

JAVA MINI PROJECT

GAS BOOKING SYSTEM:

Team Members:

Dharunika .S - 3122 22 5001 026

Dilsha Singh .D - 3122 22 5001 028

Hannah .S - 3122 22 5001 032

Problem Statement:

Our Implementation is a comprehensive Gas Bill Generating System that allows gas companies to efficiently manage billing processes, customer data, and payment transactions. The system will automate the generation of accurate and transparent gas bills, provide customers with detailed billing information, and facilitate secure payment processing. Additionally, our system is an user-friendly platform which provides a seamless experience for customers with an intuitive interface and robust functionalities.

Motivation of the problem:

To ensure a simple and secure environment for the consumer and the agencies we are making the 'Online Gas Booking System'. Through this system, we are solving the customers' problems in a lot of ways like booking their gas cylinder from home without traveling to the agency and stand in a queue to just book the gas.

On the other hand, it also makes it easier for the agency to check the number as a person has booked in a specific amount of time and the person who is booking the gas is authenticated or not whether he/she has brought the bottle within the time period, to get his gas booked.

It also prevents any error while registering the gas as if it's done manually there is always a chance of omission and oversight. It also helps the agency to move from a manual system of registering the data to storing it online which digitized the agency and reduces their overall carbon footprint.

Scope:

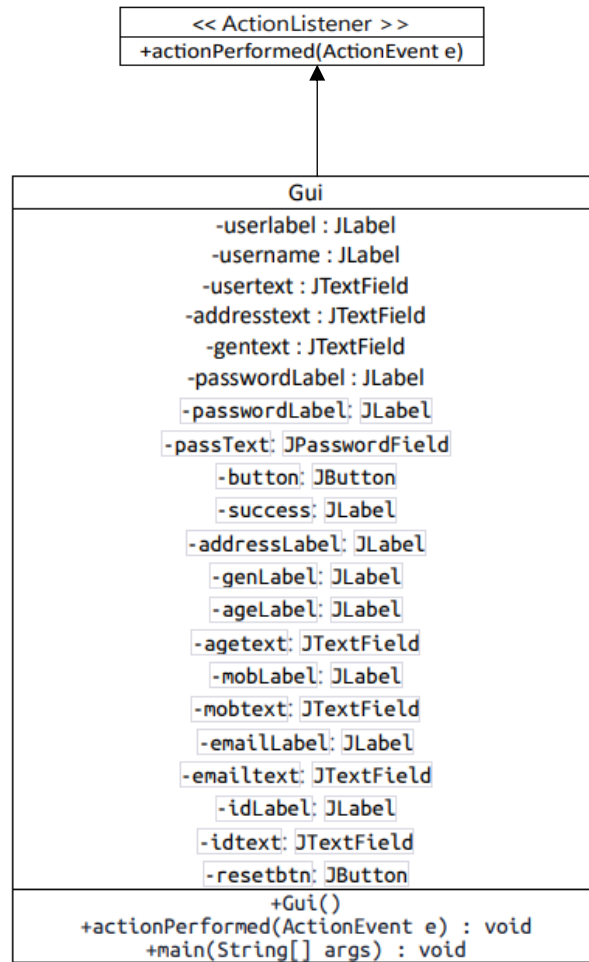
- The system will help the customers by providing a simple user interactive interface for booking gas online which will save them time and money.
- Basically, there are two types of users for the cylinders domestic and other is commercial. It gives every user a simple and secure system by authorizing the user before entering the system.
- The system will display the user the number of gas they booked online with a detailed description as there should be a limited time after which new gas can be booked

Limitation:

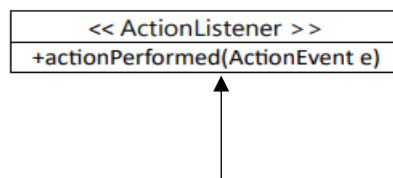
- The booking process increments the level of complexity as the number of customers coming to book gas increases.
- Information is not available globally to both customers and agencies.
- Only one customer can book a gas at a time.
- The user has to manually enter some details in which they can enter it wrongly.

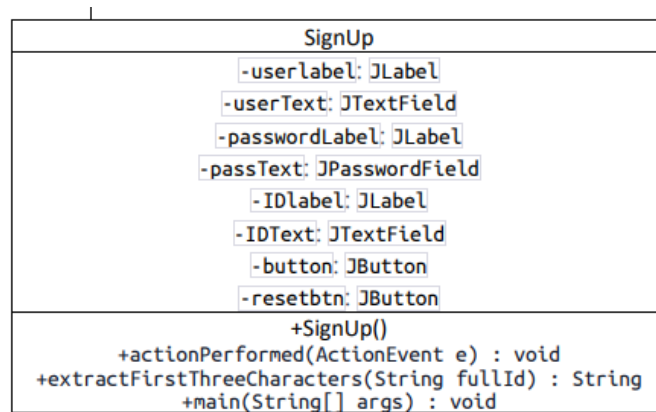
CLASS DIAGRAMS

REGISTRATION

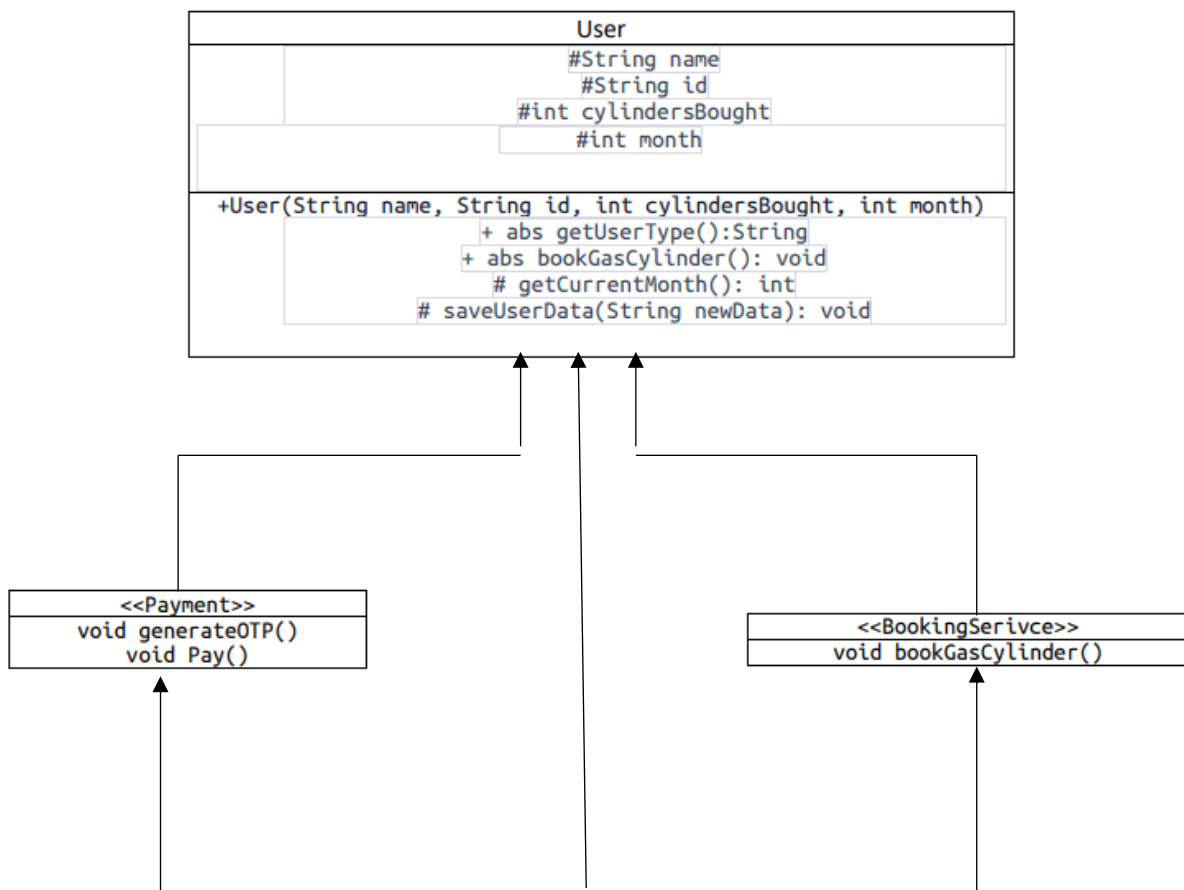


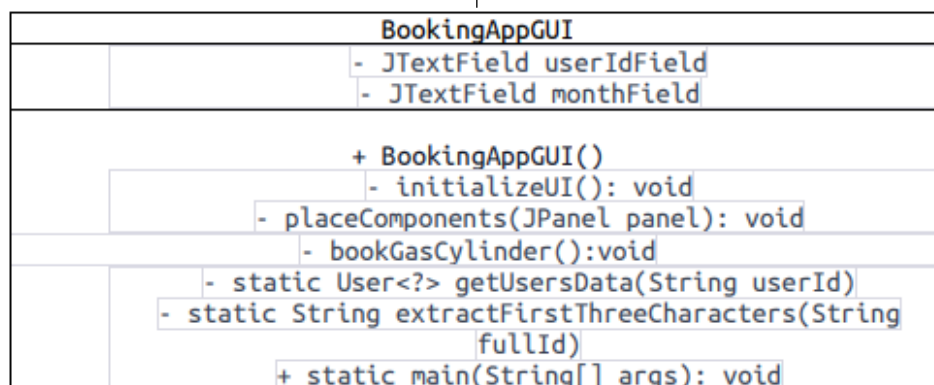
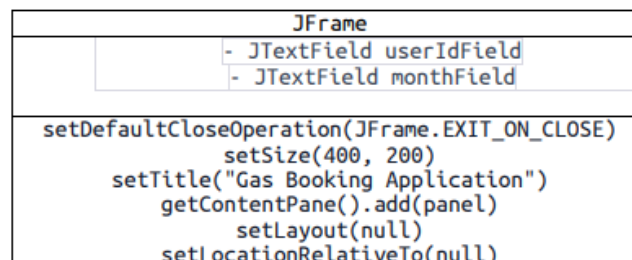
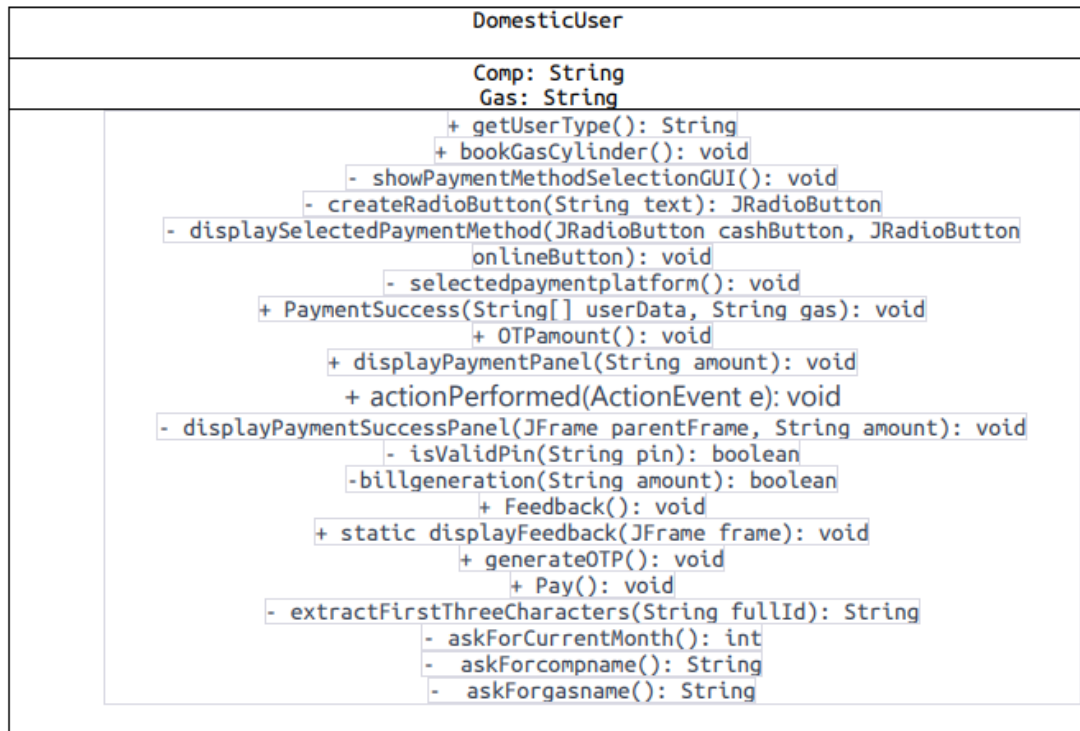
SIGN UP





BOOKING APP





Gas provider

GasProvider	
	- String company; - String ID; - String address; - String interconnector; - String shipper;
	- Map<String, String> gasdetails() + GasProvider() + saveToFile(): void - subsidy(String id, String original_amount): String - Map.Entry<String, String> get_last_entry() + static main(String[] args): void

IMPLEMENTATION

COMPANY REGISTRATION

```

package main;

import java.io.FileWriter;
import java.io.File;
import java.io.IOException;
import java.util.*;

class GasProvider
{
    private String company;
    private String ID;
    private String address;
    private String interconnector;
    private String shipper;
    private Map<String, String> gasdetails;
  
```

```
public GasProvider()
{
    gasdetails = new HashMap<>();
}
public void setCompany(String company)
{
    this.company = company;
}
public String getCompany()
{
    return company;
}
public void setID(String ID)
{
    this.ID = ID;
}
public String getID()
{
    return ID;
}
public void setAddress(String address)
{
    this.address = address;
}
public String getAddress()
{
    return address;
}
public void setInterconnector(String interconnector)
{

```

```
        this.interconnector = interconnector;
    }
    public String getInterconnector()
    {
        return interconnector;
    }
    public void setShipper(String shipper)
    {
        this.shipper = shipper;
    }
    public String getShipper()
    {
        return shipper;
    }
    public void addGas(String gas, String amount)
    {
        gasdetails.put(gas, amount);
    }
    public Map<String, String> getGasDetails()
    {
        return gasdetails;
    }
    public void saveToFile()
    {
        try
        {
            String fileName = "C:\\\\Users\\\\Hannah\\\\GasBill\\\\company.txt";
            File f = new File(fileName);
            FileWriter fw = new FileWriter(f, true);
            String id = getID();
```



```

        fw.write(id + "," + getCompany() + ",");
        for (Map.Entry<String, String> entry : gasdetails.entrySet())
        {
            String gas = entry.getKey();
            String original_amount = entry.getValue();
            String adjusted_amount = subsidy(id, original_amount);
            fw.write(gas + "," + adjusted_amount);
            if (!entry.equals(get_last_entry()))
            {
                fw.write(",");
            }
        }
        fw.write("\n");
        System.out.println("data Written");
        fw.close();

        //System.out.println("Data written to " + fileName);
    }
    catch (IOException e)
    {
        System.out.println("An error occurred: " + e.getMessage());
    }
}

private String subsidy(String id, String original_amount)
{
    if (id.startsWith("ABC"))
    {
        int amount = Integer.parseInt(original_amount) - 300;
        return String.valueOf(Math.max(amount, 0));
    }
    else if (id.startsWith("XYZ"))

```

```

        {
            return original_amount;
        }
        else
        {
            return "0";
        }
    }
}

private Map.Entry<String, String> get_last_entry()
{
    Map.Entry<String, String> last_entry = null;
    for (Map.Entry<String, String> entry : gasdetails.entrySet())
    {
        last_entry = entry;
    }
    return last_entry;
}

}

public class Gasproviders {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        GasProvider provider = new GasProvider();
        System.out.println("Welcome Gas providers");
        System.out.println("Your company name:");
        String companyName = sc.nextLine();
        provider.setCompany(companyName);
        System.out.println("Your ID:");
        provider.setID(sc.next());
        sc.nextLine();
        System.out.println("Your address:");
    }
}

```

```

        provider.setAddress(sc.nextLine());
        System.out.println("Tell us your Gas interconnector:");
        provider.setInterconnector(sc.nextLine());
        System.out.println("Tell us your Gas Shipper:");
        provider.setShipper(sc.nextLine());
        System.out.println("Enter the gases you supply along with their amounts (Eg:
gas1 amount1, gas2 amount2):");
        String gasesInput = sc.nextLine();
        String[] gas_amount_pairs = gasesInput.split(",");
        for (String pair : gas_amount_pairs)
        {
            String[] gasAndAmount = pair.trim().split(" ");
            provider.addGas(gasAndAmount[0], gasAndAmount[1]);
        }

        provider.saveToFile();
    }
}

```

COMPANY FILE

ABC1234,Indore Gas,NaturalGas,600,LPG,640
 ABC1245,Bharath,NaturalGas,600,BioGas,450,Propane,1200
 ABC1789,HP,Methane,1300,LPG,450,Propane,1200
 XYZ1453,GAIL,Oxygen,1700,Helium,900,Argon,3000
 XYZ3456,Air Liquide India,Oxygen,10000,Hydrogen,1200,Choline,9000
 XYZ1567,Inox Air
 Products,Ammonia,5000,Oxygen,10000,Argon,9000,Helium,5000ABC1267,Super
 Gas,LPG,600,Biogas,700
 ABC1298,Super Gas,Naturalgas,500,LPG,600
 ABC1289,SuperGas,BioGas,400,Natural,500

REGISTRATION

```
package main;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

import javax.sql.ConnectionEvent;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;

public class Gui implements ActionListener{

    private static JLabel userlabel;
    private static JLabel username;
    private static JTextField userText;
    private static JTextField usertext;
    private static JTextField addresstext;
    private static JTextField gentext;
```

```
private static JLabel passwordLabel;  
private static JPasswordField passText;  
private static JButton button;  
private static JLabel success;  
private static JLabel addressLabel;  
private static JLabel genLabel;  
private static JLabel ageLabel;  
private static JTextField agetext;  
private static JLabel mobLabel;  
private static JTextField mobtext;  
private static JLabel emailLabel;  
private static JTextField emailtext;  
private static JLabel idLabel;  
private static JTextField idtext;  
private static JButton resetbtn;
```

```
public Gui(){  
    //Frame is window && panel is layout  
    JPanel panel = new JPanel();  
    JFrame frame = new JFrame();  
    frame.setSize(350,200);  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    frame.add(panel);  
  
    panel.setLayout(null);  
  
    username = new JLabel("Name");  
    username.setBounds(10,20,80,25);
```

```
panel.add(username);
```

```
usertext = new JTextField(20);  
usertext.setBounds(100,20,165,25);  
panel.add(usertext);
```

```
addressLabel = new JLabel("Address");  
addressLabel.setBounds(10,50,80,25);  
panel.add(addressLabel);
```

```
addresstext = new JTextField(20);  
addresstext.setBounds(100,50,165,25);  
panel.add(addresstext);
```

```
genLabel = new JLabel("Gender");  
genLabel.setBounds(10,80,80,25);  
panel.add(genLabel);
```

```
gentext = new JTextField(20);  
gentext.setBounds(100,80,165,25);  
panel.add(gentext);
```

```
ageLabel = new JLabel("Age");  
ageLabel.setBounds(10, 110, 80, 25);  
panel.add(ageLabel);
```

```
agetext = new JTextField(20);  
agetext.setBounds(100,110,165,25);  
panel.add(agetext);
```

```
mobLabel = new JLabel("Mobile");  
mobLabel.setBounds(10,140,80,25);  
panel.add(mobLabel);
```

```
mobtext = new JTextField(20);  
mobtext.setBounds(100,140,165,25);  
panel.add(mobtext);
```

```
emailLabel = new JLabel("Email");  
emailLabel.setBounds(10,170,80,25);  
panel.add(emailLabel);
```

```
emailtext = new JTextField(20);  
emailtext.setBounds(100,170,165,25);  
panel.add(emailtext);
```

```
idLabel = new JLabel("Id");  
idLabel.setBounds(10,200,80,25);  
panel.add(idLabel);
```

```
idtext = new JTextField(20);  
idtext.setBounds(100,200,165,25);  
panel.add(idtext);
```

```
userlabel = new JLabel("User");  
userlabel.setBounds(10, 230, 80, 25);  
panel.add(userlabel);
```

```
userText = new JTextField(20);
userText.setBounds(100,230,165,25);
panel.add(userText);
```

```
passwordLabel = new JLabel("Password");
passwordLabel.setBounds(10,260,80,25);
panel.add(passwordLabel);
```

```
passText = new JPasswordField(20);
passText.setBounds(100,260,165,25);
panel.add(passText);
```

```
button = new JButton("Register");
button.setBounds(10,290,100,25);
button.addActionListener(this);
panel.add(button);
```

```
resetbtn = new JButton("Reset");
resetbtn.setBounds(150,290,80,25);
resetbtn.addActionListener(this);
panel.add(resetbtn);
```

```
success = new JLabel("");
success.setBounds(10,110,300,25);
panel.add(success);
```

```
frame.setVisible(true);
```

```
}
```

```
public void actionPerformed(ActionEvent e) {
```



```

if (e.getSource() == button) {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/users","root","Book123$")
;

        String query = "INSERT INTO registration values(?,?,?,?,?,?,?,?,?)";
        PreparedStatement ps = con.prepareStatement(query);
        ps.setString(1, usertext.getText());
        ps.setString(2, addresstext.getText());
        ps.setString(3, gentext.getText());
        ps.setString(4, agetext.getText());
        ps.setString(5, mobtext.getText());
        ps.setString(6, emailtext.getText());
        ps.setString(7, idtext.getText());
        ps.setString(8, userText.getText());
        ps.setString(9, passText.getText());

        int i = ps.executeUpdate();
        JOptionPane.showMessageDialog(button,i+" Record updated! ");
        ps.close();
        con.close();

    }catch(Exception e1){
        e1.printStackTrace();
    }

} else if (e.getSource() == resetbtn) {
    usertext.setText("");
    addresstext.setText("");
    gentext.setText("");

```

```
    agetext.setText("");
    mobtext.setText("");
    emailtext.setText("");
    idtext.setText("");
    userText.setText("");
    passText.setText("");
}
}
```

```
public static void main(String[] args)
{
    new Gui();

    SwingUtilities.invokeLater(() -> {
        JFrame frame = new JFrame("Swing Transition Example");
        frame.setSize(300, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton nextButton = new JButton("Sign Up");
        frame.add(nextButton);

        // Add ActionListener to the button
        nextButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Open the next class
                new SignUp();
                // Close the current frame if needed
                frame.dispose();
            }
        });
    });
}
```

```

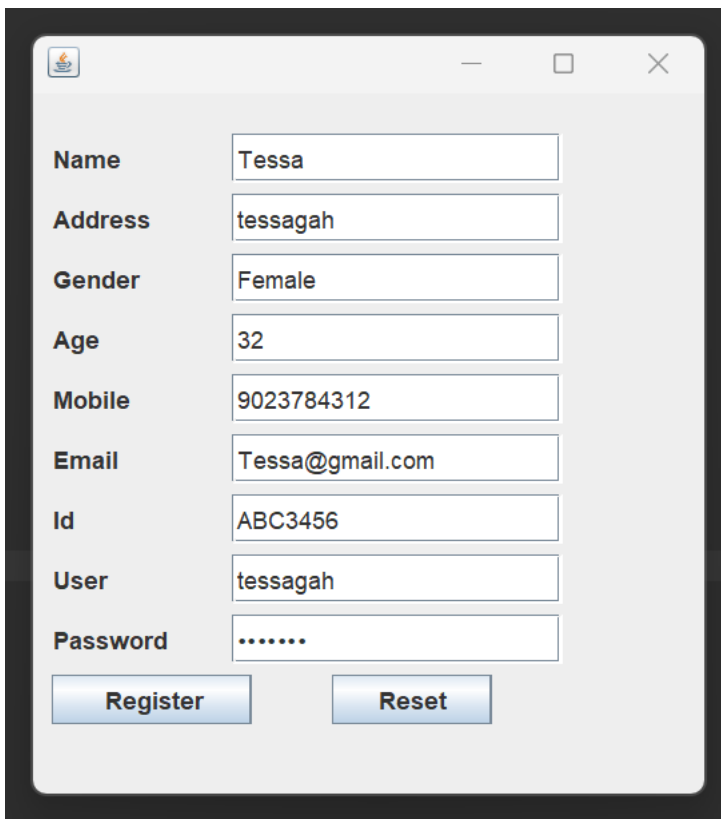
        }
    });

    frame.setVisible(true);
});
}

}

```

OUTPUT



The screenshot shows a Java Swing window titled "Register" with a registration form. The form contains the following fields and values:

Field	Value
Name	Tessa
Address	tessagah
Gender	Female
Age	32
Mobile	9023784312
Email	Tessa@gmail.com
Id	ABC3456
User	tessagah
Password

At the bottom of the form, there are two buttons: "Register" and "Reset".

DATA STORED IN SQL

SQL CODE

```
create database users;
```

```
use users;
```

```
create table registration(Name varchar(20),Address varchar(50),Gender
varchar(10),Age varchar(10),Mobile varchar(10),Email varchar(30),Id
varchar(20),User varchar(30),Password varchar(10));
```

```
desc registration;
```

```
select * from registration;
```

The screenshot displays a database management interface. At the top, a list of SQL queries is shown:

- 1 • create database users;
- 2 • use users;
- 3 • create table registration(Name varchar(20),Address varchar(50),Gender varchar(10),Age varchar(10),Mobile \
- 4 • desc registration;
- 5 • select * from registration;

Below the queries, a 'Result Grid' is visible, showing the output of the 'select * from registration;' query. The grid has columns for Name, Address, Gender, Age, Mobile, Email, Id, User, and Password. The data is as follows:

Name	Address	Gender	Age	Mobile	Email	Id	User	Password
Tessa	chaitanyapuri, warangal	Female	32	9023784312	Tessa@gmail.com	ABC3456	tessagah	procode
Lara	selayur, Chennai	Female	26	9012893452	Lara@gmail.com	XYZ9023	Laradm	code
kaira	Tirunelveli, chennai	Female	34	9023561734	kaira@gmail.com	ABC9012	kairagah	codex
Hanfi	warangal, Telangana	Female	45	8912349023	Hanfi@gmail.com	ABC7865	hanfirah	promax
Bharati	chennai	Female	20	9034567123	bharati@gmail.com	ABC4098	bharatigah	pass
Ram	selayur	Male	30	9012345678	ram@gmail.com	ABC1234	ramgah	password

Below the result grid, there is an 'Output' section with a tab labeled 'registration 1'. It shows the execution of the 'select * from registration;' query, indicating that 6 rows were returned.

SIGN UP

```
package main;
```

```
import java.awt.BorderLayout;
```

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

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

```
import java.io.BufferedReader;
```

```
import java.io.FileReader;
```

```
import java.io.IOException;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
```

```
import java.sql.Statement;
```

```
import java.awt.Image;
```

```
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;
```

```
public class SignUp implements ActionListener{
```

```
    private static JLabel userlabel;
    private static JLabel backgroundLabel;
    private static JTextField userText;
    private static JLabel passwordLabel;
    private static JPasswordField passText;
    private static JButton button;
    private static JButton resetbtn;
    private static JLabel IDlabel;
    private static JTextField IDText;
```

```
    public SignUp(){
```

```
        JPanel panel = new JPanel();
        JFrame frame = new JFrame();
```

```
frame.setSize(350,200);  
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
frame.add(panel);
```

```
panel.setLayout(null);
```

```
userlabel = new JLabel("Username");  
userlabel.setBounds(10, 20, 80, 25);  
panel.add(userlabel);
```

```
userText = new JTextField(20);  
userText.setBounds(100,20,165,25);  
panel.add(userText);
```

```
passwordLabel = new JLabel("Password");  
passwordLabel.setBounds(10,50,80,25);  
panel.add(passwordLabel);
```

```
passText = new JPasswordField(20);  
passText.setBounds(100,50,165,25);  
panel.add(passText);
```

```
IDlabel = new JLabel("ID");  
IDlabel.setBounds(10,80,80,25);  
panel.add(IDlabel);
```

```
IDText = new JTextField(20);  
IDText.setBounds(100,80,165,25);  
panel.add(IDText);
```

```
button = new JButton("Login");
button.setBounds(10,110,100,25);
button.addActionListener(this);
panel.add(button);
```

```
resetbtn = new JButton("Reset");
resetbtn.setBounds(150,110,80,25);
resetbtn.addActionListener(this);
panel.add(resetbtn);
```

```
frame.setVisible(true);
```

```
}
```

```
@Override
```

```
public void actionPerformed(ActionEvent e) {
    if(e.getSource() == button)
    {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/users","root","Book123$")
;

            Statement stmt = con.createStatement();
            String sql = "SELECT * FROM registration where
User='"+userText.getText()+"' and Password='"+passText.getText().toString()+"'";
            ResultSet rs = stmt.executeQuery(sql);
            if(rs.next())
```

```

        {
            JOptionPane.showMessageDialog(null, "Login
Sucessfull....");
        }
        else
        {
            JOptionPane.showMessageDialog(null, "Incorrect
Username and Password....");
        }
        String id = IDText.getText();
        String FILE_PATH =
"C:\\Users\\Hannah\\GasBill\\company.txt";
        String extractedId = extractFirstThreeCharacters(id);
        JTextArea textArea = new JTextArea();
        textArea.setEditable(false);

        // Move panel and frame creation outside the inner try-catch block
        JPanel panel = new JPanel();
        JFrame frame = new JFrame();
        frame.setSize(350, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(panel);
        panel.setLayout(new BorderLayout());
        panel.add(new JScrollPane(textArea), BorderLayout.CENTER);
        if(extractedId.equals("ABC"))
        {
            textArea.append("WELCOME DOMESTIC GAS USERS...");
        }
        else {
            textArea.append("WELCOME COMMERCIAL GAS USERS...");
        }
    }
}

```



```

    }

    try (BufferedReader br = new BufferedReader(new
FileReader(FILE_PATH))) {
        String line;
        while ((line = br.readLine()) != null) {
            String[] userData = line.split(",");

            String idFromFile =
extractFirstThreeCharacters(userData[0]); // Assuming id is the second element in
the line

            // Corrected condition to check if the extracted ID matches
"ABC"

            if (extractedId.equalsIgnoreCase(idFromFile)) {
                textArea.append("\n\n");
                textArea.append("COMPANY NAME : " + userData[1] +
"\n");

                for(int i=2;i<userData.length;i++)
                {
                    textArea.append("GAS NAME : " + userData[i] + "
Amount : Rs" + userData[i + 1] + "\n");
                    i++;
                }
            }
        }

        frame.setVisible(true);
    } catch (IOException | NumberFormatException e2) {
        e2.printStackTrace();
    }

    con.close();

```

```

        }catch(Exception e1){
            e1.printStackTrace();
        }
    }
    else if (e.getSource() == resetbtn) {
        userText.setText("");
        passText.setText("");
        IDText.setText("");
    }

}

public static String extractFirstThreeCharacters(String fullId) {
return fullId.substring(0, 3);
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> new SignUp());

    JFrame frame = new JFrame("Swing Transition Example");
    frame.setSize(300, 200);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JButton nextButton = new JButton("Booking APP");
    frame.add(nextButton);

    // Add ActionListener to the button
    nextButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

```

```

        // Open the next class
        BookingAppGUI bookingAppSwing = new BookingAppGUI();
        bookingAppSwing.setVisible(true);

        frame.dispose();
    }
});

frame.setVisible(true);

}

}

```

OUTPUT



BOOKING APP

```

package main;

import javax.swing.*.*;
import javax.swing.Timer;

```

```
import java.util.*;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Font;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.io.*;
import java.awt.event.ActionListener;
import java.awt.geom.RoundRectangle2D;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

// Define a generic abstract class for User
abstract class User<T> {
```

```
protected static final String FILE_PATH =
"C:\\Users\\Hannah\\GasBill\\users_data.txt";

protected String name;
protected String id;
protected int cylindersBought;
protected int month;

public User(String name, String id, int cylindersBought, int month) {
    this.name = name;
    this.id = id;
    this.cylindersBought = cylindersBought;
    this.month = month;
}

// Abstract method to get the user type
public abstract String getUserType();

// Method to book the gas cylinder
public abstract void bookGasCylinder();

// Method to get the current month (dummy implementation, replace with actual
implementation)
protected int getCurrentMonth() {
    return month; // Replace with actual implementation when available
}

// Save the user data to the file
protected void saveUserData(String newData) {
    try (BufferedWriter bw = new BufferedWriter(new FileWriter(FILE_PATH))) {
        bw.write(newData);
    } catch (IOException e) {
```

```
        e.printStackTrace();
    }
}
}
```

// Define a generic concrete class for Domestic User

```
class DomesticUser extends User<DomesticUser> implements
BookingService<DomesticUser>,payment<DomesticUser> {
```

```
    String comp;
```

```
    String gas;
```

```
    public DomesticUser(String name, String id, int cylindersBought, int month) {
        super(name, id, cylindersBought, month);
    }
```

```
@Override
```

```
public String getUserType() {
    return "Domestic";
}
```

```
@Override
```

```
public void bookGasCylinder() {
```

```
    // Ask the user for the current month
```

```
    int currentMonth = askForCurrentMonth();
```

```
    comp = askForcompname();
```

```
    gas = askForgasname();
```

```
    StringBuilder newData = new StringBuilder();
```

```
    try (BufferedReader br = new BufferedReader(new FileReader(FILE_PATH))) {
```

```
        String line;
```

```

while ((line = br.readLine()) != null) {
    String[] userData = line.split(",");
    String idFromFile = userData[1]; // Assuming id is the second element in
the line
    String extractedId = extractFirstThreeCharacters(idFromFile);

    // Corrected condition to check if the extracted ID is "ABC"
    if (extractedId.equals("ABC") && id.equals(idFromFile)) {
        // Check if the month has changed
        if (currentMonth != getCurrentMonth()) {
            cylindersBought = 0; // Reset the cylinders bought for the new month
            month = currentMonth; // Update the current month
        }

        if (cylindersBought < 3) {
            cylindersBought++;
            JOptionPane.showMessageDialog(null, "Gas cylinder booked
successfully.");
        } else {
            JOptionPane.showMessageDialog(null, "You can book only 3 cylinders
per month.");
        }

        newData.append(name).append(",").append(id).append(",").append(cylindersBought
)
            .append(",").append(month).append("\n");

        // Break out of the loop after finding the matching user
        break;
    } else if (extractedId.equals("XYZ") && id.equals(idFromFile)) {
        if (currentMonth != getCurrentMonth()) {

```

```

        cylindersBought = 0; // Reset the cylinders bought for the new month
        month = currentMonth; // Update the current month
    }

    if (cylindersBought < 3) {
        cylindersBought++;
        JOptionPane.showMessageDialog(null, "Gas cylinder booked
successfully.");
    } else {
        JOptionPane.showMessageDialog(null, "You can book only 3 cylinders
per month.");
    }

    newData.append(name).append(",").append(id).append(",").append(cylindersBought
)

        .append(",").append(month).append("\n");

    // Break out of the loop after finding the matching user
    break;
} else {
    newData.append(line).append("\n");
}

}

// Append the rest of the file content after the matching user
while ((line = br.readLine()) != null) {
    newData.append(line).append("\n");
}

saveUserData(newData.toString());
} catch (IOException e) {
    e.printStackTrace();
}

```



```

    }
    showPaymentMethodSelectionGUI();
}

private void showPaymentMethodSelectionGUI() {
    JFrame paymentFrame = new JFrame();
    paymentFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    paymentFrame.setTitle("Payment Method Selection");
    paymentFrame.setSize(300, 150);

    JPanel panel = new JPanel(new GridLayout(1, 2));

    JRadioButton cashButton = createRadioButton("Cash");
    JRadioButton onlineButton = createRadioButton("Online");

    ButtonGroup buttonGroup = new ButtonGroup();
    buttonGroup.add(cashButton);
    buttonGroup.add(onlineButton);

    panel.add(cashButton);
    panel.add(onlineButton);

    JButton submitButton = new JButton("Submit");
    submitButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            displaySelectedPaymentMethod(cashButton, onlineButton);
            paymentFrame.dispose(); // Close the payment frame after selection
        }
    });
}

```

```
JPanel buttonPanel = new JPanel();  
buttonPanel.add(submitButton);
```

```
paymentFrame.setLayout(new BorderLayout());  
paymentFrame.add(panel, BorderLayout.CENTER);  
paymentFrame.add(buttonPanel, BorderLayout.SOUTH);
```

```
paymentFrame.setLocationRelativeTo(null);  
paymentFrame.setVisible(true);  
}
```

```
private JRadioButton createRadioButton(String text) {  
    JRadioButton radioButton = new JRadioButton(text);  
    radioButton.setFont(new Font("Arial", Font.PLAIN, 14));  
    radioButton.setHorizontalAlignment(SwingConstants.CENTER);  
    return radioButton;  
}
```

```
private void displaySelectedPaymentMethod(JRadioButton cashButton,  
JRadioButton onlineButton) {  
    if (cashButton.isSelected()) {  
        JOptionPane.showMessageDialog(null, "Selected Payment Method:  
Cash");  
        generateOTP();  
    } else if (onlineButton.isSelected()) {  
        JOptionPane.showMessageDialog(null, "Selected Payment Method:  
Online");  
        Pay();  
    } else {  
        JOptionPane.showMessageDialog(null, "Please select a payment  
method.");  
    }  
}
```

```

    }
}

private void selectedpaymentplatform() {
    String FILE_PATH1 = "C:\\Users\\Hannah\\GasBill\\company.txt";
    try (BufferedReader b = new BufferedReader(new FileReader(FILE_PATH1)))
    {
        String line;
        while ((line = b.readLine()) != null) {
            String[] userData = line.split(",");
            if(comp.equalsIgnoreCase(userData[1]));
            {
                PaymentSuccess(userData,gas);
            }

        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void PaymentSuccess(String[] userData,String gas)
{
    for(int i=2;i<userData.length;i++)
    {
        if(userData[i].equalsIgnoreCase(gas))
        {
            String amount = userData[i + 1];
            displayPaymentPanel(amount);
            break;
        }
    }
}

```

```

        }
        i++;
    }
}

public void OTPamount() {

    String FILE_PATH1 = "C:\\Users\\Hannah\\GasBill\\company.txt";
    try (BufferedReader b = new BufferedReader(new FileReader(FILE_PATH1)))
    {
        String line;
        while ((line = b.readLine()) != null) {
            String[] userData = line.split(",");
            if(comp.equalsIgnoreCase(userData[1]));
            {
                for(int i=2;i<userData.length;i++)
                {
                    if(userData[i].equalsIgnoreCase(gas))
                    {
                        billgeneration(userData[i+1]);
                        break;
                    }
                    i++;
                }
            }

        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

```
public void displayPaymentPanel(String amount) {  
    JFrame paymentFrame = new JFrame();  
    paymentFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);  
    paymentFrame.setTitle("Payment");  
  
    JPanel panel = new JPanel(null);  
  
    JLabel amountLabel = new JLabel("Amount: $" + amount);  
    amountLabel.setBounds(10, 20, 165, 25);  
    panel.add(amountLabel);  
  
    JLabel pinLabel = new JLabel("Enter PIN:");  
    pinLabel.setBounds(10, 50, 80, 25);  
    panel.add(pinLabel);  
  
    JPasswordField pinField = new JPasswordField();  
    pinField.setBounds(100, 50, 165, 25);  
    panel.add(pinField);  
  
    JButton payButton = new JButton("Pay");  
    payButton.setBounds(10, 80, 150, 25);  
    payButton.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            String enteredPin = new String(pinField.getPassword());  
            if (isValidPin(enteredPin)) {
```

```
        displayPaymentSuccessPanel(paymentFrame,amount);
    } else {
        JOptionPane.showMessageDialog(paymentFrame, "Invalid PIN.
Please try again.");
    }
}
});
panel.add(payButton);
```

```
paymentFrame.getContentPane().setLayout(new BorderLayout());
paymentFrame.getContentPane().add(panel, BorderLayout.CENTER);
```

```
paymentFrame.setSize(300, 150);
paymentFrame.setLocationRelativeTo(null);
paymentFrame.setVisible(true);
}
```

```
private void displayPaymentSuccessPanel(JFrame parentFrame,String amount)
{
```

```
    JFrame successFrame = new JFrame();
    successFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    successFrame.setTitle("Payment Success");
```

```
    JPanel panel = new JPanel();
    JLabel successLabel = new JLabel("Payment Successful!");
    successLabel.setFont(new Font("Arial", Font.BOLD, 16));
    successLabel.setForeground(Color.GREEN);
    panel.add(successLabel);
```

```
    successFrame.getContentPane().setLayout(new BorderLayout());
    successFrame.getContentPane().add(panel, BorderLayout.CENTER);
```

```
successFrame.setSize(250, 100);
successFrame.setLocationRelativeTo(parentFrame);
successFrame.setVisible(true);
```

```
int delay = 2000;
```

```
Timer timer = new Timer(delay, new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        successFrame.dispose();
        billgeneration(amount);
    }
});
```

```
// Start the timer
timer.setRepeats(false); // Set to false to run only once
timer.start();
```

```
}
```

```
private boolean isValidPin(String pin) {

    return pin.length() == 6 && pin.matches("\\d+");
}
```

```
void billgeneration(String amount)
```

```

{
    JFrame billFrame = new JFrame();
    billFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    billFrame.setTitle("Bill Generation");

    // Create and configure main panel
    JPanel mainPanel = new JPanel() {
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            Graphics2D g2d = (Graphics2D) g.create();
            int w = getWidth();
            int h = getHeight();
            GradientPaint gp = new GradientPaint(0, 0, new Color(52, 152, 219), 0,
h, new Color(41, 128, 185));
            g2d.setPaint(gp);
            g2d.fill(new RoundRectangle2D.Double(0, 0, w, h, 20, 20));
            g2d.dispose();
        }
    };
    mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y_AXIS));

    // Company Name Label
    BookingAppGUI obj = new BookingAppGUI();
    obj.setVisible(true); // Make sure the GUI is visible before trying to get data

    // Later in your code...
    String userId = obj.userId;

```



```
JLabel companyLabel = new JLabel("<html><div style='text-align: center; font-size: 25px;padding-left: 60px; font-weight: bold; padding-top: 10px;padding-bottom: 10px; " +
```

```
"color: #ffffff;'>"
```

```
+ comp +" Company "+"</div></html>");
```

```
companyLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
```

```
mainPanel.add(companyLabel);
```

```
try {
```

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

```
    Connection con =
```

```
    DriverManager.getConnection("jdbc:mysql://localhost:3306/users", "root",  
    "Book123$");
```

```
    String sql = "SELECT Id, Name, Address FROM registration WHERE  
    Id=?";
```

```
    PreparedStatement preparedStatement = con.prepareStatement(sql);
```

```
    preparedStatement.setString(1, userId);
```

```
    ResultSet resultSet = preparedStatement.executeQuery();
```

```
    if (resultSet.next()) {
```

```
        // Retrieve the Id, Name, and Address if there is a matching record
```

```
        String idFromDatabase = resultSet.getString("Id");
```

```
        String name = resultSet.getString("Name");
```

```
        String address = resultSet.getString("Address");
```

```
        // Check if provided idtext matches the Id in the database
```

```
        if ((userId).equals(idFromDatabase)) {
```

```
            // Display the Name and Address
```

```
            JLabel nameLabel = new JLabel("<html><div style='text-align:  
            center;font-size: 15px; font-weight: bold; padding: 10px; " +
```

```
            "color: #ffffff;'>Name: " + name + "</div></html>");
```

```
            nameLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
```

```
mainPanel.add(nameLabel);

JLabel addressLabel = new JLabel("<html><div style='text-align:
center;font-size: 15px; font-weight: bold; padding: 10px; " +
    "color: #ffffff;'>Address: " + address + "</div></html>");
addressLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
mainPanel.add(addressLabel);
    } else {
        JOptionPane.showMessageDialog(null, "Idtext and ID do not
match.");
    }
    } else {
        JOptionPane.showMessageDialog(null, "No matching record found for
Id: " + userId);
    }

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
JLabel gasLabel = new JLabel("<html><div style='text-align: center;font-size:
15px; font-weight: bold; padding: 10px; " +
    "color: #ffffff;'>Gas Name: " + gas + "</div></html>");
gasLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
mainPanel.add(gasLabel);
```

```
JLabel amountLabel = new JLabel("<html><div style='text-align: center;font-size: 15px; font-weight: bold; padding: 10px; " +
```

```
"color: #ffffff;'>Amount Paid: $" + amount + "</div></html>");
```

```
amountLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
```

```
mainPanel.add(amountLabel);
```

```
JLabel onlinePaymentLabel = new JLabel("<html><div style='text-align: center; font-size: 10px; padding-left: 50px; padding-top:10px;padding-bottom:10px;" +
```

```
"color: #ffffff;'>"
```

```
+ "&#9733;&#9733;&#9733;&#9733;&#9733 Payment  
&#9733;&#9733;&#9733;&#9733;&#9733;" + "</div></html>");
```

```
onlinePaymentLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
```

```
mainPanel.add(onlinePaymentLabel);
```

```
mainPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
```

```
billFrame.getContentPane().setLayout(new BorderLayout());
```

```
billFrame.getContentPane().add(mainPanel, BorderLayout.CENTER);
```

```
billFrame.setSize(400, 250);
```

```
billFrame.setLocationRelativeTo(null);
```

```
billFrame.setVisible(true);
```

```
Feedback();
```

```
}
```

```
void Feedback()
```

```
{  
    JFrame frame = new JFrame("Feedback Form");  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    frame.setSize(300, 200);  
  
    JPanel panel = new JPanel();  
    panel.setLayout(new GridLayout(3, 2));  
  
    JLabel idLabel = new JLabel("Enter your ID:");  
    idLabel.setBounds(10,20,80,25);  
    JTextField idField = new JTextField();  
    idField.setBounds(100,20,165,25);  
  
    JLabel feedbackLabel = new JLabel("Enter your feedback:");  
    feedbackLabel.setBounds(10,60,80,25);  
    JTextArea feedbackArea = new JTextArea();  
    feedbackArea.setBounds(100,60,165,80);  
  
    JButton submitButton = new JButton("Submit");  
    submitButton.setBounds(10,120,150,25);  
  
    submitButton.addActionListener(new ActionListener() {  
        @Override  
        public void actionPerformed(ActionEvent e) {  
            try {  
                displayFeedback(frame);  
            } catch (NumberFormatException ex) {  
                JOptionPane.showMessageDialog(frame, "Please enter a valid ID.");  
            }  
        }  
    })  
}
```

```

});

panel.add(idLabel);
panel.add(idField);
panel.add(feedbackLabel);
panel.add(new JScrollPane(feedbackArea));
panel.add(new JLabel()); // Empty label for spacing
panel.add(submitButton);

frame.add(panel);
frame.setLocationRelativeTo(null);
frame.setVisible(true);
}

public static void displayFeedback(JFrame frame) {
    JOptionPane.showMessageDialog(frame, "Feedback submitted
successfully!");
}

public void generateOTP()
{
    int otp = 1000 + new Random().nextInt(9000);
    JOptionPane.showMessageDialog(null, "Generated OTP: " + otp + "\n\n");
    JOptionPane.showMessageDialog(null, "The selected gas will be delivered on
time."
        + "\n\nNote: Additional charges will be for delivery");

    int delay = 2000;

```

```

Timer timer = new Timer(delay, new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        OTPamount();
    }
});

// Start the timer
timer.setRepeats(false); // Set to false to run only once
timer.start();
}

public void Pay()
{
    JFrame paymentFrame = new JFrame();
    paymentFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    paymentFrame.setTitle("Online Payment");
    paymentFrame.setSize(300, 150);

    JPanel panel = new JPanel(new GridLayout(1, 2));

    JRadioButton gpayButton = createRadioButton("GPay");
    JRadioButton paytmButton = createRadioButton("Paytm");

    ButtonGroup buttonGroup = new ButtonGroup();
    buttonGroup.add(gpayButton);
    buttonGroup.add(paytmButton);

    panel.add(gpayButton);

```

```
panel.add(paymentButton);
```

```
JButton submitButton = new JButton("Submit");  
submitButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        selectedPaymentPlatform();  
        paymentFrame.dispose(); // Close the payment frame after selection  
    }  
});
```

```
JPanel buttonPanel = new JPanel();  
buttonPanel.add(submitButton);
```

```
paymentFrame.setLayout(new BorderLayout());  
paymentFrame.add(panel, BorderLayout.CENTER);  
paymentFrame.add(buttonPanel, BorderLayout.SOUTH);
```

```
paymentFrame.setLocationRelativeTo(null);  
paymentFrame.setVisible(true);  
}
```

```
// Method to extract the first three characters from the ID  
private String extractFirstThreeCharacters(String fullId) {  
    return fullId.substring(0, 3);  
}
```

```
// Method to ask the user for the current month  
private int askForCurrentMonth() {  
    String input = JOptionPane.showInputDialog("Enter the current month (1-12):");
```

```

        JLabel monthLabel = new JLabel("Enter Month:");
        return Integer.parseInt(input);
    }
    private String askForcompname() {
        String input = JOptionPane.showInputDialog("Enter the company name:");
        JLabel compLabel = new JLabel("choose company:");
        return (input);
    }
    private String askForgasname() {
        String input = JOptionPane.showInputDialog("Enter the gas name:");
        JLabel gasLabel = new JLabel("Choose gas:");
        return (input);
    }
}

// Define a generic interface for BookingService
interface payment<T extends User<?>>{
    void generateOTP();
    void Pay();
}

interface BookingService<T extends User<?>> {
    void bookGasCylinder();
}

public class BookingAppGUI extends JFrame{

```



```
private JTextField userIdField;
private JTextField monthField;
public String userId;

public BookingAppGUI() {
    initializeUI();
}

private void initializeUI() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(400, 200);
    setTitle("Gas Booking Application");

    JPanel panel = new JPanel();
    getContentPane().add(panel);
    placeComponents(panel);

    setLocationRelativeTo(null);
}

private void placeComponents(JPanel panel) {

    JLabel userLabel = new JLabel("Enter User ID:");
    userLabel.setBounds(10, 20, 80, 25);
    panel.add(userLabel);

    userIdField = new JTextField(20);
    userIdField.setBounds(100, 20, 165, 25);
    panel.add(userIdField);
```

```
JLabel monthLabel = new JLabel("Enter Month:");
monthLabel.setBounds(10, 50, 80, 25);
panel.add(monthLabel);

monthField = new JTextField(20);
monthField.setBounds(100, 50, 165, 25);
panel.add(monthField);

JButton bookButton = new JButton("Book Gas Cylinder");
bookButton.setBounds(10, 80, 150, 25);
bookButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        bookGasCylinder();
    }
});
panel.add(bookButton);

}

private void bookGasCylinder() {
    userId = userIdField.getText();
    String monthStr = monthField.getText();

    if (userId.isEmpty() || monthStr.isEmpty()) {
```

```
        JOptionPane.showMessageDialog(this, "Please enter both User ID and  
Month.");  
        return;  
    }
```

```
    try {  
        int month = Integer.parseInt(monthStr);  
  
        // Check if the id and month are valid  
        User<?> user = getUsersData(userId);  
  
        if (user != null) {  
            BookingService<?> bookingService = (BookingService<?>) user;  
            bookingService.bookGasCylinder();  
        } else {  
            JOptionPane.showMessageDialog(this, "Invalid user ID.");  
        }  
    } catch (NumberFormatException e) {  
        JOptionPane.showMessageDialog(this, "Please enter a valid month  
(numeric).");  
    }  
}
```

```
// Method to get user data from the file  
private static User<?> getUsersData(String userId) {  
    try (BufferedReader br = new BufferedReader(new  
        FileReader(User.FILE_PATH))) {  
        String line;  
        while ((line = br.readLine()) != null) {  
            String[] userData = line.split(",");  
            String idFromFile = userData[1]; // Assuming id is the second element in  
the line
```

```

String extractedId = extractFirstThreeCharacters(idFromFile);

// Corrected condition to check if the extracted ID matches "ABC"
if (extractedId.equals("ABC") && userId.equals(idFromFile)) {
    String name = userData[0];

    int cylindersBought = Integer.parseInt(userData[2]); // Assuming
cylinders bought is the third element

    int month = Integer.parseInt(userData[3]); // Assuming month is the
fourth element

    return new DomesticUser(name, userId, cylindersBought, month);
}

else if(extractedId.equals("XYZ") && userId.equals(idFromFile)){
    String name = userData[0];

    int cylindersBought = Integer.parseInt(userData[2]); // Assuming
cylinders bought is the third element

    int month = Integer.parseInt(userData[3]); // Assuming month is the
fourth element

    return new DomesticUser(name, userId, cylindersBought, month);
}
}

} catch (IOException | NumberFormatException e) {
    e.printStackTrace();
}

return null; // Id not found
}

// Method to extract the first three characters from the ID
private static String extractFirstThreeCharacters(String fullId) {
    return fullId.substring(0, 3);
}

public static void main(String[] args) {

```

```

SwingUtilities.invokeLater(() -> {

    BookingAppGUI bookingAppSwing = new BookingAppGUI();
    bookingAppSwing.setVisible(true);

});

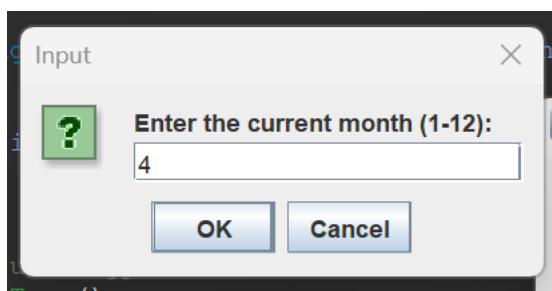
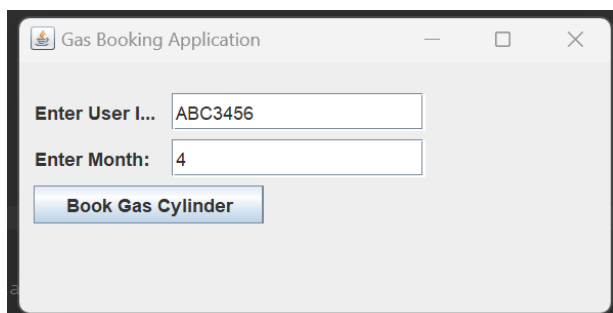
}
}

```


USERSDATA FILE

Tessa,ABC3456,1,3
 Lara,XYZ9023,1,3
 kaira,ABC9012,1,3
 Mara,XYZ9053,1,3
 Hanfi,ABC7865,1,2
 Ram,ABC1234,1,4


OUTPUT




Input

 Enter the company name:

Input

 Enter the gas name:

Message


 Gas cylinder booked successfully.

IF ONLINE PAYMENT

Payment Method Se...

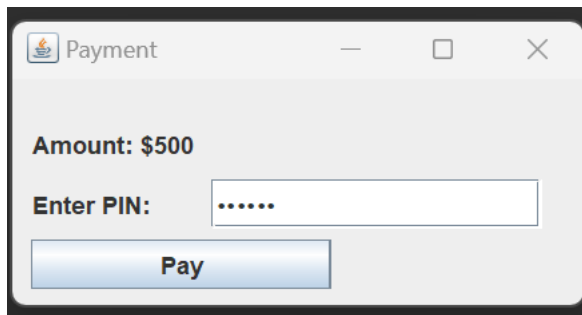
☐ Cash ☒ Online

Message

 Selected Payment Method: Online

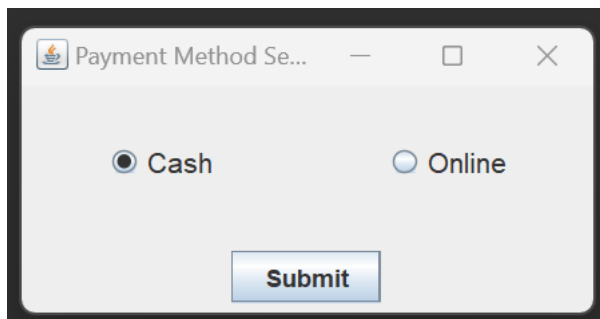
Online Payment

☒ GPay ☐ Paytm

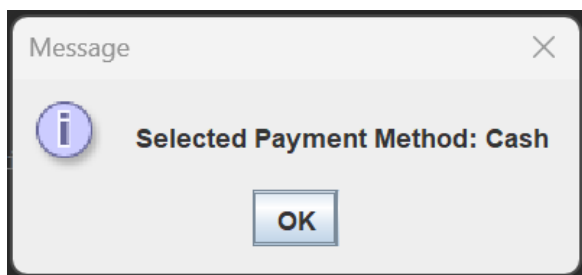


A screenshot of a 'Payment' dialog box. It features a title bar with a flame icon and the text 'Payment'. The main content area displays 'Amount: \$500' and 'Enter PIN:' followed by a text input field containing six dots. A 'Pay' button is located at the bottom.

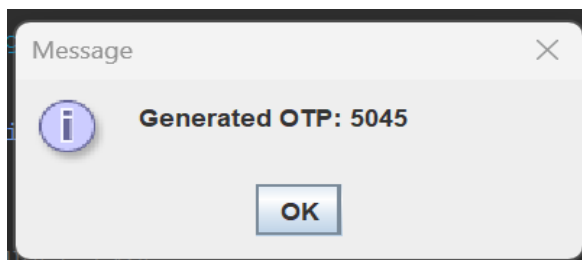
IF CASH ON DELIVERY



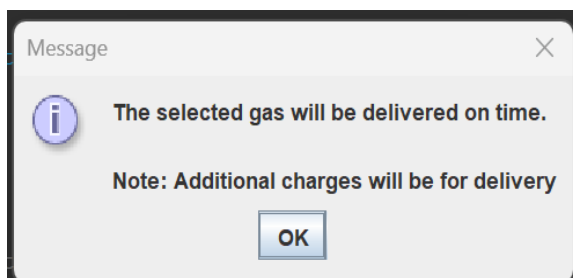
A screenshot of a 'Payment Method Selection' dialog box. It has a title bar with a flame icon and the text 'Payment Method Se...'. The main content area shows two radio buttons: 'Cash' (which is selected) and 'Online'. A 'Submit' button is positioned at the bottom.



A screenshot of a 'Message' dialog box. It has a title bar with the text 'Message' and a close button. The main content area features an information icon (i) and the text 'Selected Payment Method: Cash'. An 'OK' button is at the bottom.

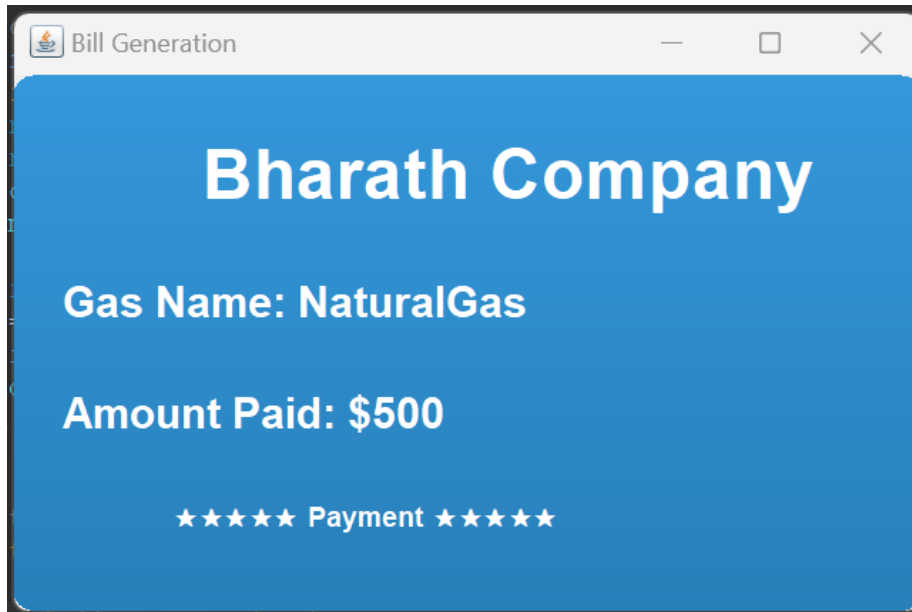


A screenshot of a 'Message' dialog box. It has a title bar with the text 'Message' and a close button. The main content area features an information icon (i) and the text 'Generated OTP: 5045'. An 'OK' button is at the bottom.

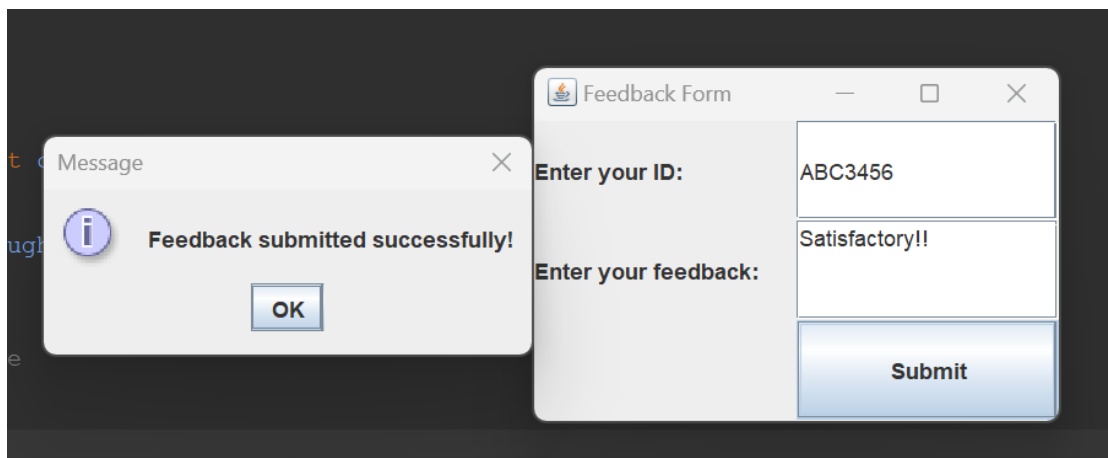


A screenshot of a 'Message' dialog box. It has a title bar with the text 'Message' and a close button. The main content area features an information icon (i) and the text 'The selected gas will be delivered on time.' followed by 'Note: Additional charges will be for delivery'. An 'OK' button is at the bottom.

BILL GENERATION



FEED BACK



Object-oriented-features used:

Interface:

In Java, an interface specifies the behavior of a class by providing an abstract type. Interfaces are used in Java to achieve abstraction.

- **Payment**
- **Booking Service**
- **Action Listener**

Collection Framework:

The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects. Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion. Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

We are using Hash-map in order to store the gas names and their respective amount entered by gas providers. Among all the collection framework, we chose hashmap, because we can store as key, value pair.

- `public Map<String, String> getGasDetails()`

Generics:

Generics means parameterized types. The idea is to allow type (Integer, String, ... etc., and user-defined types) to be a parameter to methods, classes, and interfaces. Using Generics, it is possible to create classes that work with different data types. An entity such as class, interface, or method that operates on a parameterized type is a generic entity.

We are using lower bound generics and extended it in an abstract type subclass

- `payment<T extends User<?>>`
- `BookingService<T extends User<?>>`

Abstract class:

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

We used abstract class, where another class can extend it ensuring that it is being overridden (**INHERITANCE**), and also internally many override functions are there, since we used JFrame.

Files:

The File class is an abstract representation of file and directory pathname. A pathname can be either absolute or relative. The File class has several methods for working with directories and files such as creating new directories or files, deleting and renaming directories or files, listing the contents of a directory etc.

The various details of the gas providers are being stored at files for the future use, while the details of the user are stored in **SQL**.

Encapsulation:

Encapsulation in Java is the process by which data (variables) and the code that acts upon them (methods) are integrated as a single unit. By encapsulating a class's variables, other classes cannot access them, and only the methods of the class can access them.

We employed encapsulation to safeguard the user's privacy by keeping all of their personal information confidential.

Exception Handling:

The Exception Handling in Java is one of the powerful *mechanism to handle the* runtime errors so that the normal flow of the application can be maintained. Since it raises many exceptions, we have defined user-defined exceptions, and also there are many checked and unchecked exceptions.

FUTURE EXTENSION:

- We can give more advance software for Online Gas Booking System including more facilities.
- To make it available to everyone, we can host it on web servers.
- Here in this project, only one customer can book the project but in future, we would develop the design that multiple customers can book this at the time.