**School of Information Technology &Engineering**

**MCA(Master of Computer Application)**

**Programming in Java**

# Topic: Stock Management System – Facade Pattern

**Submitted by,**

**Karthik S(20MCA0097)**

## CONTENTS

## INTRODUCTION

Stock Management System helps the company to reduce the manual work and increase their efficiency of the system. Stock Management system helps the company to know about the product details of various products. Product details consists of product name, No.of.Order, Description etc. Customer details consists of order Id, Customer name etc. Then the order details consists of No.of.order, date expired etc. So we designed the stock management system in a unified interface by the use of Façade Pattern. Façade pattern provide the simple interface for the stock managemen system. Clients will be able use the system effectively and easily.

## PROBLEM STATEMENT

Stock management system is a system for tracking the product the product details like validation of the product, date expires, remainder and on sales. It also used to track the Order Status with the help of Order ID, date order, Item order and number of items. It is used to check the stock level of the product. It gives information about the reorder level, Out of Stock. It helps the customer to pay the amount in Different payment modes. The system also helps to calculate the taxes, Total, Sub-Total with respect to discount. It provides a unified interface for all the subsystem. It is the simpler and easy to use. So Façade pattern is suitable for the Stock Management System.

## KEYWORDS

Stock Management System – Facade- Unified Interface- simpler- Easy to Use.

## HARDWARE REQUIREMENTS

Desktop

8 GB RAM

## SOFTWARE REQUIREMENTS

Language- JAVA

Software- Java Netbeans

Database- Php myadmin.

## MODULE DESCRIPTION

### PRODUCT DETAILS

Product Module helps to know about the Product. It displays the Product Id, Product Name, Description, Stock Level, Out of Stock and Reorder Level.

### CUSTOMER DETAILS

This Module helps to know what product are customer ordered with the help of Order Id. It contains details of the customer like Name, Customer Id, address.

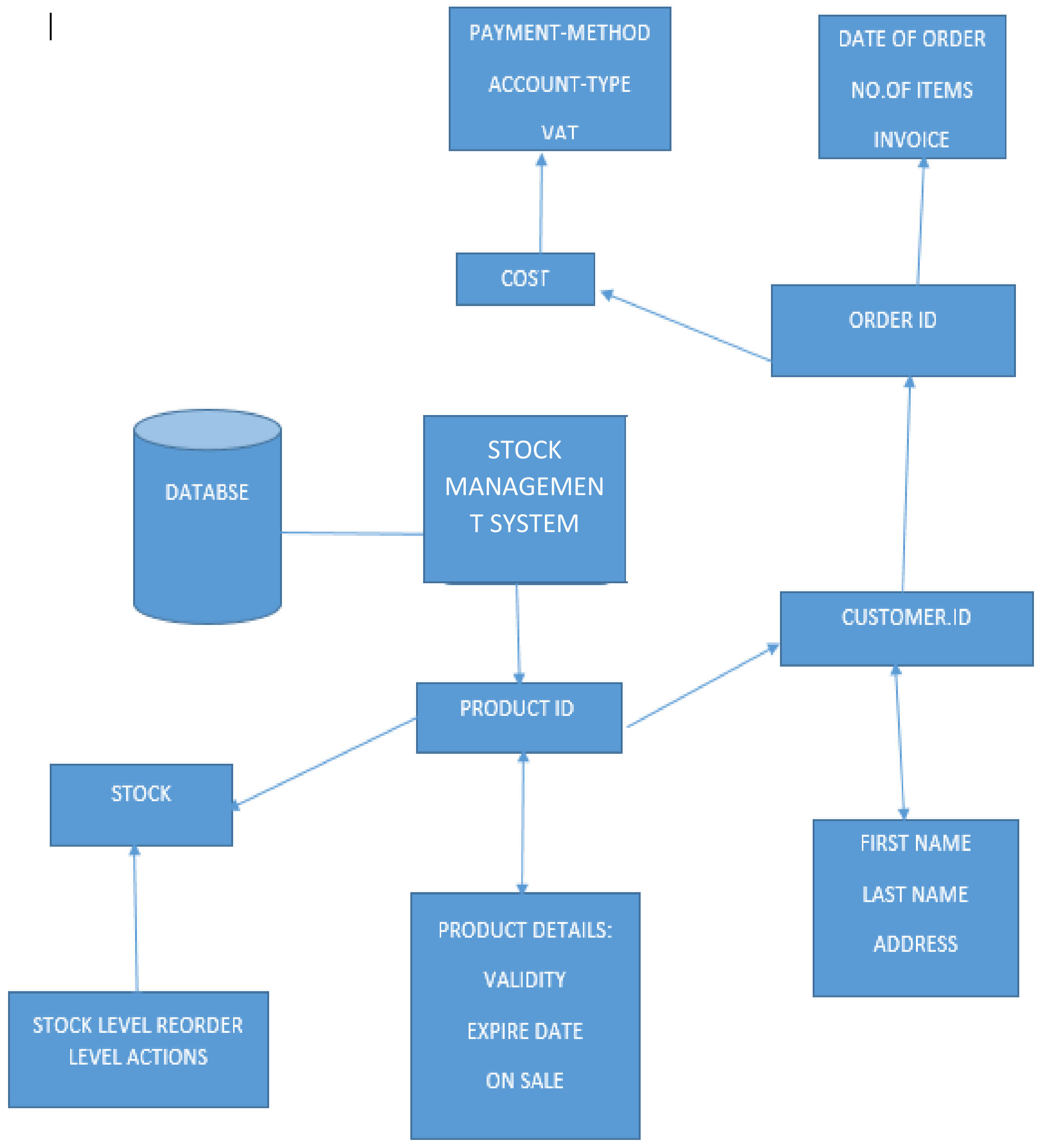
### ORDER DETAILS

This module contains the details of No. Of. Orders, Total, Tax, VAT, Account Type etc.

### VALIDITY

This module contains the Date of expires of the product, Validity, Date Order, Item Order.

## ARCHITECTURE OF SYSTEM



## FAÇADE

A façade pattern says that “just provide a unified and simplified interface to set of interface in a subsystem ,Therefore it hides the complexity of the subsystem from the client “.

In other words, Façade pattern describes a higher-level interface that makes the sub-system easier to use.

Practically every Abstract Factory is a type of Façade.

## INTENT

Provide a unified interface to a set of interfaces in a subsystem. Façade defines a higher level interface that makes the subsystem easier to use.

In the stock management system all the interfaces(Product Details, Customer Details, Tax, Validity Of Product) are combined in a unified interface and make the system easier to use.

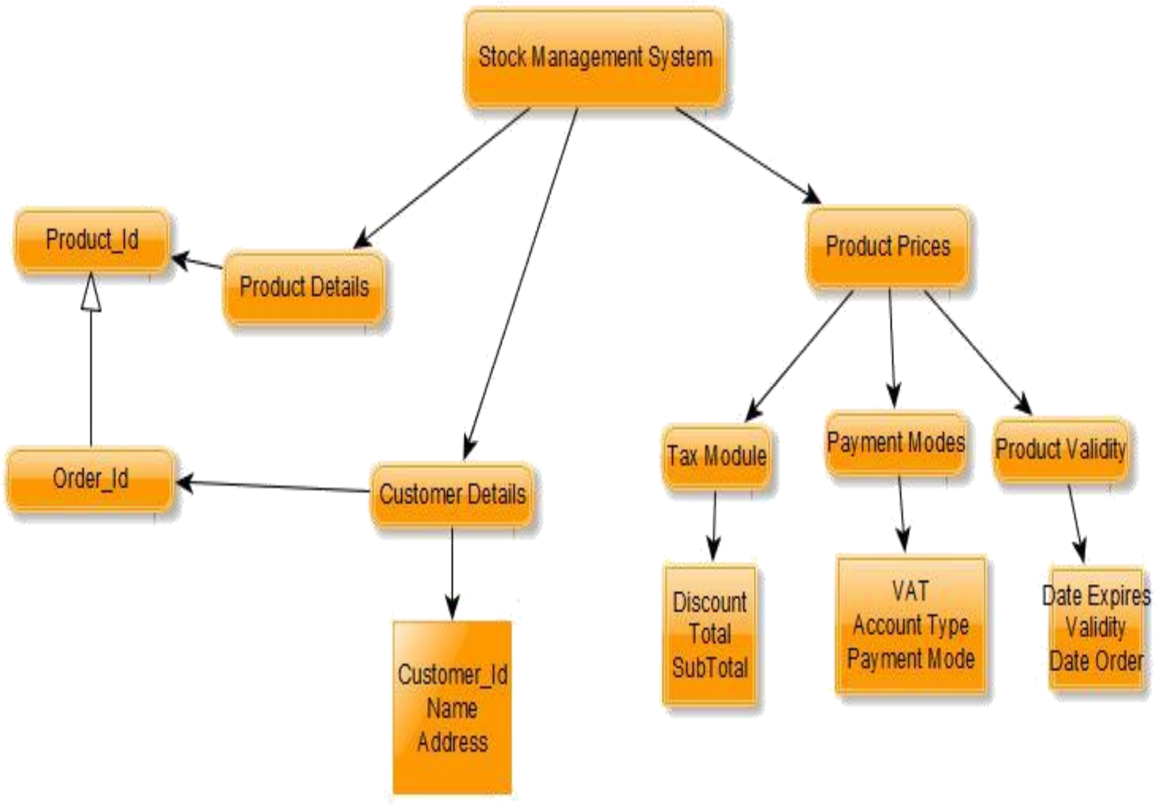
## APPLICABILITY

Façade pattern is applicable when we have a complex system that we want to expose to clients in a simplified way.

We want to Layer our Subsystems.

There are many dependcies between clients and the implemenatation classes of abstraction.

## STRUCTURE



## COLLOBORATIONS

Clients communicate with the subsystem by sending requests to facade, which forwards them to the appropriate subsytems object(s) .Although the subsystem objects perform the actual work, the facade may have to do work of its own to translate its interface to its subsystem interface.

## CONSEQUENCES:

(i) It shields client from subsystem components ,thereby reducing the number of objects that clients deal with and making the subsystem carier to use.
(ii)It peromotes weak coupling between the subsystem and its clients.
(iii) Help in layering the systems helps elements circular dependancy

## SAMPLE CODE

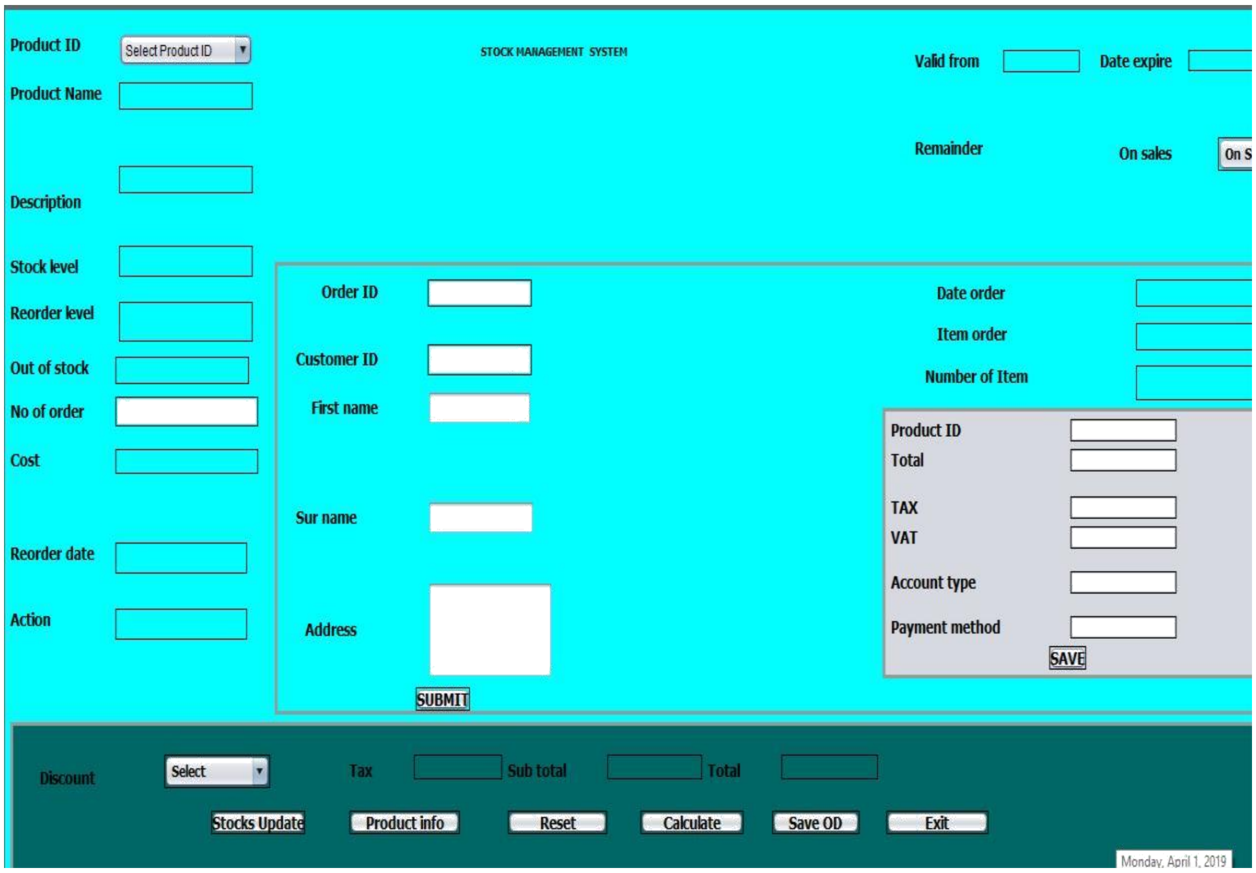
```
package ERPs;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import javax.swing.JOptionPane;

public class ERP extends javax.swing.JFrame {

    double[] product = new double[10];
    Connection con;

    PreparedStatement st;

    ResultSet rs;


    /**

     * Creates new form ERP

     */

    public ERP() {

        initComponents();

        try

        {

            Class.forName("com.mysql.jdbc.Driver");


con=DriverManager.getConnection("jdbc:mysql://localhost:3306/erpstock?zeroDa
teTimeBehavior=convertToNull","root","");

            JOptionPane.showMessageDialog(null,"Database connected");

        }

        catch(Exception e)

        {

            JOptionPane.showMessageDialog(null,"Database not connected");

        }
```

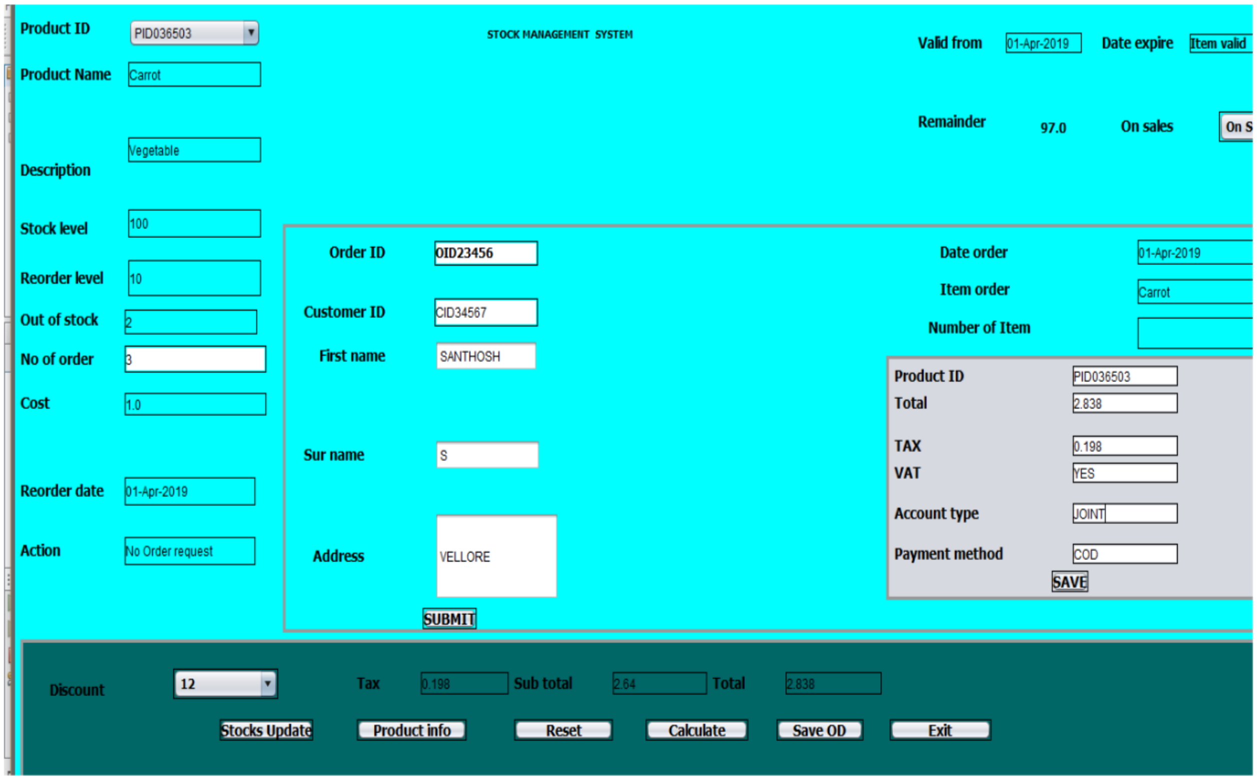
```
}


/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated
Code"> private void initComponents() {


    jPanel1 = new javax.swing.JPanel(); jPanel2 =
    new javax.swing.JPanel(); jLabel19 = new
    javax.swing.JLabel(); jComboBox2 = new
    javax.swing.JComboBox<>(); jLabel20 = new
    javax.swing.JLabel(); jLabel21 = new
    javax.swing.JLabel();
    jLabel22 = new javax.swing.JLabel();
    jLabel23 = new javax.swing.JLabel();
    jLabel24 = new javax.swing.JLabel();
```

```
jLabel25 = new javax.swing.JLabel();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jButton3 = new javax.swing.JButton();
jButton4 = new javax.swing.JButton();
jButton6 = new javax.swing.JButton();
jButton7 = new javax.swing.JButton();
jPanel3 = new javax.swing.JPanel(); jPanel4
= new javax.swing.JPanel(); jLabel44 = new
javax.swing.JLabel(); jLabel45 = new
javax.swing.JLabel(); jLabel46 = new
javax.swing.JLabel(); jLabel50 = new
javax.swing.JLabel(); jTextField7 = new
javax.swing.JTextField(); jLabel51 = new
javax.swing.JLabel(); jTextField8 = new
javax.swing.JTextField(); jTextField9 = new
javax.swing.JTextField(); jTextField10 = new
javax.swing.JTextField(); jTextField11 = new
javax.swing.JTextField(); jButton5 = new
javax.swing.JButton();
```

```
TAX = new javax.swing.JLabel();

jTextField12 = new javax.swing.JTextField();

jLabel33 = new javax.swing.JLabel();

jLabel34 = new javax.swing.JLabel();
```

## SCREENSHOTS





Product ID: PID036503
Product Name: Carrot
Description: Vegetable
Stock level: 100
Reorder level: 10
Out of stock: 2
No of order: 3
Cost: 1.0
Reorder date: 01-Apr-2019
Action: No Order request

Valid from: 01-Apr-2019
Date expire: Item valid
Remainder: 97.0
On sales: On S

Order ID: OID23456
Customer ID: CID34567
First name: SANTHOSH
Sur name: S
Address: VELLORE

Date order: 01-Apr-2019
Item order: Carrot
Number of Item:

Product ID: PID036503
Total: 2.838
TAX: 0.198
VAT: YES
Account type: JOINT
Payment method: COD

Discount: 12
Tax: 0.198
Sub total: 2.64
Total: 2.838

## PROS AND USAGE

(i) It shields the client from the complexity of the sub-system components.

(ii) It promotes loose coupling between subsystems and its client.

## USAGE:

(i) When you want to provide simple interface to a complex sub-system

(ii) When several dependencies exist between clients and the implementation classes of an abstraction.

## CONCLUSION

Stock Management system provide the unified interface. Customer can easily umderstand the system. Reduces the coupling between the client and the subsystem by the Façade Pattern. It provides the simple interface for the stock management system.