# JDBC + SWING:

1. Create a table called “Employee” which will store employee: Name, ID, Age, Gender, Department no. (You may decide the data types ☺)

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

public class CreateTableExample {

public static void main(String[] args) {

String url = "jdbc:mysql://localhost:3306/your\_database\_name";

String username = "your\_username";

String password = "your\_password";

try (Connection connection = DriverManager.getConnection(url, username, password);

Statement statement = connection.createStatement()) {

String createTableQuery = "CREATE TABLE Employee ("

+ "ID INT PRIMARY KEY,"

+ "Name VARCHAR(100) NOT NULL,"

+ "Age INT,"

+ "Gender VARCHAR(10),"

+ "DepartmentNo INT"

+ ")";

statement.executeUpdate(createTableQuery);

System.out.println("Employee table created successfully!");

} catch (SQLException e) {

System.err.println("Error creating the table: " + e.getMessage());

}

}

}

1. Then create a suitable swing based UI to enter required data to the Employee DB. Note that the user needs to insert all the five details to insert a new employee record to the database. (Insert)

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class EmployeeInsertUI extends JFrame {

private JTextField txtName, txtID, txtAge, txtGender, txtDeptNo;

private JButton btnInsert;

public EmployeeInsertUI() {

setTitle("Insert Employee Record");

setSize(300, 220);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new GridLayout(6, 2, 5, 5));

JLabel lblName = new JLabel("Name:");

txtName = new JTextField();

JLabel lblID = new JLabel("ID:");

txtID = new JTextField();

JLabel lblAge = new JLabel("Age:");

txtAge = new JTextField();

JLabel lblGender = new JLabel("Gender:");

txtGender = new JTextField();

JLabel lblDeptNo = new JLabel("Department No:");

txtDeptNo = new JTextField();

btnInsert = new JButton("Insert");

btnInsert.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

insertEmployeeRecord();

}

});

add(lblName);

add(txtName);

add(lblID);

add(txtID);

add(lblAge);

add(txtAge);

add(lblGender);

add(txtGender);

add(lblDeptNo);

add(txtDeptNo);

add(new JLabel());

add(btnInsert);

}

private void insertEmployeeRecord() {

String name = txtName.getText();

int id = Integer.parseInt(txtID.getText());

int age = Integer.parseInt(txtAge.getText());

String gender = txtGender.getText();

int deptNo = Integer.parseInt(txtDeptNo.getText());

String url = "jdbc:mysql://localhost:3306/your\_database\_name";

String username = "your\_username";

String password = "your\_password";

try (Connection connection = DriverManager.getConnection(url, username, password)) {

String insertQuery = "INSERT INTO Employee (ID, Name, Age, Gender, DepartmentNo) VALUES (?, ?, ?, ?, ?)";

PreparedStatement preparedStatement = connection.prepareStatement(insertQuery);

preparedStatement.setInt(1, id);

preparedStatement.setString(2, name);

preparedStatement.setInt(3, age);

preparedStatement.setString(4, gender);

preparedStatement.setInt(5, deptNo);

int rowsInserted = preparedStatement.executeUpdate();

if (rowsInserted > 0) {

JOptionPane.showMessageDialog(this, "Employee record inserted successfully!");

clearFields();

} else {

JOptionPane.showMessageDialog(this, "Failed to insert employee record.");

}

}

catch (SQLException ex) {

JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());

}

}

private void clearFields() {

txtName.setText("");

txtID.setText("");

txtAge.setText("");

txtGender.setText("");

txtDeptNo.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

EmployeeInsertUI insertUI = new EmployeeInsertUI();

insertUI.setVisible(true);

});

}

}

1. Insert 5 records to the table using the UI.
2. In the same UI create a another internal frame to perform search/update and delete operations.
3. Enable searching with employee ID, load and display the record in an table structure. (Search)
4. Enable editing an existing record and save that to the DB. (Update)
5. Enable selected records to be deleted from the table. (Delete)

**Discussion Questions:**

1. Discuss how exceptions are handled during the process.

The program demonstrated how to handle “FileNotFoundException” while attempting to read a file and “SQLException” during database interactions using try-catch blocks.

“NumberFormatException” was handled when parsing user input data to convert them into appropriate data types.

1. Discuss the main issues occurred during the JDBC and Swing connection process.

Common JDBC issues include database connectivity problems and SQL injection vulnerability. The provided program addressed these concerns by ensuring proper connection and using prepared statements to prevent SQL injection attacks.

For Swing UI, it is important to access and modify components on the Event Dispatch Thread to maintain thread safety and prevent UI freeze.

1. Discuss how you can use threads in this program.

While the provided program didn't heavily utilize threads, we discussed potential use cases for multithreading, such as running long-running tasks, background database operations, and asynchronous tasks

Proper synchronization and careful handling of shared resources are essential when using threads.

Overall, the program demonstrated basic exception handling techniques and database interactions using JDBC, while the Swing UI allowed users to input employee details for insertion into the database. For more complex applications, additional data validation, error handling, and advanced thread usage would be necessary to ensure robustness and efficiency.