

Electricity Billing System

Lecturer

ANAND SIR

RAJATHI MA'AM

The team

ARUN KUMAR SAH

CH.EN.U4CSE21266

ADARSH KARTHIKEYAN

CH.EN.U4CSE21201

ANIRUDDHA MUKHERJEE

CH.EN.U4CSE21233

BINOD MANDAL

CH.U4CSE21208

Abstract

The purpose of electricity billing system is to automate the already existing manual system by the help of computerized equipments and full-fledged computer software, fulfilling their requirement so that their valuable data can be stored for a longer period with easy accessing and manipulation of the same

Abstract

Electricity Billing System, as described above can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on record keeping. Thus, it will help the organization in better utilization of resources. the organization can maintain computerized records without redundant entries. thus, one need not be distracted by the information that is not relevant, while being able to reach the information.

Electric Billing System

Manager

Customer

Billing and Payment

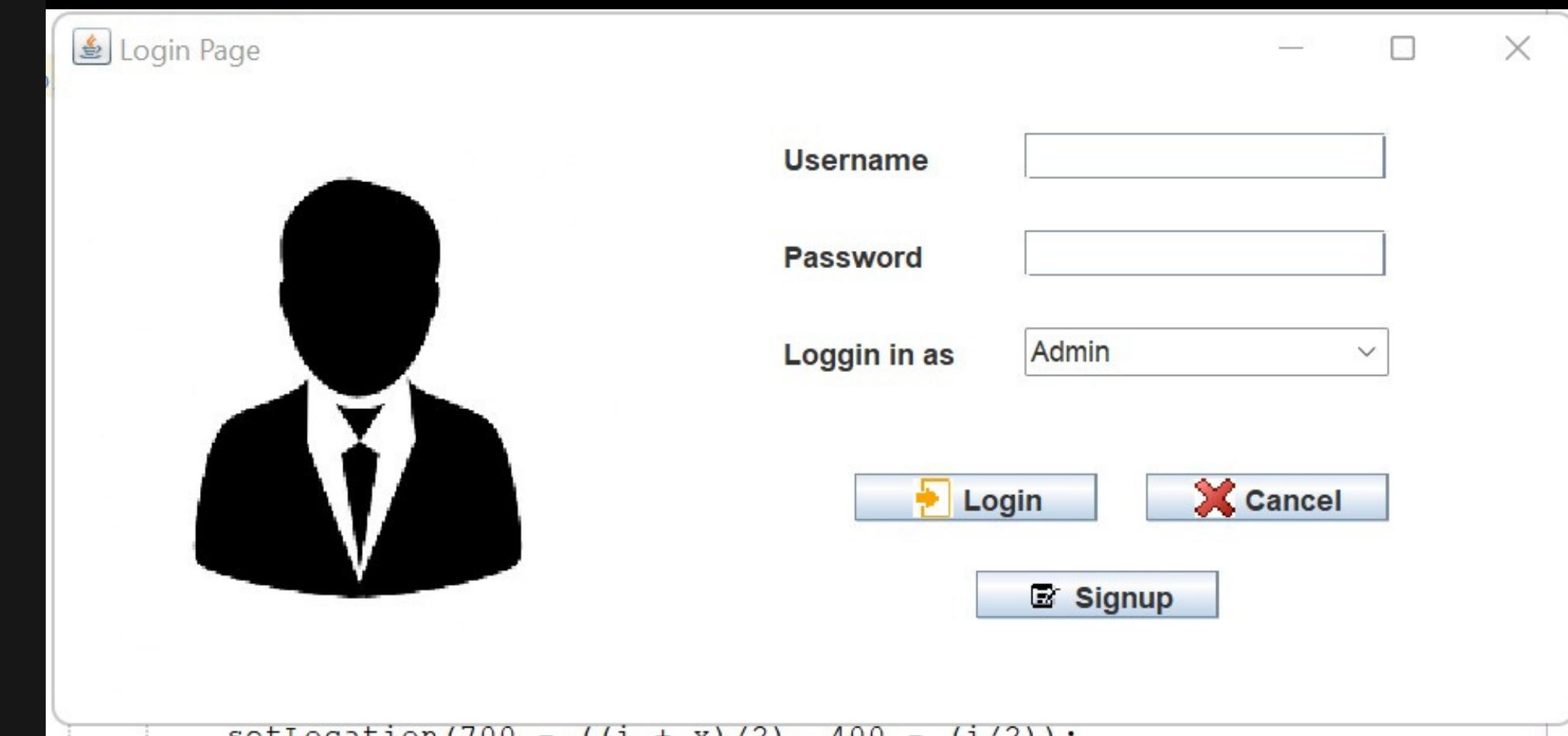
Extra Utilities

Information Management

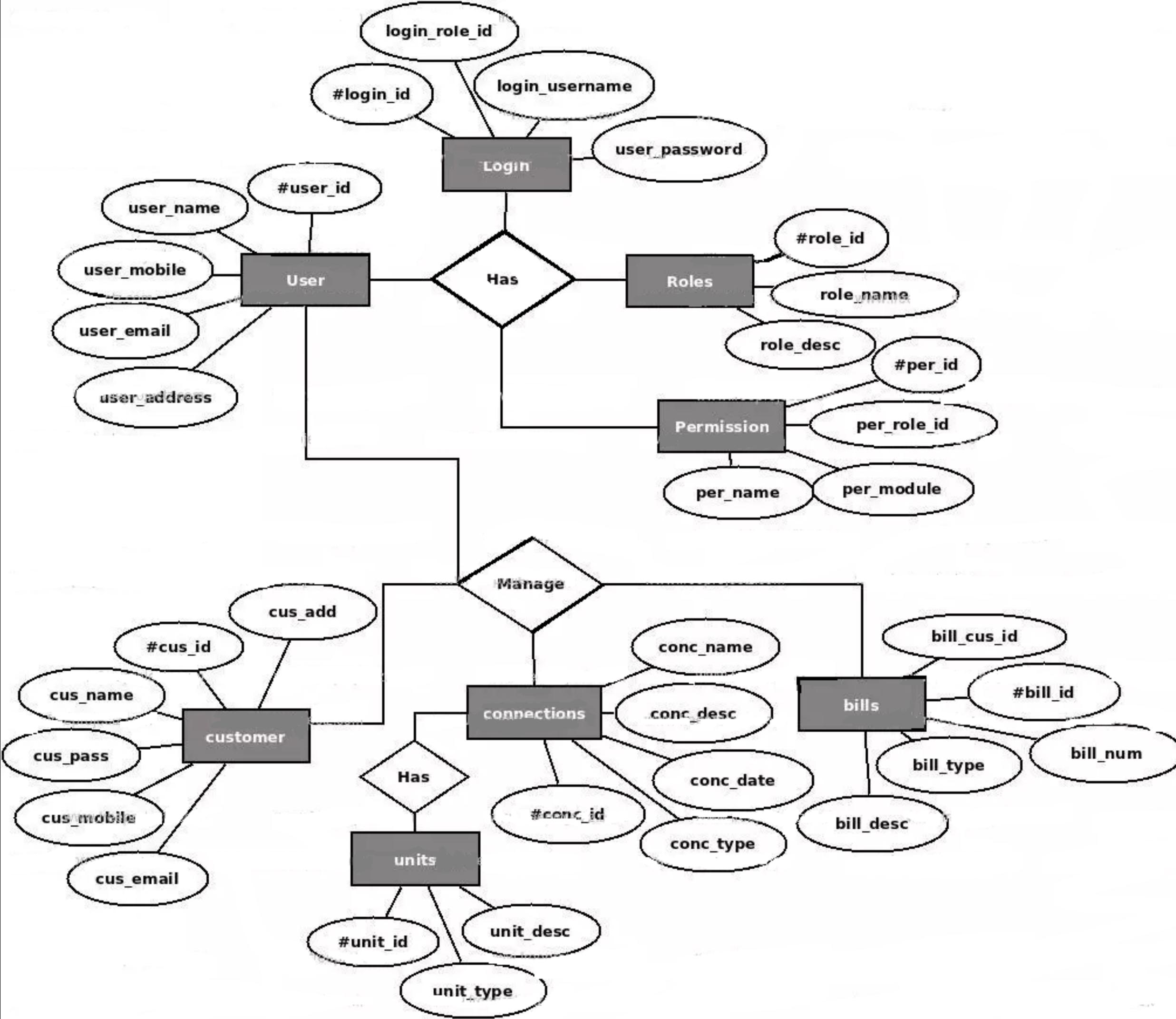
SQL Server Connect

Objective

To Build a GUI connected to SQL that can store, update, delete data as well as manage access to the entire database

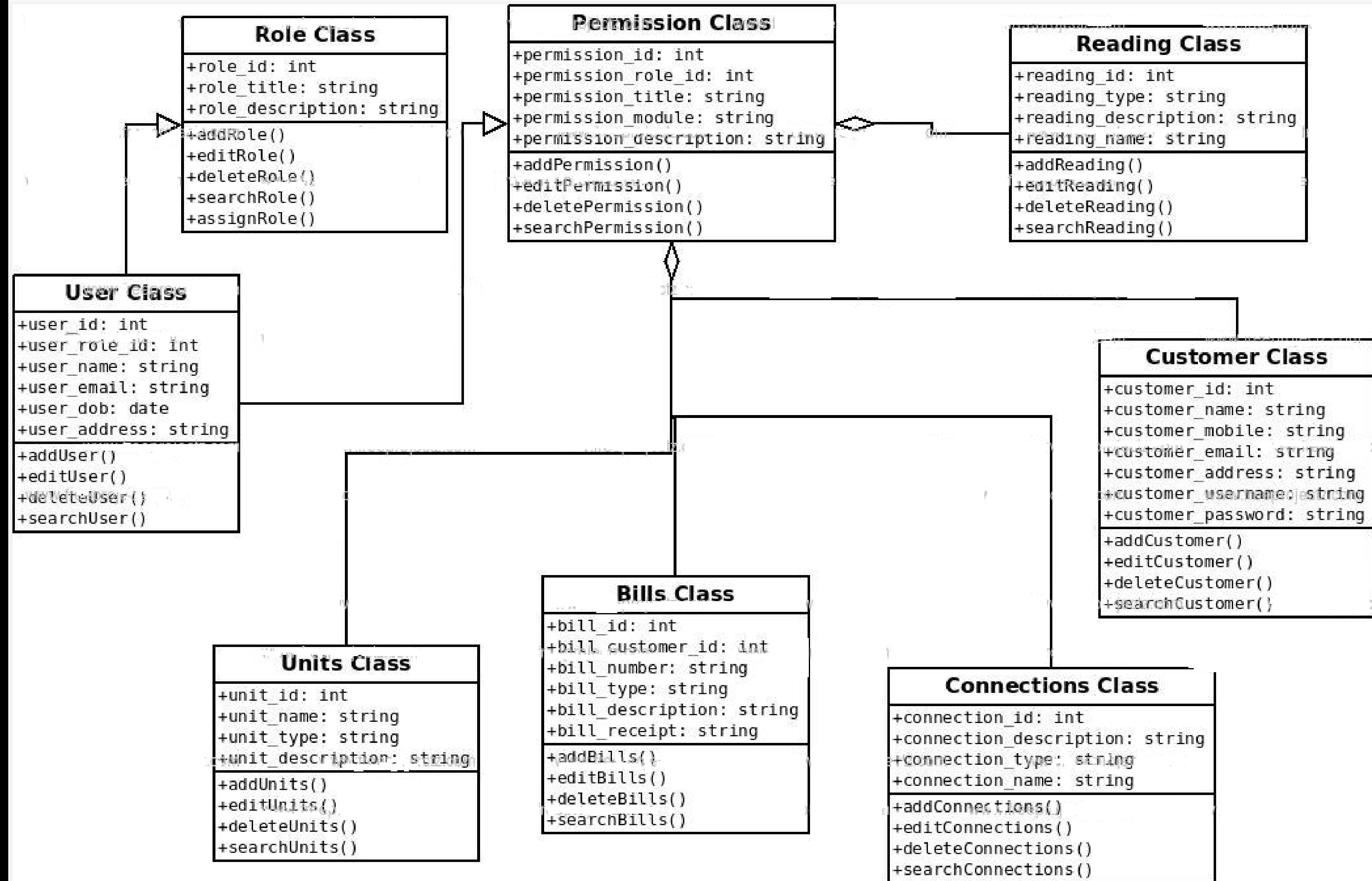


ER Diagram



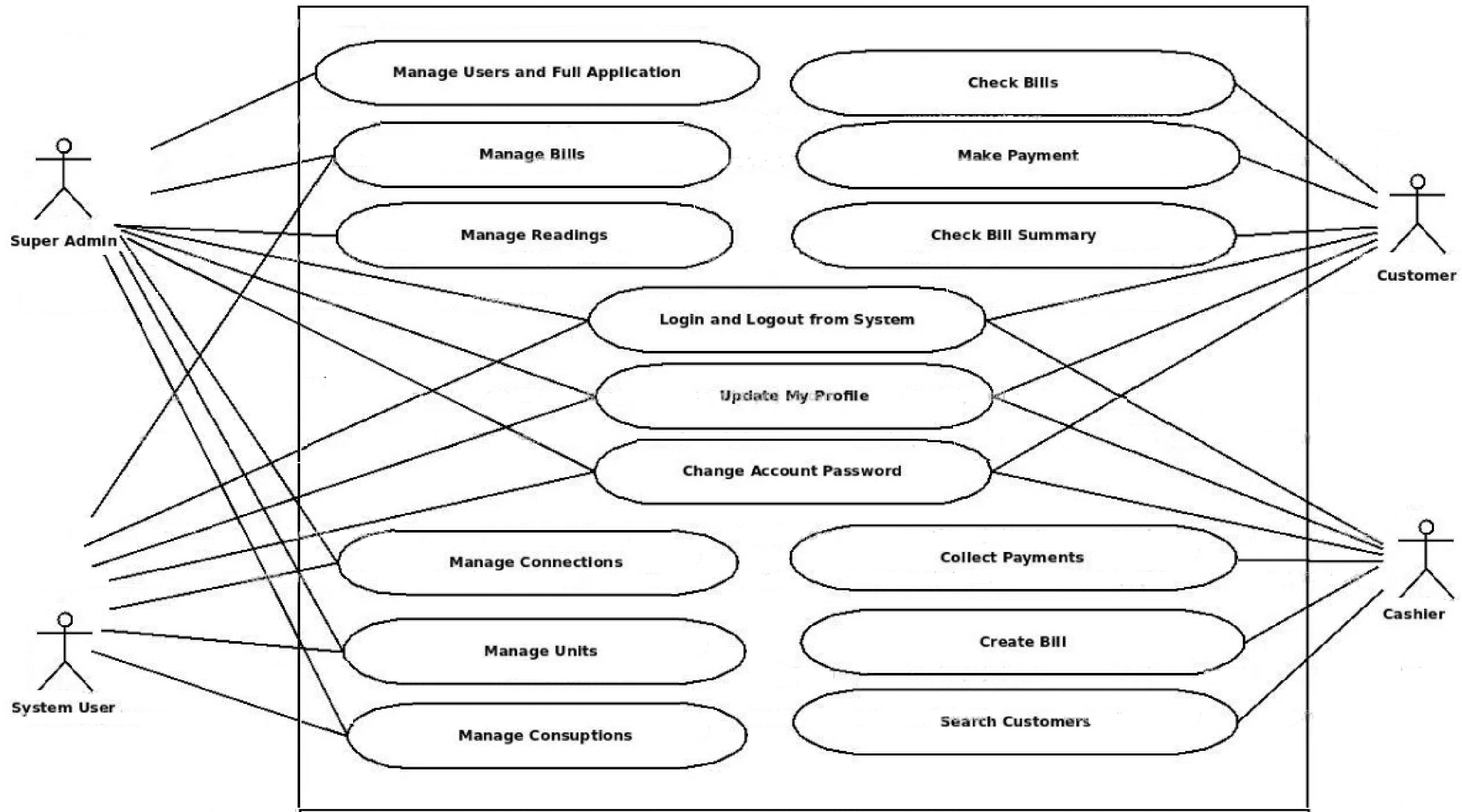
UML Diagrams

Class Diagram

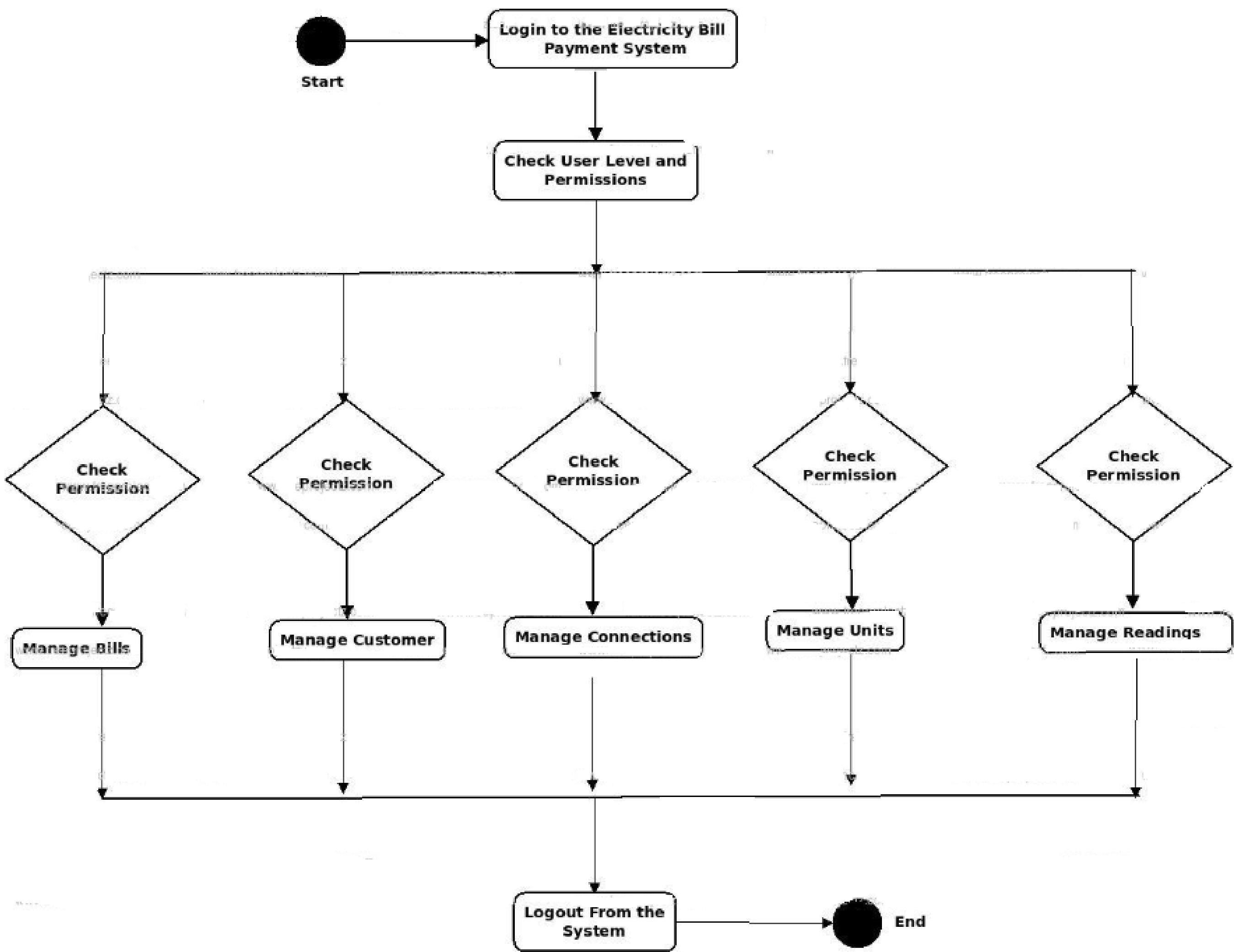


Use Case Diagram

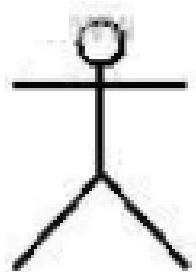
Use Case Diagram of Electricity Bill Payment System :



Activity Diagram



Sequence Diagram



Admin



Sequence flow: Admin → Login to Page (solid arrow) → Fogot Password (rectangle) → Check Login Details (rectangle).

Fogot Password

Check Login Details

Check Authenticity for access

Provide Authorization for access

Valid Login Details

Check Security
Question and
Answer

Send email to user to reset Password

Create Session
in and store
in Database

Allow user to
Access the Internal
Pages on base of
User Token and
Session

Invalid Login Details

Allow User to Access the Pages

Logout from application

Login Successfully

Destroy Session
and tokens
from database

The screenshot of the codes.

Splash Java

```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public class Splash extends JFrame implements Runnable {
7     Thread t;
8
9     Splash() {
10
11         ImageIcon i1 = new ImageIcon(location: ClassLoader.getSystemResource(name: "icon/elect.jpg"));
12         Image i2 = i1.getImage().getScaledInstance(width: 730, height: 550, hints: Image.SCALE_DEFAULT);
13         ImageIcon i3 = new ImageIcon(image: i2);
14         JLabel image = new JLabel(image: i3);
15         add(comp: image);
16
17         setVisible(b: true);
18
19         int x = 1;
20         for (int i = 2; i < 600; i+=4, x+=1) {
21             setSize(i + x, height: i);
22             setLocation(700 - ((i + x)/2), 400 - (i/2));
23             try {
24                 Thread.sleep(millis: 5);
25             } catch (Exception e) {
26                 e.printStackTrace();
27             }
28         }
29     }
30
31     public void run() {
32         while (true) {
33             if (t.isAlive())
34                 break;
35         }
36         dispose();
37     }
38 }
```

The screenshot of the codes.

Sign Up Java

A screenshot of a Java Integrated Development Environment (IDE) showing the code for a Java Swing application named "Signup". The code is part of the "electricity.billing.system" package and implements the ActionListener interface. It defines several components including JButton, Choice, and JTextField, and sets up a JPanel for the main window. The code uses various Java Swing and AWT libraries for its functionality.

```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import javax.swing.border.*;
5 import java.awt.*;
6 import java.awt.event.*;
7 import java.sql.*;
8
9 public class Signup extends JFrame implements ActionListener{
10
11     JButton create, back;
12     Choice accountType;
13     JTextField meter, username, name, password;
14     Signup(){
15
16         setBounds( x: 450, y:150, width:700, height:400);
17         getContentPane().setBackground( c:Color.WHITE );
18         setLayout( manager: null );
19
20         JPanel panel = new JPanel();
21         panel.setBounds( x:30, y:30, width:650, height:300 );
22         panel.setBorder( new TitledBorder( new LineBorder( new Color( r:173, g:216, b:230 ), thickness: 2 ), title: "Create-Account" ) );
23         panel.setBackground( bg:Color.WHITE );
24         panel.setLayout( mgr: null );
25         panel.setForeground( new Color( r:34, g:139, b:34 ) );
26         add( comp:panel );
27 }
```

The screenshot of the codes.

Project Java

The screenshot shows a Java IDE interface with the title "Project Java". The top menu bar has tabs for "Source" and "History", along with various tool icons. Below the menu is a toolbar with icons for file operations like open, save, and search. The main area displays the source code for "Project.java". The code is as follows:

```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.*;
6
7 public class Project extends JFrame implements ActionListener{
8
9     String atype, meter;
10    Project(String atype, String meter) {
11        this.atype = atype;
12        this.meter = meter;
13        setExtendedState( state(JFrame.MAXIMIZED_BOTH);
14
15        ImageIcon i1 = new ImageIcon( location:ClassLoader.getSystemResource( name: "icon/elect1.jpg"));
16        Image i2 = i1.getImage().getScaledInstance( width:1550, height: 850, hints:Image.SCALE_DEFAULT);
17        ImageIcon i3 = new ImageIcon( image:i2);
18        JLabel image = new JLabel( image:i3);
19        add( comp:image);
20
21        JMenuBar mb = new JMenuBar();
22        setJMenuBar( menubar:mb);
23
24        JMenu master = new JMenu( s:"Master");
25        master.setForeground( fg:Color.BLUE);
```

The screenshot of the codes.

Update Java

A screenshot of a Java Integrated Development Environment (IDE) showing the source code for a Java class named `UpdateInformation`. The code is part of the `electricity.billing.system` package and implements the `ActionListener` interface. The code defines several fields for customer information and sets up the window's appearance and layout. It includes labels for the heading, name, address, state, city, email, and phone number, along with buttons for update and cancel.

```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6 import java.awt.event.*;
7
8 public class UpdateInformation extends JFrame implements ActionListener{
9
10    JTextField tfaddress, tfstate, tfcity, tfemail, tfphone;
11    JButton update, cancel;
12    String meter;
13    JLabel name;
14    UpdateInformation(String meter) {
15        this.meter = meter;
16        setBounds( x: 300, y:150, width:1050, height:450);
17        getContentPane().setBackground( c:Color.WHITE );
18        setLayout( manager: null );
19
20        JLabel heading = new JLabel( text:"UPDATE CUSTOMER INFORMATION");
21        heading.setBounds( x:110, y:0, width:400, height:30);
22        heading.setFont( new Font( name: "Tahoma", style:Font.PLAIN, size:20));
23        add( comp:heading );
24
25        JLabel lblname = new JLabel( text:"Name");
26        lblname.setBounds( x:30, y:70, width:100, height:20);
27        add( comp:lblname );
```

The screenshot of the codes.

View Java

A screenshot of a Java IDE interface. The title bar says "View Java". The menu bar has "File", "Edit", "Source", "History", and "Help". The toolbar includes icons for opening files, saving, undo, redo, cut, copy, paste, find, search, and others. The tab bar shows multiple open files: "CustomerDetails.java X", "Splash.java X", "ViewInformation.java X" (which is the active tab), "Login.java X", and "UpdateInfor...". The main area is a code editor with the following Java code:

```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6 import java.awt.event.*;
7
8 public class ViewInformation extends JFrame implements ActionListener{
9
10    JButton cancel;
11    ViewInformation(String meter) {
12        setBounds( x: 350, y: 150, width: 850, height: 650);
13        getContentPane().setBackground( c:Color.WHITE );
14        setLayout( manager: null );
15
16
17        JLabel heading = new JLabel( text:"VIEW CUSTOMER INFORMATION");
18        heading.setBounds( x: 250, y:0, width: 500, height: 40);
19        heading.setFont( new Font( name: "Tahoma", style:Font.PLAIN, size:20));
20        add( comp:heading);
21
22        JLabel lblname = new JLabel( text:"Name");
23        lblname.setBounds( x: 70, y:80, width:100, height:20);
24        add( comp:lblname);
```

The screenshot of the codes.

Pay Bill Java

A screenshot of a Java Integrated Development Environment (IDE) showing the code for a Java application named 'Pay Bill Java'. The code is written in Java and defines a class 'PayBill' that extends 'JFrame' and implements 'ActionListener'. The code includes imports for javax.swing, java.awt, java.sql, and java.awt.event, and initializes components like a choice menu for months, two buttons for pay and back, and labels for meter number and electricity bill. The code uses setLayout(null) and setBounds to define the window's dimensions and position.

```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6 import java.awt.event.*;
7
8 public class PayBill extends JFrame implements ActionListener{
9
10    Choice cmonth;
11    JButton pay, back;
12    String meter;
13    PayBill(String meter) {
14        this.meter = meter;
15        setLayout(null);
16        setBounds( x: 300, y:150, width:900, height:600 );
17
18        JLabel heading = new JLabel( text: "Electricity Bill" );
19        heading.setFont( new Font( name: "Tahoma", style:Font.BOLD, size:24 ) );
20        heading.setBounds( x: 120, y:5, width:400, height:30 );
21        add( comp:heading );
22
23        JLabel lblmeternumber = new JLabel( text: "Meter Number" );
24        lblmeternumber.setBounds( x: 35, y:80, width:200, height:20 );
25        add( comp:lblmeternumber );
26
27        pay = new JButton( text: "Pay" );
28        pay.setBounds( x: 250, y:150, width:100, height:40 );
29        pay.addActionListener( this );
30        add( comp:pay );
31
32        back = new JButton( text: "Back" );
33        back.setBounds( x: 350, y:150, width:100, height:40 );
34        back.addActionListener( this );
35        add( comp:back );
36
37    }
38
39    public void actionPerformed( ActionEvent e ) {
40        if( e.getSource() == pay ) {
41            System.out.println( "Pay button clicked" );
42        }
43        if( e.getSource() == back ) {
44            System.out.println( "Back button clicked" );
45        }
46    }
47}
```

The screenshot of the codes.

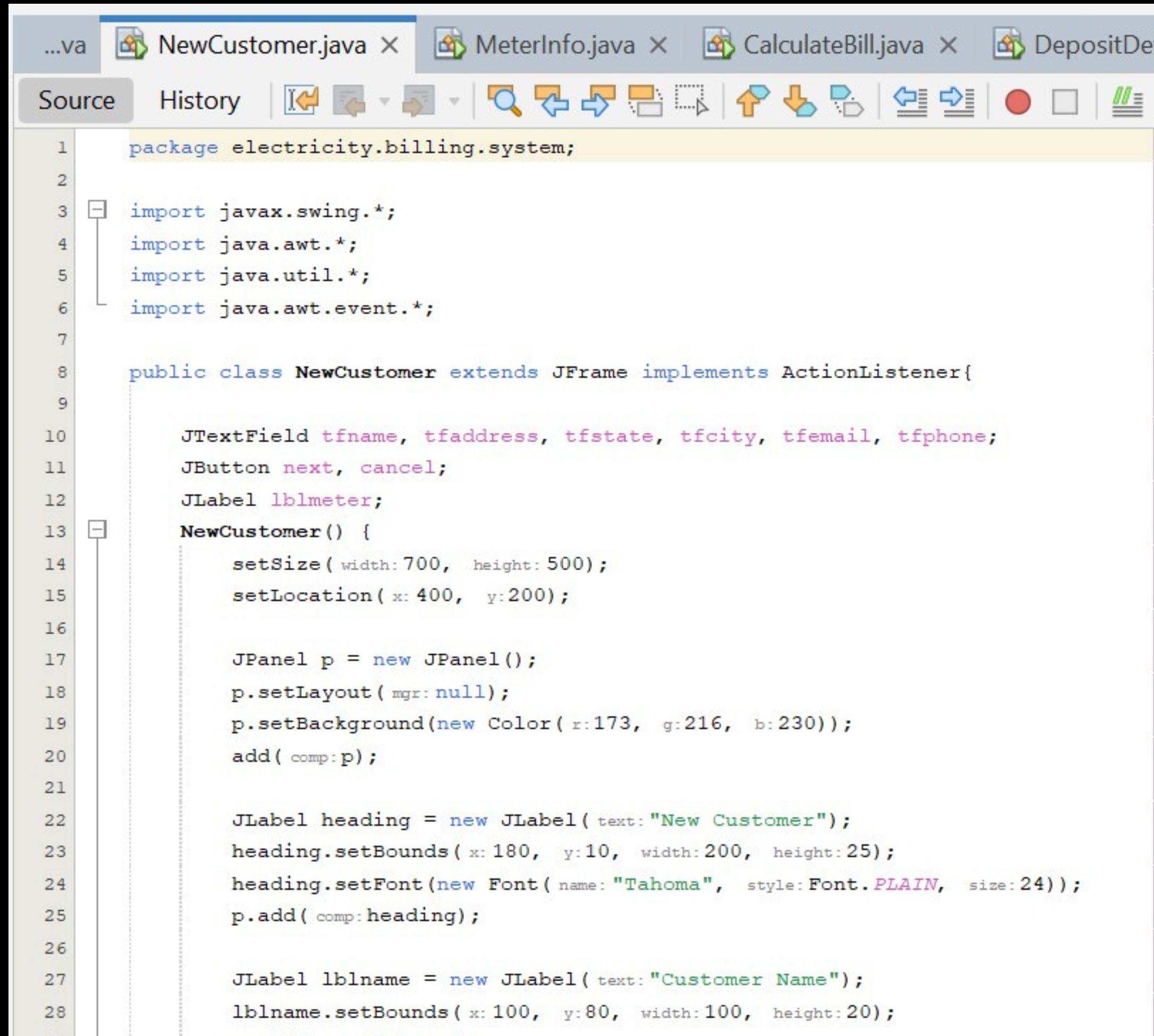
Paytm Java

A screenshot of a Java Integrated Development Environment (IDE) showing the code for a Java application named `Paytm`. The code is part of the `electricity.billing.system` package and implements the `ActionListener` interface. It uses `JFrame`, `JButton`, and `JEditorPane` components from the Java Swing library. The code attempts to load a URL from Paytm's online payments page into a `JEditorPane` component. The IDE interface includes tabs for other files like `UpdateInformation.java`, `BillDetails.java`, and `PayBill.java`, and various toolbars and status bars.

```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.*;
6
7 public class Paytm extends JFrame implements ActionListener{
8
9     String meter;
10    JButton back;
11    Paytm(String meter) {
12        this.meter = meter;
13
14        JEditorPane j = new JEditorPane();
15        j.setEditable( b: false );
16
17        try {
18            j.setPage( url: "https://paytm.com/online-payments" );
19        } catch (Exception e) {
20            j.setContentType( type: "text/html" );
21            j.setText( t: "<html>Could not load</html>" );
22
23        }
24    }
25}
```

The screenshot of the codes.

Add Customer Java



```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.util.*;
6 import java.awt.event.*;
7
8 public class NewCustomer extends JFrame implements ActionListener{
9
10    JTextField tfname, tfaddress, tfstate, tfcity, tfemail, tfphone;
11    JButton next, cancel;
12    JLabel lblmeter;
13    NewCustomer() {
14        setSize( width: 700, height: 500 );
15        setLocation( x: 400, y:200 );
16
17        JPanel p = new JPanel();
18        p.setLayout( mgr: null );
19        p.setBackground( new Color( r:173, g:216, b:230 ) );
20        add( comp:p );
21
22        JLabel heading = new JLabel( text:"New Customer" );
23        heading.setBounds( x: 180, y:10, width:200, height:25 );
24        heading.setFont( new Font( name: "Tahoma", style:Font.PLAIN, size:24 ) );
25        p.add( comp:heading );
26
27        JLabel lblname = new JLabel( text:"Customer Name" );
28        lblname.setBounds( x: 100, y:80, width:100, height:20 );
```

The screenshot of the codes.

Login Java

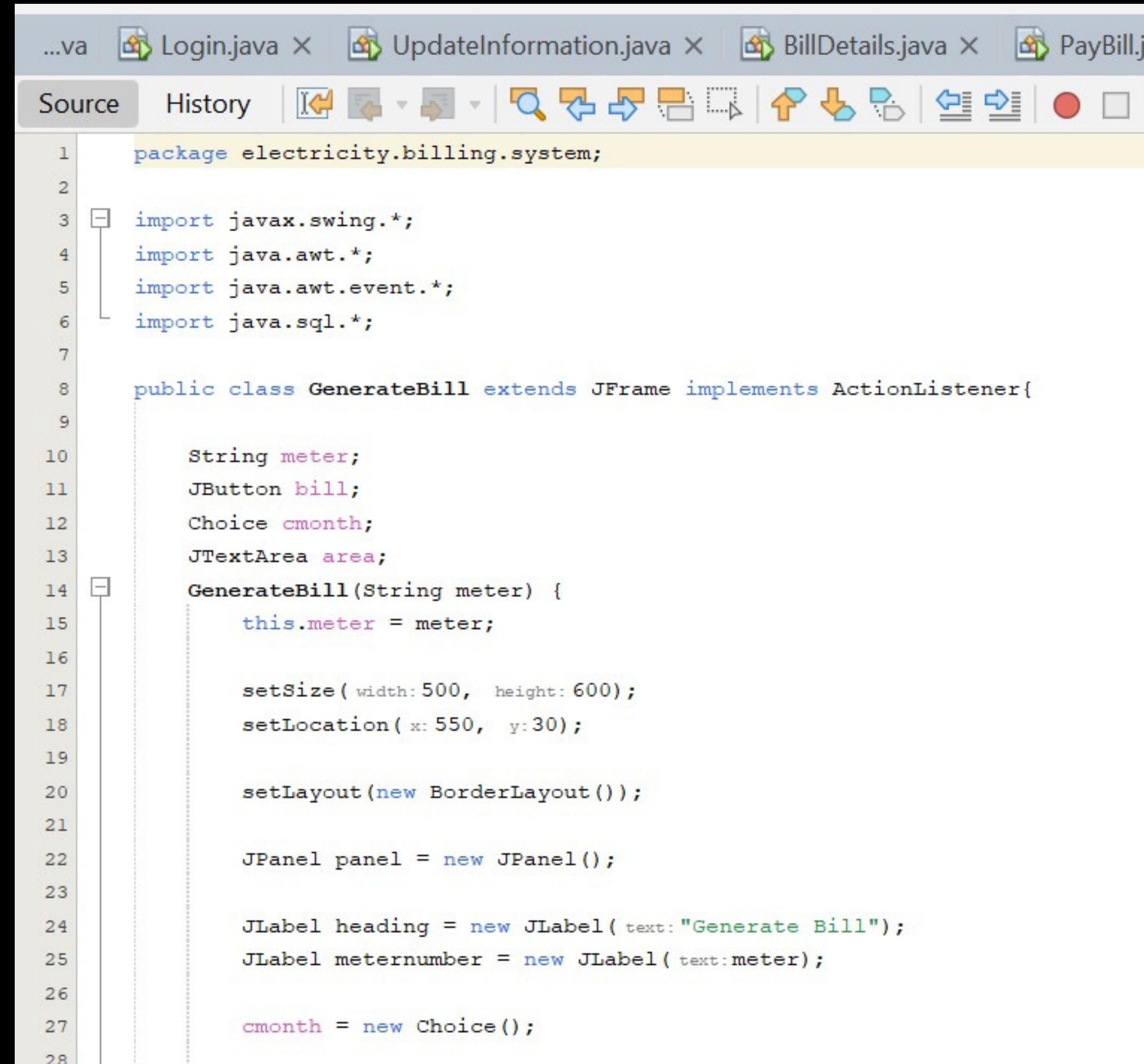
The screenshot shows a Java code editor interface with the following details:

- Title Bar:** The title bar displays "Login Java".
- Project Explorer:** On the left, there is a project tree showing "Electricity Billing System" expanded, with "src" and "bin" nodes.
- Code Editor:** The main area contains the source code for the "Login" class. The code is color-coded for syntax highlighting.
- Toolbars:** There are two toolbars at the top of the code editor.
- Status Bar:** The status bar at the bottom shows the path "src\main\java\electricity\billing\system\Login.java" and the word "File".

```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.*;
6 import java.sql.*;
7
8 public class Login extends JFrame implements ActionListener{
9
10    JButton login, cancel, signup;
11    JTextField username, password;
12    Choice logginin;
13    Login() {
14        super( title:"Login Page");
15        getContentPane().setBackground( c:Color.WHITE);
16        setLayout( manager: null);
17
18        JLabel lblusername = new JLabel( text:"Username");
19        lblusername.setBounds( x: 300, y:20, width:100, height:20);
20        add( comp:lblusername);
21
22        username = new JTextField();
23        username.setBounds( x: 400, y:20, width:150, height:20);
24        add( comp:username);
```

The screenshot of the codes.

Generate Bill Java



The screenshot shows a Java IDE interface with the title "Generate Bill Java". The top menu bar has tabs for "Source" and "History", along with various toolbars and icons. Below the toolbar, several Java files are listed in the tab bar: "...va", "Login.java X", "UpdateInformation.java X", "BillDetails.java X", and "PayBill.java". The main code editor area displays the following Java code:

```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.*;
6 import java.sql.*;
7
8 public class GenerateBill extends JFrame implements ActionListener{
9
10    String meter;
11    JButton bill;
12    Choice cmonth;
13    JTextArea area;
14    GenerateBill(String meter) {
15        this.meter = meter;
16
17        setSize( width:500, height:600 );
18        setLocation( x: 550, y:30 );
19
20        setLayout( new BorderLayout() );
21
22        JPanel panel = new JPanel();
23
24        JLabel heading = new JLabel( text:"Generate Bill" );
25        JLabel meternumber = new JLabel( text:meter );
26
27        cmonth = new Choice();
28    }
29
30    public void actionPerformed( ActionEvent e ) {
31        if ( e.getSource() == bill ) {
32            area.setText( "Generating Bill for Meter Number: " + meternumber.getText() );
33        }
34    }
35}
```

The screenshot of the codes.

Deposit DetailsJava

A screenshot of a Java Integrated Development Environment (IDE) showing the source code for a Java application named 'Deposit Details'. The code is written in Java and uses Swing for the graphical user interface. The IDE interface includes tabs for other files like CustomerDetails.java, Splash.java, and ViewInformation, and various toolbars for file operations and navigation.

```
1 package electricity.billing.system;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import java.sql.*;
6 import net.proteanit.sql.DbUtils;
7 import java.awt.event.*;
8
9 public class DepositDetails extends JFrame implements ActionListener{
10
11     Choice meternumber, cmonth;
12     JTable table;
13     JButton search, print;
14
15     DepositDetails(){
16         super( title:"Deposit Details");
17
18         setSize( width: 700, height: 700);
19         setLocation( x: 400, y:100);
20
21         setLayout( manager: null);
22         getContentPane().setBackground( c:Color.WHITE);
23
24         JLabel lblmeternumber = new JLabel( text:"Search By Meter Number");
25         lblmeternumber.setBounds( x:20, y:20, width:150, height:20);
26         add( comp:lblmeternumber);
27 }
```

The screenshot of the codes.

Customer Details Java

The screenshot shows a Java IDE interface with the title "Customer Details Java". The top menu bar includes tabs for "Source" and "History", along with various toolbars and icons. Below the toolbar, the code editor displays the "CustomerDetails.java" file. The code is written in Java and defines a class "CustomerDetails" that extends "JFrame" and implements "ActionListener". The class contains several instance variables (Choice, JTable, JButton) and a constructor that initializes the frame, sets its size and location, creates a table, and attempts to connect to a database and populate the table with customer data. The code uses imports from java.awt, javax.swing, java.sql, net.proteanit.sql.DbUtils, and java.awt.event.

```
1 package electricity.billing.system;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import java.sql.*;
6 import net.proteanit.sql.DbUtils;
7 import java.awt.event.*;
8
9 public class CustomerDetails extends JFrame implements ActionListener{
10
11     Choice meternumber, smonth;
12     JTable table;
13     JButton search, print;
14
15     CustomerDetails(){
16         super( title:"Customer Details");
17
18         setSize( width:1000, height:500);
19         setLocation( x: 100, y:110);
20
21         table = new JTable();
22
23         try {
24             Conn c = new Conn();
25             ResultSet rs = c.s.executeQuery( string:"select * from customer");
26
27             table.setModel( dataModel:DbUtils.resultSetToTableModel(rs));
28
29         } catch (Exception e) {
30             e.printStackTrace();
31         }
32     }
33
34     public void actionPerformed(ActionEvent ae) {
35         if(ae.getSource() == search) {
36             String meterNo = meternumber.getSelectedItem().toString();
37             String month = smonth.getSelectedItem().toString();
38
39             Conn c = new Conn();
40             ResultSet rs = c.s.executeQuery("select * from customer where meterNumber = "+meterNo+" and month = "+month);
41
42             DefaultTableModel dtm = (DefaultTableModel) table.getModel();
43             dtm.setRowCount(0);
44
45             while(rs.next()) {
46                 dtm.addRow(new Object[]{rs.getString("meterNumber"), rs.getString("month"), rs.getString("customerName"), rs.getString("unitConsumed"), rs.getString("amountDue")});
47             }
48
49         }
50     }
51
52     public static void main(String args[]) {
53         new CustomerDetails();
54     }
55 }
```

The screenshot of the codes.

Connection Java

The screenshot shows a Java IDE interface with the following details:

- Toolbar:** Includes tabs for Source (selected), History, and various navigation icons (back, forward, search, etc.).
- Project Bar:** Shows multiple open files: ...va, BillDetails.java, PayBill.java, GenerateBill.java, Signup.java, Paytm.java, and Conn.java (the active file).
- Code Editor:** Displays the Conn.java code. The code is a class named Conn that establishes a connection to a MySQL database using JDBC. It includes imports for java.sql.* and java.util.*. The code uses Connection, Statement, and Exception classes from the java.sql package. It attempts to connect to the database with URL "jdbc:mysql://ebs", user "root", and password "2058". If an exception occurs, it prints the stack trace.
- Status Bar:** Shows line numbers 1 through 18 at the bottom of the code editor.

```
1 package electricity.billing.system;
2
3 import java.sql.*;
4
5 public class Conn {
6
7     Connection c;
8     Statement s;
9     Conn() {
10         try {
11             c = DriverManager.getConnection(url: "jdbc:mysql://ebs", user: "root", password: "2058");
12             s = c.createStatement();
13         } catch (Exception e) {
14             e.printStackTrace();
15         }
16     }
17 }
18
```

The screenshot of the codes.

Calculate Bill Java

A screenshot of a Java IDE interface. The title bar shows "Calculate Bill Java". The menu bar includes "File", "Edit", "Source", "History", and "Tools". The "Source" tab is selected. The code editor displays the following Java code:

```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.util.*;
6 import java.awt.event.*;
7 import java.sql.*;
8
9 public class CalculateBill extends JFrame implements ActionListener{
10
11     JTextField tfname, tfaddress, tfstate, tfunits, tfemail, tfphone;
12     JButton next, cancel;
13     JLabel lblname, labeladdress;
14     Choice meternumber, cmonth;
15     CalculateBill() {
16         setSize( width: 700, height: 500 );
17         setLocation( x: 400, y:150 );
18
19         JPanel p = new JPanel();
20         p.setLayout( mgr: null );
21         p.setBackground( new Color( r:173, g:216, b:230 ) );
22         add( comp:p );
23
24         JLabel heading = new JLabel( text:"Calculate Electricity Bill" );
25         heading.setBounds( x: 100, y:10, width:400, height:25 );
26         heading.setFont( new Font( name: "Tahoma", style:Font.PLAIN, size:24 ) );
27         p.add( comp:heading );
```

The screenshot of the codes.

Bill Details Java

A screenshot of a Java Integrated Development Environment (IDE) showing the source code for a Java application named "Bill Details". The code is written in Java and uses JDBC to interact with a database. The IDE interface includes a toolbar with various icons for file operations, a menu bar, and a status bar at the bottom.

```
1 package electricity.billing.system;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6 import net.proteanit.sql.DbUtils;
7
8 public class BillDetails extends JFrame{
9
10    BillDetails(String meter) {
11
12        setSize(width: 700, height: 650);
13        setLocation(x: 400, y:150);
14
15        getContentPane().setBackground(c:Color.WHITE);
16
17        JTable table = new JTable();
18
19        try {
20            Conn c = new Conn();
21            String query = "select * from bill where meter_no = '"+meter+"'";
22            ResultSet rs = c.s.executeQuery(string:query);
23
24            table.setModel(dataModel:DbUtils.resultSetToTableModel(rs));
25        } catch (Exception e) {
26            e.printStackTrace();
27        }
28
29    }
30}
```

SQL Code

```
create database ebs;  
  
use ebs;  
  
create table login(meter_no varchar(20),  
username varchar(30),  
name varchar(30), password varchar(20), user  
varchar(30));  
  
select * from login;  
  
create table customer(name varchar(20) ,  
meter_no varchar(20),  
address varchar(50), city varchar(30),state  
varchar(30),email varchar(40),phone  
varchar(20));
```

```
select * from login;  
select * from customer;  
create table meter_info(meter_no varchar(20),  
meter_location  
varchar(20), meter_type varchar(20), phase_code  
varchar(20),  
bill_type varchar(20),days varchar(20));  
select * from meter_info;  
  
create table tax(cost_per_unit varchar(20), meter_rent  
varchar(20), service_charge varchar(20), service_tax  
varchar(20), swacch_bharat_cess varchar(20), fixed_tax  
varchar(20));  
insert into tax values('9','47','22','57','6','18');  
select * from tax;  
  
create table bill(meter_no varchar(20), month varchar(30),  
units varchar(20), totalbill varchar(20), status varchar(20));  
select * from bill
```

New Customer

Customer Name:

Meter Number: 494918

Address:

City:

State:

Email:

Phone Number:

Next **Cancel**

RESULTS

meter_no	month	units	totalbill	status
729168	January	120	1230	Paid
748941	October	125	1275	Not Paid
748941	January	178	1752	Paid
590075	January	12	258	Not Paid
590075	June	250	2400	Not Paid

ELECTRICITY BILL GENERATED FOR THE MONTH
OF January, 2022

Search By Meter Number

Search By Month

Search
Print

Customer Name: binod

Meter Number : 729168

Address : dhanusha

City : birgunj

State : province 2

Email : ben10@gmail.com

Phone : 9814818394

Meter Location: Outside

Meter Type : Solar Meter

Phase Code : 033

Bill Type : Industrial

Days : 30

Calculate Electricity Bill

Meter Number:

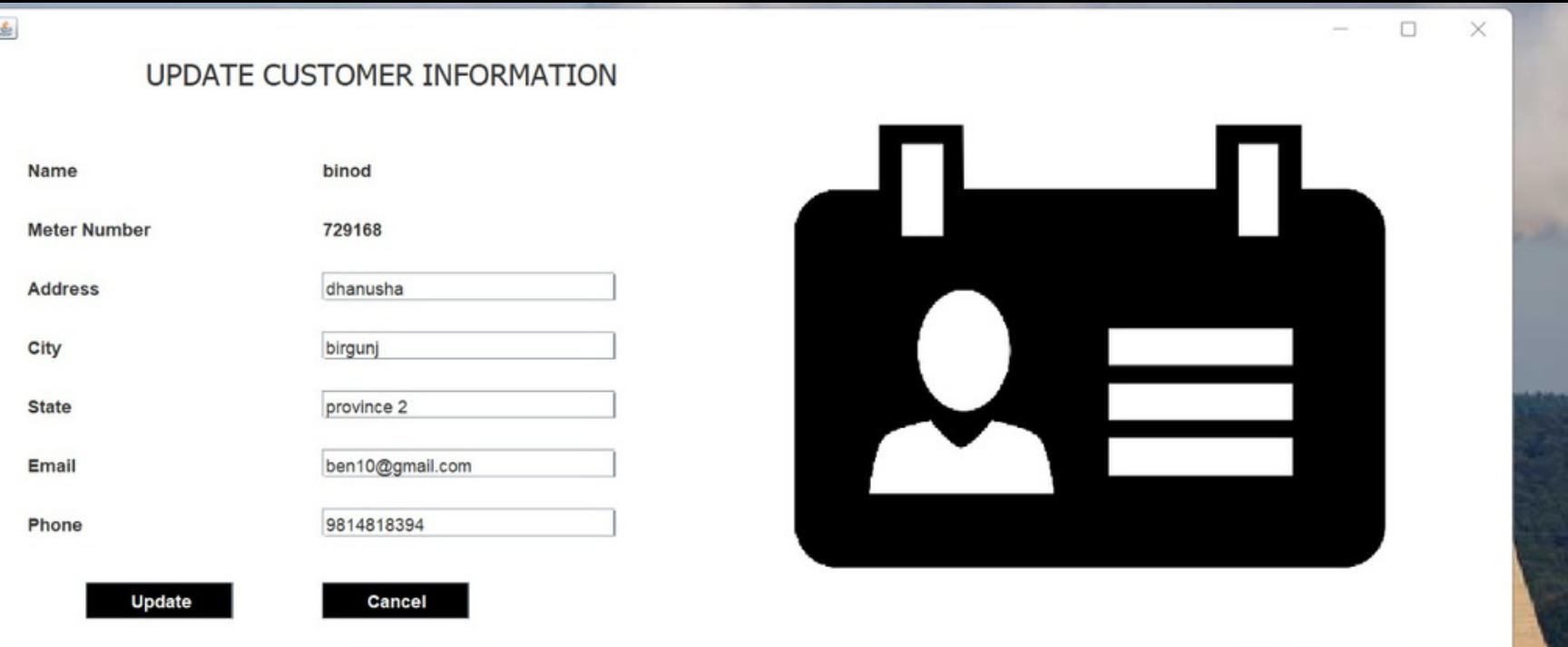
Name:

Address:

Units Consumed:

Month:

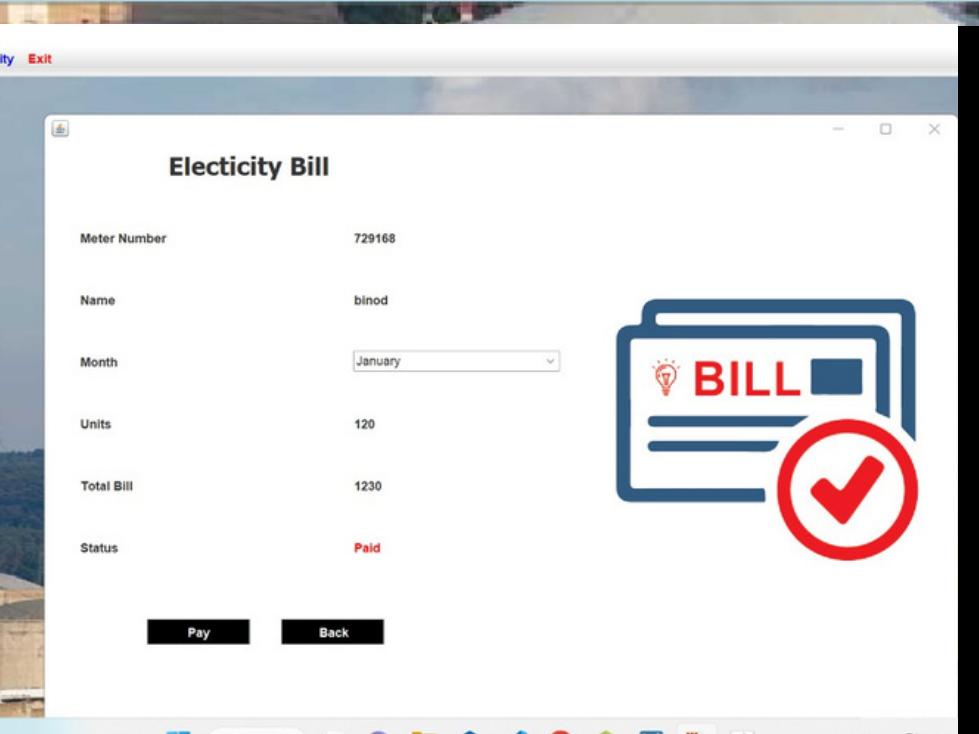
Submit **Cancel**



UPDATE CUSTOMER INFORMATION

Name	binod
Meter Number	729168
Address	<input type="text" value="dhanusha"/>
City	<input type="text" value="birgunj"/>
State	<input type="text" value="province 2"/>
Email	<input type="text" value="ben10@gmail.com"/>
Phone	<input type="text" value="9814818394"/>

Update **Cancel**



Electricity Bill

Meter Number	729168
Name	binod
Month	January
Units	120
Total Bill	1230
Status	Paid

Pay **Back**

BILL

Customer Details

name	meter_no	address	city	state	email	phone
adarsh	590075	tea nagar	coimbatore	tamil nadu	iloveubaby@gmai...	9003587451
binod	729168	dhanusha	birgunj	province 2	ben10@gmail.com	9814818394
anirudh	748941	anna nagar	delhi	ap	anirodh@gmail.com	9293042100
binod	828211	jeetpur	bara	3	abc@gmail.com	383435366

Print

General **Page Setup** **Appearance**

Print Service

Name: Microsoft Print to PDF

Status: Accepting jobs

Type:

Info: Print To File

Print Range

All
 Pages To

Copies

Number of copies:

Collate

Java Sql × SQL File 3*

```
20 • insert into tax values('9','47','22','57','6','18');  
21 • select * from tax;  
22  
23 • create table bill(meter_no varchar(20), month varchar(30), units varchar(20), totalbill varchar(20), status  
24 • select * from bill;  
25
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

	meter_no	month	units	totalbill	status
▶	729168	January	120	1230	Paid
	748941	October	125	1275	Not Paid
	748941	January	178	1752	Paid
	590075	January	12	258	Not Paid
	590075	June	250	2400	Not Paid

bill 2 × Read Only

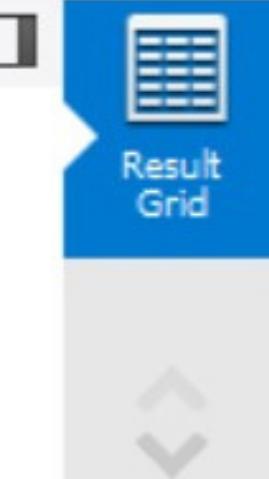
Java Sql × SQL File 3*



```
13 •     select * from customer;
14 •     create table meter_info(meter_no varchar(20), meter_location
15         varchar(20), meter_type varchar(20), phase_code varchar(20),
16         bill_type varchar(20),days varchar(20));
17 •     select * from meter_info;
18
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

	meter_no	meter_location	meter_type	phase_code	bill_type	days
▶	590075	Outside	Smart Meter	055	Normal	30
	729168	Outside	Solar Meter	033	Industial	30
	748941	Outside	Solar Meter	077	Industial	30
	828211	Outside	Electric Meter	011	Normal	30



Result Grid

meter_info 6 ×

Read Only

Output

Conclusion

- We used Net Beans IDE to execute the program
- We have Used Swing Graphical User Interface Toolkit
- We imported SQL connecter package to connect SQL and JAVA
- The GUI we have designed is very user friendly and have many functionalities

Thank
you!