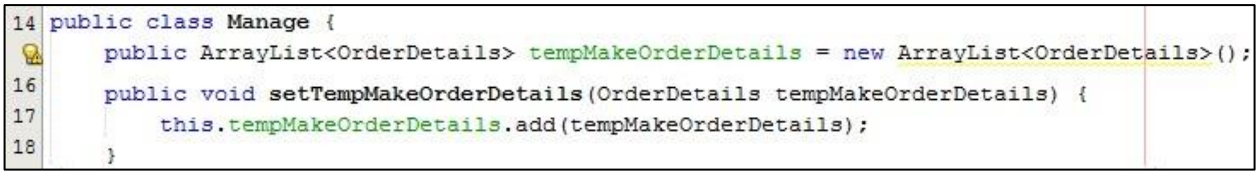


```java
private static int generateNum(int min, int max){
    Random rand = new Random();
    return min + rand.nextInt((max - min) + 1);
}
```

Figure: It represents the polymorphism technique.

4.5 ArrayList

ArrayList is a part of standard library and it is available through **java.util.ArrayList** package. An ArrayList provides dynamic non-primitive arrays, and it is clear that it is slower than normal primitive arrays but ArrayList can be helpful in programs to manipulate sequences of non-primitive objects. In this assignment, student implemented library ArrayList to store temporary objects, such as list of ordered objects (Figure).

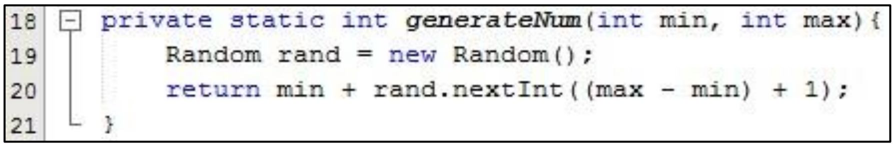


```
public class Manage {
    public ArrayList<OrderDetails> tempMakeOrderDetails = new ArrayList<OrderDetails>();
    public void setTempMakeOrderDetails(OrderDetails tempMakeOrderDetails) {
        this.tempMakeOrderDetails.add(tempMakeOrderDetails);
    }
```

Figure: It represents implementation of ArrayList for Order Details

4.6 Generate Random Numbers

In this assignment, students have to use unique values to describe objects, such as customer id, product id and order id. In this case, students should use java library collections to generate random number. This advantage helps with processing time and saves extra lines of code. If developer wants to use a certain feature, such as generating a random number, first need to import **java.util.Random** library into the code, then need to declare few lines of code at the beginning of the class or before the main function (Figure).

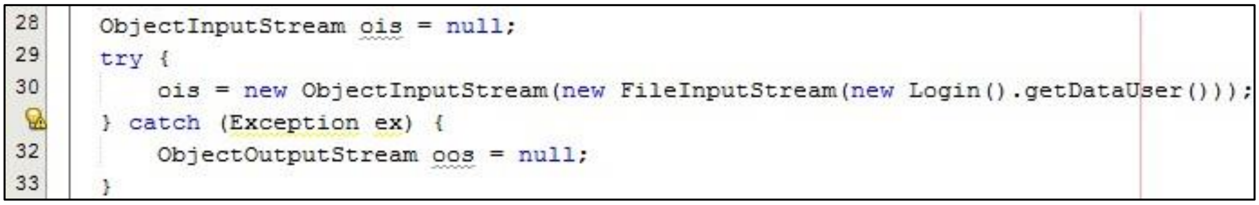


```java
private static int generateNum(int min, int max){
    Random rand = new Random();
    return min + rand.nextInt((max - min) + 1);
}
```

Figure: It represents a generateNum function.

4.7 ObjectInputStream & ObjectOutputStream

The ObjectInputStream & ObjectOutputStream is a java class, and both of them are available under the standard java library collection (**java.io.ObjectOutputStream**). It enables users to store java objects to an OutputStream instead of just primitive bytes, and the InputStream is used to call the objects again. In this assignment, student used ObjectStream features to get access to output files, which hold user objects, product objects, order objects and order details objects (Figure).

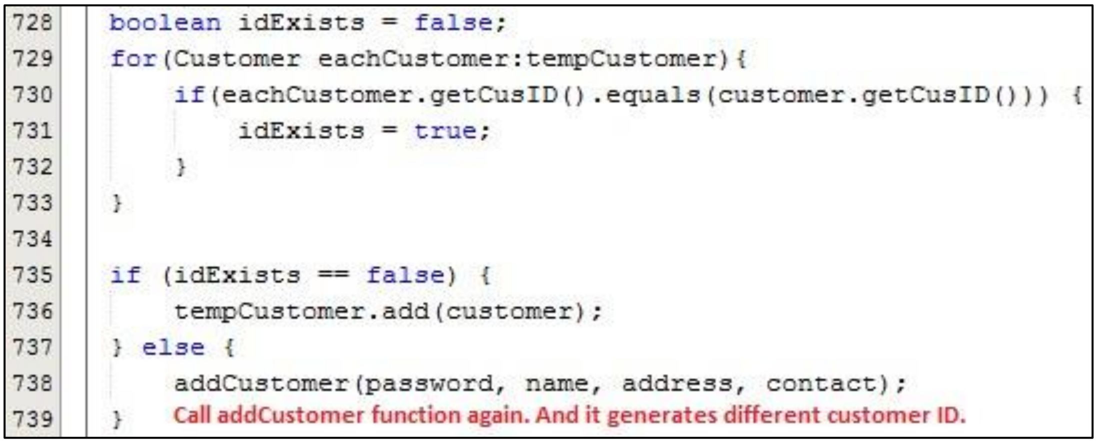


```
ObjectInputStream ois = null;
try {
    ois = new ObjectInputStream(new FileInputStream(new Login().getDataUser()));
} catch (Exception ex) {
    ObjectOutputStream oos = null;
}
```

Figure: It represents implementation of ObjectInputStream, which is expected to read customer data.

4.8 No Duplication of IDs

Within the system, unique identification numbers are implemented to differentiate objects. Such as customer id (**UID000000**), product id (**PID000000**) and order id (**OID000000**). Each value is 9 characters long, 3 capital letters which based on object names, and 6 random generated numbers. During the registration process, if generated identification number is already used for describing certain objects, then if-else statement recalls whole function again, in result, system generates new object identification number.



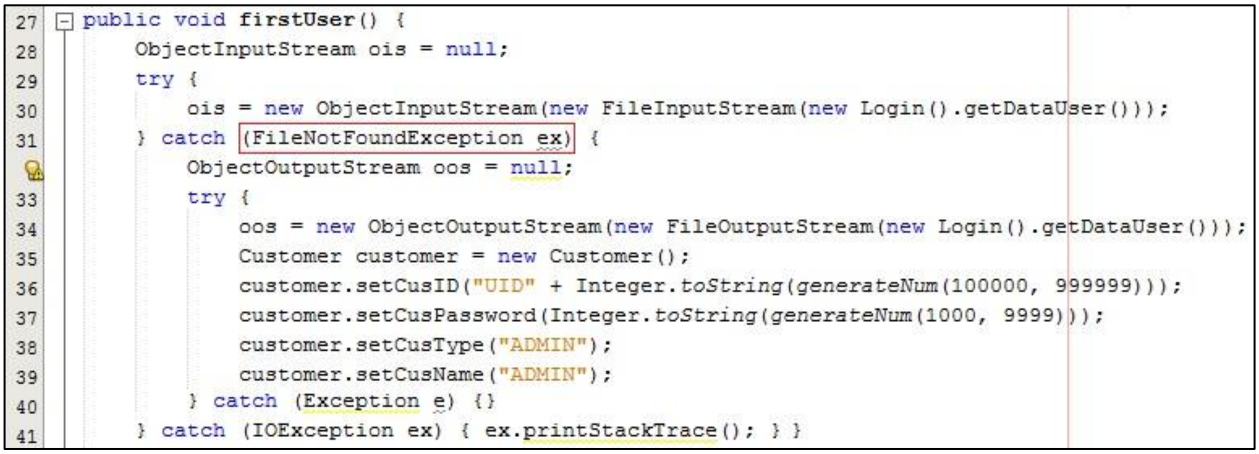
```
boolean idExists = false;
for(Customer eachCustomer:tempCustomer){
    if(eachCustomer.getCusID().equals(customer.getCusID())) {
        idExists = true;
    }
}

if (idExists == false) {
    tempCustomer.add(customer);
} else {
    addCustomer(password, name, address, contact);
}
```

Call addCustomer function again. And it generates different customer ID.

Figure: The simple logic is implemented to prevent duplication of object identification number.

4.9 First User is Auto Generated

The figure in the below represents simple logic which designed to generate first user or an admin. At the beginning, logic checks that is there an output file or not which contains user data (via **ObjectInputStream** and **FileNotFoundException**), if not exists then creates new output file, and stores auto generated first user, if an output file is already exists, then skips this procedure. During the generation process, system randomly creates credentials, such as user id and user password (format: UID000000, 0000).

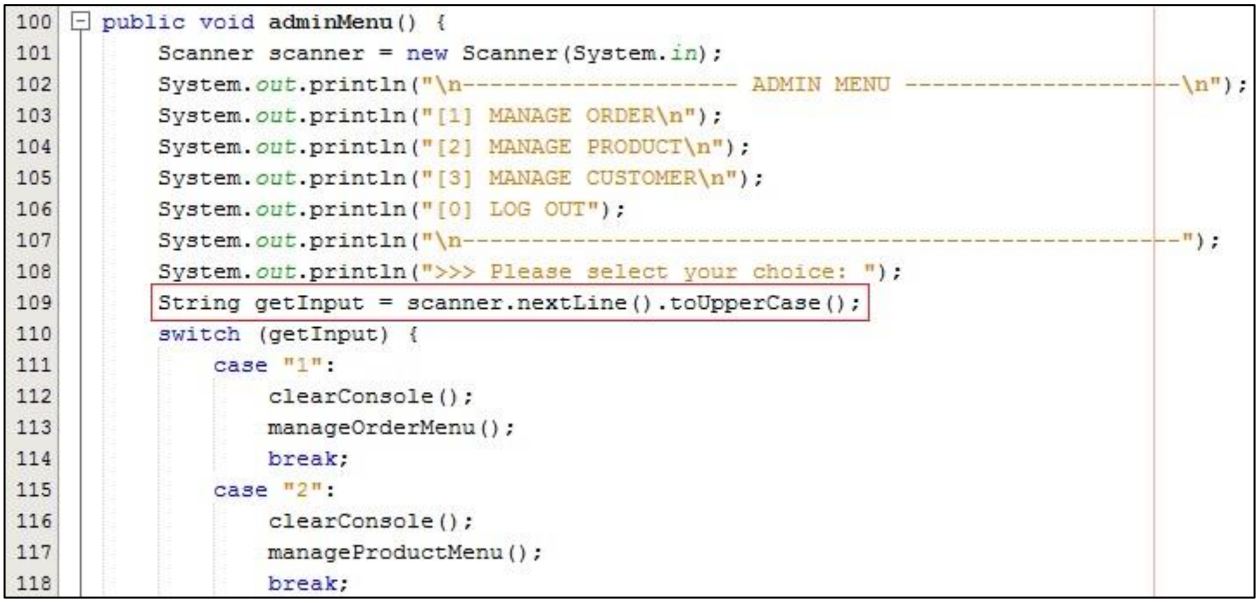


```
public void firstUser() {
    ObjectInputStream ois = null;
    try {
        ois = new ObjectInputStream(new FileInputStream(new Login().getDataUser()));
    } catch (FileNotFoundException ex) {
        ObjectOutputStream oos = null;
        try {
            oos = new ObjectOutputStream(new FileOutputStream(new Login().getDataUser()));
            Customer customer = new Customer();
            customer.setCusID("UID" + Integer.toString(generateNum(100000, 999999)));
            customer.setCusPassword(Integer.toString(generateNum(1000, 9999)));
            customer.setCusType("ADMIN");
            customer.setCusName("ADMIN");
        } catch (Exception e) {}
    } catch (IOException ex) { ex.printStackTrace(); } }
```

Figure: The simple logic is implemented to create first user (admin) if system has no user.

4.10 No Case Sensitive for Input Values

In this assignment, system is designed as no case sensitive for all input values. For example, "admin1" and “ADMIN1” are same inputs. Because system automatically converts all inputs to uppercase letters before processing or storing it an output file (Figure). In this case, users do not have to think about difference between uppercase and lowercase letters, in one word, system is designed as user friendly, and easy to use.

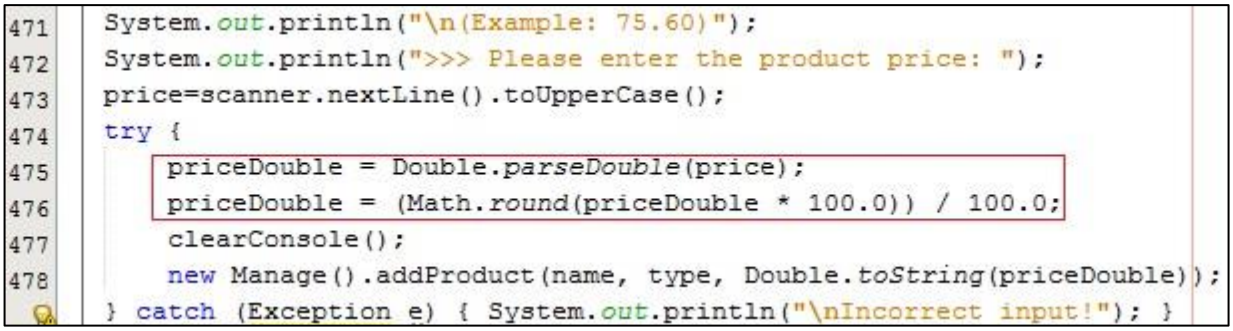


```java
public void adminMenu() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("\n-------------------- ADMIN MENU -------------------\n");
    System.out.println("[1] MANAGE ORDER\n");
    System.out.println("[2] MANAGE PRODUCT\n");
    System.out.println("[3] MANAGE CUSTOMER\n");
    System.out.println("[0] LOG OUT");
    System.out.println("\n----------------------------------------------------");
    System.out.println(">>> Please select your choice: ");
    String getInput = scanner.nextLine().toUpperCase();
    switch (getInput) {
        case "1":
            clearConsole();
            manageOrderMenu();
            break;
        case "2":
            clearConsole();
            manageProductMenu();
            break;
```

Figure: It represents admin menu.

4.11 Math.Round()

In this assignment, a Math.round() method is implemented to round product prices in 0.00 format (Math.round() is available under the standard java libraries). Because, if user enters unrealistic values such as 111.1111 or 12.99998 then system automatically converts it to double format, which is 111.11 and 13.0 respectively. In result, system become more accurate in calculations and become more user friendly.

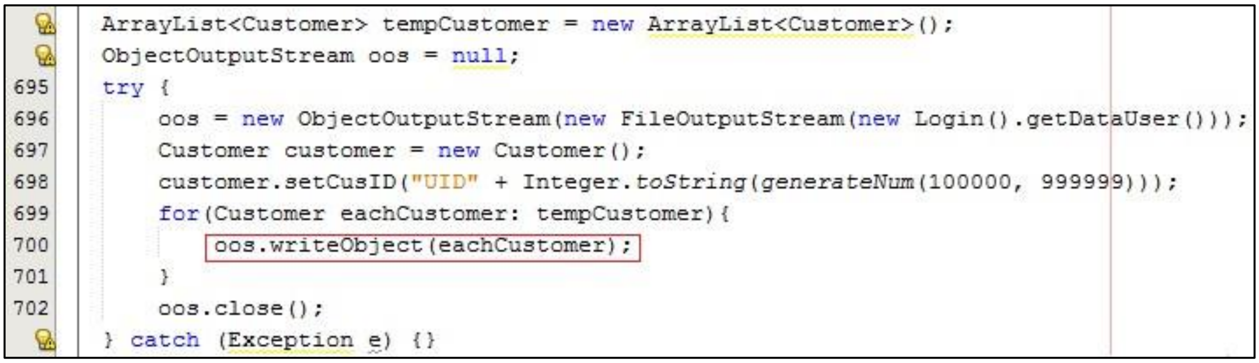


```
System.out.println("\n(Example: 75.60)");
System.out.println(">>> Please enter the product price: ");
price=scanner.nextLine().toUpperCase();
try {
    priceDouble = Double.parseDouble(price);
    priceDouble = (Math.round(priceDouble * 100.0)) / 100.0;
    clearConsole();
    new Manage().addProduct(name, type, Double.toString(priceDouble));
} catch (Exception e) { System.out.println("\nIncorrect input!"); }
```

Figure: It represents the implementation of Math.round method.

4.12 Objects are Implemented to Store Outputs

Objects are vital functionalities of object-oriented programing. Within the program, 4 different objects are implemented to support system (customer object, product object, order object and order details object), and they can be treated as variables, just need to declare it with the ***new*** keyword. The keyword invokes matched constructor defined in the class to create an object. Objects have fields to store values and methods for action. Fields and methods of one object are distinct from other objects of the same class.

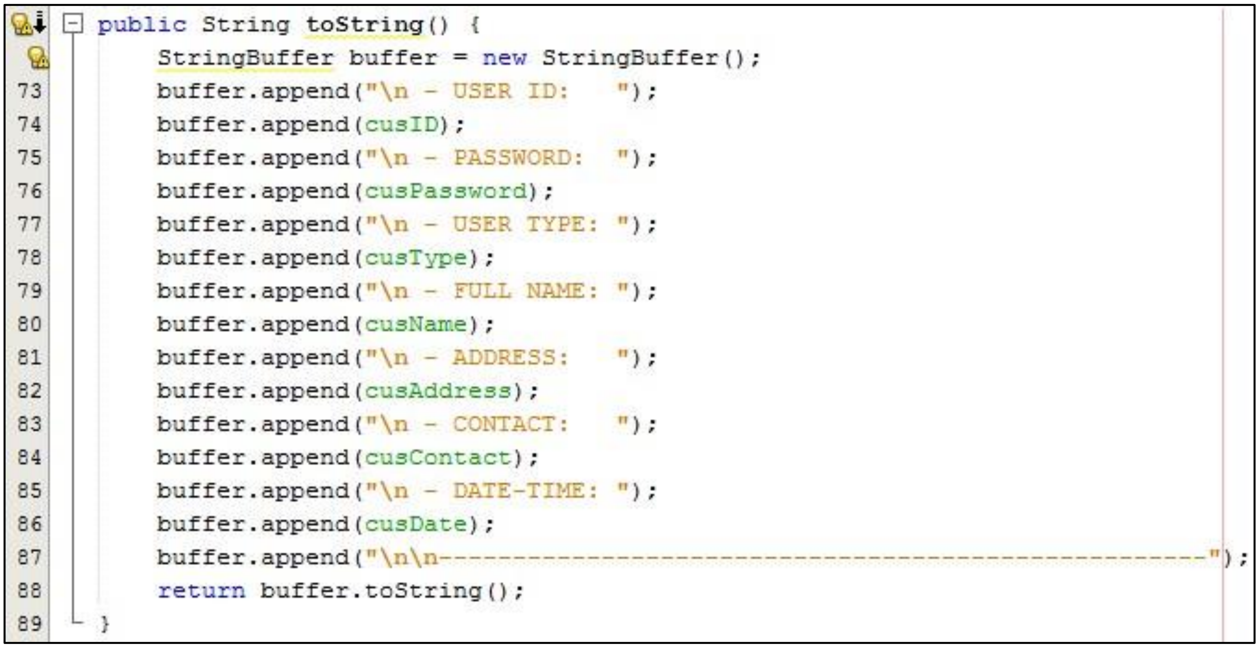


```java
ArrayList<Customer> tempCustomer = new ArrayList<Customer>();
ObjectOutputStream oos = null;
try {
    oos = new ObjectOutputStream(new FileOutputStream(new Login().getDataUser()));
    Customer customer = new Customer();
    customer.setCusID("UID" + Integer.toString(generateNum(100000, 999999)));
    for(Customer eachCustomer: tempCustomer){
        oos.writeObject(eachCustomer);
    }
    oos.close();
} catch (Exception e) {}
```

Figure: It represents the procedure of writing objects to an output file.

4.13 ToString() Method is Implemented

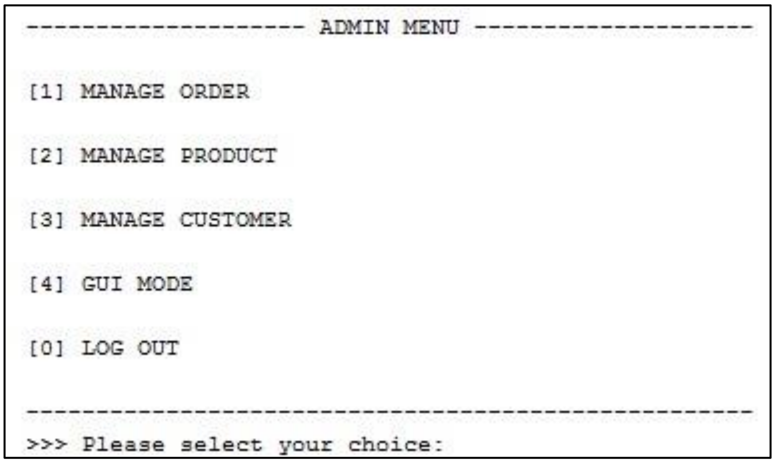
The java toString() method is useful when user need a string representation of an object. It is declared in Object class. And toString() method can be overridden to update or change the String representation of the Object. The figure in the below represents implementation of toString() method, and method is located within the Customer class.



```java
public String toString() {
    StringBuffer buffer = new StringBuffer();
    buffer.append("\n - USER ID:    ");
    buffer.append(cusID);
    buffer.append("\n - PASSWORD:  ");
    buffer.append(cusPassword);
    buffer.append("\n - USER TYPE: ");
    buffer.append(cusType);
    buffer.append("\n - FULL NAME: ");
    buffer.append(cusName);
    buffer.append("\n - ADDRESS:    ");
    buffer.append(cusAddress);
    buffer.append("\n - CONTACT:    ");
    buffer.append(cusContact);
    buffer.append("\n - DATE-TIME: ");
    buffer.append(cusDate);
    buffer.append("\n\n-----------------------------------------------------");
    return buffer.toString();
}
```

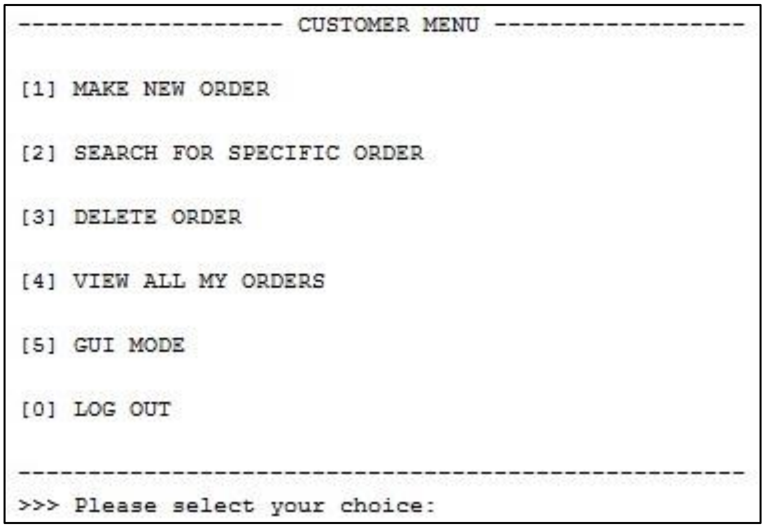
Figure: It represents toString() method which is implemented to display customer details.

4.14 Terminal User Interface



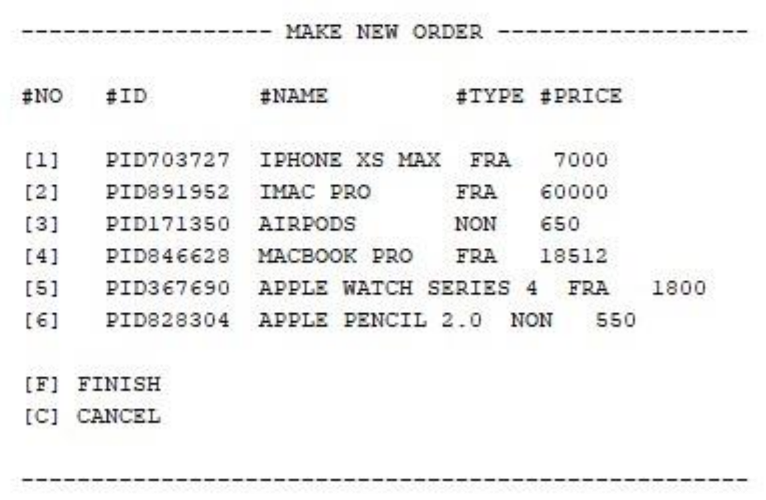
```
-------------------- ADMIN MENU --------------------

[1] MANAGE ORDER

[2] MANAGE PRODUCT

[3] MANAGE CUSTOMER

[4] GUI MODE

[0] LOG OUT

----------------------------------------------------
>>> Please select your choice:
```

Figure: Admin Menu



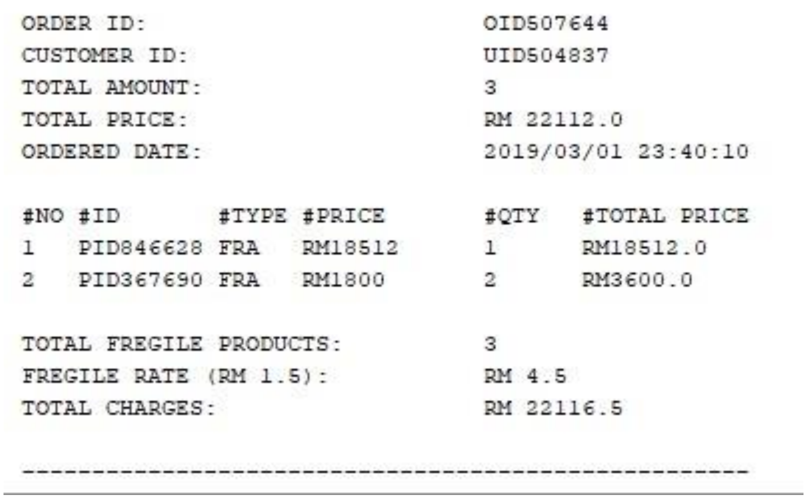
```
------------------- CUSTOMER MENU -------------------

[1] MAKE NEW ORDER

[2] SEARCH FOR SPECIFIC ORDER

[3] DELETE ORDER

[4] VIEW ALL MY ORDERS

[5] GUI MODE

[0] LOG OUT

-----------------------------------------------------
>>> Please select your choice:
```

Figure: Customer Menu



```
------------------ MAKE NEW ORDER ------------------

#NO   #ID        #NAME         #TYPE #PRICE

[1]   PID703727  IPHONE XS MAX  FRA   7000
[2]   PID891952  IMAC PRO      FRA   60000
[3]   PID171350  AIRPODS       NON   650
[4]   PID846628  MACBOOK PRO   FRA   18512
[5]   PID367690  APPLE WATCH SERIES 4  FRA   1800
[6]   PID828304  APPLE PENCIL 2.0  NON   550

[F] FINISH
[C] CANCEL

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

Figure: Make New Order



```
ORDER ID:                         OID507644
CUSTOMER ID:                      UID504837
TOTAL AMOUNT:                     3
TOTAL PRICE:                      RM 22112.0
ORDERED DATE:                     2019/03/01 23:40:10

#NO #ID       #TYPE #PRICE        #QTY   #TOTAL PRICE
1   PID846628 FRA   RM18512       1      RM18512.0
2   PID367690 FRA   RM1800        2      RM3600.0

TOTAL FREGILE PRODUCTS:           3
FREGILE RATE (RM 1.5):            RM 4.5
TOTAL CHARGES:                    RM 22116.5

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

Figure: Report Page

4.15 Graphical User Interface

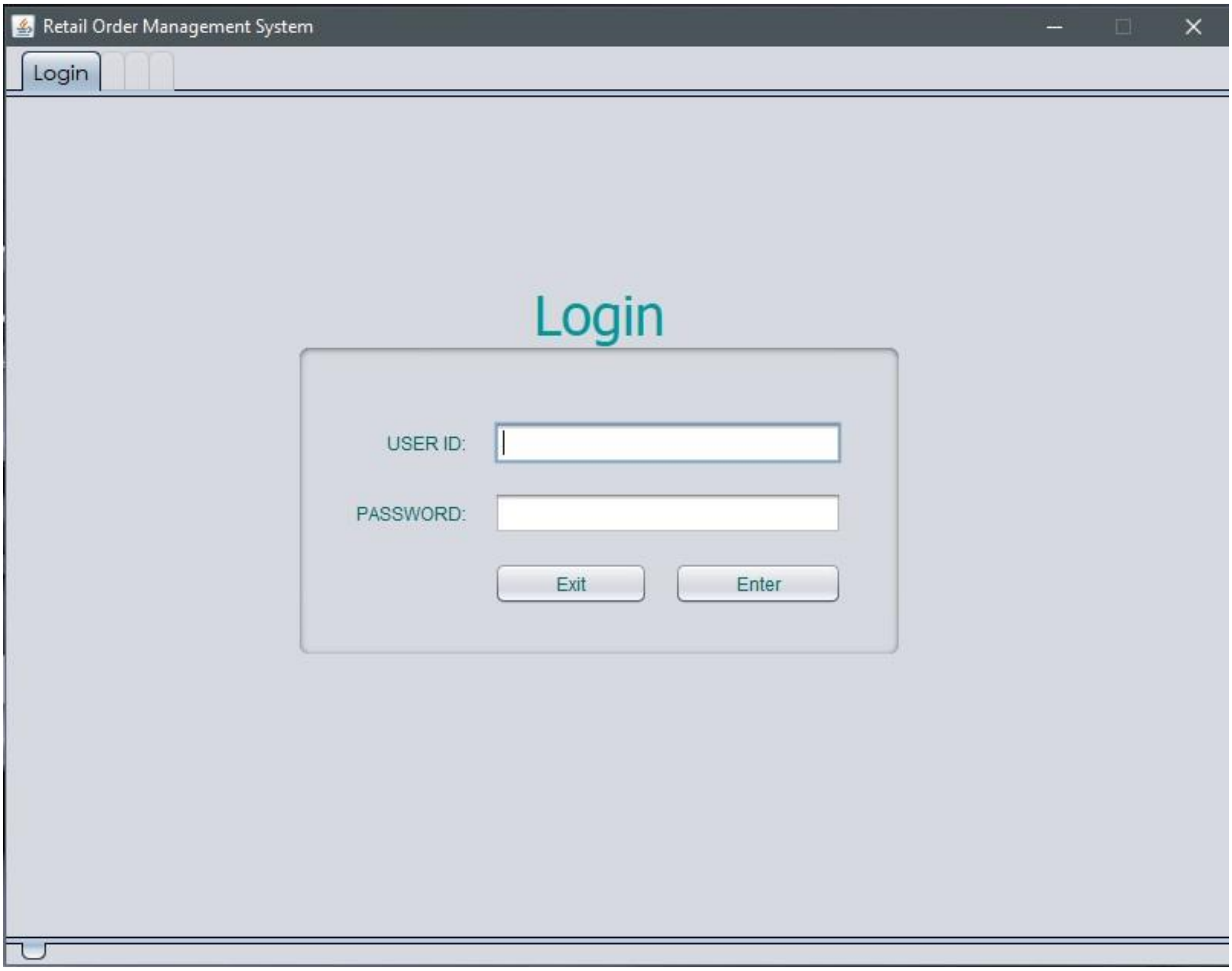
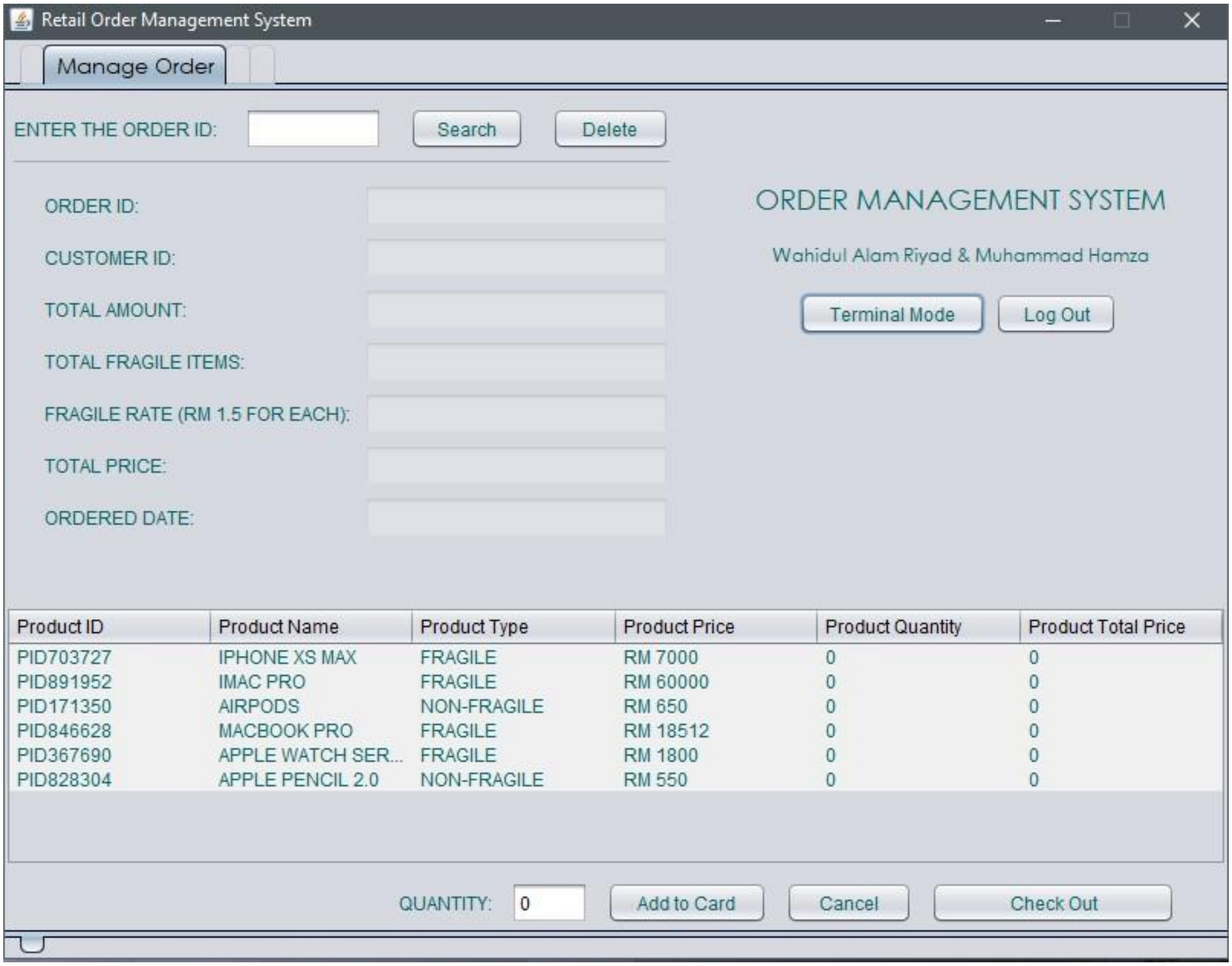


Figure: Login Page (GUI Mode)

This is the Login page in GUI mode. Here the user can login based on the user type, such as Admin or Customer. Based on the user type, the menu panel will show.

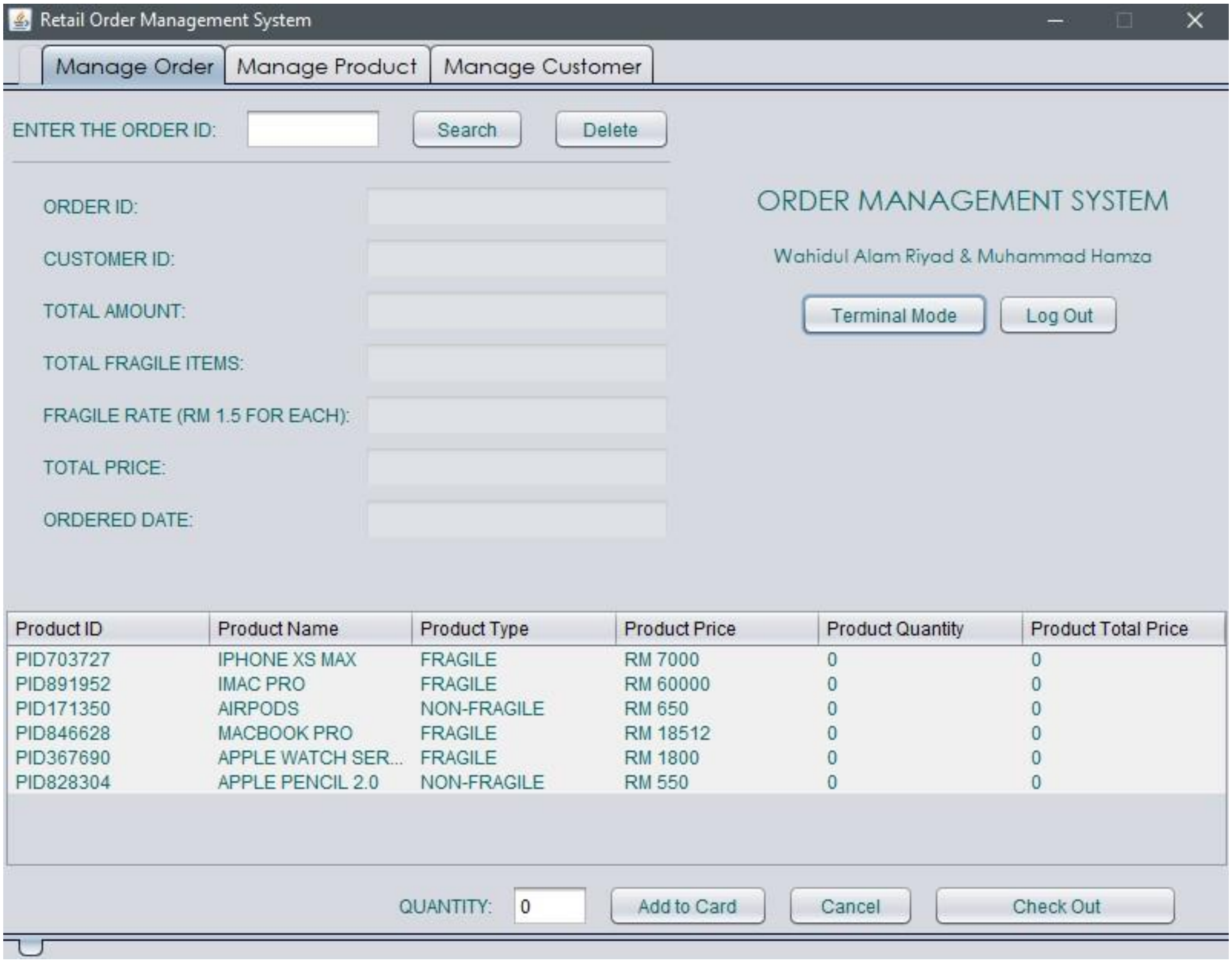


Product ID	Product Name	Product Type	Product Price	Product Quantity	Product Total Price
PID703727	IPHONE XS MAX	FRAGILE	RM 7000	0	0
PID891952	IMAC PRO	FRAGILE	RM 60000	0	0
PID171350	AIRPODS	NON-FRAGILE	RM 650	0	0
PID846628	MACBOOK PRO	FRAGILE	RM 18512	0	0
PID367690	APPLE WATCH SER...	FRAGILE	RM 1800	0	0
PID828304	APPLE PENCIL 2.0	NON-FRAGILE	RM 550	0	0

QUANTITY: 0

Figure: Customer Page (GUI Mode)

This is the Customer page where customer can manage order. The customer can order products, search order, delete order and display bill. Terminal mode can also be activated from here.

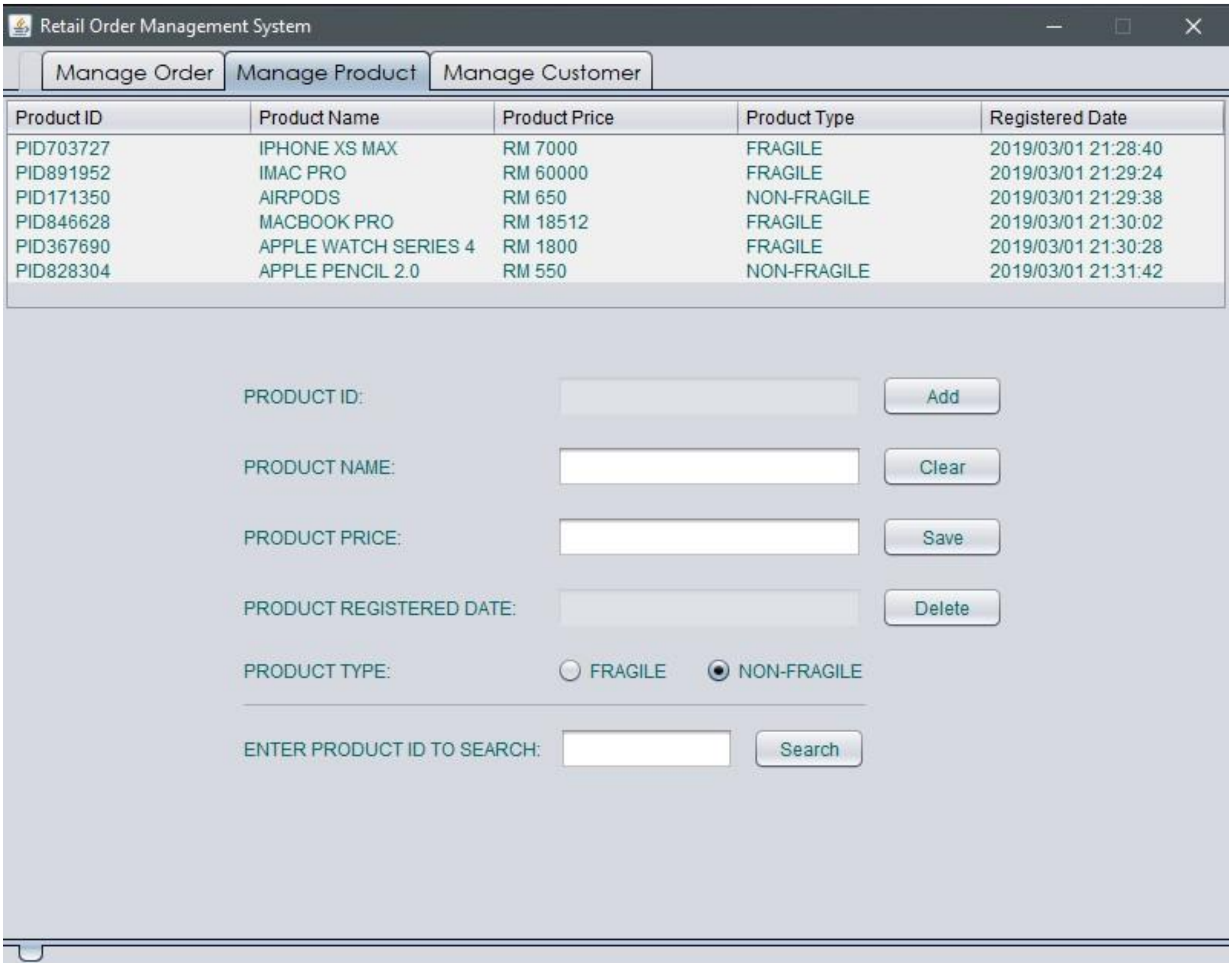


Product ID	Product Name	Product Type	Product Price	Product Quantity	Product Total Price
PID703727	IPHONE XS MAX	FRAGILE	RM 7000	0	0
PID891952	IMAC PRO	FRAGILE	RM 60000	0	0
PID171350	AIRPODS	NON-FRAGILE	RM 650	0	0
PID846628	MACBOOK PRO	FRAGILE	RM 18512	0	0
PID367690	APPLE WATCH SER...	FRAGILE	RM 1800	0	0
PID828304	APPLE PENCIL 2.0	NON-FRAGILE	RM 550	0	0

QUANTITY: 0

Figure: Manage Order Page (GUI Mode)

This is the Admin page where admin can manage order. The admin can order products, search order, delete order and display bill. Terminal mode can also be activated from here.



Product ID	Product Name	Product Price	Product Type	Registered Date
PID703727	IPHONE XS MAX	RM 7000	FRAGILE	2019/03/01 21:28:40
PID891952	IMAC PRO	RM 60000	FRAGILE	2019/03/01 21:29:24
PID171350	AIRPODS	RM 650	NON-FRAGILE	2019/03/01 21:29:38
PID846628	MACBOOK PRO	RM 18512	FRAGILE	2019/03/01 21:30:02
PID367690	APPLE WATCH SERIES 4	RM 1800	FRAGILE	2019/03/01 21:30:28
PID828304	APPLE PENCIL 2.0	RM 550	NON-FRAGILE	2019/03/01 21:31:42

PRODUCT ID:

PRODUCT NAME:

PRODUCT PRICE:

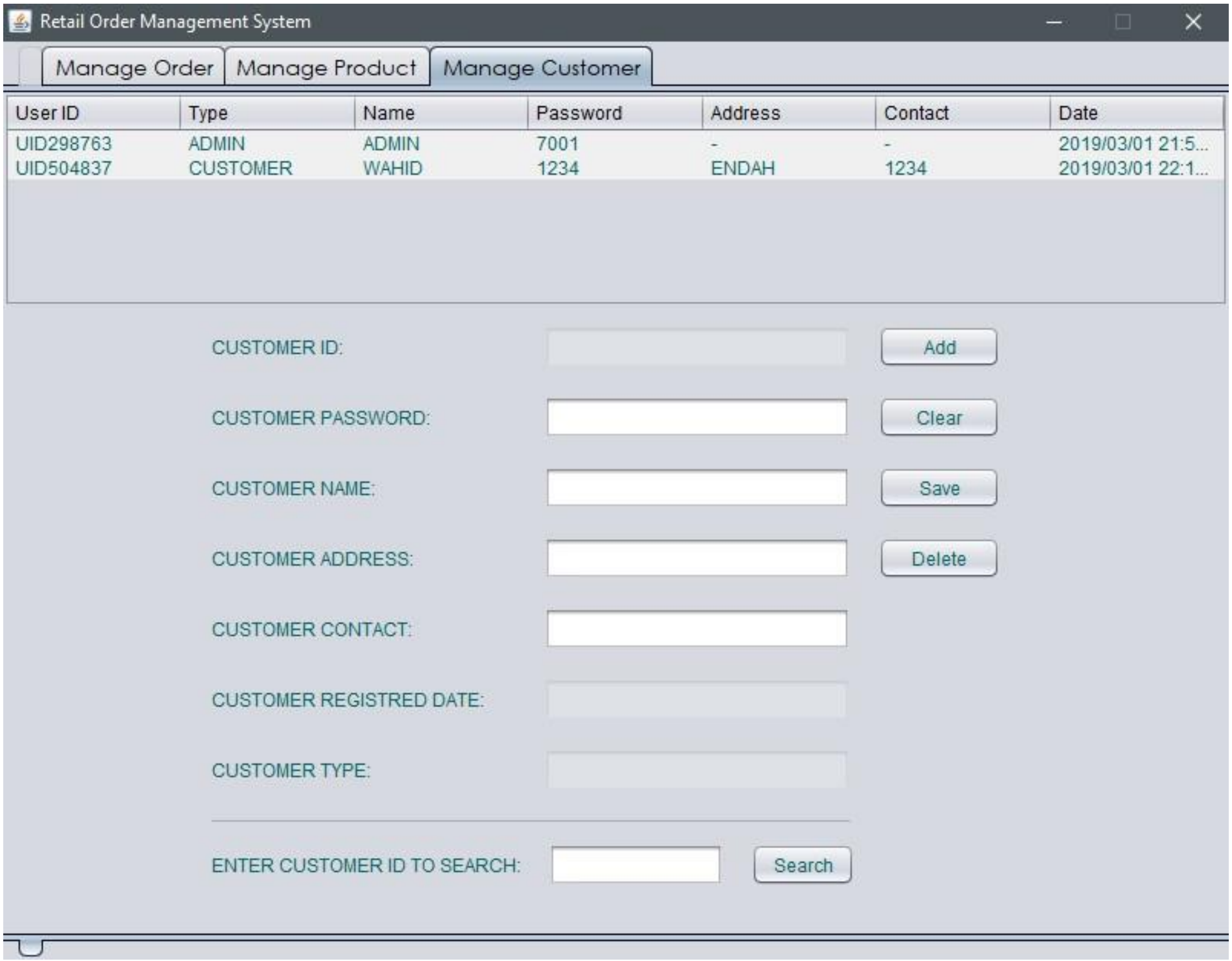
PRODUCT REGISTERED DATE:

PRODUCT TYPE: ○ FRAGILE ● NON-FRAGILE

ENTER PRODUCT ID TO SEARCH:

Figure: Manage Product Page (GUI Mode)

This is the panel for admin where admin can add new products, search products, update and delete products. For each new product, new product ID is created.



User ID	Type	Name	Password	Address	Contact	Date
UID298763	ADMIN	ADMIN	7001	-	-	2019/03/01 21:5...
UID504837	CUSTOMER	WAHID	1234	ENDAH	1234	2019/03/01 22:1...

Figure: Manage Customer Page (GUI Mode)

This is the panel for admin to create new customers by filling their details. For each customer, there is a unique ID which later helps them to login the system. Here admin can add, search, update and delete customers.

5.0 Conclusion

There is no doubt that object-oriented programming has vital functionalities in designing large systems. It allows to developers to divide tasks in small unities, then it gives possibilities to reuse it or integrate it with other part of system. In this assignment, students learn basic fundamentals of object-oriented programming, such as inheritance, polymorphism, encapsulation, and abstraction. On the other hand, students got chance to differentiate between variety data types and its unique features, for example Vectors and ArrayList, then implemented it to their assignments. As well as, students learn how to allocate resources effectively (memory), or how to design an optimal algorithm (timely efficient). And it was clear that all researches and practices are definitely helpful for future software engineers (Object, 2019).

6.0 Workload Matrix

TASK ID	TASK NAME	Wahid TP043338	Hamza TP047153
1	USE CASE DIAGRAM	50%	50%
2	CLASS DIAGRAM	50%	50%
3	USER LEVEL ACCESS	50%	50%
4	PRODUCT & ORDER MANAGEMENT	50%	50%
5	ORDER PRODUCTS & DISPLAY BILL	50%	50%
6	PROGRAM DOCUMENTATION	50%	50%
7	REPORT FORMAT & REFERENCES	50%	50%
	SIGNATURE		

References

Object, A., 2019. *Atomic Object.* [Online]

Available at: https://atomicobject.com/resources/oo-programming/oo-development-process

[Accessed 2 March 2019].

Rouse, M., 2019. *WhatIs.* [Online]

Available at: https://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP

[Accessed 2 March 2019].

Stackoverflow.com. (2009) *What is Object Serialization?* [Online]

Available at: https://stackoverflow.com/questions/447898/what-is-object-serialization

[Accessed: 10/01/2018].

Martin, G. (2017) Random Number Generation [Online].

Available at: https://study.com/academy/lesson/java-generate-random-number-between-1-100.html

[Accessed: 27/02/2018].