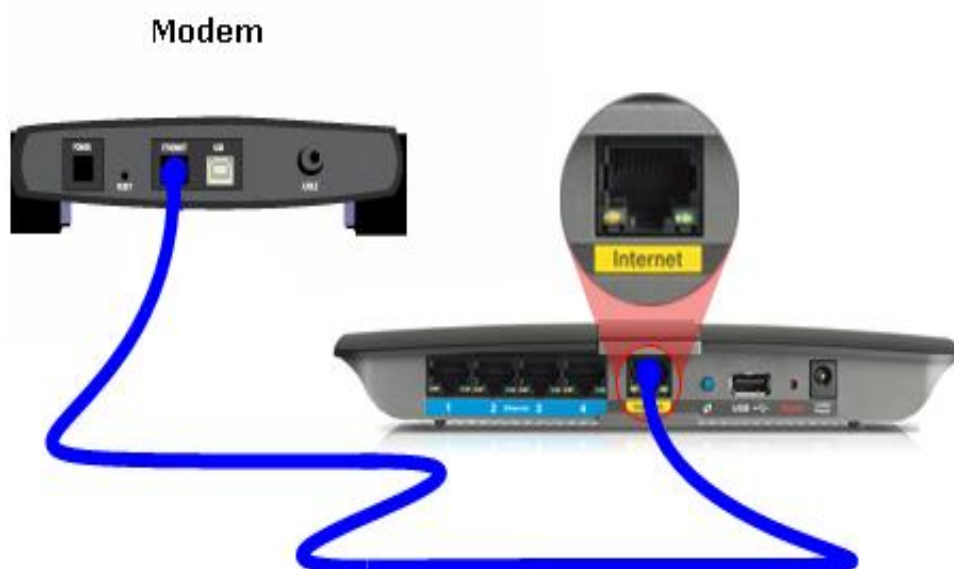


< Mini-Projet />

< Partie IG & API-JDBC />



Réalisée par :

ZENAGUI Anas

Encadrée par :

Pr. EN-NAIMI El Mokhtar

Définition de JAVA SWING :

- PSwing est une bibliothèque graphique pour le langage de programmation Java, faisant partie du package Java Foundation Classes (JFC), inclus dans J2SE.

❖ Objectifs

- Création d'une application qui gère des produits et les commandes de pharmacie.

Partie 1 : Création de la base de transaction de données

- ✓ Dans cette partie j'ai créé une base de données s'appelle « BaseTransaction » comme une interface pour l'implémenter par les autres classes qu'on va les voir

```
✓ package ma.fstt.model;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;
```

```
import java.util.List;

public abstract class BaseTranscation<T> {

    protected Connection cnx ;

    protected Statement statement ;

    protected PreparedStatement preparedStatement ;

    protected ResultSet resultSet ;

    // private String url = "jdbc:mysql://localhost:3306/GPharmacie";
    // private String login = "root";
    // private String pass = "";

    public BaseTranscation() throws SQLException {

        this.cnx =
        DriverManager.getConnection("jdbc:mysql://localhost:3306/GPharmacie",
        "root", "");

    }

    public abstract void save(T object) throws SQLException ;
    public abstract void update(T object) throws SQLException ;
    public abstract void delete(T object) throws SQLException ;
    public abstract List<T> getAll() throws SQLException ;
    public abstract T getOne(Integer id) throws SQLException ;

}
```

Partie 2 : Créations des classes important :

- Premièrement j'ai créé une classe s'appelle client avec leur méthodes et attributs.

```
- package ma.fstt.model;

import java.util.List;

// java bean
public class Client {

    private Integer id_client ;

    private String nom ;

    private String email ;

    private String genre ;

    private String password;

    // one to many
    private List<Commande> lsitcmds ;


    public List<Commande> getLsitcmds() {
        return lsitcmds;
    }
}
```

```
public void setLsitemds(List<Commande> lsitemds) {  
    this.lsitemds = lsitemds;  
}
```

```
public Client(int id_client, String nom, String email, String genre,  
String password) {  
    super();  
    this.id_client = id_client;  
    this.nom = nom;  
    this.email = email;  
    this.genre = genre;  
    this.password = password;  
}
```

```
public Client() {  
    super();  
    // TODO Auto-generated constructor stub  
}
```

```
public Integer getId_client() {  
    return id_client;  
}
```

```
public void setId_client(Integer id_client) {  
    this.id_client = id_client;  
}
```

```
public String getNom() {  
    return nom;  
}
```

```
public void setNom(String nom) {  
    this.nom = nom;  
}
```

```
public String getEmail() {  
    return email;  
}
```

```
public void setEmail(String email) {  
    this.email = email;  
}
```

```
public String getGenre() {  
    return genre;  
}
```

```
public void setGenre(String genre) {  
    this.genre = genre;  
}
```

```
public String getPassword() {  
    return password;  
}
```

```
public void setPassword(String password) {  
    this.password = password;  
}
```

```
@Override  
public String toString() {  
    return "Client [id_client=" + id_client + ", nom=" + nom + ",
```

```

        email=" + email + ", genre=" + genre + "]];
    }

}

```

et son classe de transaction est :

```

package ma.fstt.model;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class ClientTransaction extends BaseTranscation<Client>{

    private CommandeTransaction commandeTransaction ;

    public ClientTransaction() throws SQLException {
        super();
        // TODO Auto-generated constructor stub

        commandeTransaction = new CommandeTransaction();
    }

    @Override
    public void save(Client object) throws SQLException {

        String request= "insert into client (nom , email, genre) values ( ? , ? , ?)" ;
    }
}

```

```

        this.preparedStatement = this.cnx.prepareStatement(request);

        // maooing relatipon et objet table client et objet client
        this.preparedStatement.setString(1, object.getNom());
        this.preparedStatement.setString(2, object.getEmail());
        this.preparedStatement.setString(3, object.getGenre());

        this.preparedStatement.execute();

    }

    @Override
    public void update(Client object) throws SQLException {
        // TODO Auto-generated method stub

    }

    @Override
    public void delete(Client object) throws SQLException {

        String request= "delete from client where id_client=?" ;

        this.preparedStatement = this.cnx.prepareStatement(request);

        this.preparedStatement.setInt(1, object.getId_client());

        this.preparedStatement.execute();

    }

    @Override
    public List<Client> getAll() throws SQLException {
        // TODO Auto-generated method stub

        List<Client> malist = new ArrayList<Client>();
    }

```



```

String request= "select * from client " ;

this.statement = this.cnx.createStatement();

this.resultSet = this.statement.executeQuery(request);

while (this.resultSet.next()) {

    malist.add(new Client( this.resultSet.getInt(1),
this.resultSet.getString(2), this.resultSet.getString(3),
this.resultSet.getString(4), this.resultSet.getString(5)));

}

return malist;
}

@Override
public Client getOne(Integer id) throws SQLException {
    // TODO Auto-generated method stub

    String request= "select * from client where id_client =?" ;

    this.preparedStatement = this.cnx.prepareStatement(request);

    this.preparedStatement.setInt(1, id);

    this.resultSet = this.preparedStatement.executeQuery();

    while (this.resultSet.next()) {

```

```

        Client cli = new Client( this.resultSet.getInt(1),
this.resultSet.getString(2), this.resultSet.getString(3),
this.resultSet.getString(4), this.resultSet.getString(5));

        cli.setLsitemds(commandeTransaction.getByClient(cli));

        return cli ;

    }

    return null;
}
}

```

- 2^e classe c'est le classe Product :

```

- package ma.fstt.model;

public class Product {

    private int id;
    private String name;
    private float price;
    private String addDate;
    private final byte[] image;
    private Categorie categorie;

    public Product(int id, String name, float price, String addDate,
byte[] image)
    {
        this.id = id;
        this.name = name;
        this.price = price;
        this.addDate = addDate;
        this.image = image;
    }
}

```

```
}
```

```
public int getId()  
{  
    return id;  
}
```

```
public String getName()  
{  
    return name;  
}
```

```
public float getPrice()  
{  
    return price;  
}
```

```
public String getAddDate()  
{  
    return addDate;  
}
```

```
public byte[] getImage()  
{  
    return image;  
}
```

```
public void setId(int id) {  
    this.id = id;  
}
```

```
public void setName(String name) {  
    this.name = name;  
}
```

```
public void setPrice(float price) {  
    this.price = price;  
}
```

```

    public void setAddDate(String addDate) {
        this.addDate = addDate;
    }

    public void setCategorie(Categorie categorie) {
        this.categorie = categorie;
    }
}

```

- **et son classe de transaction**

```

package ma.fstt.model;

import java.sql.SQLException;
import java.util.List;

public class ProductTransaction extends BaseTranscation<Product> {

    public ProductTransaction() throws Exception{
        super();
    }

    @Override
    public void save(Product object) throws SQLException {

    }

    @Override
    public void update(Product object) throws SQLException {

    }

    @Override
    public void delete(Product object) throws SQLException {

    }
}

```

```

@Override
public List<Product> getAll() throws SQLException {
    return null;
}

@Override
public Product getOne(Integer id) throws SQLException {
    return null;
}
}

```

- 3e c'est la classe commande

```

- package ma.fstt.model;

public class Commande {

    private Integer id_commande;
    private String date;
    private int qtePrd;

    // Many to one
    private Client client ;
    private Product product;

    public Client getClient() {
        return client;
    }
    public void setClient(Client client) {
        this.client = client;
    }
}

```

```
public Commande() {
    super();
    // TODO Auto-generated constructor stub
}

public Commande(Integer id_commande, String date, int qtePrd , int
cl, int pr) {
    super();
    this.id_commande = id_commande;
    this.date = date;
    this.client.setId_client(cl);
    this.product.setId(pr);
    this.qtePrd = qtePrd;
}

public Integer getId_commande() {
    return id_commande;
}

public void setId_commande(Integer id_commande) {
    this.id_commande = id_commande;
}

public String getDate() {
    return date;
}

public void setDate(String date) {
    this.date = date;
}

public int getQtePrd() {
    return qtePrd;
}

public void setQtePrd(int qtePrd) {
    this.qtePrd = qtePrd;
}

public Product getProduct() {
    return product;
}
```

```

    public void setProduct(Product product) {
        this.product = product;
    }

    @Override
    public String toString() {
        return "Commande [id_commande=" + id_commande + ", date="
+ date + ", client=" + client
        + ", Produit=" + product + "]\n";
    }
}

```

et son classe de transaction

```

package ma.fstt.model;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class CommandeTransaction extends BaseTranscation<Commande>
{

    public CommandeTransaction() throws SQLException {
        super();
        // TODO Auto-generated constructor stub
    }

    @Override
    public void save(Commande object) throws SQLException {

        String request = "INSERT INTO commande( numero, date , id_client )

```

```
VALUES (?,? , ?) ";
```

```
this.preparedStatement = this.cnx.prepareStatement(request);
```

```
// mapping objet relation
```

```
this.preparedStatement.setString(2, object.getDate());
```

```
this.preparedStatement.setInt(3, object.getClient().getId_client());
```

```
this.preparedStatement.execute();
```

```
}
```

```
@Override
```

```
public void update(Commande object) throws SQLException {
```

```
    // TODO Auto-generated method stub
```

```
}
```

```
@Override
```

```
public void delete(Commande object) throws SQLException {
```

```
    // TODO Auto-generated method stub
```

```
}
```

```
@Override
```

```
public List<Commande> getAll() throws SQLException {
```

```
    // TODO Auto-generated method stub
```

```
List<Commande> liste = new ArrayList<Commande>();
```

```
String request = "select * from commande ";
```

```
this.statement = this.cnx.createStatement();
```

```
this.resultSet = this.statement.executeQuery(request);
```



```

        while ( this.resultSet.next()) {

            liste.add(new Commande(resultSet.getInt("id_commande"),
resultSet.getString("date"), resultSet.getInt("qte"),
resultSet.getInt("id_client"), resultSet.getInt("id_prod")));
        }

        return liste;
    }

    @Override
    public Commande getOne(Integer id) throws SQLException {
        // TODO Auto-generated method stub
        String request = "select * from commande where id_commande= ?";

        this.preparedStatement = this.cnx.prepareStatement(request);

        this.preparedStatement.setInt(1, id);

        this.resultSet = this.preparedStatement.executeQuery();

        while ( this.resultSet.next()) {

            new Commande(resultSet.getInt("id_commande"),
resultSet.getString("date"), resultSet.getInt("qte"),
resultSet.getInt("id_client"), resultSet.getInt("id_prod"));

        }

        return null;
    }

    public List<Commande> getByClient(Client client) throws SQLException

```

```
{  
    // TODO Auto-generated method stub  
    List<Commande> liste = new ArrayList<Commande>();  
  
    String request = "select * from commande where id_client = ? ";  
  
    this.preparedStatement = this.cnx.prepareStatement(request);  
  
    this.preparedStatement.setInt(1, client.getId_client());  
  
    this.resultSet = this.preparedStatement.executeQuery();  
  
    while ( this.resultSet.next()) {  
  
        liste.add(new Commande(resultSet.getInt("id_commande"),  
resultSet.getString("date"), resultSet.getInt("qte"),  
resultSet.getInt("id_client"), resultSet.getInt("id_prod")));  
  
    }  
  
    return liste;  
}  
}
```

Partie 3 : Configurer et vérifier l'accès client sans fil

- cette partie s'intéresse à l'interface graphique c'est-à-dire, on va créer pour chaque classe qu'on a créée une classe qui développe l'interface graphique
- la 1^{er} c'est la classe Client :

```
- package ma.fstt.ihm;

import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.table.AbstractTableModel;

import ma.fstt.model.ClientTransaction;

import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.JLabel;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.awt.event.ActionEvent;
import javax.swing.JTable;

import java.awt.*;
import javax.swing.*;

import com.toedter.calendar.JDateChooser;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
```

```
import java.awt.event.KeyListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
```

```
public class Client extends JFrame implements DBInfo,
ActionListener {
```

```
    JPanel panel;
    JLabel idLabel;
    JLabel nameLabel;
    JLabel emailLabel;
    JLabel genreLabel;
    JLabel dateLabel;
    JLabel searchLabel;
    JTextField idField;
    JTextField nameField;
    JTextField emailField;
    JTextField genreField;
    JTextField searchField;
    JDateChooser dateField;
```

```

JButton updateImageButton;
JButton insertButton;
JButton updateButton;
JButton deleteButton;
JButton exitButton;
JButton backButton;
JTable table;
JScrollPane tableScroller;
DefaultTableModel model;
String currentImagePath = null;
AddNewProductDialog addProductDialog;

public Client() {
    this.createAndShowGUI();
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new Client();
        }
    });
}

private void createAndShowGUI() {
    try {
        UIManager.LookAndFeelInfo[] var1 =
        UIManager.getInstalledLookAndFeels();
        int var2 = var1.length;

        for(int var3 = 0; var3 < var2; ++var3) {
            UIManager.LookAndFeelInfo info = var1[var3];
            if ("Nimbus".equals(info.getName())) {
                UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (IllegalAccessException | InstantiationException |
    UnsupportedLookAndFeelException | ClassNotFoundException var6)
    {

```

```

    ;
}

this.panel = new JPanel((LayoutManager)null);
this.model = new DefaultTableModel();
this.table = new JTable(this.model);
this.tableScroller = new JScrollPane(this.table);
this.idLabel = new JLabel("ID");
this.nameLabel = new JLabel("Name");
this.emailLabel = new JLabel("Email");
this.genreLabel = new JLabel("Genre");
this.dateLabel = new JLabel("Date");
this.searchLabel = new JLabel();
this.idField = new JTextField();
this.nameField = new JTextField();
this.emailField = new JTextField();
this.genreField = new JTextField();
this.dateField = new JDateChooser();
this.searchField = new JTextField();
this.insertButton = new JButton("Add New", new
ImageIcon(this.getClass().getResource("/images/insert.png")));
this.updateButton = new JButton("Update", new
ImageIcon(this.getClass().getResource("/images/update.png")));
this.deleteButton = new JButton("Delete", new
ImageIcon(this.getClass().getResource("/images/delete.png")));
this.searchLabel = new JLabel("Search");
this.backButton = new JButton("Back", new
ImageIcon(this.getClass().getResource("/images/previous.png")));
this.exitButton = new JButton("Exit", new
ImageIcon(this.getClass().getResource("/images/exit.png")));
this.addProductDialog = new AddNewProductDialog(this,
this.model);
this.idLabel.setBounds(20, 200, 50, 40);
this.idField.setBounds(80, 200, 270, 40);
this.nameLabel.setBounds(20, 300, 50, 40);
this.nameField.setBounds(80, 300, 270, 40);
this.emailLabel.setBounds(20, 400, 50, 40);
this.emailField.setBounds(80, 400, 270, 40);
this.genreLabel.setBounds(20, 500, 50, 40);
this.genreField.setBounds(80, 500, 270, 40);

```

```
this.deleteButton.setBounds(80, 575, 130, 40);
this.updateButton.setBounds(220, 575, 130, 40);
this.tableScroller.setBounds(377, 40, 520, 505);
this.searchField.setBounds(530, 577, 255, 36);
this.searchLabel.setBounds(460, 575, 115, 40);
this.insertButton.setBounds(920, 40, 130, 60);
this.backButton.setBounds(920, 350, 130, 40);
this.exitButton.setBounds(920, 575, 130, 40);
this.idLabel.setFont(new Font("Arial", 1, 16));
this.idField.setFont(new Font("Arial", 1, 15));
this.nameLabel.setFont(new Font("Arial", 1, 16));
this.nameField.setFont(new Font("Arial", 1, 15));
this.emailLabel.setFont(new Font("Arial", 1, 16));
this.emailField.setFont(new Font("Arial", 1, 15));
this.genreLabel.setFont(new Font("Arial", 1, 16));
this.genreField.setFont(new Font("Arial", 1, 15));
this.dateLabel.setFont(new Font("Arial", 1, 16));
this.dateField.setFont(new Font("Arial", 1, 13));
this.deleteButton.setFont(new Font("Arial", 1, 16));
this.updateButton.setFont(new Font("Arial", 1, 16));
this.insertButton.setFont(new Font("Arial", 1, 16));
this.searchField.setFont(new Font("Arial", 1, 15));
this.searchLabel.setFont(new Font("Arial", 1, 17));
this.backButton.setFont(new Font("Arial", 1, 16));
this.exitButton.setFont(new Font("Arial", 1, 16));
```

```
this.idField.setBorder(BorderFactory.createLineBorder(Color.gray, 2,
true));
```

```
this.nameField.setBorder(BorderFactory.createLineBorder(Color.gray
, 2, true));
```

```
this.emailField.setBorder(BorderFactory.createLineBorder(Color.gray
, 2, true));
```

```
this.genreField.setBorder(BorderFactory.createLineBorder(Color.gray
, 2, true));
```

```
this.searchField.setBorder(BorderFactory.createLineBorder(Color.gra
y, 2, true));
```

```
this.idField.setEditable(false);
this.idField.setBackground(new Color(240, 240, 240));
this.table.setColumnSelectionAllowed(false);
this.table.getParent().setBackground(Color.white);
this.tableScroller.setViewportViewView(this.table);
this.model.addColumn("ID");
this.model.addColumn("Name");
this.model.addColumn("Email");
this.model.addColumn("Genre");

try {
    this.viewProductsInTheTable();
} catch (Exception var5) {
    ;
}

this.table.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent evt) {
        int index = Client.this.table.getSelectedRow();
        Client.this.showClient(index);
        Client.this.currentImagePath = null;
    }
});
this.table.addKeyListener(new KeyListener() {
    public void keyReleased(KeyEvent e) {
        if (e.getKeyCode() == 38 || e.getKeyCode() == 40) {
            Client.this.showClient(Client.this.table.getSelectedRow());
        }
    }

    public void keyTyped(KeyEvent e) {
    }

    public void keyPressed(KeyEvent e) {
    }
});
this.searchField.addKeyListener(new KeyListener() {
    public void keyReleased(KeyEvent e) {
        Client.this.search();
    }
});
```



```
}

public void keyTyped(KeyEvent e) {
}

public void keyPressed(KeyEvent e) {
}
});

this.insertButton.addActionListener(this);
this.updateButton.addActionListener(this);
this.deleteButton.addActionListener(this);
this.backButton.addActionListener(this);
this.exitButton.addActionListener(this);
this.panel.add(this.idLabel);
this.panel.add(this.idField);
this.panel.add(this.idField);
this.panel.add(this.nameLabel);
this.panel.add(this.nameField);
this.panel.add(this.emailLabel);
this.panel.add(this.emailField);
this.panel.add(this.genreLabel);
this.panel.add(this.genreField);
this.panel.add(this.dateLabel);
this.panel.add(this.dateField);
this.panel.add(this.insertButton);
this.panel.add(this.updateButton);
this.panel.add(this.deleteButton);
this.panel.add(this.tableScroller);
this.panel.add(this.searchField);
this.panel.add(this.searchLabel);
this.panel.add(this.backButton);
this.panel.add(this.exitButton);
this.panel.setPreferredSize(new Dimension(1070, 640));
this.panel.setMinimumSize(new Dimension(1070, 640));
this.setContentPane(new JPanel(new GridBagLayout()));
this.add(this.panel);
this.setTitle("Pharmacy Products");
this.setDefaultCloseOperation(3);
this.pack();
```

```

        this.setLocationRelativeTo((Component)null);
        this.setVisible(true);
    }

    private Connection getConnection() {
        try {
            Connection con =
                DriverManager.getConnection("jdbc:mysql://localhost/gpharmacie?useUnicode=yes&characterEncoding=UTF-8", "root", "");
            return con;
        } catch (SQLException var3) {
            JOptionPane.showMessageDialog(this, var3.getMessage(),
                "Connection Error", 0);
            return null;
        }
    }

    private void viewProductsInTheTable() {
        ArrayList<ma.fstt.model.Client> clients = new ArrayList<>();
        Connection con = this.getConnection();
        String query = "SELECT * FROM client";

        try {
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery(query);

            while(rs.next()) {
                ma.fstt.model.Client client = new
                ma.fstt.model.Client(rs.getInt("id_client"), rs.getString("nom"),
                rs.getString("email"), rs.getString("genre"), "");
                clients.add(client);
            }

            con.close();
        } catch (SQLException var8) {
            JOptionPane.showMessageDialog(this, var8.getMessage(),
                "Error", 0);
        }

        this.model.setRowCount(0);
    }

```

```

Object[] row = new Object[4];

for(int i = 0; i < clients.size(); ++i) {
    row[0] = (clients.get(i)).getId_client();
    row[1] = (clients.get(i)).getNom();
    row[2] = (clients.get(i)).getEmail();
    row[3] = (clients.get(i)).getGenre();
    this.model.addRow(row);
}

}

private boolean checkInputs() {
    if (!this.nameField.getText().equals("") &&
!this.emailField.getText().equals("") &&
!this.genreField.getText().equals("")) {
        try {
            this.emailField.getText();
            return true;
        } catch (NumberFormatException var2) {
            return false;
        }
    } else {
        JOptionPane.showMessageDialog(this, "Product information are
not updated because one or more fields are empty", "Error", 0);
        return false;
    }
}

private void addNewProduct() {
    this.addProductDialog.show();
}

private void updateClient() {
    if (this.checkInputs() && this.idField.getText() != null) {
        if (this.currentImagePath != null) {
            try {
                InputStream img = new FileInputStream(new

```

```

File(this.currentImagePath));
        String query = "UPDATE client SET nom = ?, email = ?,
genre = ? WHERE id_client = ?";
        Connection con = this.getConnection();
        PreparedStatement ps = con.prepareStatement(query);
        ps.setString(1, this.nameField.getText());
        ps.setString(2, this.emailField.getText());
        ps.setString(3, this.genreField.getText());
        ps.setInt(4, Integer.parseInt(this.idField.getText()));
        ps.executeUpdate();
        con.close();
        this.viewProductsInTheTable();
        JOptionPane.showMessageDialog(this, "Product information
has been successfully updated");
    } catch (FileNotFoundException | NumberFormatException |
SQLException | HeadlessException var8) {
        JOptionPane.showMessageDialog(this, var8.getMessage(),
"Error", 0);
    }
    } else {
        try {
            String query = "UPDATE client SET nom = ?, email = ?,
genre = ? WHERE id_client = ?";
            Connection con = this.getConnection();
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, this.nameField.getText());
            ps.setString(2, this.emailField.getText());
            ps.setString(3, this.genreField.getText());
            ps.setInt(4, Integer.parseInt(this.idField.getText()));
            ps.executeUpdate();
            con.close();
            this.viewProductsInTheTable();
            JOptionPane.showMessageDialog(this, "Product information
are successfully updated");
        } catch (NumberFormatException | SQLException |
HeadlessException var7) {
            JOptionPane.showMessageDialog(this, var7.getMessage(),
"Error", 0);
        }
    }
}

```

```

    }

}

private ArrayList<ma.fstt.model.Client> getClientList() {
    ArrayList<ma.fstt.model.Client> clients = new ArrayList<>();
    Connection con = this.getConnection();
    String query = "SELECT * FROM client";

    try {
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery(query);

        while(rs.next()) {
            ma.fstt.model.Client client = new
ma.fstt.model.Client(rs.getInt("id_client"), rs.getString("nom"),
rs.getString("email"), rs.getString("genre", ""));
            clients.add(client);
        }

        con.close();
    } catch (SQLException var7) {
        JOptionPane.showMessageDialog(this, var7.getMessage(),
"Error", 0);
    }

    return clients;
}

private void showClient(int index) {

this.idField.setText(Integer.toString((this.getClientList().get(index)).g
etId_client()));
    this.nameField.setText((this.getClientList().get(index)).getNom());

this.emailField.setText((this.getClientList().get(index)).getEmail());

this.genreField.setText((this.getClientList().get(index)).getGenre());

}

```

```

private void deleteClient() {
    if (this.table.getSelectedRow() == -1) {
        JOptionPane.showMessageDialog(this, "Please select the
product that you want to delete from the table and try again");
    } else {
        try {
            Connection con = this.getConnection();
            PreparedStatement ps = con.prepareStatement("DELETE
FROM client WHERE id = ?");
            int id = Integer.parseInt(this.idField.getText());
            ps.setInt(1, id);
            ps.executeUpdate();
            con.close();
            int nextSelectedRowIndex = this.table.getSelectedRow();
            this.viewProductsInTheTable();
            if (this.table.getRowCount() == 1) {
                this.table.setRowSelectionInterval(0, 0);
                this.showClient(0);
            } else if (this.table.getRowCount() > 1 &&
nextSelectedRowIndex < this.table.getRowCount()) {
                this.table.setRowSelectionInterval(nextSelectedRowIndex,
nextSelectedRowIndex);
                this.showClient(nextSelectedRowIndex);
            } else if (this.table.getRowCount() > 1 &&
nextSelectedRowIndex == this.table.getRowCount()) {
                --nextSelectedRowIndex;
                this.table.setRowSelectionInterval(nextSelectedRowIndex,
nextSelectedRowIndex);
                this.showClient(nextSelectedRowIndex);
            }

            if (this.table.getRowCount() == 0) {
                this.idField.setText("");
                this.nameField.setText("");
                this.emailField.setText("");
                this.genreField.setText("");
            }
        } catch (SQLException var5) {
            JOptionPane.showMessageDialog(this, var5.getMessage(),

```

```

        "Error", 0);
    }
}

}

private void showPreviousPage() {
//    if (this.table.getSelectedRow() > 0) {
//        int currentSelectedRow = this.table.getSelectedRow() - 1;
//        this.table.setRowSelectionInterval(currentSelectedRow,
currentSelectedRow);
//        this.showProduct(currentSelectedRow);
//    }

}

private void search() {
    String keyword = this.searchField.getText();
    TableRowSorter<DefaultTableModel> tr = new
TableRowSorter(this.model);
    this.table.setRowSorter(tr);
    tr.setRowFilter(RowFilter.regexFilter(keyword, new int[0]));
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == this.updateImageButton) {
    } else if (e.getSource() == this.insertButton) {
        this.addNewProduct();
    } else if (e.getSource() == this.updateButton) {
        this.updateClient();
    } else if (e.getSource() == this.deleteButton) {
        this.deleteClient();
    } else if (e.getSource() == this.backButton) {
        this.showPreviousPage();
    } else if (e.getSource() == this.exitButton) {
        System.exit(0);
    }
}
}

```

}



ID	Name	Email	Genre
3	Mohamed	mohamed@fstt.ma	M
4	Liela	dfdf	F
5	M	yassine@gmail.com	
6	Test	Test@gmail.com	M
7	Moahmed	mohamed@gmail.c...	M
8	toto	toto@gmail.com	F
9	yyyy	yyyy@yyy.com	F
10	LST	LST@yyy.com	M
11	sjdfjsd	toto@gmail.com	F
12	Lotfi	lotfi@gmail.com	H
13	Zenagui Anas	anas.zenagui@gm...	M
14	Zenagui Anas	anas.zenagui@gm...	M
15	Zenagui Anas	anas.zenagui@gm...	M
16	Zenagui Anas	anas.zenagui@gm...	M
17	Zenagui Anas	anas.zenagui@gm...	M

ID


Name


Email


Genre

 Delete  Update

Search

 Add New

 Back

 Exit

- la 2^e c'est la classe product :

- package ma.fstt.ihm;

import java.awt.*;

import javax.swing.*;

import com.toedter.calendar.JDateChooser;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseAdapter;


```
import java.awt.event.MouseEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;
```

```
public class Product extends JFrame implements DBInfo,
ActionListener {
```

```
    JPanel panel;
    JLabel image;
    JLabel idLabel;
    JLabel nameLabel;
    JLabel priceLabel;
    JLabel dateLabel;
    JLabel searchLabel;
    JTextField idField;
    JTextField nameField;
    JTextField priceField;
    JTextField searchField;
    JDateChooser dateField;
    JButton updateImageButton;
    JButton insertButton;
    JButton updateButton;
    JButton deleteButton;
```

```

JButton exitButton;
JButton backButton;
JTable table;
JScrollPane tableScroller;
DefaultTableModel model;
String currentImagePath = null;
AddNewProductDialog addProductDialog;

public Product() {
    this.createAndShowGUI();
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new Product();
        }
    });
}

private void createAndShowGUI() {
    try {
        UIManager.LookAndFeelInfo[] var1 =
        UIManager.getInstalledLookAndFeels();
        int var2 = var1.length;

        for(int var3 = 0; var3 < var2; ++var3) {
            UIManager.LookAndFeelInfo info = var1[var3];
            if ("Nimbus".equals(info.getName())) {
                UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (IllegalAccessException | InstantiationException |
    UnsupportedLookAndFeelException | ClassNotFoundException var6)
    {
        ;
    }

    this.panel = new JPanel((LayoutManager)null);

```

```
this.model = new DefaultTableModel();
this.table = new JTable(this.model);
this.tableScroller = new JScrollPane(this.table);
this.image = new JLabel();
this.idLabel = new JLabel("ID");
this.nameLabel = new JLabel("Name");
this.priceLabel = new JLabel("Price");
this.dateLabel = new JLabel("Date");
this.searchLabel = new JLabel();
this.idField = new JTextField();
this.nameField = new JTextField();
this.priceField = new JTextField();
this.dateField = new JDateChooser();
this.searchField = new JTextField();
this.updateImageButton = new JButton("Update Image");
this.insertButton = new JButton("Add New", new
ImageIcon(this.getClass().getResource("/images/insert.png")));
this.updateButton = new JButton("Update", new
ImageIcon(this.getClass().getResource("/images/update.png")));
this.deleteButton = new JButton("Delete", new
ImageIcon(this.getClass().getResource("/images/delete.png")));
this.searchLabel = new JLabel("Search");
this.backButton = new JButton("Back", new
ImageIcon(this.getClass().getResource("/images/previous.png")));
this.exitButton = new JButton("Exit", new
ImageIcon(this.getClass().getResource("/images/exit.png")));
this.addProductDialog = new AddNewProductDialog(this,
this.model);
this.image.setBounds(80, 41, 270, 250);
this.updateImageButton.setBounds(150, 300, 130, 34);
this.idLabel.setBounds(20, 355, 50, 40);
this.idField.setBounds(80, 355, 270, 40);
this.nameLabel.setBounds(20, 405, 50, 40);
this.nameField.setBounds(80, 405, 270, 40);
this.priceLabel.setBounds(20, 455, 50, 40);
this.priceField.setBounds(80, 455, 270, 40);
this.dateLabel.setBounds(20, 505, 50, 40);
this.dateField.setBounds(80, 505, 270, 40);
this.deleteButton.setBounds(80, 575, 130, 40);
this.updateButton.setBounds(220, 575, 130, 40);
```

```
this.tableScroller.setBounds(377, 40, 520, 505);
this.searchField.setBounds(530, 577, 255, 36);
this.searchLabel.setBounds(460, 575, 115, 40);
this.insertButton.setBounds(920, 40, 130, 60);
this.backButton.setBounds(920, 350, 130, 40);
this.exitButton.setBounds(920, 575, 130, 40);
this.updateImageButton.setFont(new Font("Arial", 1, 14));
this.idLabel.setFont(new Font("Arial", 1, 16));
this.idField.setFont(new Font("Arial", 1, 15));
this.nameLabel.setFont(new Font("Arial", 1, 16));
this.nameField.setFont(new Font("Arial", 1, 15));
this.priceLabel.setFont(new Font("Arial", 1, 16));
this.priceField.setFont(new Font("Arial", 1, 15));
this.dateLabel.setFont(new Font("Arial", 1, 16));
this.dateField.setFont(new Font("Arial", 1, 13));
this.deleteButton.setFont(new Font("Arial", 1, 16));
this.updateButton.setFont(new Font("Arial", 1, 16));
this.insertButton.setFont(new Font("Arial", 1, 16));
this.searchField.setFont(new Font("Arial", 1, 15));
this.searchLabel.setFont(new Font("Arial", 1, 17));
this.backButton.setFont(new Font("Arial", 1, 16));
this.exitButton.setFont(new Font("Arial", 1, 16));
```

```
this.image.setBorder(BorderFactory.createLineBorder(Color.gray, 1,
true));
```

```
this.idField.setBorder(BorderFactory.createLineBorder(Color.gray, 2,
true));
```

```
this.nameField.setBorder(BorderFactory.createLineBorder(Color.gray
, 2, true));
```

```
this.priceField.setBorder(BorderFactory.createLineBorder(Color.gray,
2, true));
```

```
this.searchField.setBorder(BorderFactory.createLineBorder(Color.gra
y, 2, true));
```

```
    this.idField.setEditable(false);
```

```
    this.idField.setBackground(new Color(240, 240, 240));
```

```
    this.dateField.setDateFormatString("yyyy-MM-dd");
```

```

        this.dateField.setBackground(Color.gray);
        this.dateField.getCalendarButton().setIcon(new
ImageIcon(this.getClass().getResource("/images/calendar.png")));
        this.dateField.getCalendarButton().setBackground(Color.gray);
        this.table.setColumnSelectionAllowed(false);
        this.table.getParent().setBackground(Color.white);
        this.tableScroller.setViewportView(this.table);
        this.model.addColumn("ID");
        this.model.addColumn("Name");
        this.model.addColumn("Price ($)");
        this.model.addColumn("Date Of Add");

        try {
            this.viewProductsInTheTable();
        } catch (Exception var5) {
            ;
        }

        this.table.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent evt) {
                int index = Product.this.table.getSelectedRow();
                Product.this.showProduct(index);
                Product.this.currentImagePath = null;
            }
        });
        this.table.addKeyListener(new KeyListener() {
            public void keyReleased(KeyEvent e) {
                if (e.getKeyCode() == 38 || e.getKeyCode() == 40) {
Product.this.showProduct(Product.this.table.getSelectedRow());
                }

            }

            public void keyTyped(KeyEvent e) {
            }

            public void keyPressed(KeyEvent e) {
            }
        });

```

```
this.searchField.addKeyListener(new KeyListener() {  
    public void keyReleased(KeyEvent e) {  
        Product.this.search();  
    }  
  
    public void keyTyped(KeyEvent e) {  
    }  
  
    public void keyPressed(KeyEvent e) {  
    }  
});  
this.updateImageButton.addActionListener(this);  
this.insertButton.addActionListener(this);  
this.updateButton.addActionListener(this);  
this.deleteButton.addActionListener(this);  
this.backButton.addActionListener(this);  
this.exitButton.addActionListener(this);  
this.panel.add(this.image);  
this.panel.add(this.updateImageButton);  
this.panel.add(this.idLabel);  
this.panel.add(this.idField);  
this.panel.add(this.idField);  
this.panel.add(this.nameLabel);  
this.panel.add(this.nameField);  
this.panel.add(this.priceLabel);  
this.panel.add(this.priceField);  
this.panel.add(this.dateLabel);  
this.panel.add(this.dateField);  
this.panel.add(this.insertButton);  
this.panel.add(this.updateButton);  
this.panel.add(this.deleteButton);  
this.panel.add(this.tableScroller);  
this.panel.add(this.searchField);  
this.panel.add(this.searchLabel);  
this.panel.add(this.backButton);  
this.panel.add(this.exitButton);  
this.panel.setPreferredSize(new Dimension(1070, 640));  
this.panel.setMinimumSize(new Dimension(1070, 640));  
this.setContentPane(new JPanel(new GridBagLayout()));  
this.add(this.panel);
```

```

        this.setTitle("Pharmacy Products");
        this.setDefaultCloseOperation(3);
        this.pack();
        this.setLocationRelativeTo((Component)null);
        this.setVisible(true);
    }

    private Connection getConnection() {
        try {
            Connection con =
                DriverManager.getConnection("jdbc:mysql://localhost/gpharmacie?useUnicode=yes&characterEncoding=UTF-8", "root", "");
            return con;
        } catch (SQLException var3) {
            JOptionPane.showMessageDialog(this, var3.getMessage(),
                "Connection Error", 0);
            return null;
        }
    }

    private void viewProductsInTheTable() {
        ArrayList<ma.fstt.model.Product> productList = new
        ArrayList<>();
        Connection con = this.getConnection();
        String query = "SELECT * FROM products";

        try {
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery(query);

            while(rs.next()) {
                ma.fstt.model.Product product = new
                ma.fstt.model.Product(rs.getInt("id"), rs.getString("name"),
                Float.parseFloat(rs.getString("price")), rs.getString("add_date"),
                rs.getBytes("image"));
                productList.add(product);
            }

            con.close();
        } catch (SQLException var8) {

```

```

        JOptionPane.showMessageDialog(this, var8.getMessage(),
"Error", 0);
    }

    this.model.setRowCount(0);
    Object[] row = new Object[4];

    for(int i = 0; i < productList.size(); ++i) {
        row[0] = ((ma.fstt.model.Product)productList.get(i)).getId();
        row[1] =
((ma.fstt.model.Product)productList.get(i)).getName();
        row[2] =
((ma.fstt.model.Product)productList.get(i)).getPrice();
        row[3] =
((ma.fstt.model.Product)productList.get(i)).getAddDate();
        this.model.addRow(row);
    }

}

private boolean checkInputs() {
    if (!this.nameField.getText().equals("") &&
!this.priceField.getText().equals("") && this.dateField.getDate() !=
null) {
        try {
            Float.parseFloat(this.priceField.getText());
            return true;
        } catch (NumberFormatException var2) {
            return false;
        }
    } else {
        JOptionPane.showMessageDialog(this, "Product information
are not updated because one or more fields are empty", "Error", 0);
        return false;
    }
}

private ImageIcon resizeImage(byte[] pic) {
    ImageIcon myImage;
    if (pic == null) {

```



```
        myImage = new  
        ImageIcon(this.getClass().getResource("/images/no-image.jpg"));  
    } else {  
        myImage = new ImageIcon(pic);  
    }
```

```
        Image tempImage =  
        myImage.getImage().getScaledInstance(this.image.getWidth(),  
        this.image.getHeight(), 4);  
        return new ImageIcon(tempImage);  
    }
```

```
    private void updateImage() {  
        JFileChooser file = new JFileChooser();  
        file.setCurrentDirectory(new  
        File(System.getProperty("user.home")));  
        FileNameExtensionFilter filter = new  
        FileNameExtensionFilter("Select a .JPG .PNG .GIF image", new  
        String[]{"jpg", "png", "gif"});  
        file.setFileFilter(filter);  
        int result = file.showOpenDialog(this);  
        if (result == 0) {  
            try {  
                byte[] selectedImage =  
                Files.readAllBytes(file.getSelectedFile().toPath());  
                this.image.setIcon(this.resizeImage(selectedImage));  
                this.currentImagePath =  
                file.getSelectedFile().toPath().toString();  
            } catch (IOException var5) {  
                this.image.setIcon(this.resizeImage((byte[])null));  
            }  
        }  
    }
```

```
    private void addNewProduct() {  
        this.addProductDialog.show();  
    }
```

```
    private void updateProduct() {
```

```

        if (this.checkInputs() && this.idField.getText() != null) {
            if (this.currentImagePath != null) {
                try {
                    InputStream img = new FileInputStream(new
File(this.currentImagePath));
                    String query = "UPDATE products SET name = ?, price
= ?, add_date = ?, image = ? WHERE id = ?";
                    Connection con = this.getConnection();
                    PreparedStatement ps = con.prepareStatement(query);
                    ps.setString(1, this.nameField.getText());
                    ps.setString(2, this.priceField.getText());
                    SimpleDateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd");
                    String addDate =
dateFormat.format(this.dateField.getDate());
                    ps.setString(3, addDate);
                    ps.setBlob(4, img);
                    ps.setInt(5, Integer.parseInt(this.idField.getText()));
                    ps.executeUpdate();
                    con.close();
                    this.viewProductsInTheTable();
                    JOptionPane.showMessageDialog(this, "Product
information has been successfully updated");
                } catch (FileNotFoundException | NumberFormatException
| SQLException | HeadlessException var8) {
                    JOptionPane.showMessageDialog(this,
var8.getMessage(), "Error", 0);
                }
            } else {
                try {
                    String query = "UPDATE products SET name = ?, price
= ?, add_date = ? WHERE id = ?";
                    Connection con = this.getConnection();
                    PreparedStatement ps = con.prepareStatement(query);
                    ps.setString(1, this.nameField.getText());
                    ps.setString(2, this.priceField.getText());
                    SimpleDateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd");
                    String addDate =
dateFormat.format(this.dateField.getDate());

```

```

        ps.setString(3, addDate);
        ps.setInt(4, Integer.parseInt(this.idField.getText()));
        ps.executeUpdate();
        con.close();
        this.viewProductsInTheTable();
        JOptionPane.showMessageDialog(this, "Product
information are successfully updated");
    } catch (NumberFormatException | SQLException |
HeadlessException var7) {
        JOptionPane.showMessageDialog(this,
var7.getMessage(), "Error", 0);
    }
}

}

}

}

private ArrayList<ma.fstt.model.Product> getProductList() {
    ArrayList<ma.fstt.model.Product> productList = new
ArrayList();
    Connection con = this.getConnection();
    String query = "SELECT * FROM products";

    try {
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery(query);

        while(rs.next()) {
            ma.fstt.model.Product product = new
ma.fstt.model.Product(rs.getInt("id"), rs.getString("name"),
Float.parseFloat(rs.getString("price")), rs.getString("add_date"),
rs.getBytes("image"));
            productList.add(product);
        }

        con.close();
    } catch (SQLException var7) {
        JOptionPane.showMessageDialog(this, var7.getMessage(),
"Error", 0);
    }
}

```

```

        return productList;
    }

    private void showProduct(int index) {

        this.idField.setText(Integer.toString(((ma.fstt.model.Product)this.getProductList().get(index)).getId()));

        this.nameField.setText(((ma.fstt.model.Product)this.getProductList().get(index)).getName());

        this.priceField.setText(Float.toString(((ma.fstt.model.Product)this.getProductList().get(index)).getPrice()));

        try {
            Date addDate = (new SimpleDateFormat("yyyy-MM-dd")).parse(((ma.fstt.model.Product)this.getProductList().get(index)).getAddDate());
            this.dateField.setDate(addDate);
        } catch (ParseException var3) {
            JOptionPane.showMessageDialog(this, var3.getMessage(), "Error", 0);
        }

        byte[] theImage =
            ((ma.fstt.model.Product)this.getProductList().get(index)).getImage();
        this.image.setIcon(this.resizeImage(theImage));
    }

    private void deleteProduct() {
        if (this.table.getSelectedRow() == -1) {
            JOptionPane.showMessageDialog(this, "Please select the product that you want to delete from the table and try again");
        } else {
            try {
                Connection con = this.getConnection();
                PreparedStatement ps = con.prepareStatement("DELETE FROM products WHERE id = ?");
                int id = Integer.parseInt(this.idField.getText());
            }
        }
    }

```

```

        ps.setInt(1, id);
        ps.executeUpdate();
        con.close();
        int nextSelectedRowIndex = this.table.getSelectedRow();
        this.viewProductsInTheTable();
        if (this.table.getRowCount() == 1) {
            this.table.setRowSelectionInterval(0, 0);
            this.showProduct(0);
        } else if (this.table.getRowCount() > 1 &&
nextSelectedRowIndex < this.table.getRowCount()) {

        this.table.setRowSelectionInterval(nextSelectedRowIndex,
nextSelectedRowIndex);
            this.showProduct(nextSelectedRowIndex);
        } else if (this.table.getRowCount() > 1 &&
nextSelectedRowIndex == this.table.getRowCount()) {
            --nextSelectedRowIndex;

        this.table.setRowSelectionInterval(nextSelectedRowIndex,
nextSelectedRowIndex);
            this.showProduct(nextSelectedRowIndex);
        }

        if (this.table.getRowCount() == 0) {
            this.image.setIcon((Icon)null);
            this.idField.setText("");
            this.nameField.setText("");
            this.priceField.setText("");
            this.dateField.setDate((Date)null);
        }
    } catch (SQLException var5) {
        JOptionPane.showMessageDialog(this, var5.getMessage(),
"Error", 0);
    }
}

}

private void showPreviousPage() {

```

```

//      if (this.table.getSelectedRow() > 0) {
//          int currentSelectedRow = this.table.getSelectedRow() - 1;
//          this.table.setRowSelectionInterval(currentSelectedRow,
currentSelectedRow);
//          this.showProduct(currentSelectedRow);
//      }

    }

    private void search() {
        String keyword = this.searchField.getText();
        TableRowSorter<DefaultTableModel> tr = new
TableRowSorter(this.model);
        this.table.setRowSorter(tr);
        tr.setRowFilter(RowFilter.regexFilter(keyword, new int[0]));
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == this.updateImageButton) {
            this.updateImage();
        } else if (e.getSource() == this.insertButton) {
            this.addNewProduct();
        } else if (e.getSource() == this.updateButton) {
            this.updateProduct();
        } else if (e.getSource() == this.deleteButton) {
            this.deleteProduct();
        } else if (e.getSource() == this.backButton) {
            this.showPreviousPage();
        } else if (e.getSource() == this.exitButton) {
            System.exit(0);
        }
    }
}

```

Pharmacy Products


ID	Name	Price (\$)	Date Of Add
----	------	------------	-------------



Update Image

ID


Name


Price


Date 

 Delete  Update

Search

 Add New

 Back

 Exit

- et la 3^e c'est la classe Commande

```
package ma.fstt.ihm;  
  
import java.awt.*;  
import javax.swing.*;
```

```
import com.toedter.calendar.JDateChooser;
import ma.fstt.model.Client;
import ma.fstt.model.Product;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableRowSorter;

public class Commande extends JFrame implements DBInfo,
ActionListener {
    JPanel panel;
    JLabel image;
    JLabel idLabel;
    JLabel dateLabel;
    JLabel qtePrdLabel;
    JLabel clientLabel;
    JLabel productLabel;
    JLabel searchLabel;
```



```

    JTextField idField;
    JTextField dateField;
    JTextField qtePrdField;
    JTextField clientField;
    JTextField productField;
    JTextField searchField;
    JDateChooser date_Field;
    JButton updateImageButton;
    JButton insertButton;
    JButton updateButton;
    JButton deleteButton;
    JButton exitButton;
    JButton backButton;
    JTable table;
    JScrollPane tableScroller;
    DefaultTableModel model;
    String currentImagePath = null;
    AddNewProductDialog addProductDialog;

    public Commande() {
        this.createAndShowGUI();
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new Commande();
            }
        });
    }

    private void createAndShowGUI() {
        try {
            UIManager.LookAndFeelInfo[] var1 =
UIManager.getInstalledLookAndFeels();
            int var2 = var1.length;

            for(int var3 = 0; var3 < var2; ++var3) {
                UIManager.LookAndFeelInfo info = var1[var3];
                if ("Nimbus".equals(info.getName())) {

```

```

        UIManager.setLookAndFeel(info.getClassName());
        break;
    }
}
} catch (IllegalAccessException | InstantiationException |
UnsupportedLookAndFeelException | ClassNotFoundException var6) {
    ;
}

this.panel = new JPanel((LayoutManager)null);
this.model = new DefaultTableModel();
this.table = new JTable(this.model);
this.tableScroller = new JScrollPane(this.table);
this.image = new JLabel();
this.idLabel = new JLabel("ID");
this.productLabel = new JLabel("Product Name");
this.clientLabel = new JLabel("Client Name");
this.qtePrdLabel = new JLabel("QTE");
this.dateLabel = new JLabel("Date");
this.searchLabel = new JLabel();
this.idField = new JTextField();
this.productField = new JTextField();
this.clientField = new JTextField();
this.qtePrdField = new JTextField();
this.date_Field = new JDateChooser();
this.searchField = new JTextField();
this.updateImageButton = new JButton("Update Image");
this.insertButton = new JButton("Add New", new
ImageIcon(this.getClass().getResource("/images/insert.png")));
this.updateButton = new JButton("Update", new
ImageIcon(this.getClass().getResource("/images/update.png")));
this.deleteButton = new JButton("Delete", new
ImageIcon(this.getClass().getResource("/images/delete.png")));
this.searchLabel = new JLabel("Search");
this.backButton = new JButton("Back", new
ImageIcon(this.getClass().getResource("/images/previous.png")));
this.exitButton = new JButton("Exit", new
ImageIcon(this.getClass().getResource("/images/exit.png")));
//      this.addProductDialog = new AddNewProductDialog(this,
this.model);

```

```
this.image.setBounds(80, 41, 270, 250);
this.updateImageButton.setBounds(150, 300, 130, 34);
this.idLabel.setBounds(20, 355, 50, 40);
this.idField.setBounds(80, 355, 270, 40);
this.productLabel.setBounds(20, 405, 50, 40);
this.productField.setBounds(80, 405, 270, 40);
this.clientLabel.setBounds(20, 455, 50, 40);
this.clientField.setBounds(80, 455, 270, 40);
this.qtePrdLabel.setBounds(20, 455, 50, 40);
this.qtePrdField.setBounds(80, 455, 270, 40);
this.dateLabel.setBounds(20, 505, 50, 40);
this.deleteButton.setBounds(80, 575, 130, 40);
this.updateButton.setBounds(220, 575, 130, 40);
this.tableScroller.setBounds(377, 40, 520, 505);
this.searchField.setBounds(530, 577, 255, 36);
this.searchLabel.setBounds(460, 575, 115, 40);
this.insertButton.setBounds(920, 40, 130, 60);
this.backButton.setBounds(920, 350, 130, 40);
this.exitButton.setBounds(920, 575, 130, 40);
this.updateImageButton.setFont(new Font("Arial", 1, 14));
this.idLabel.setFont(new Font("Arial", 1, 16));
this.idField.setFont(new Font("Arial", 1, 15));
this.productLabel.setFont(new Font("Arial", 1, 16));
this.productField.setFont(new Font("Arial", 1, 15));
this.clientLabel.setFont(new Font("Arial", 1, 16));
this.clientField.setFont(new Font("Arial", 1, 15));
this.qtePrdLabel.setFont(new Font("Arial", 1, 16));
this.qtePrdField.setFont(new Font("Arial", 1, 15));
this.dateLabel.setFont(new Font("Arial", 1, 16));
this.deleteButton.setFont(new Font("Arial", 1, 16));
this.updateButton.setFont(new Font("Arial", 1, 16));
this.insertButton.setFont(new Font("Arial", 1, 16));
this.searchField.setFont(new Font("Arial", 1, 15));
this.searchLabel.setFont(new Font("Arial", 1, 17));
this.backButton.setFont(new Font("Arial", 1, 16));
this.exitButton.setFont(new Font("Arial", 1, 16));
this.image.setBorder(BorderFactory.createLineBorder(Color.gray, 1,
true));
this.idField.setBorder(BorderFactory.createLineBorder(Color.gray, 2,
true));
```

```
this.productField.setBorder(BorderFactory.createLineBorder(Color.gray, 2,
true));
    this.clientField.setBorder(BorderFactory.createLineBorder(Color.gray,
2, true));

this.qtePrdField.setBorder(BorderFactory.createLineBorder(Color.gray, 2,
true));

this.searchField.setBorder(BorderFactory.createLineBorder(Color.gray, 2,
true));
    this.idField.setEditable(false);
    this.idField.setBackground(new Color(240, 240, 240));
    this.table.setColumnSelectionAllowed(false);
    this.table.getParent().setBackground(Color.white);
    this.tableScroller.setViewPortView(this.table);
    this.model.addColumn("ID");
    this.model.addColumn("Name");
    this.model.addColumn("Price ($)");
    this.model.addColumn("Date Of Add");

    try {
        this.viewCommandesInTheTable();
    } catch (Exception var5) {
        ;
    }

    this.table.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent evt) {
            int index = Commande.this.table.getSelectedRow();
            Commande.this.currentImagePath = null;
        }
    });
    this.table.addKeyListener(new KeyListener() {
        public void keyReleased(KeyEvent e) {
            if (e.getKeyCode() == 38 || e.getKeyCode() == 40) {
            }

        }
    })
```

```
public void keyTyped(KeyEvent e) {
}

public void keyPressed(KeyEvent e) {
}
});
this.searchField.addKeyListener(new KeyListener() {
    public void keyReleased(KeyEvent e) {
        Commande.this.search();
    }

    public void keyTyped(KeyEvent e) {
    }

    public void keyPressed(KeyEvent e) {
    }
});
this.updateImageButton.addActionListener(this);
this.insertButton.addActionListener(this);
this.updateButton.addActionListener(this);
this.deleteButton.addActionListener(this);
this.backButton.addActionListener(this);
this.exitButton.addActionListener(this);
this.panel.add(this.image);
this.panel.add(this.updateImageButton);
this.panel.add(this.idLabel);
this.panel.add(this.idField);
this.panel.add(this.productLabel);
this.panel.add(this.productField);
this.panel.add(this.clientLabel);
this.panel.add(this.clientField);
this.panel.add(this.qtePrdLabel);
this.panel.add(this.qtePrdField);
this.panel.add(this.insertButton);
this.panel.add(this.updateButton);
this.panel.add(this.deleteButton);
this.panel.add(this.tableScroller);
this.panel.add(this.searchField);
this.panel.add(this.searchLabel);
this.panel.add(this.backButton);
```

```

        this.panel.add(this.exitButton);
        this.panel.setPreferredSize(new Dimension(1070, 640));
        this.panel.setMinimumSize(new Dimension(1070, 640));
        this.setContentPane(new JPanel(new GridBagLayout()));
        this.add(this.panel);
        this.setTitle("Pharmacy Orders");
        this.setDefaultCloseOperation(3);
        this.pack();
        this.setLocationRelativeTo((Component)null);
        this.setVisible(true);
    }

    private Connection getConnection() {
        try {
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/gpharmacie?useUnicode=yes&characterEncoding=UTF-8", "root", "");
            return con;
        } catch (SQLException var3) {
            JOptionPane.showMessageDialog(this, var3.getMessage(),
"Connection Error", 0);
            return null;
        }
    }

    private void viewCommandesInTheTable() {
        ArrayList<ma.fstt.model.Commande> cmdList = new ArrayList<>();
        Connection con = this.getConnection();
        String query = "SELECT * FROM commande";

        try {
            Statement st = con.createStatement();
            ResultSet rs = st.executeQuery(query);

            while(rs.next()) {
                ma.fstt.model.Commande cmd = new
ma.fstt.model.Commande(rs.getInt("id_commande"), rs.getString("date"),
rs.getInt("qte"), rs.getInt("id_client"), rs.getInt("id_prod"));
                cmdList.add(cmd);
            }
        }
    }

```

```

        con.close();
    } catch (SQLException var8) {
        JOptionPane.showMessageDialog(this, var8.getMessage(), "Error",
0);
    }

    this.model.setRowCount(0);
    Object[] row = new Object[5];

    for(int i = 0; i < cmdList.size(); ++i) {
        row[0] =
((ma.fstt.model.Commande)cmdList.get(i)).getId_commande();
        row[1] = ((ma.fstt.model.Commande)cmdList.get(i)).getDate();
        row[2] = ((ma.fstt.model.Commande)cmdList.get(i)).getQtePrd();
        row[3] = ((ma.fstt.model.Commande)cmdList.get(i)).getProduct();
        row[4] = ((ma.fstt.model.Commande)cmdList.get(i)).getClient();
        this.model.addRow(row);
    }

}

private boolean checkInputs() {
    if (!this.clientField.getText().equals("") &&
!this.productField.getText().equals("") && this.dateField.getText() != null)
{
        try {
            Integer.parseInt(this.qtePrdField.getText());
            return true;
        } catch (NumberFormatException var2) {
            return false;
        }
    } else {
        JOptionPane.showMessageDialog(this, "Commande information are
not updated because one or more fields are empty", "Error", 0);
        return false;
    }
}

private ImageIcon resizeImage(byte[] pic) {

```

```

        ImageIcon myImage;
        if (pic == null) {
            myImage = new
ImageIcon(this.getClass().getResource("/images/no-image.jpg"));
        } else {
            myImage = new ImageIcon(pic);
        }

        Image tempImage =
myImage.getImage().getScaledInstance(this.image.getWidth(),
this.image.getHeight(), 4);
        return new ImageIcon(tempImage);
    }

    private void addNewProduct() {
        this.addProductDialog.show();
    }

    private void updateProduct() {
        if (this.checkInputs() && this.idField.getText() != null) {
            if (this.currentImagePath != null) {
                try {
                    InputStream img = new FileInputStream(new
File(this.currentImagePath));
                    String query = "UPDATE commande SET qte = ?, date = ?,
id_client = ?, id_prod = ? WHERE id = ?";
                    Connection con = this.getConnection();
                    PreparedStatement ps = con.prepareStatement(query);
                    ps.setString(1, this.qtePrdField.getText());
//                    ps.setString(2, this.dateField.getText());
                    ps.setString(3, this.clientField.getText());
                    ps.setString(4, this.productField.getText());
                    SimpleDateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd");
                    String addDate = dateFormat.format(this.dateField.getText());
                    ps.setString(2, addDate);
                    ps.executeUpdate();
                    con.close();
                    this.viewCommandesInTheTable();
                }
            }
        }
    }

```



```

        JOptionPane.showMessageDialog(this, "Commande
information has been successfully updated");
    } catch (FileNotFoundException | NumberFormatException |
SQLException | HeadlessException var8) {
        JOptionPane.showMessageDialog(this, var8.getMessage(),
"Error", 0);
    }
    } else {
        try {
            String query = "UPDATE commande SET qte = ?, date = ?,
id_client = ?, id_prod = ? WHERE id = ?";
            Connection con = this.getConnection();
            PreparedStatement ps = con.prepareStatement(query);
            ps.setString(1, this.qtePrdField.getText());
            // ps.setString(2, this.dateField.getText());
            ps.setString(3, this.clientField.getText());
            ps.setString(4, this.productField.getText());
            SimpleDateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd");
            String addDate = dateFormat.format(this.dateField.getText());
            ps.setString(2, addDate);
            ps.executeUpdate();
            con.close();
            this.viewCommandesInTheTable();
            JOptionPane.showMessageDialog(this, "Commande
information are successfully updated");
        } catch (NumberFormatException | SQLException |
HeadlessException var7) {
            JOptionPane.showMessageDialog(this, var7.getMessage(),
"Error", 0);
        }
    }
}

private ArrayList<ma.fstt.model.Commande> getCommandeList() {
    ArrayList<ma.fstt.model.Commande> cmdList = new ArrayList();
    Connection con = this.getConnection();
    String query = "SELECT * FROM commande";

```

```

try {
    Statement st = con.createStatement();
    ResultSet rs = st.executeQuery(query);

    while(rs.next()) {
        ma.fstt.model.Commande cmd = new
ma.fstt.model.Commande(rs.getInt("id_commande"), rs.getString("date"),
rs.getInt("qte"), rs.getInt("id_client"), rs.getInt("id_prod"));
        cmdList.add(cmd);
    }

    con.close();
} catch (SQLException var7) {
    JOptionPane.showMessageDialog(this, var7.getMessage(), "Error",
0);
}

return cmdList;
}

private void deleteProduct() {
    if (this.table.getSelectedRow() == -1) {
        JOptionPane.showMessageDialog(this, "Please select the
Commande that you want to delete from the table and try again");
    } else {
        try {
            Connection con = this.getConnection();
            PreparedStatement ps = con.prepareStatement("DELETE FROM
commande WHERE id = ?");
            int id = Integer.parseInt(this.idField.getText());
            ps.setInt(1, id);
            ps.executeUpdate();
            con.close();
            int nextSelectedRowIndex = this.table.getSelectedRow();
            this.viewCommandesInTheTable();
            if (this.table.getRowCount() == 1) {
                this.table.setRowSelectionInterval(0, 0);
            } else if (this.table.getRowCount() > 1 &&

```

```

nextSelectedRowIndex < this.table.getRowCount()) {
    this.table.setRowSelectionInterval(nextSelectedRowIndex,
nextSelectedRowIndex);
} else if (this.table.getRowCount() > 1 &&
nextSelectedRowIndex == this.table.getRowCount()) {
    --nextSelectedRowIndex;
    this.table.setRowSelectionInterval(nextSelectedRowIndex,
nextSelectedRowIndex);
}

} catch (SQLException var5) {
    JOptionPane.showMessageDialog(this, var5.getMessage(),
"Error", 0);
}
}
}

```

```

private void showPreviousPage() {
//    if (this.table.getSelectedRow() > 0) {
//        int currentSelectedRow = this.table.getSelectedRow() - 1;
//        this.table.setRowSelectionInterval(currentSelectedRow,
currentSelectedRow);
//        this.showProduct(currentSelectedRow);
//    }
}

```

```

private void search() {
    String keyword = this.searchField.getText();
    TableRowSorter<DefaultTableModel> tr = new
TableRowSorter(this.model);
    this.table.setRowSorter(tr);
    tr.setRowFilter(RowFilter.regexFilter(keyword, new int[0]));
}

```

```














































public void actionPerformed(ActionEvent e) {
    if (e.getSource() == this.insertButton) {
        this.addNewProduct();
    }
}

```




```
    } else if (e.getSource() == this.updateButton) {  
        this.updateProduct();  
    } else if (e.getSource() == this.deleteButton) {  
        this.deleteProduct();  
    } else if (e.getSource() == this.backButton) {  
        this.showPreviousPage();  
    } else if (e.getSource() == this.exitButton) {  
        System.exit(0);  
    }  
}  
}
```

Partie 4 : Créations des BDs :

- Pour la classe Client:

← T →				id_client	nom	email	genre
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	3	Mohamed	mohamed@fstt.ma	M
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	4	Liela	dfdf	F
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	5	M	yassine@gmail.com	NULL
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	6	Test	Test@gmail.com	M
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	7	Moahmed	mohamed@gmail.com	M
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	8	toto	toto@gmail.com	F
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	9	yyyy	yyyy@yyy.com	F
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	10	LST	LST@yyy.com	M
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	11	sjdfjsd	toto@gmail.com	F
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	12	Lotfi	lotfi@gmail.com	H
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	13	Zenagui Anas	anas.zenagui@gmail.ma	M
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	14	Zenagui Anas	anas.zenagui@gmail.ma	M
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	15	Zenagui Anas	anas.zenagui@gmail.ma	M
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	16	Zenagui Anas	anas.zenagui@gmail.ma	M
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	17	Zenagui Anas	anas.zenagui@gmail.ma	M

- Pour la classe Commande:

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
<input type="checkbox"/>	1	id_commande 	int(11)			Non	Aucun(e)		AUTO_INCREMENT
<input type="checkbox"/>	2	date	varchar(255)	utf8mb4_general_ci		Non	Aucun(e)		
<input type="checkbox"/>	3	qte	int(100)			Non	Aucun(e)		
<input type="checkbox"/>	4	id_client 	int(11)			Non	Aucun(e)		
<input type="checkbox"/>	5	id_prod 	int(11)			Non	Aucun(e)		

- Pour la classe Product:

+ Options