INDIVIDUAL ASSIGNMENT

CT038-3-2 OODJ

Object Oriented Development with Java

Hand-Out Date: 1st December 2017

Hand-In Date: 5th March 2018

Lecturer: Miss Minnu Joseph

UC2F1711IT-ISS

Contents

1. Introduction

Developed Application is Retail Order Management System through which a customer can order multiple products online. This keeps the record of the customers and their order details in a database (text-file). It gives the functionality to the admin who can access through customer details and keep track, update and check the records of customer details whenever it is required. The application has been developed with java programing language which is an object-oriented programing language.

As of 2016, Java is one of the most popular programming languages in use particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them. Java is a general-purpose computer programming language that is concurrent, class-based, Object-Oriented, and specifically designed to have as few implementation dependencies as possible. Object-Oriented Programming (OOP) uses "objects" to model real-world objects. Object-Oriented Programming (OOP) consist of some important concepts namely Encapsulation, Polymorphism, Inheritance and Abstraction. These features are generally referred to as the OOPS concepts which has been used in this application as well.

2. Assumption

In this application, there are basically three users who can access the system. Firstly, admin, who has the privilege to add, delete, modify, view and search customers as well as products whose details are stored in the database (text-file). Admin can also view the customer order details by accessing through their customer ID. Another user are customers who can place order with particular order ID which is auto generated while placing an order. They can select multiple products from the product list and add to cart to view their selected products and their details (type and rates) and total amount. These order details are stored in database (text-file). The application provide couple of features for payment i.e. either pay cash on delivery or payment via customer’s A.T.M. card. Lastly, if there is a new customer who wants to view the products and place an order then he needs to register first by filling the register form to become a member and to have access in the system.

3. UML Diagrams

3.1. Use-Case Diagram

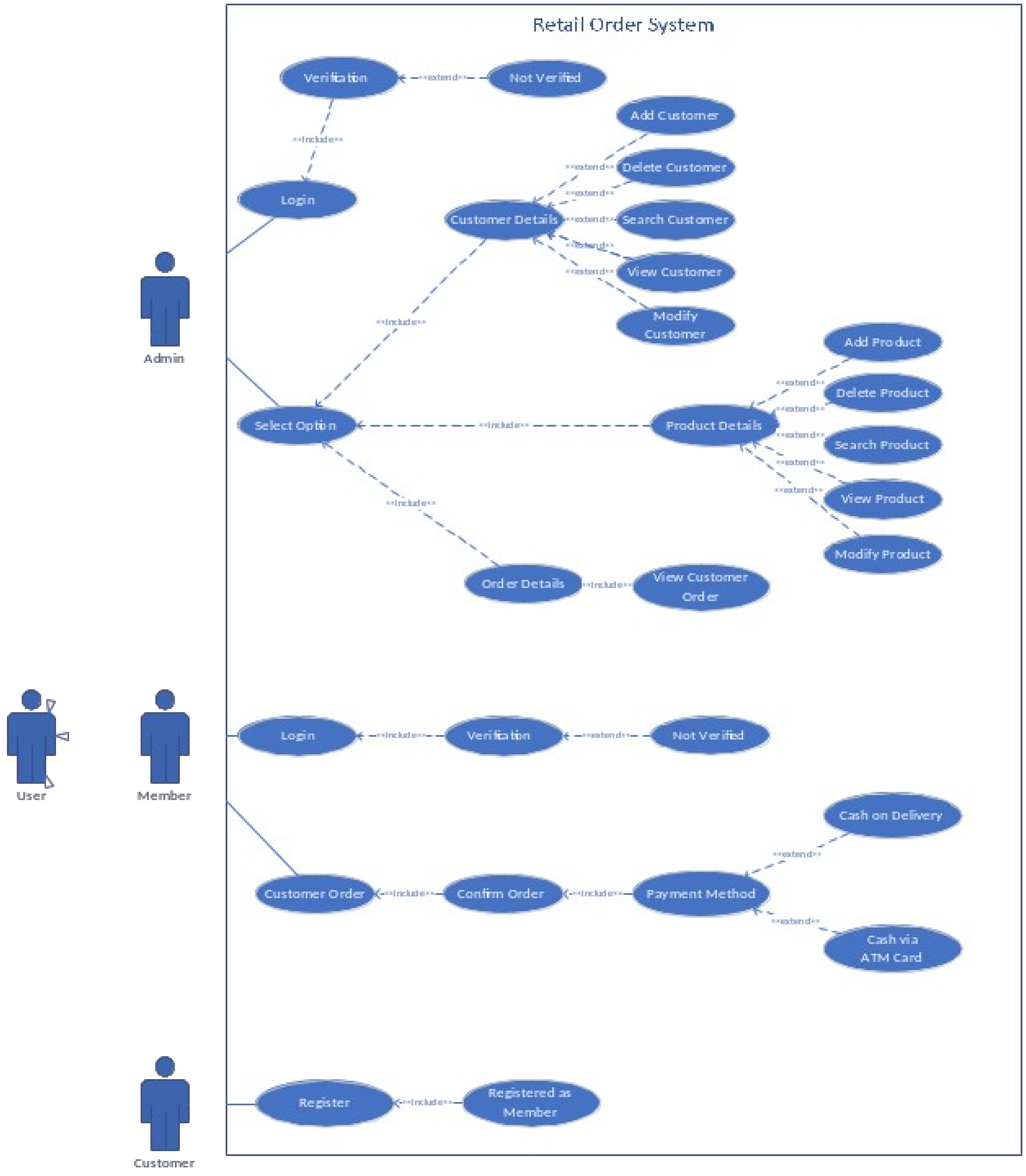
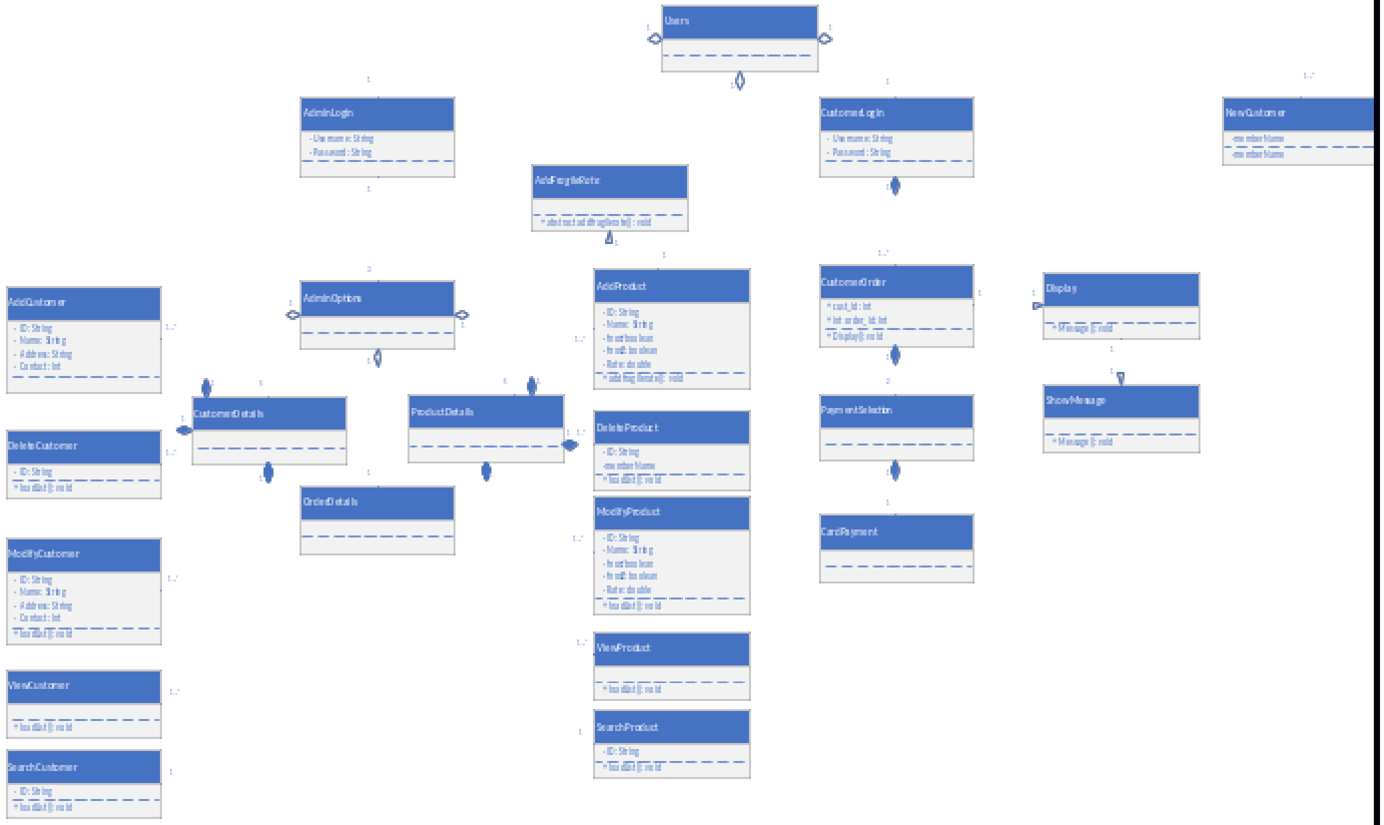


Fig 3.1.1: Use-Case Diagram

3.2. Class Diagram



3.3. Activity Diagram

3.3.1. Admin Login

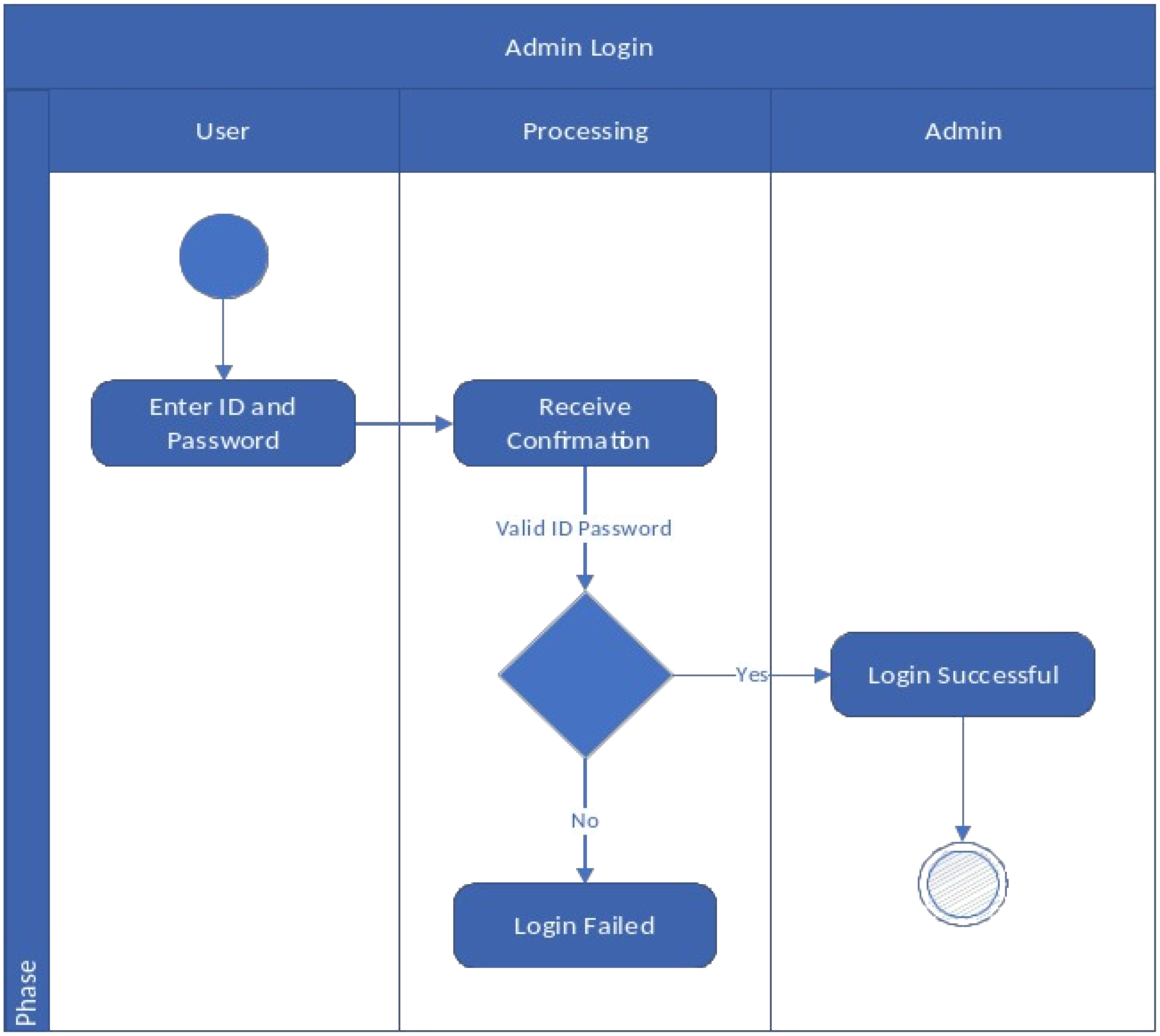


Fig 3.3.1 Admin Login

3.3.2. Add Customer

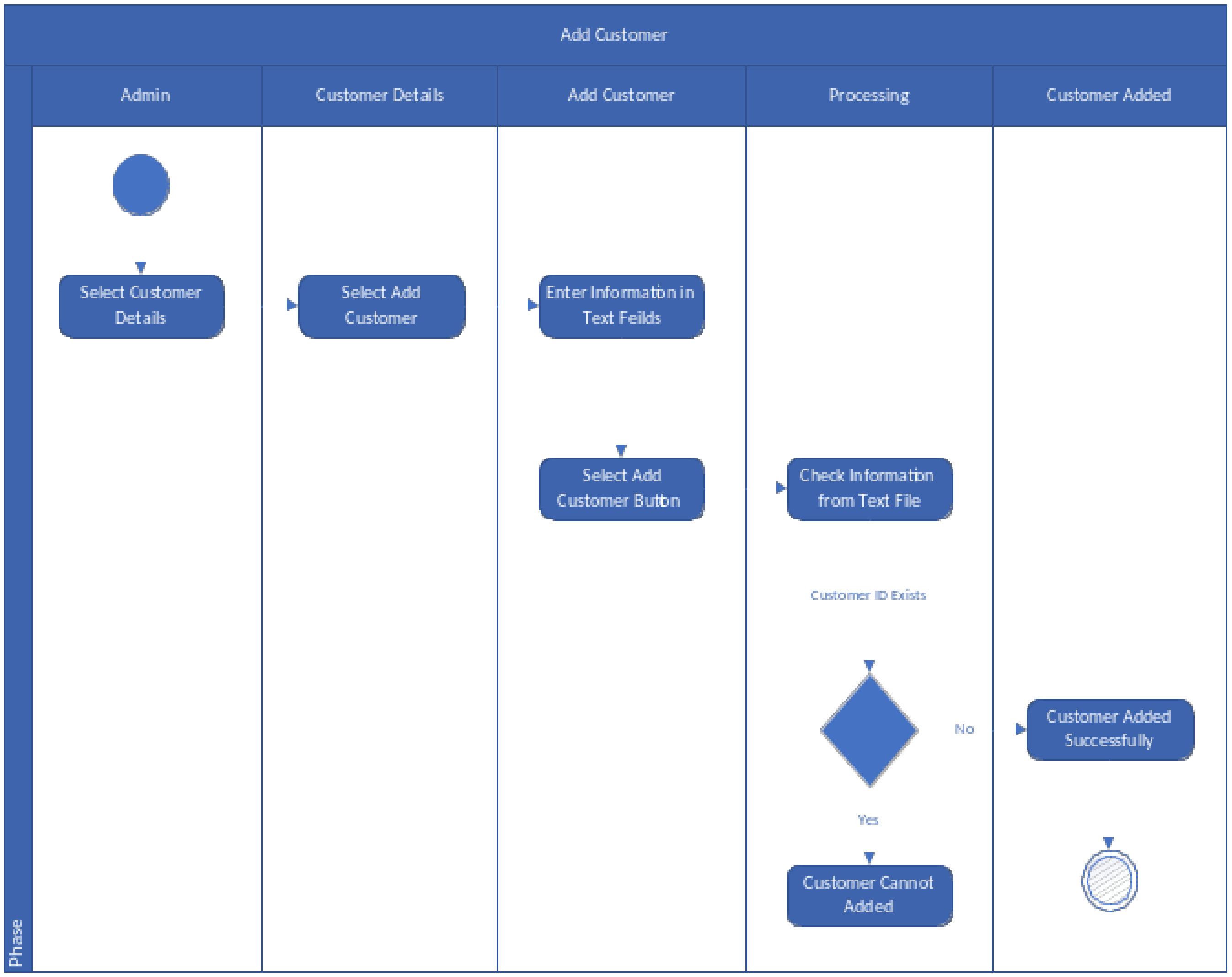


Fig 3.3.2 Add Customer

3.3.3. Delete Product

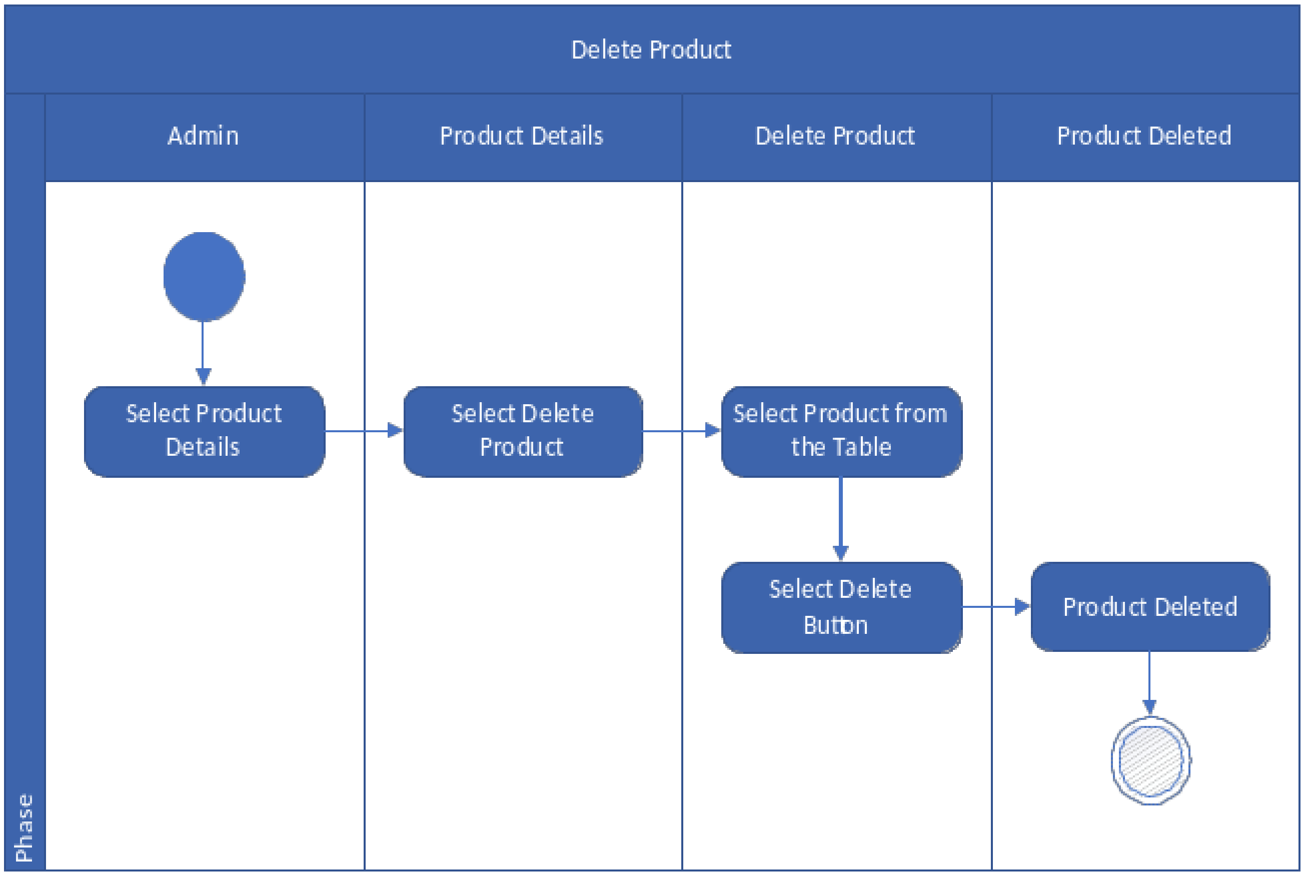


Fig 3.3.3 Delete Product

3.3.4. Modify Product

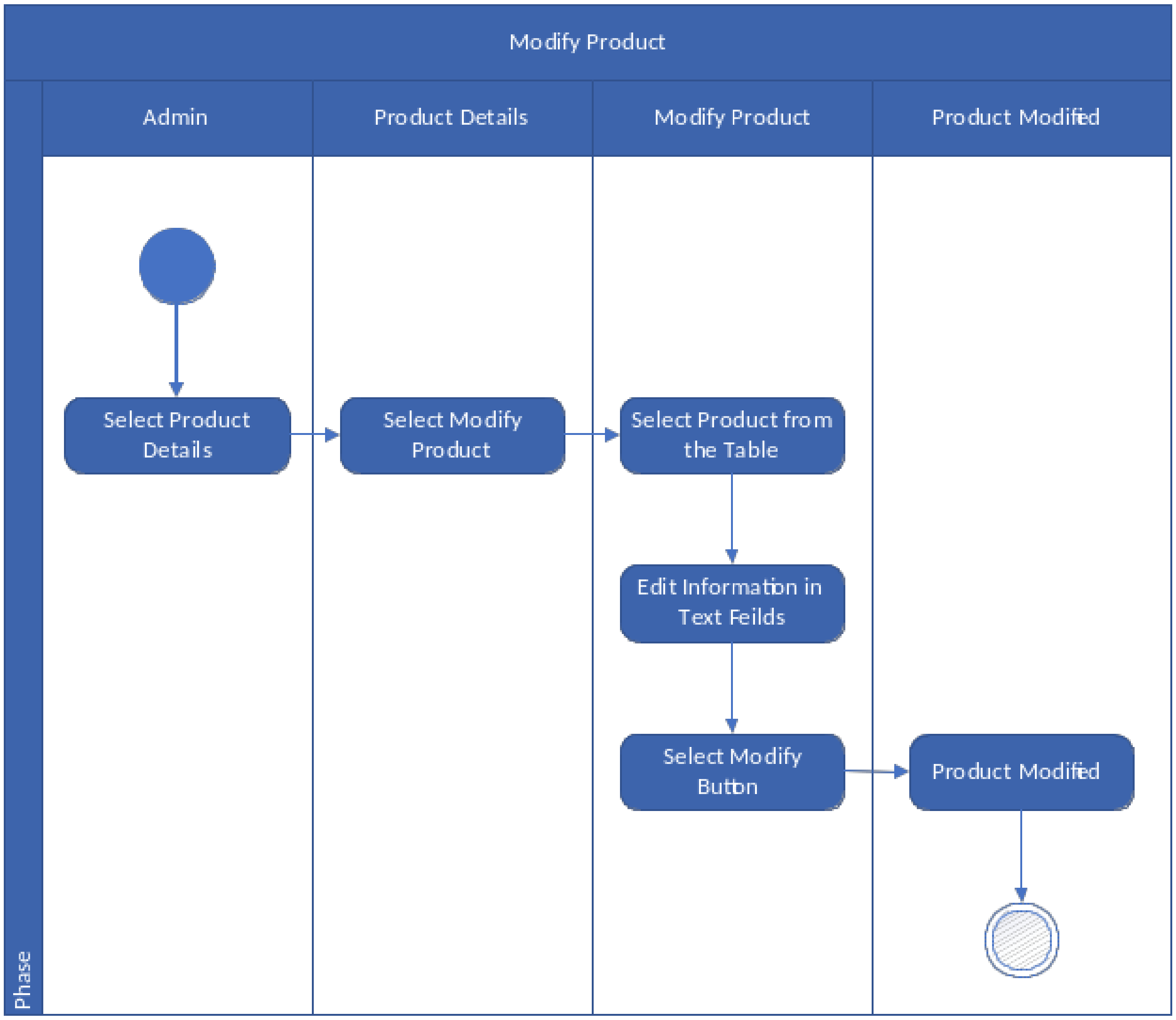


Fig 3.3.4 Modify Product

3.3.5. Customer Order

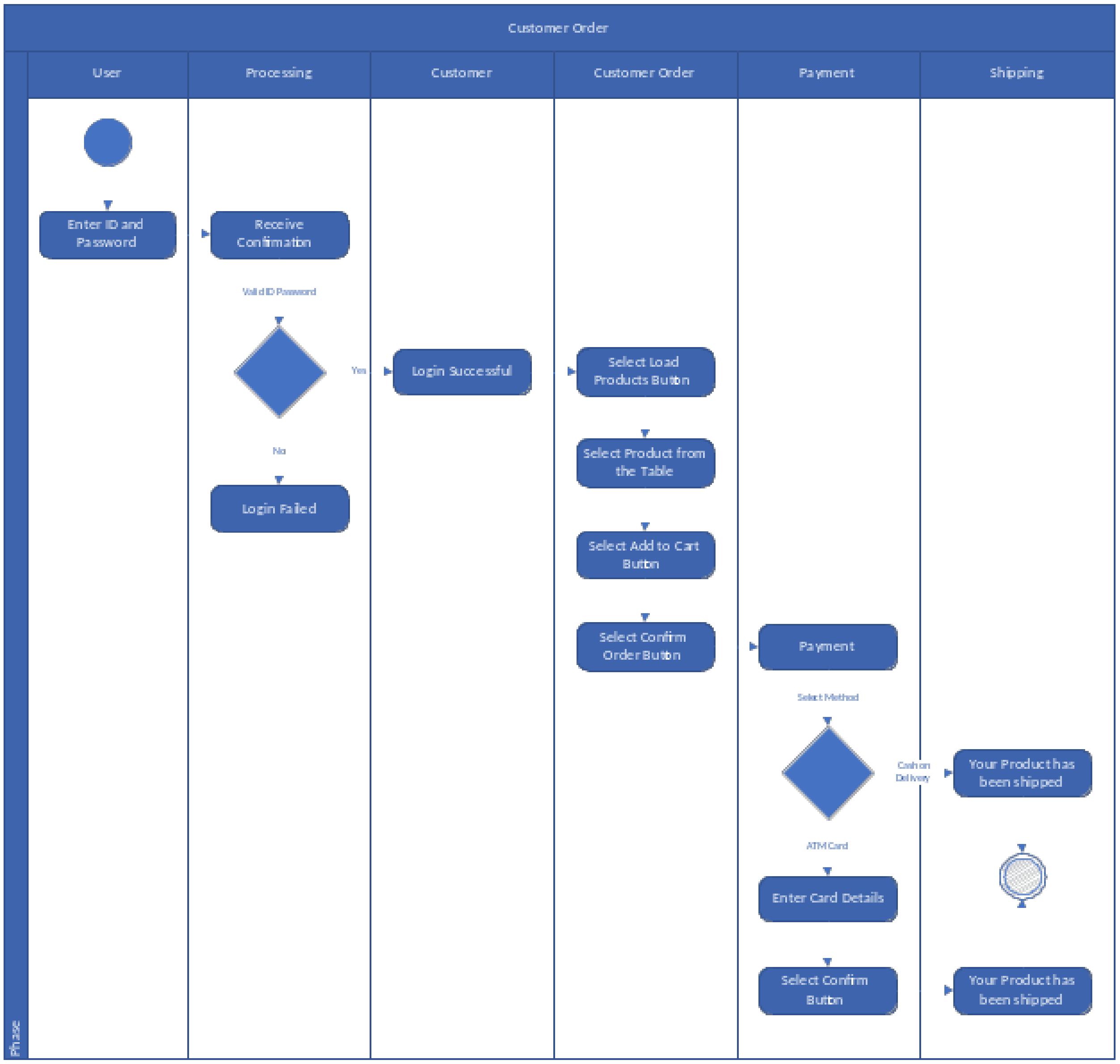
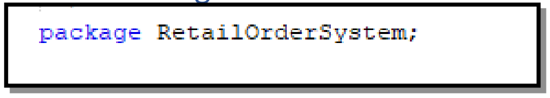


Fig 3.3.5 Customer Order

4. Sample of Code

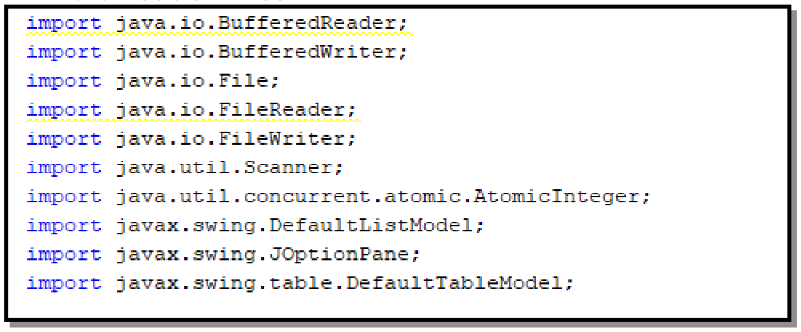
4.1. Package



```java
package RetailOrderSystem;
```

A java package is a group of similar types of classes, interfaces and sub-packages. Package in java can be categorized in two form, built-in package and user-defined package.

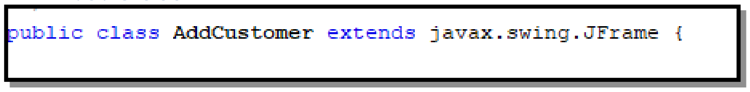
4.2. Header Files



```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.Scanner;
import java.util.concurrent.atomic.AtomicInteger;
import javax.swing.DefaultListModel;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

The java function is written in different classes which usually are imported from a library, to use each function you need to import the library to the class by using the “import” keyword following the class path in the top of the class after package name.

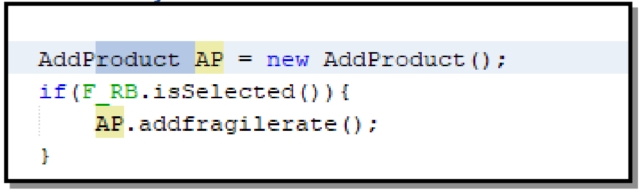
4.3. Class



```java
public class AddCustomer extends javax.swing.JFrame {
```

The java application is created by different classes the function should be inside the class. Java class objects exhibit the properties and behaviors defined by its class. A class can contain fields and methods to describe the behavior of an object.

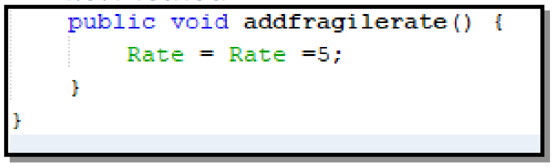
4.4. Object



```java
AddProduct AP = new AddProduct();
if(F_RB.isSelected()){
    AP.addfragilerate();
}
```

Any entity that has state and behavior is known as an object. Object refers to a instance of a class where the object can be a combination of variables, functions, and data structures.

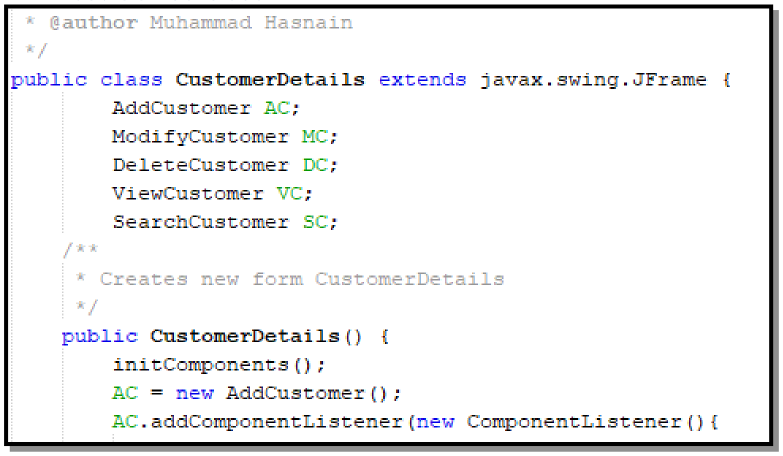
4.5. Method



```
    public void addfragilerate() {
        Rate = Rate =5;
    }
}
```

A method in Java is a block of statements that has a name and can be executed by calling (also called invoking) it from some other place in your program. Along with fields, methods are one of the two elements that are considered members of a class.

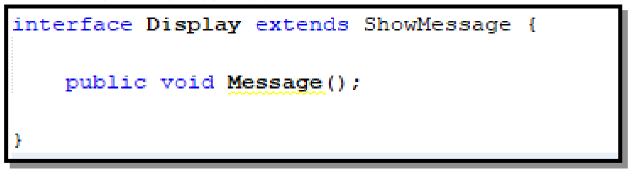
4.6. Constructor



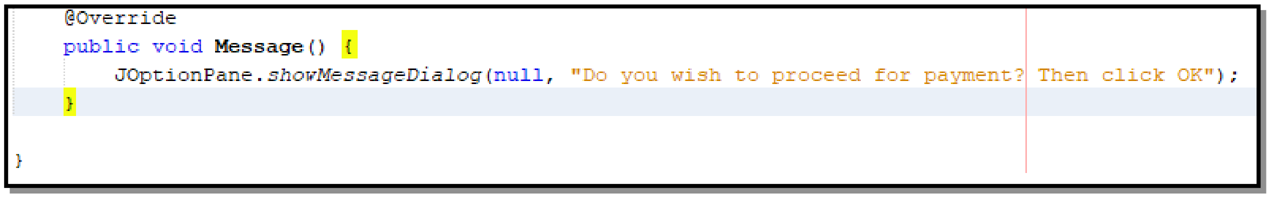
```
 * @author Muhammad Hasnain
 */
public class CustomerDetails extends javax.swing.JFrame {
        AddCustomer AC;
        ModifyCustomer MC;
        DeleteCustomer DC;
        ViewCustomer VC;
        SearchCustomer SC;
    /**
     * Creates new form CustomerDetails
     */
    public CustomerDetails() {
        initComponents();
        AC = new AddCustomer();
        AC.addComponentListener(new ComponentListener(){
```

A constructor in Java is a block of code like a method that's called when an instance of an object is created. Here are the key differences between a constructor and a method: A constructor doesn't have a return type. The name of the constructor must be the same as the name of the class.

4.7. Inheritance



```
interface Display extends ShowMessage {

    public void Message();

}
```

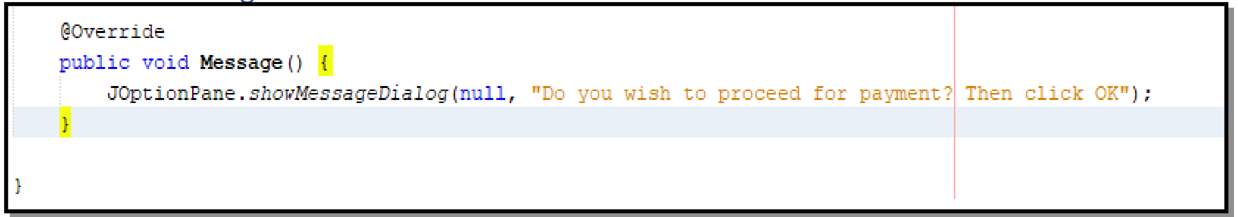


```
    @Override
    public void Message() {
        JOptionPane.showMessageDialog(null, "Do you wish to proceed for payment? Then click OK");
    }

}
```

When one object acquires all the properties and behaviors of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

4.8. Polymorphism

4.8.1. Over Riding



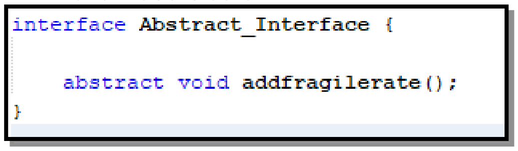
```
    @Override
    public void Message() {
        JOptionPane.showMessageDialog(null, "Do you wish to proceed for payment? Then click OK");
    }

}
```

In method overriding, the child class can use the OOP polymorphism concept to override a method of its parent class.

4.8.2. Over Loading

In method overloading, a single method may perform different functions depending on the context in which it’s called.

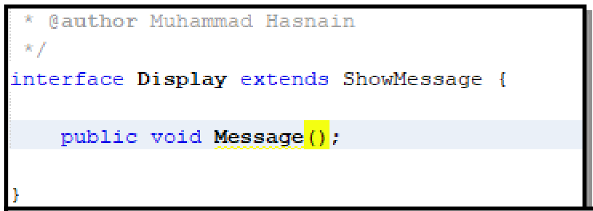
4.9. Abstraction



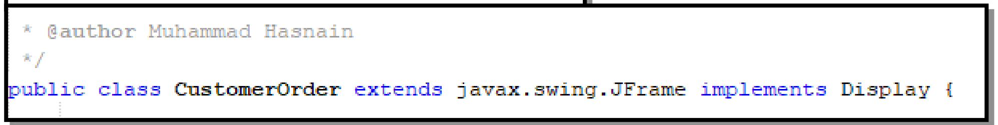
```
interface Abstract_Interface {

    abstract void addfragilerate();
}
```

Abstraction means simple things like objects, classes, and variables represent more complex underlying code and data. This is important because it avoid repeating same work multiple times.

4.10. Interface



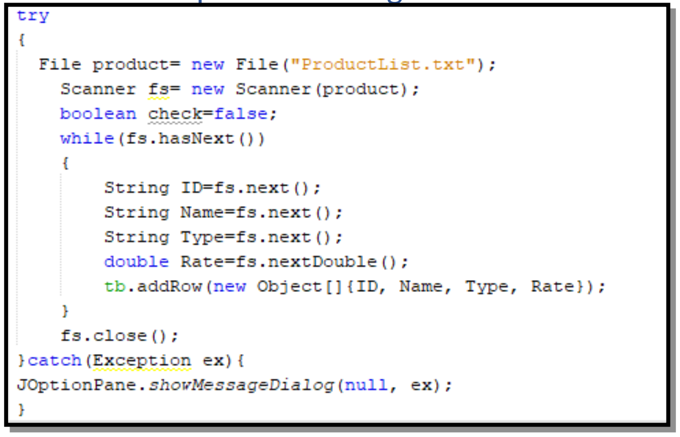
```
 * @author Muhammad Hasnain
 */
interface Display extends ShowMessage {

    public void Message();

}
```



```
 * @author Muhammad Hasnain
 */
public class CustomerOrder extends javax.swing.JFrame implements Display {
```

Encapsulation means that the internal representation of an object is generally hidden from view outside of the object's definition. This is the practice of keeping fields within a class private, then providing access to them via public methods.

4.11. Exception Handling



```
try
{
  File product= new File("ProductList.txt");
    Scanner fs= new Scanner(product);
    boolean check=false;
    while(fs.hasNext())
    {
        String ID=fs.next();
        String Name=fs.next();
        String Type=fs.next();
        double Rate=fs.nextDouble();
        tb.addRow(new Object[]{ID, Name, Type, Rate});
    }
    fs.close();
}catch(Exception ex){
JOptionPane.showMessageDialog(null, ex);
}
```

Java provides a powerful mechanism which allows you to handle the exceptional event where it occurred. You can either use the try-catch-finally approach to handle all kinds of exceptions.

(Chaudhari, 2015) (dummies, 2016) (Stackify, 2016) (javatpoint, 2017)

5. Application Output

5.1. Application Users

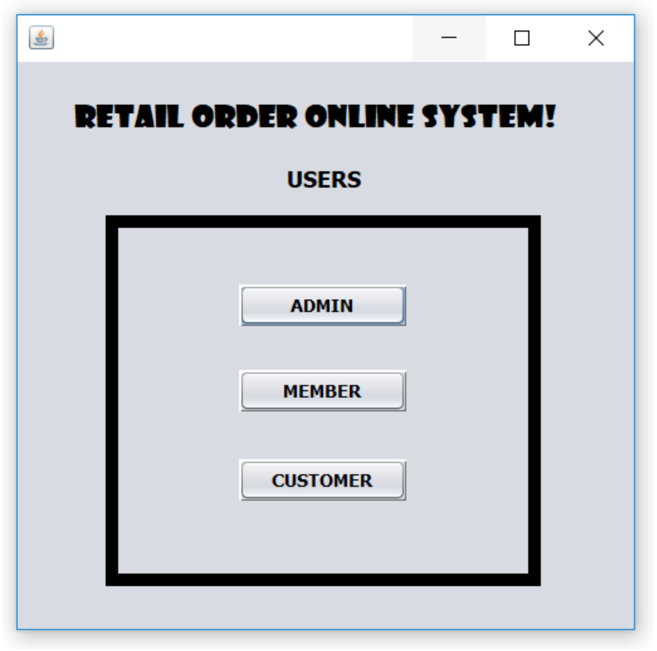
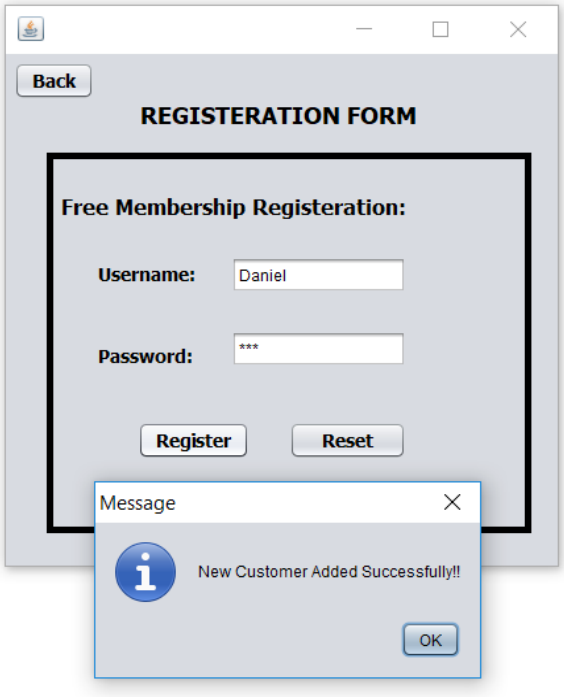


Fig 5.1.1: Application Users

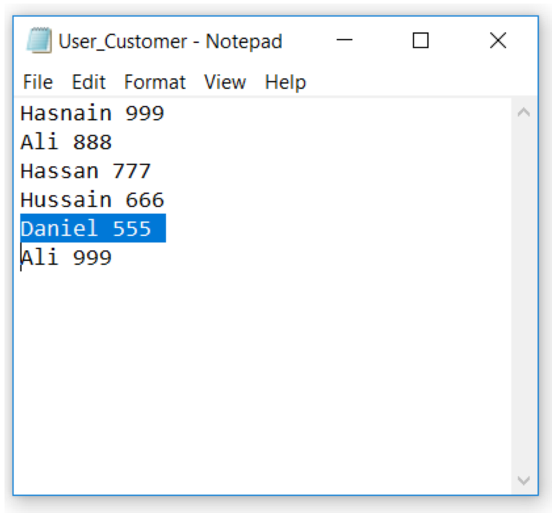
5.2. New Customer Register



Username: Daniel

Password: ***

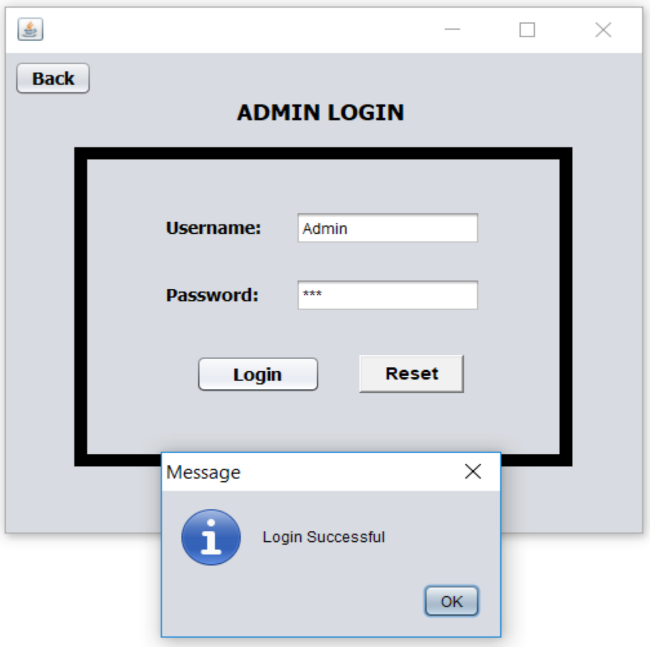
New Customer Added Successfully!!



```
Hasnain 999
Ali 888
Hassan 777
Hussain 666
Daniel 555
Ali 999
```

Fig 5.3.1 & 5.3.2: New Customer Register with Text-File

5.3. Login



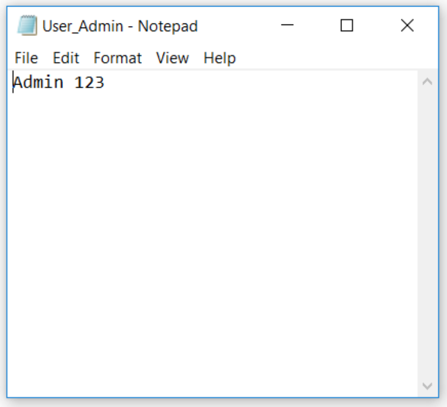
ADMIN LOGIN

Username: Admin

Password: ***

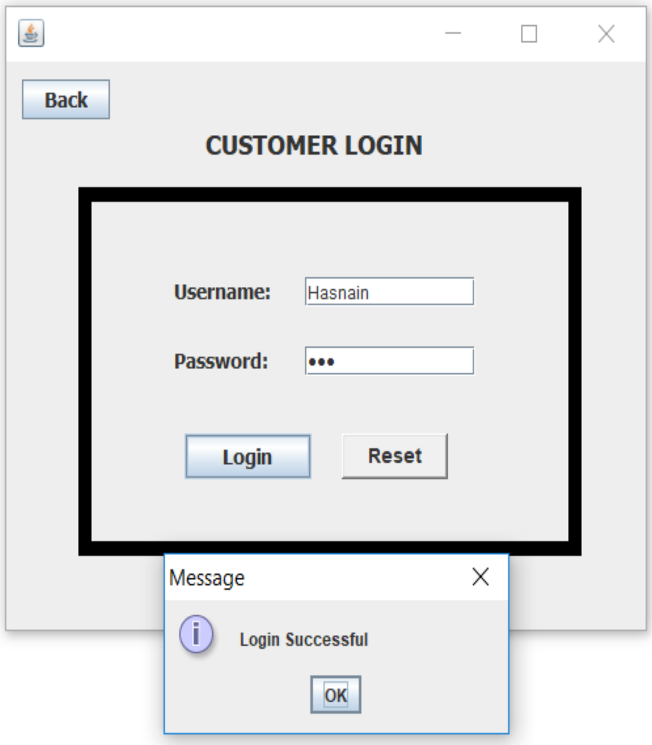
Message

Login Successful



```
Admin 123
```

Fig 5.2.1 & 5.2.2: Admin Login with Text-File



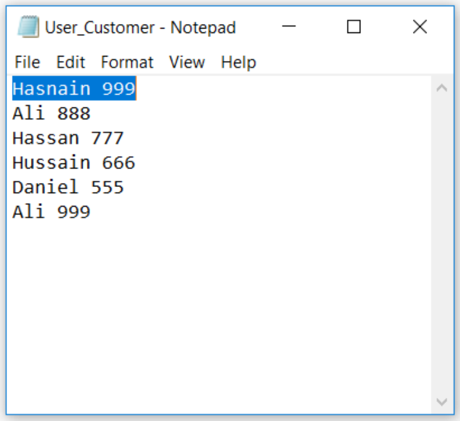
CUSTOMER LOGIN

Username: Hasnain

Password: •••

Message

Login Successful



```
Hasnain 999
Ali 888
Hassan 777
Hussain 666
Daniel 555
Ali 999
```

Fig 5.2.3 & 5.2.4: Customer Login with Text-File

5.4.Admin Options

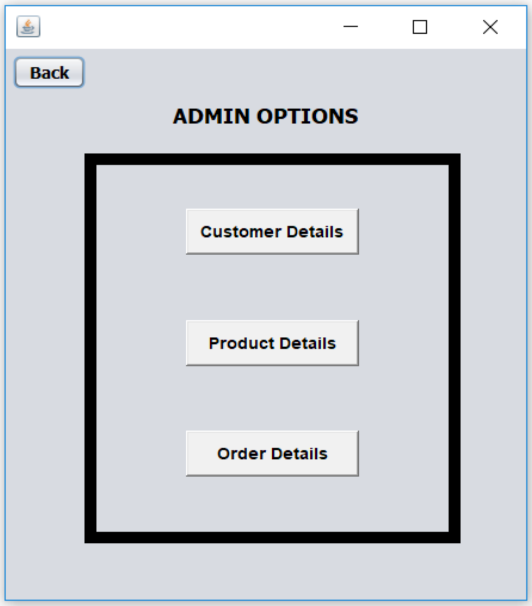


Fig 5.4.1: Admin Options

5.5. Customer Details

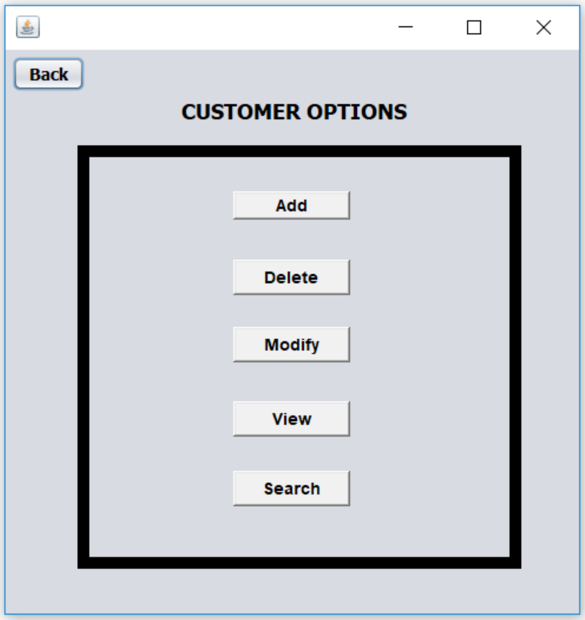
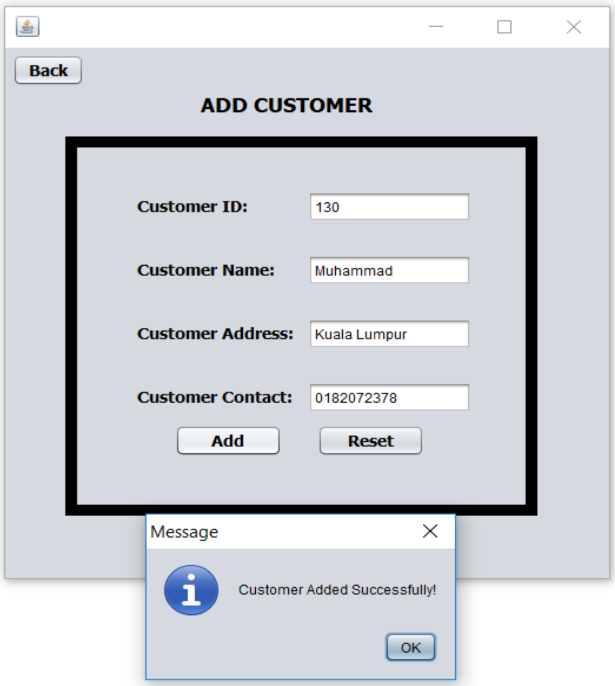


Fig 5.5.1: Customer Details

5.6. Customer Options


Back

ADD CUSTOMER

Customer ID: 130

Customer Name: Muhammad

Customer Address: Kuala Lumpur

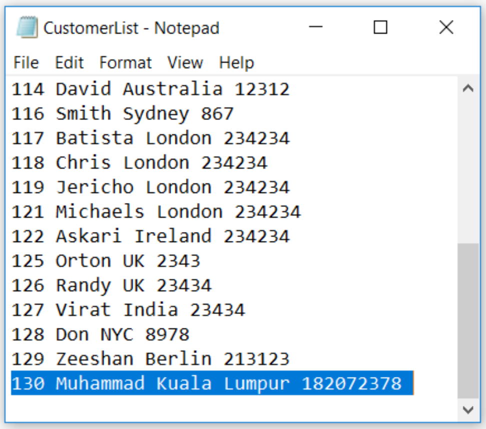
Customer Contact: 0182072378

Add Reset

Message

Customer Added Successfully!

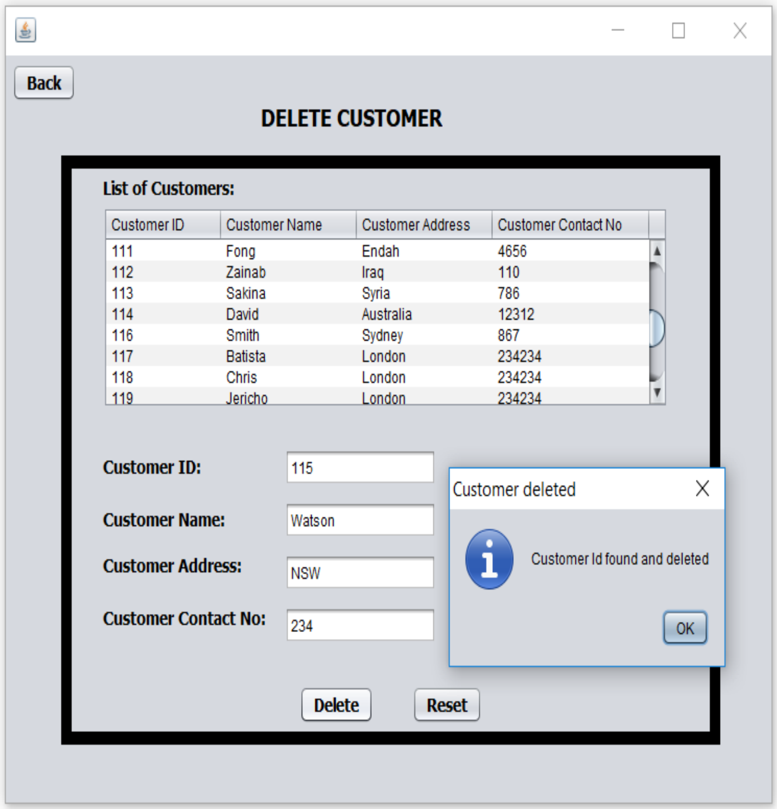
OK


CustomerList - Notepad

File Edit Format View Help

```
114 David Australia 12312
116 Smith Sydney 867
117 Batista London 234234
118 Chris London 234234
119 Jericho London 234234
121 Michaels London 234234
122 Askari Ireland 234234
125 Orton UK 2343
126 Randy UK 23434
127 Virat India 23434
128 Don NYC 8978
129 Zeeshan Berlin 213123
130 Muhammad Kuala Lumpur 182072378
```

Fig 5.6.1 & 5.6.2: Add Customer with Text-File


Back

DELETE CUSTOMER

List of Customers:

Customer ID	Customer Name	Customer Address	Customer Contact No
111	Fong	Endah	4656
112	Zainab	Iraq	110
113	Sakina	Syria	786
114	David	Australia	12312
116	Smith	Sydney	867
117	Batista	London	234234
118	Chris	London	234234
119	Jericho	London	234234

Customer ID: 115

Customer Name: Watson

Customer Address: NSW

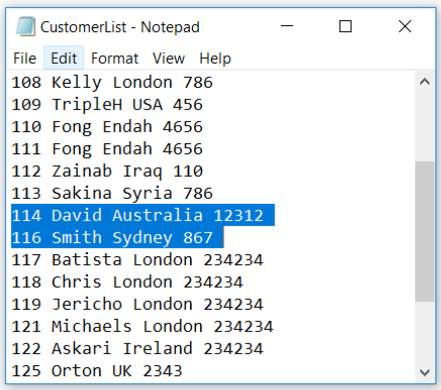
Customer Contact No: 234

Delete Reset

Customer deleted

Customer Id found and deleted

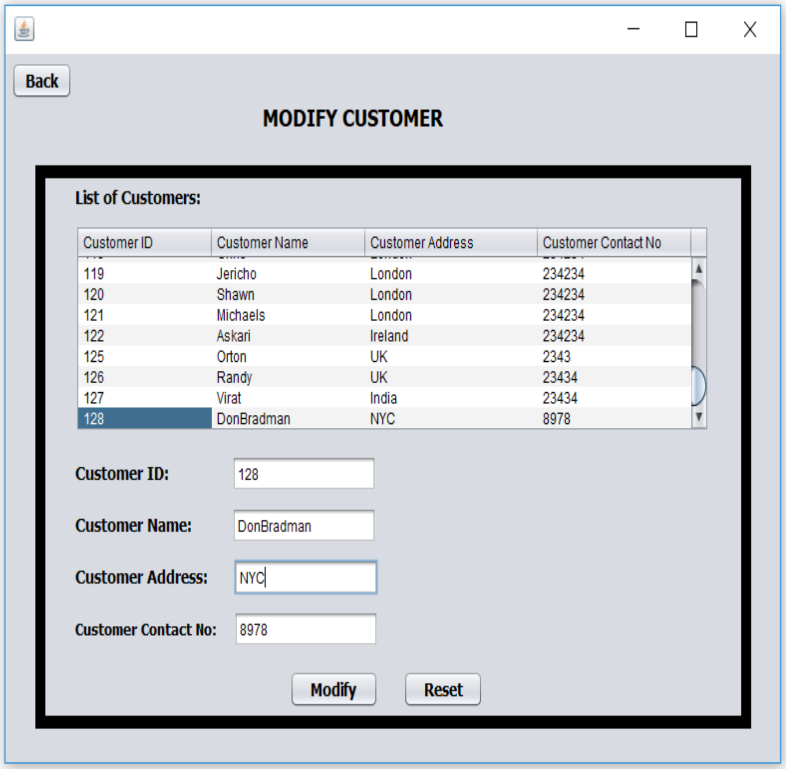
OK


CustomerList - Notepad

File Edit Format View Help

```
108 Kelly London 786
109 TripleH USA 456
110 Fong Endah 4656
111 Fong Endah 4656
112 Zainab Iraq 110
113 Sakina Syria 786
114 David Australia 12312
116 Smith Sydney 867
117 Batista London 234234
118 Chris London 234234
119 Jericho London 234234
121 Michaels London 234234
122 Askari Ireland 234234
125 Orton UK 2343
```

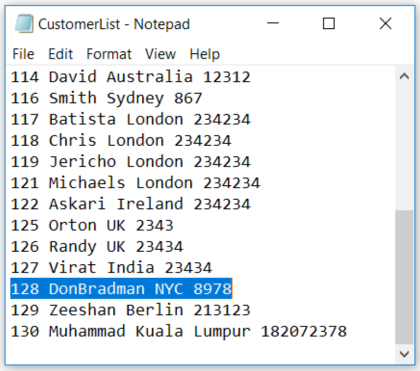
Fig 5.6.3 & 5.6.4: Delete Customer with Text-File



List of Customers:

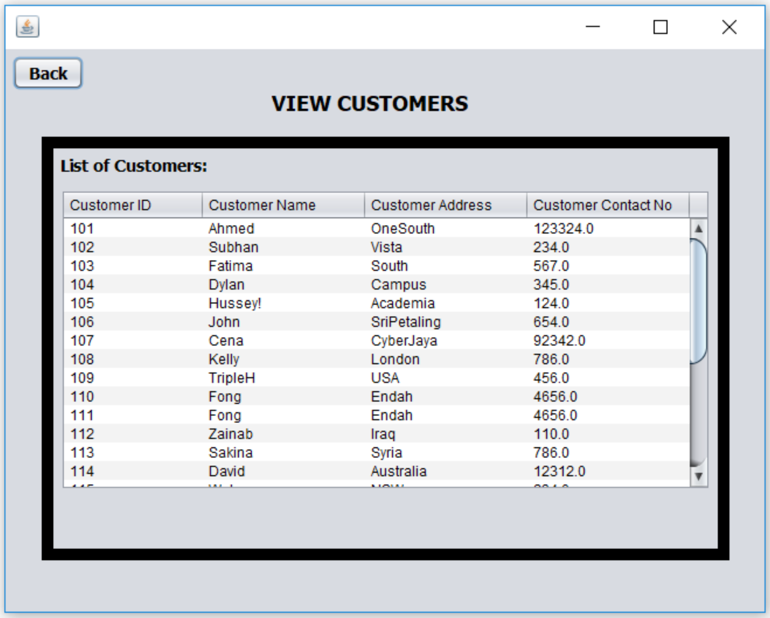
Customer ID	Customer Name	Customer Address	Customer Contact No
119	Jericho	London	234234
120	Shawn	London	234234
121	Michaels	London	234234
122	Askari	Ireland	234234
125	Orton	UK	2343
126	Randy	UK	23434
127	Virat	India	23434
128	DonBradman	NYC	8978

Customer ID:	128
Customer Name:	DonBradman
Customer Address:	NYC
Customer Contact No:	8978



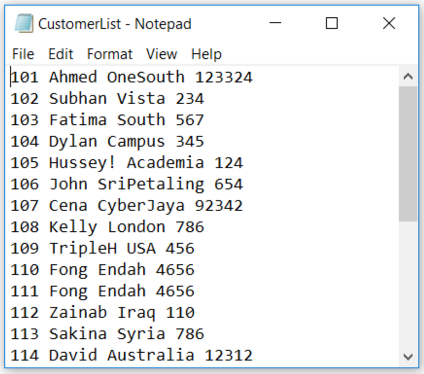
```
114 David Australia 12312
116 Smith Sydney 867
117 Batista London 234234
118 Chris London 234234
119 Jericho London 234234
121 Michaels London 234234
122 Askari Ireland 234234
125 Orton UK 2343
126 Randy UK 23434
127 Virat India 23434
128 DonBradman NYC 8978
129 Zeeshan Berlin 213123
130 Muhammad Kuala Lumpur 182072378
```

Fig 5.6.5 & 5.6.6: Modify Customer with Text-File



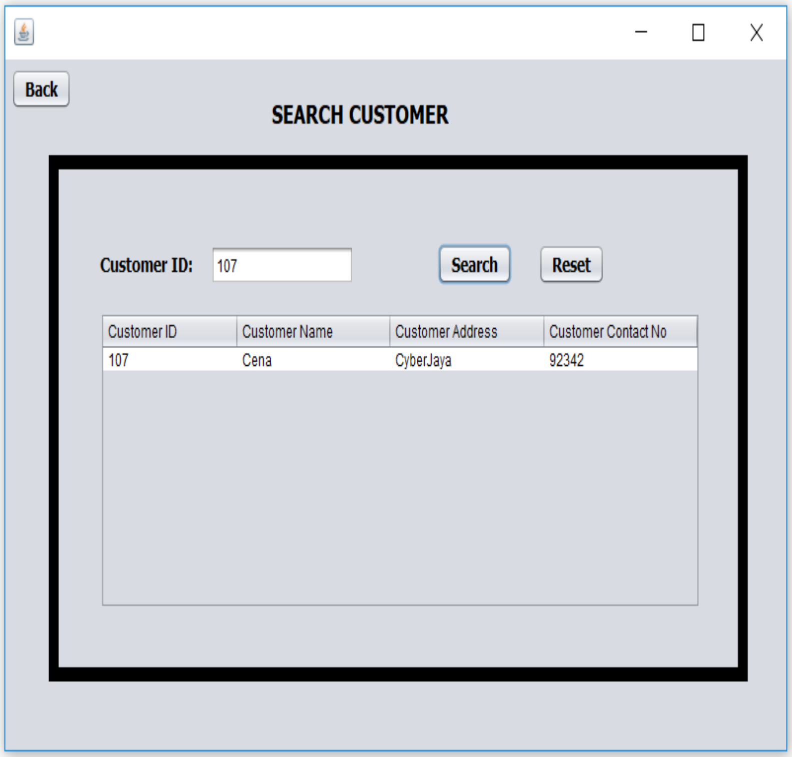
List of Customers:

Customer ID	Customer Name	Customer Address	Customer Contact No
101	Ahmed	OneSouth	123324.0
102	Subhan	Vista	234.0
103	Fatima	South	567.0
104	Dylan	Campus	345.0
105	Hussey!	Academia	124.0
106	John	SriPetaling	654.0
107	Cena	CyberJaya	92342.0
108	Kelly	London	786.0
109	TripleH	USA	456.0
110	Fong	Endah	4656.0
111	Fong	Endah	4656.0
112	Zainab	Iraq	110.0
113	Sakina	Syria	786.0
114	David	Australia	12312.0



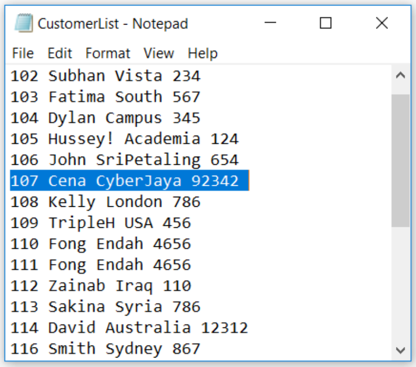
```
101 Ahmed OneSouth 123324
102 Subhan Vista 234
103 Fatima South 567
104 Dylan Campus 345
105 Hussey! Academia 124
106 John SriPetaling 654
107 Cena CyberJaya 92342
108 Kelly London 786
109 TripleH USA 456
110 Fong Endah 4656
111 Fong Endah 4656
112 Zainab Iraq 110
113 Sakina Syria 786
114 David Australia 12312
```

Fig 5.6.7 & 5.6.8: View Customers with Text-File



Customer ID: 107

Customer ID	Customer Name	Customer Address	Customer Contact No
107	Cena	CyberJaya	92342



```
102 Subhan Vista 234
103 Fatima South 567
104 Dylan Campus 345
105 Hussey! Academia 124
106 John SriPetaling 654
107 Cena CyberJaya 92342
108 Kelly London 786
109 TripleH USA 456
110 Fong Endah 4656
111 Fong Endah 4656
112 Zainab Iraq 110
113 Sakina Syria 786
114 David Australia 12312
116 Smith Sydney 867
```

Fig 5.6.9 & 5.6.10: Search Customer with Text-File

5.7. Product Details

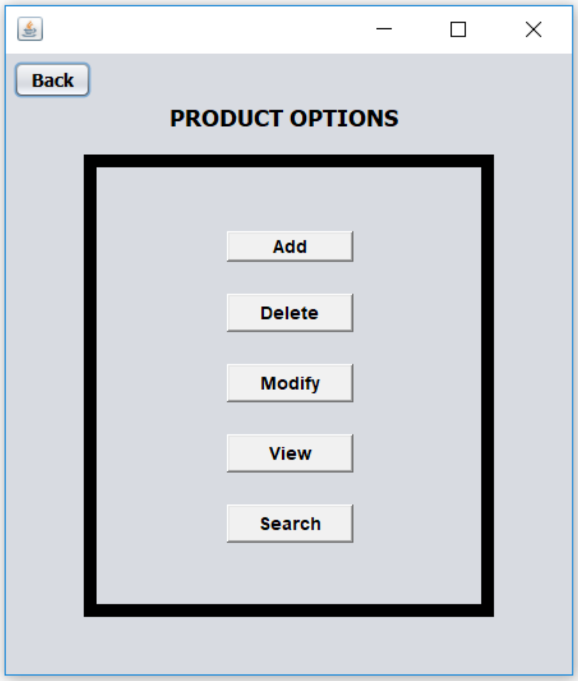
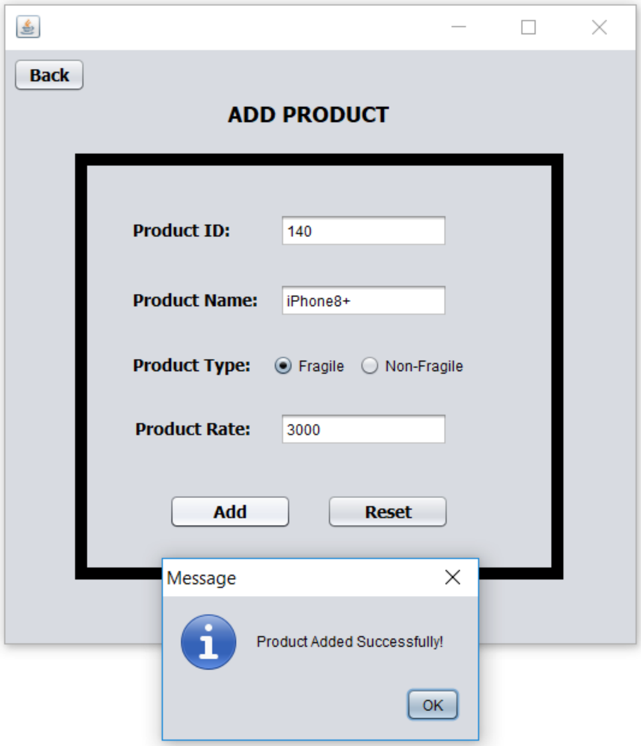


Fig 5.7.1: Product Details

5.8. Product Options



Back

ADD PRODUCT

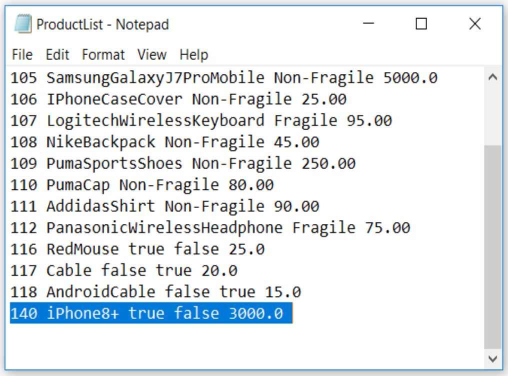
Product ID:	140
Product Name:	iPhone8+
Product Type:	[✓ Fragile] Non-Fragile
Product Rate:	3000

Add Reset

Message

Product Added Successfully!

OK

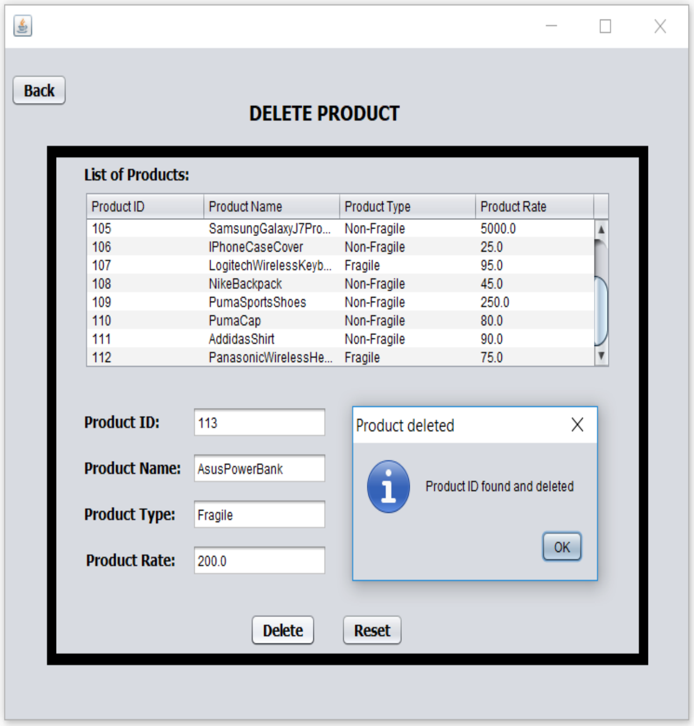


ProductList - Notepad

File Edit Format View Help

```
105 SamsungGalaxyJ7ProMobile Non-Fragile 5000.0
106 IPhoneCaseCover Non-Fragile 25.00
107 LogitechWirelessKeyboard Fragile 95.00
108 NikeBackpack Non-Fragile 45.00
109 PumaSportsShoes Non-Fragile 250.00
110 PumaCap Non-Fragile 80.00
111 AddidasShirt Non-Fragile 90.00
112 PanasonicWirelessHeadphone Fragile 75.00
116 RedMouse true false 25.0
117 Cable false true 20.0
118 AndroidCable false true 15.0
140 iPhone8+ true false 3000.0
```

Fig 5.8.1 & 5.8.2: Add Product with Text-File



Back

DELETE PRODUCT

List of Products:

Product ID	Product Name	Product Type	Product Rate
105	SamsungGalaxyJ7Pro...	Non-Fragile	5000.0
106	IPhoneCaseCover	Non-Fragile	25.0
107	LogitechWirelessKeyb...	Fragile	95.0
108	NikeBackpack	Non-Fragile	45.0
109	PumaSportsShoes	Non-Fragile	250.0
110	PumaCap	Non-Fragile	80.0
111	AddidasShirt	Non-Fragile	90.0
112	PanasonicWirelessHe...	Fragile	75.0

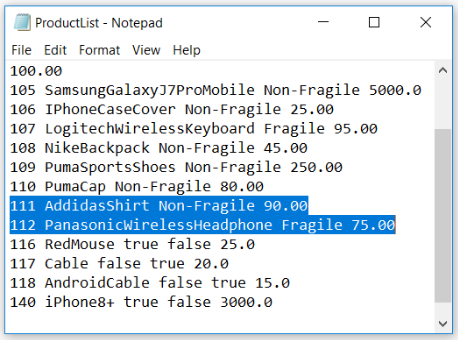
Product ID:	113
Product Name:	AsusPowerBank
Product Type:	Fragile
Product Rate:	200.0

Delete Reset

Product deleted

Product ID found and deleted

OK

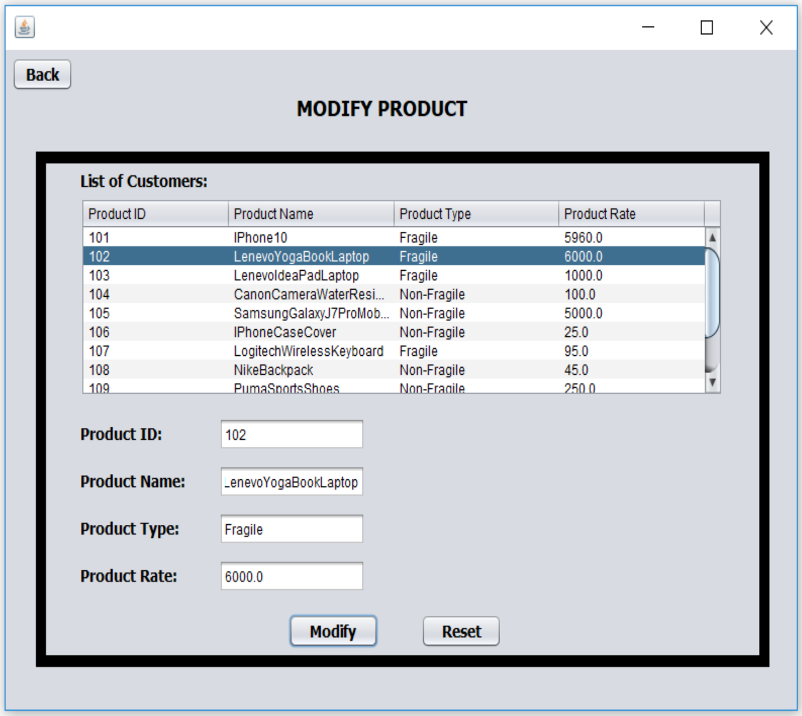


ProductList - Notepad

File Edit Format View Help

```
100.00
105 SamsungGalaxyJ7ProMobile Non-Fragile 5000.0
106 IPhoneCaseCover Non-Fragile 25.00
107 LogitechWirelessKeyboard Fragile 95.00
108 NikeBackpack Non-Fragile 45.00
109 PumaSportsShoes Non-Fragile 250.00
110 PumaCap Non-Fragile 80.00
111 AddidasShirt Non-Fragile 90.00
112 PanasonicWirelessHeadphone Fragile 75.00
116 RedMouse true false 25.0
117 Cable false true 20.0
118 AndroidCable false true 15.0
140 iPhone8+ true false 3000.0
```

Fig 5.8.3 & 5.8.4: Delete Product with Text-File



List of Customers:

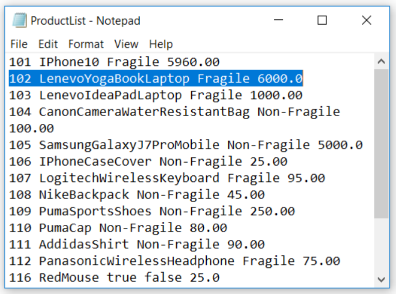
Product ID	Product Name	Product Type	Product Rate
101	IPhone10	Fragile	5960.0
102	LenevoYogaBookLaptop	Fragile	6000.0
103	LenevoIdeaPadLaptop	Fragile	1000.0
104	CanonCameraWaterResi...	Non-Fragile	100.0
105	SamsungGalaxyJ7ProMob...	Non-Fragile	5000.0
106	IPhoneCaseCover	Non-Fragile	25.0
107	LogitechWirelessKeyboard	Fragile	95.0
108	NikeBackpack	Non-Fragile	45.0
109	PumaSportsShoes	Non-Fragile	250.0

Product ID: 102

Product Name: _enevoYogaBookLaptop

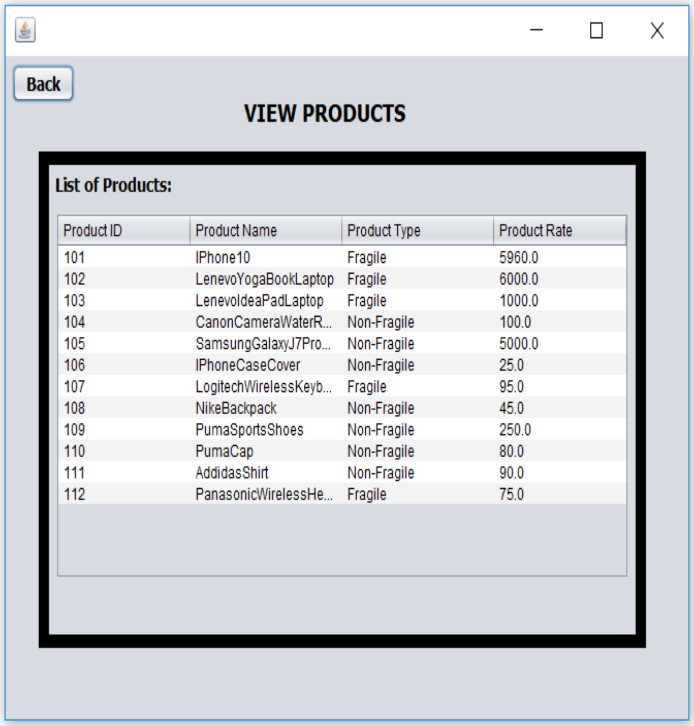
Product Type: Fragile

Product Rate: 6000.0



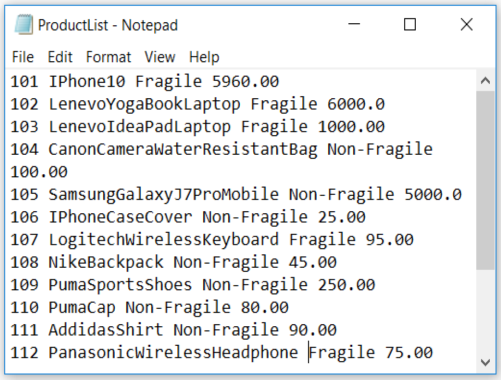
```
101 IPhone10 Fragile 5960.00
102 LenevoYogaBookLaptop Fragile 6000.0
103 LenevoIdeaPadLaptop Fragile 1000.00
104 CanonCameraWaterResistantBag Non-Fragile
100.00
105 SamsungGalaxyJ7ProMobile Non-Fragile 5000.0
106 IPhoneCaseCover Non-Fragile 25.00
107 LogitechWirelessKeyboard Fragile 95.00
108 NikeBackpack Non-Fragile 45.00
109 PumaSportsShoes Non-Fragile 250.00
110 PumaCap Non-Fragile 80.00
111 AddidasShirt Non-Fragile 90.00
112 PanasonicWirelessHeadphone Fragile 75.00
116 RedMouse true false 25.0
```

Fig 5.8.5 & 5.8.6: Modify Product with Text-File



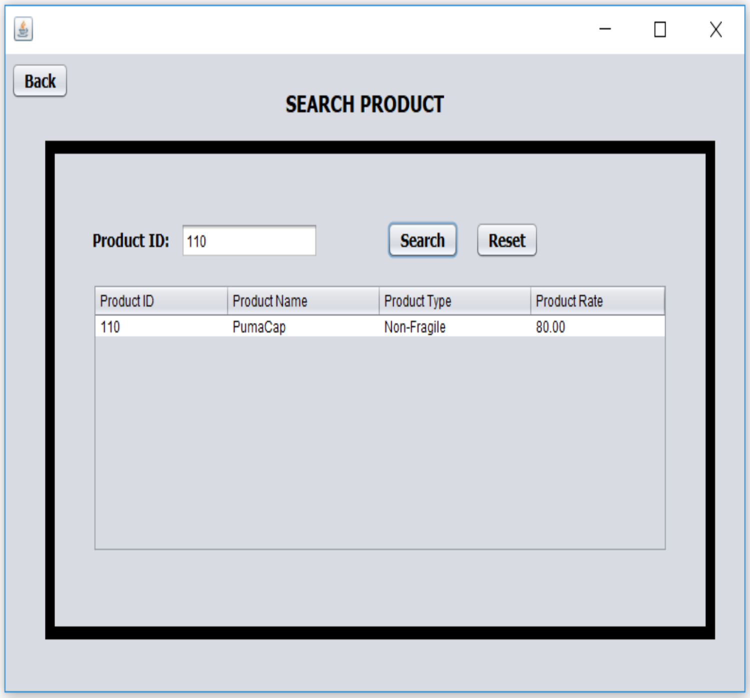
List of Products:

Product ID	Product Name	Product Type	Product Rate
101	IPhone10	Fragile	5960.0
102	LenevoYogaBookLaptop	Fragile	6000.0
103	LenevoIdeaPadLaptop	Fragile	1000.0
104	CanonCameraWaterR...	Non-Fragile	100.0
105	SamsungGalaxyJ7Pro...	Non-Fragile	5000.0
106	IPhoneCaseCover	Non-Fragile	25.0
107	LogitechWirelessKeyb...	Fragile	95.0
108	NikeBackpack	Non-Fragile	45.0
109	PumaSportsShoes	Non-Fragile	250.0
110	PumaCap	Non-Fragile	80.0
111	AddidasShirt	Non-Fragile	90.0
112	PanasonicWirelessHe...	Fragile	75.0



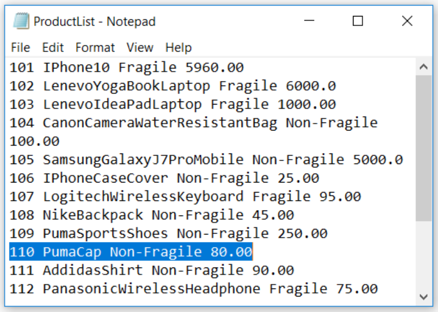
```
101 IPhone10 Fragile 5960.00
102 LenevoYogaBookLaptop Fragile 6000.0
103 LenevoIdeaPadLaptop Fragile 1000.00
104 CanonCameraWaterResistantBag Non-Fragile
100.00
105 SamsungGalaxyJ7ProMobile Non-Fragile 5000.0
106 IPhoneCaseCover Non-Fragile 25.00
107 LogitechWirelessKeyboard Fragile 95.00
108 NikeBackpack Non-Fragile 45.00
109 PumaSportsShoes Non-Fragile 250.00
110 PumaCap Non-Fragile 80.00
111 AddidasShirt Non-Fragile 90.00
112 PanasonicWirelessHeadphone Fragile 75.00
```

Fig 5.8.7 & 5.8.8: View Products with Text-File



Product ID: 110

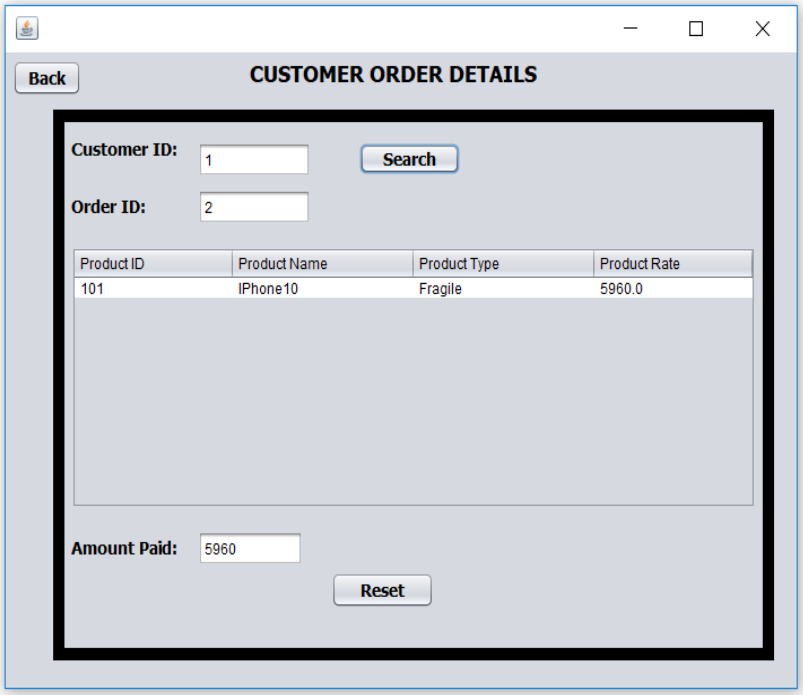
Product ID	Product Name	Product Type	Product Rate
110	PumaCap	Non-Fragile	80.00



```
101 IPhone10 Fragile 5960.00
102 LenevoYogaBookLaptop Fragile 6000.0
103 LenevoIdeaPadLaptop Fragile 1000.00
104 CanonCameraWaterResistantBag Non-Fragile
100.00
105 SamsungGalaxyJ7ProMobile Non-Fragile 5000.0
106 IPhoneCaseCover Non-Fragile 25.00
107 LogitechWirelessKeyboard Fragile 95.00
108 NikeBackpack Non-Fragile 45.00
109 PumaSportsShoes Non-Fragile 250.00
110 PumaCap Non-Fragile 80.00
111 AddidasShirt Non-Fragile 90.00
112 PanasonicWirelessHeadphone Fragile 75.00
```

Fig 5.8.9 & 5.8.10: Search Product with Text-File

5.9. Order Details

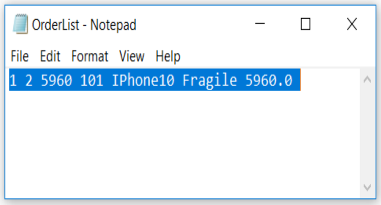


Customer ID: 1

Order ID: 2

Product ID	Product Name	Product Type	Product Rate
101	IPhone10	Fragile	5960.0

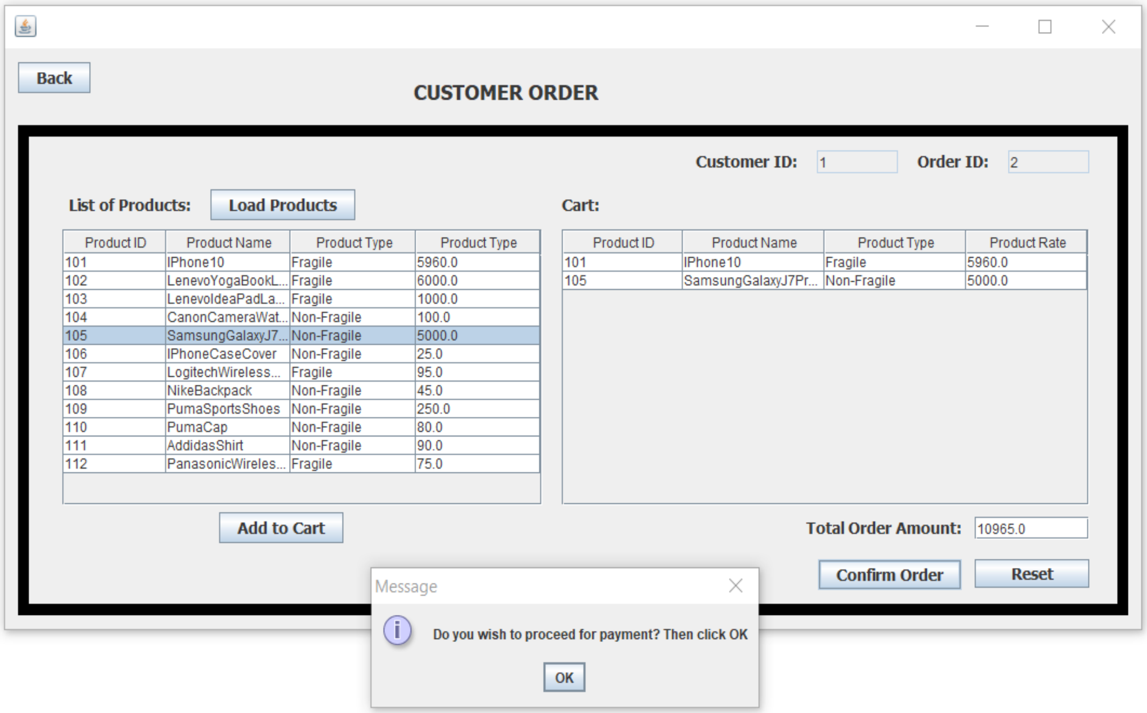
Amount Paid: 5960



```
1 2 5960 101 IPhone10 Fragile 5960.0
```

Fig 5.9.1 & 5.9.2: Customer Order Details with Text-File

5.10. Customer Order



CUSTOMER ORDER

Customer ID: 1 Order ID: 2

List of Products:

Product ID	Product Name	Product Type	Product Type
101	IPhone10	Fragile	5960.0
102	LenevoYogaBookL...	Fragile	6000.0
103	LenevoldeaPadLa...	Fragile	1000.0
104	CanonCameraWat...	Non-Fragile	100.0
105	SamsungGalaxyJ7...	Non-Fragile	5000.0
106	IPhoneCaseCover	Non-Fragile	25.0
107	LogitechWireless...	Fragile	95.0
108	NikeBackpack	Non-Fragile	45.0
109	PumaSportsShoes	Non-Fragile	250.0
110	PumaCap	Non-Fragile	80.0
111	AddidasShirt	Non-Fragile	90.0
112	PanasonicWireles...	Fragile	75.0

Cart:

Product ID	Product Name	Product Type	Product Rate
101	IPhone10	Fragile	5960.0
105	SamsungGalaxyJ7Pr...	Non-Fragile	5000.0

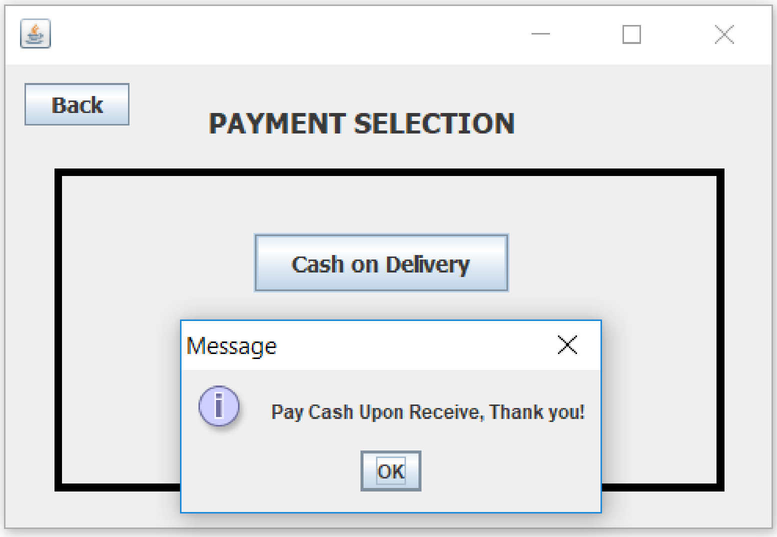
Total Order Amount: 10965.0

Message

Do you wish to proceed for payment? Then click OK

Fig 5.10.1: Customer Order

5.11. Payment Selection

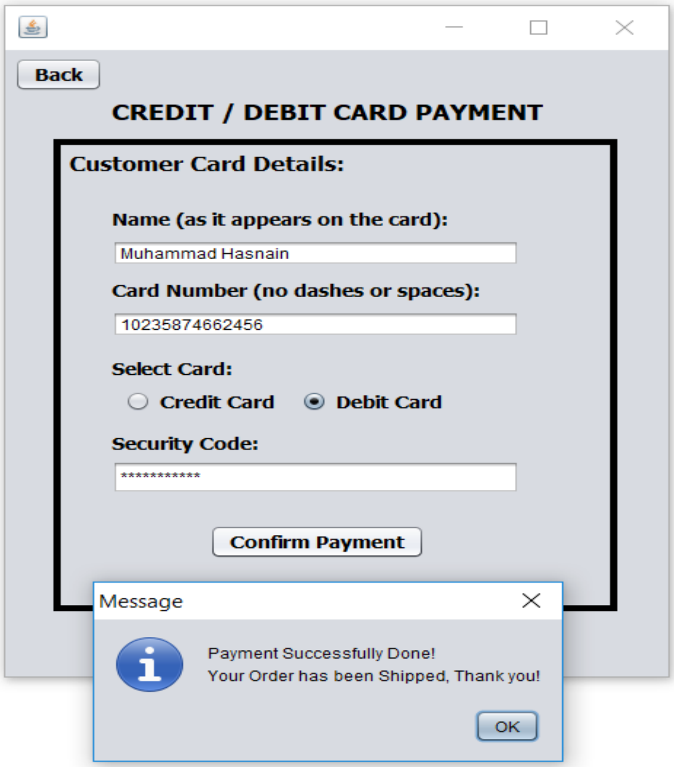


PAYMENT SELECTION

Message

Pay Cash Upon Receive, Thank you!

Fig 5.11.1: Pay Cash on Delivery



CREDIT / DEBIT CARD PAYMENT

Customer Card Details:

Name (as it appears on the card): Muhammad Hasnain

Card Number (no dashes or spaces): 10235874662456

Select Card: Credit Card [✓ Debit Card]

Security Code: ***********

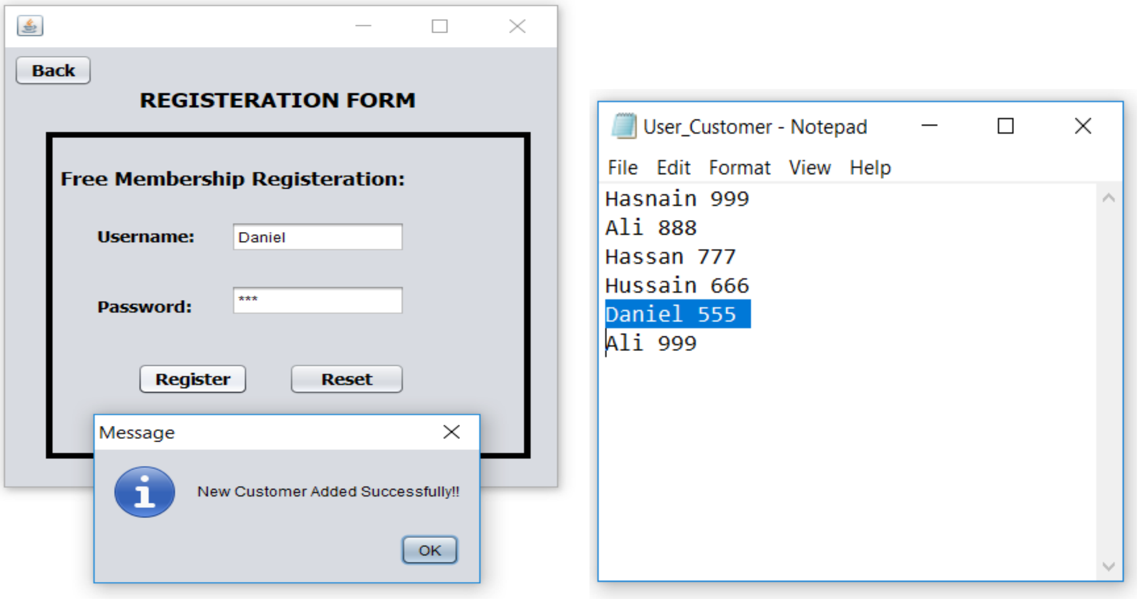
Message

Payment Successfully Done!
Your Order has been Shipped, Thank you!

Fig 5.11.2: Cash by Credit/Debit Card

6. Additional Features

6.1. New Customer Register



REGISTERATION FORM

Free Membership Registeration:

Username: Daniel

Password: ***

Message

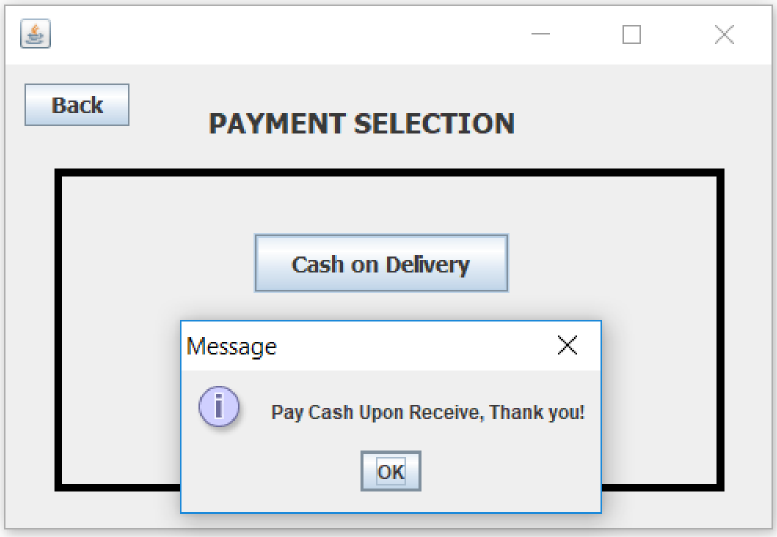
New Customer Added Successfully!!

User_Customer - Notepad

```
Hasnain 999
Ali 888
Hassan 777
Hussain 666
Daniel 555
Ali 999
```

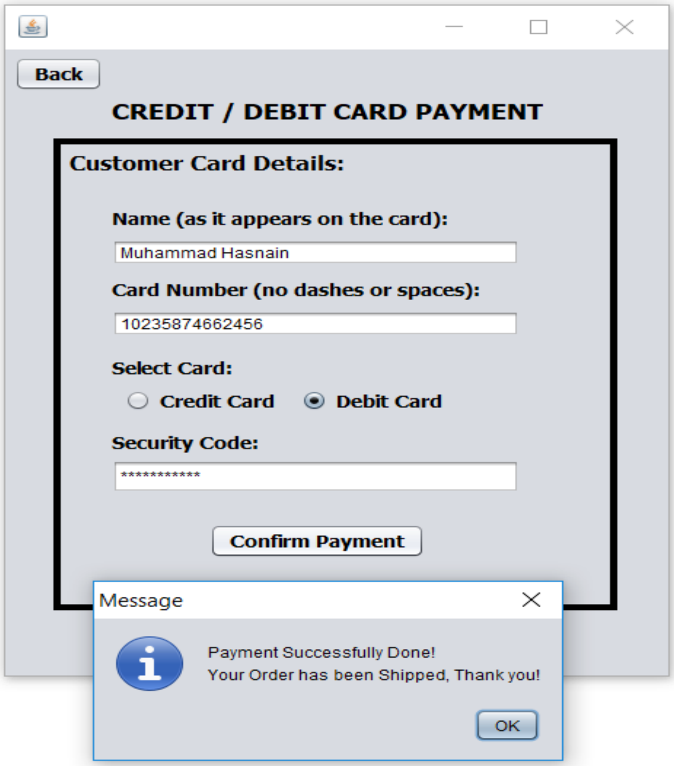
Fig 6.1.1 & 6.1.2: New Customer Register with Text-File

6.2. Payment Selection



Pay Cash Upon Receive, Thank you!

Fig 6.2.1: Pay Cash on Delivery



CREDIT / DEBIT CARD PAYMENT

Customer Card Details:

Name (as it appears on the card): Muhammad Hasnain

Card Number (no dashes or spaces): 10235874662456

Select Card: ○ Credit Card [✓ Debit Card]

Security Code: ***********

Payment Successfully Done!
Your Order has been Shipped, Thank you!

Fig 6.2.2: Cash by Credit/Debit Card

7. Conclusion

The OODJ module includes several topics and consists of basic knowledge of object-oriented programming. Which includes the understanding of the class, method, object, constructors, inheritance, polymorphism, abstraction, encapsulation and exception handling.

From this subject, I understand about beware of the origins of Java, understand the reasons for its wide use, understand and use block structure, understand the distinction between class, an object, understand difference between a method and a constructor, understand that how to create new classes that share some of the attributes of existing classes.

I have learned practical knowledge of Java by developing and designing the Retail Order Management System. The system is built in accordance with the assignment question requirements. I draw the fully detailed UML Diagrams including Use-Case, Class and Activity diagrams. I apply object-oriented programming concepts in my assignment which I have learned in OODJ module. From this assignment, I understand about the OOP in brief. With practical skill and knowledge, I test the whole program and test the codes.

In the end, I would like to thank **Miss Minnu Joseph** for being such a kind and supportive throughout my assignment. Without her help and guidelines, I would not be able to develop this system.

8. References

- Chaudhari, S. (2015). Object Oriented Programming Concepts - CodeProject. [online] Codeproject.com. Available at: https://www.codeproject.com/Articles/27775/Object-Oriented-Programming-Concepts [Accessed 27 Feb. 2018].
- dummies. (2016). How to Use a Constructor in Java - dummies. [online] Available at: http://www.dummies.com/programming/java/how-to-use-a-constructor-in-java/ [Accessed 27 Feb. 2018].
- Object-Oriented Programming Languages: Tools for Effective Communication on Application. (2016). International Journal of Science and Research (IJSR), 5(8), pp.406-411.
- Stackify. (2016). What Are OOPs Concepts in Java? The Four Main OOPs Concepts in Java, How They Work, Examples, and More. [online] Available at: https://stackify.com/oops-concepts-in-java/ [Accessed 27 Feb. 2018].
- Javatpoint. (2017). Java OOPs Concepts - Javatpoint. [online] Available at: https://www.javatpoint.com/java-oops-concepts [Accessed 26 Feb. 2018].