NAME: ARYA WARUDKAR
YEAR/SEM: 4th Year / 7th Sem
ROLL NO: 04
BRANCH: (ELECTRONICS & TELECOMMUNICATION ENGINEERING)

INPUT CODE

```
package assignment;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.query.Query;

public class assignment1 extends JFrame {
    private static final long serialVersionUID = 1L;
    private JTextField txtId;
    private JButton btnInsert, btnUpdate, btnDelete, btnViewLast;
    private JTextArea textArea;

    public assignment1() {
        setTitle("Customer Management System");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new GridLayout(0, 2, 10, 10));

        // Initialize components
        txtId = new JTextField();

        btnInsert = new JButton("Insert");
        btnUpdate = new JButton("Update");
        btnDelete = new JButton("Delete");
        btnViewLast = new JButton("View Last");

        textArea = new JTextArea();
        textArea.setEditable(false);

        // Add components to the frame
```

```
        add(new JLabel("ID (for Update/Delete):"));
        add(txtId);

        add(btnInsert);
        add(btnUpdate);
        add(btnDelete);
        add(btnViewLast);

        add(new JScrollPane(textArea));

        // Add action listeners
        btnInsert.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                showInsertDialog();
            }
        });

        btnUpdate.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                showUpdateDialog();
            }
        });

        btnDelete.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                deleteRecord();
            }
        });

        btnViewLast.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                viewLastRecord();
            }
        });
    }

    private void showInsertDialog() {
        JTextField txtName = new JTextField();
        JTextField txtAddress = new JTextField();
        JTextField txtContact = new JTextField();
        JTextField txtCity = new JTextField();
        JTextField txtPostalCode = new JTextField();
        JTextField txtCountry = new JTextField();
```

```
        Object[] message = {
            "Name:", txtName,
            "Address:", txtAddress,
            "Contact:", txtContact,
            "City:", txtCity,
            "Postal Code:", txtPostalCode,
            "Country:", txtCountry
        };

        int option = JOptionPane.showConfirmDialog(this, message, "Enter Customer Details",
JOptionPane.OK_CANCEL_OPTION);
        if (option == JOptionPane.OK_OPTION) {
            Session session = HibernateUtil.getSessionFactory().openSession();
            Transaction tx = session.beginTransaction();
            try {
                Customer customer = new Customer();
                customer.setName(txtName.getText());
                customer.setAddress(txtAddress.getText());
                customer.setContact(txtContact.getText());
                customer.setCity(txtCity.getText());
                customer.setPostalCode(txtPostalCode.getText());
                customer.setCountry(txtCountry.getText());

                session.save(customer);
                tx.commit();
                JOptionPane.showMessageDialog(this, "Record inserted successfully.");
            } catch (Exception ex) {
                if (tx != null) tx.rollback();
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Error inserting record.");
            } finally {
                session.close();
            }
        }
    }

    private void showUpdateDialog() {
        int id;
        try {
            id = Integer.parseInt(txtId.getText());
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Invalid ID format.");
```

```
            return;
        }

        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        try {
            Customer customer = session.get(Customer.class, id);
            if (customer != null) {
                JTextField txtName = new JTextField(customer.getName());
                JTextField txtAddress = new JTextField(customer.getAddress());
                JTextField txtContact = new JTextField(customer.getContact());
                JTextField txtCity = new JTextField(customer.getCity());
                JTextField txtPostalCode = new JTextField(customer.getPostalCode());
                JTextField txtCountry = new JTextField(customer.getCountry());

                Object[] message = {
                    "Name:", txtName,
                    "Address:", txtAddress,
                    "Contact:", txtContact,
                    "City:", txtCity,
                    "Postal Code:", txtPostalCode,
                    "Country:", txtCountry
                };

                int option = JOptionPane.showConfirmDialog(this, message, "Update Customer Details", JOptionPane.OK_CANCEL_OPTION);
                if (option == JOptionPane.OK_OPTION) {
                    customer.setName(txtName.getText());
                    customer.setAddress(txtAddress.getText());
                    customer.setContact(txtContact.getText());
                    customer.setCity(txtCity.getText());
                    customer.setPostalCode(txtPostalCode.getText());
                    customer.setCountry(txtCountry.getText());

                    session.update(customer);
                    tx.commit();
                    JOptionPane.showMessageDialog(this, "Record updated successfully.");
                }
            } else {
                JOptionPane.showMessageDialog(this, "Record not found.");
            }
        } catch (Exception ex) {
            if (tx != null) tx.rollback();
```

```
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error updating record.");
        } finally {
            session.close();
        }
    }

    private void deleteRecord() {
        int id;
        try {
            id = Integer.parseInt(txtId.getText());
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Invalid ID format.");
            return;
        }

        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        try {
            Customer customer = session.get(Customer.class, id);
            if (customer != null) {
                session.delete(customer);
                tx.commit();
                JOptionPane.showMessageDialog(this, "Record deleted successfully.");
            } else {
                JOptionPane.showMessageDialog(this, "Record not found.");
            }
        } catch (Exception ex) {
            if (tx != null) tx.rollback();
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error deleting record.");
        } finally {
            session.close();
        }
    }

    private void viewLastRecord() {
        Session session = HibernateUtil.getSessionFactory().openSession();
        try {
            Query<Customer> query = session.createQuery("FROM Customer ORDER BY id
DESC", Customer.class);
            query.setMaxResults(1);
            Customer customer = query.uniqueResult();
```

```
            if (customer != null) {
                String record = String.format(
                    "ID: %d\nName: %s\nAddress: %s\nContact: %s\nCity: %s\nPostal Code: %s\nCountry: %s",
                    customer.getId(),
                    customer.getName(),
                    customer.getAddress(),
                    customer.getContact(),
                    customer.getCity(),
                    customer.getPostalCode(),
                    customer.getCountry()
                );
                JOptionPane.showMessageDialog(this, record, "Last Record", JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this, "No records found.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error fetching record.");
        } finally {
            session.close();
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            assignment1 app = new assignment1();
            app.setVisible(true);
        });
    }
}
```

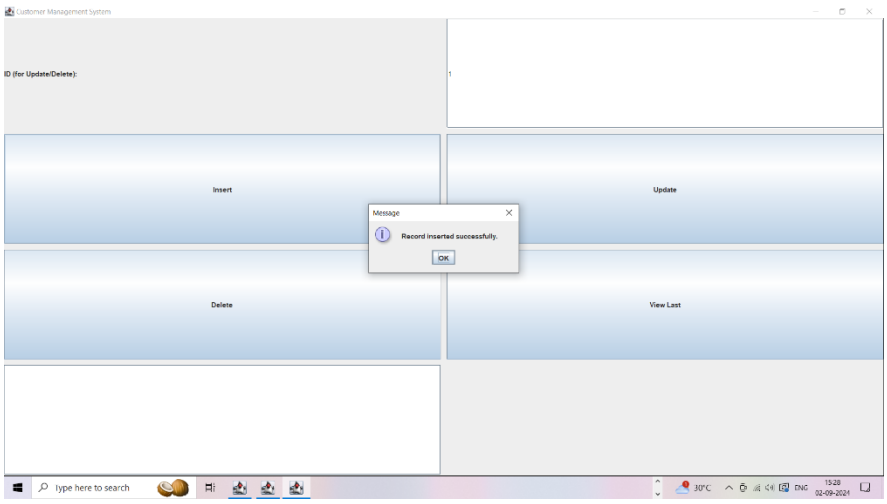
OUTPUT OF INSERT, VIEW, UPDATE AND DELETE ACTION: -

INSERT ACTION

```
    private void showInsertDialog() {
        JTextField txtName = new JTextField();
        JTextField txtAddress = new JTextField();
```

```
        JTextField txtContact = new JTextField();
        JTextField txtCity = new JTextField();
        JTextField txtPostalCode = new JTextField();
        JTextField txtCountry = new JTextField();

        Object[] message = {
            "Name:", txtName,
            "Address:", txtAddress,
            "Contact:", txtContact,
            "City:", txtCity,
            "Postal Code:", txtPostalCode,
            "Country:", txtCountry
        };

        int option = JOptionPane.showConfirmDialog(this, message, "Enter Customer Details",
JOptionPane.OK_CANCEL_OPTION);
        if (option == JOptionPane.OK_OPTION) {
            Session session = HibernateUtil.getSessionFactory().openSession();
            Transaction tx = session.beginTransaction();
            try {
                Customer customer = new Customer();
                customer.setName(txtName.getText());
                customer.setAddress(txtAddress.getText());
                customer.setContact(txtContact.getText());
                customer.setCity(txtCity.getText());
                customer.setPostalCode(txtPostalCode.getText());
                customer.setCountry(txtCountry.getText());

                session.save(customer);
                tx.commit();
                JOptionPane.showMessageDialog(this, "Record inserted successfully.");
            } catch (Exception ex) {
                if (tx != null) tx.rollback();
                ex.printStackTrace();
                JOptionPane.showMessageDialog(this, "Error inserting record.");
            } finally {
                session.close();
            }
        }
    }
```

OUTPUT:

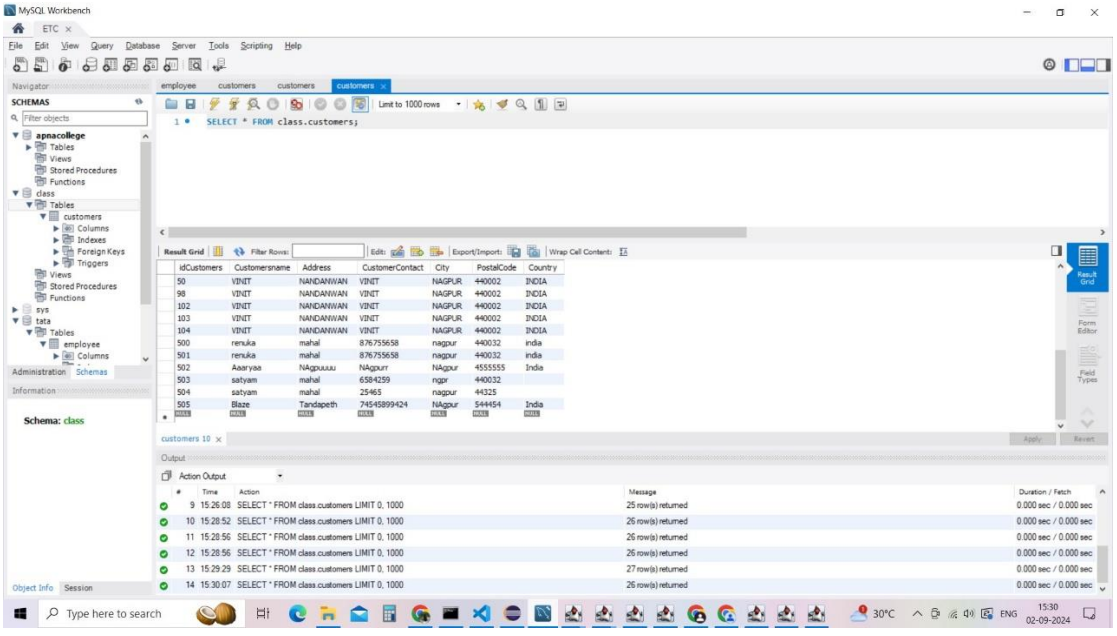


VIEW ACTION

```
 private void viewLastRecord() {
      Session session = HibernateUtil.getSessionFactory().openSession();
      try {
         Query<Customer> query = session.createQuery("FROM Customer ORDER BY id DESC", Customer.class);
         query.setMaxResults(1);
         Customer customer = query.uniqueResult();

         if (customer != null) {
            String record = String.format(
               "ID: %d\nName: %s\nAddress: %s\nContact: %s\nCity: %s\nPostal Code: %s\nCountry: %s",
               customer.getId(),
               customer.getName(),
               customer.getAddress(),
               customer.getContact(),
               customer.getCity(),
               customer.getPostalCode(),
               customer.getCountry()
            );
            JOptionPane.showMessageDialog(this, record, "Last Record", JOptionPane.INFORMATION_MESSAGE);
         } else {
```

```
            JOptionPane.showMessageDialog(this, "No records found.", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error fetching record.");
        } finally {
            session.close();
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            assignment1 app = new assignment1();
            app.setVisible(true);
        });
    }
}
```

OUTPUT:

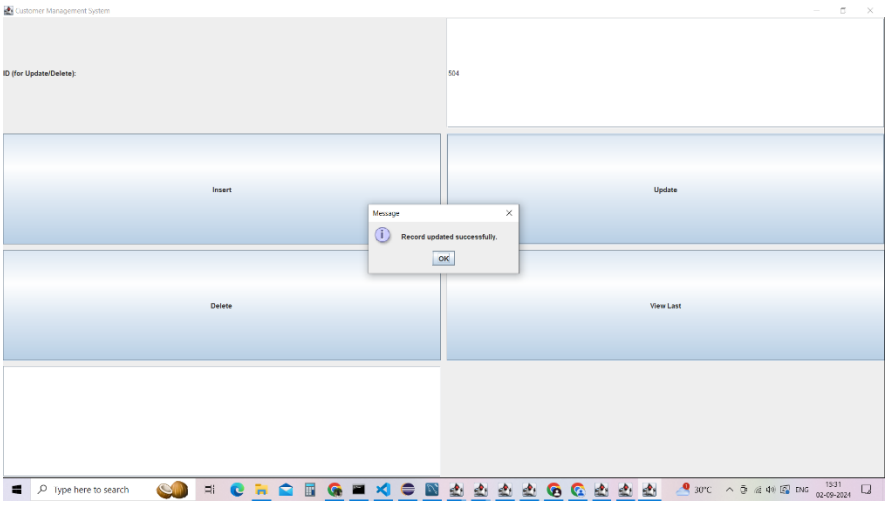


UPDATE ACTION

```
    private void showUpdateDialog() {
        int id;
        try {
            id = Integer.parseInt(txtId.getText());
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Invalid ID format.");
            return;
        }

        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        try {
            Customer customer = session.get(Customer.class, id);
            if (customer != null) {
                JTextField txtName = new JTextField(customer.getName());
                JTextField txtAddress = new JTextField(customer.getAddress());
                JTextField txtContact = new JTextField(customer.getContact());
                JTextField txtCity = new JTextField(customer.getCity());
                JTextField txtPostalCode = new JTextField(customer.getPostalCode());
                JTextField txtCountry = new JTextField(customer.getCountry());

                Object[] message = {
                    "Name:", txtName,
                    "Address:", txtAddress,
                    "Contact:", txtContact,
                    "City:", txtCity,
                    "Postal Code:", txtPostalCode,
                    "Country:", txtCountry
                };

                int option = JOptionPane.showConfirmDialog(this, message, "Update Customer
Details", JOptionPane.OK_CANCEL_OPTION);
                if (option == JOptionPane.OK_OPTION) {
                    customer.setName(txtName.getText());
                    customer.setAddress(txtAddress.getText());
```

```
                customer.setContact(txtContact.getText());
                customer.setCity(txtCity.getText());
                customer.setPostalCode(txtPostalCode.getText());
                customer.setCountry(txtCountry.getText());

                session.update(customer);
                tx.commit();
                JOptionPane.showMessageDialog(this, "Record updated successfully.");
            }
        } else {
            JOptionPane.showMessageDialog(this, "Record not found.");
        }
    } catch (Exception ex) {
        if (tx != null) tx.rollback();
        ex.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error updating record.");
    } finally {
        session.close();
    }
}
```

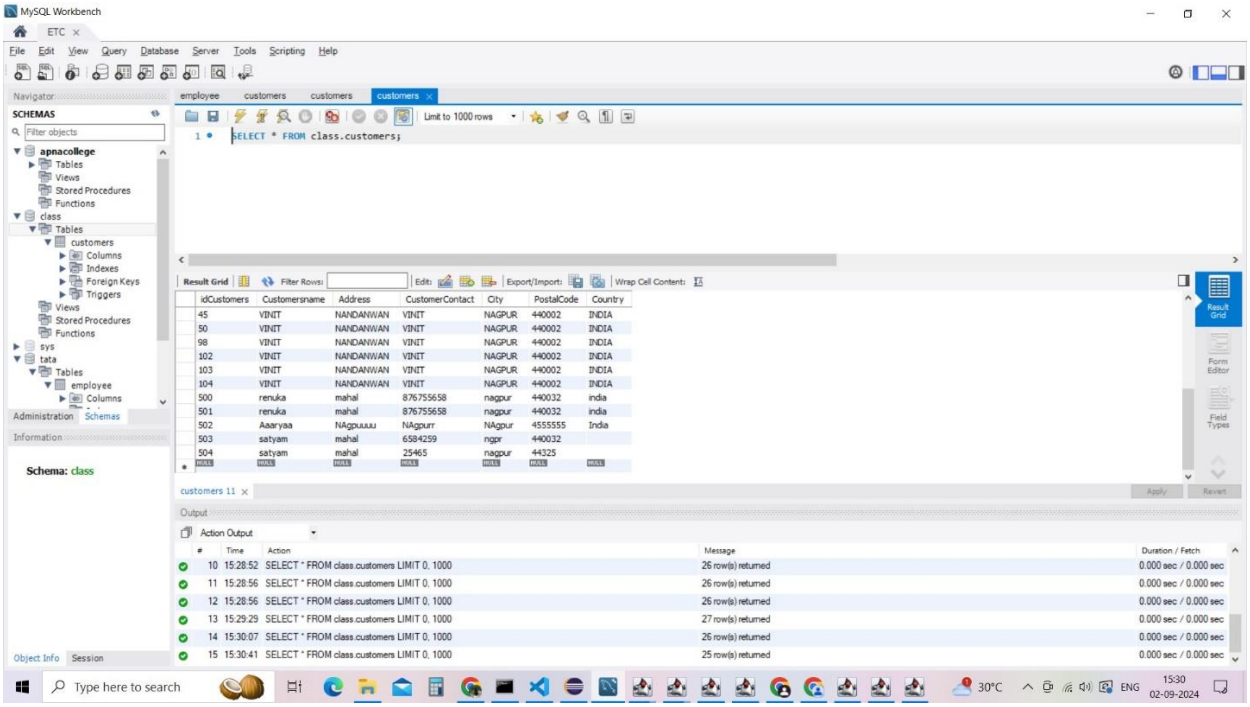
OUTPUT:



DELETE ACTION

```
    private void deleteRecord() {
        int id;
        try {
            id = Integer.parseInt(txtId.getText());
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Invalid ID format.");
            return;
        }

        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        try {
            Customer customer = session.get(Customer.class, id);
            if (customer != null) {
                session.delete(customer);
                tx.commit();
                JOptionPane.showMessageDialog(this, "Record deleted successfully.");
            } else {
                JOptionPane.showMessageDialog(this, "Record not found.");
            }
        } catch (Exception ex) {
            if (tx != null) tx.rollback();
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error deleting record.");
        } finally {
            session.close();
        }
    }
```

OUTPUT:


MySQL Workbench
ETC
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
apnacollege
Tables
Views
Stored Procedures
Functions
class
Tables
customers
Columns
Indexes
Foreign Keys
Triggers
Views
Stored Procedures
Functions
sys
tata
Tables
employee
Columns
Administration Schemas
Information
Schema: class
Object Info Session
employee customers customers customers
Limit to 1000 rows
1 SELECT * FROM class.customers;
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
idCustomers Customersname Address CustomerContact City PostalCode Country
45 VINIT NANDANWAN VINIT NAGPUR 440002 INDIA
50 VINIT NANDANWAN VINIT NAGPUR 440002 INDIA
98 VINIT NANDANWAN VINIT NAGPUR 440002 INDIA
102 VINIT NANDANWAN VINIT NAGPUR 440002 INDIA
103 VINIT NANDANWAN VINIT NAGPUR 440002 INDIA
104 VINIT NANDANWAN VINIT NAGPUR 440002 INDIA
500 renuka mahal 876755658 nagpur 440032 india
501 renuka mahal 876755658 nagpur 440032 india
502 Aaryaa NAgpuuuu NAgpurr NAgpur 4555555 India
503 satyam mahal 6584259 ngpr 440032
504 satyam mahal 25465 nagpur 44325
Result Grid
Form Editor
Field Types
Apply Revert
customers 11
Output
Action Output
Time Action Message Duration / Fetch
10 15:28:52 SELECT * FROM class.customers LIMIT 0, 1000 26 row(s) returned 0.000 sec / 0.000 sec
11 15:28:56 SELECT * FROM class.customers LIMIT 0, 1000 26 row(s) returned 0.000 sec / 0.000 sec
12 15:28:56 SELECT * FROM class.customers LIMIT 0, 1000 26 row(s) returned 0.000 sec / 0.000 sec
13 15:29:29 SELECT * FROM class.customers LIMIT 0, 1000 27 row(s) returned 0.000 sec / 0.000 sec
14 15:30:07 SELECT * FROM class.customers LIMIT 0, 1000 26 row(s) returned 0.000 sec / 0.000 sec
15 15:30:41 SELECT * FROM class.customers LIMIT 0, 1000 25 row(s) returned 0.000 sec / 0.000 sec
Type here to search
30°C
ENG
15:30
02-09-2024