**A**

**LAB REPORT**

**ON**

**ADVance java**

**By**

**Prajwal Dahal**

**TU REG no:6-2-388-03-2019**



**Submitted to:**

**Mahendra Aryal**

**Lecturer**

**Kantipur College of Management and Information Technology**

In partial fulfillment of the requirements for the Course

Advance Java

Mid Baneshwor, Kathmandu

June 2023

LAB 1

# Write a program to create login form using swing framework.

Objective:

To create a login frame using swing

Source code:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class LoginForm extends JFrame implements ActionListener {

private JTextField usernameField;

private JPasswordField passwordField;

private JButton loginButton;

public LoginForm() {

setTitle("Login Form");

setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(3, 2));

JLabel usernameLabel = new JLabel("Username:");

usernameField = new JTextField();

JLabel passwordLabel = new JLabel("Password:");

passwordField = new JPasswordField();

loginButton = new JButton("Login");

loginButton.addActionListener(this);

panel.add(usernameLabel);

panel.add(usernameField);

panel.add(passwordLabel);

panel.add(passwordField);

panel.add(new JLabel()); // Empty label for alignment

panel.add(loginButton);

add(panel);

setVisible(true);

}

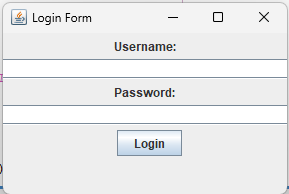
public static void main(String[] args) {

SwingUtilities.invokeLater(LoginForm::new);

}

}

Output



Conclusion:

We have created login form using swing framework.

LAB 2

# Write a program to enable Action to JButton which should fetch username and password from a login form.

Objective:

To create event handler to fetch username and password from login form

Source code:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class LoginForm extends JFrame {

private JTextField usernameField;

private JPasswordField passwordField;

private JButton loginButton;

public LoginForm() {

setTitle("Login Form");

setSize(300, 200);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

JPanel panel = new JPanel();

JLabel usernameLabel = new JLabel("Username:");

usernameField = new JTextField(30);

JLabel passwordLabel = new JLabel("Password:");

passwordField = new JPasswordField(30);

loginButton = new JButton("Login");

loginButton.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

loginButtonActionPerformed();

}

});

panel.add(usernameLabel);

panel.add(usernameField);

panel.add(passwordLabel);

panel.add(passwordField);

panel.add(new JLabel()); // Empty label for alignment

panel.add(loginButton);

add(panel);

setVisible(true);

}

public static void main(String[] args) {

new LoginForm();

}

private void loginButtonActionPerformed() {

String uname=usernameField.getText();

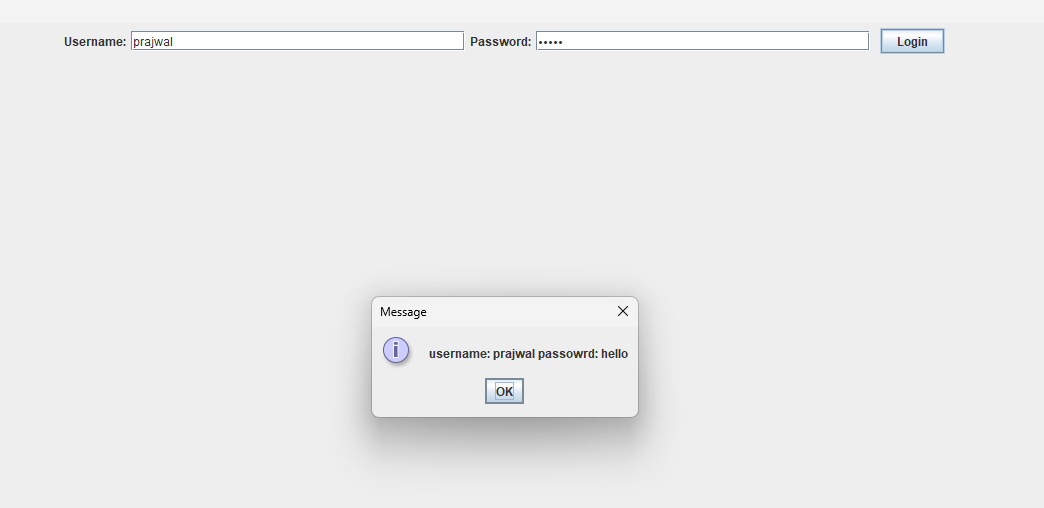
String pwd=passwordField.getText();

JOptionPane.showMessageDialog(null, "username: "+uname+" passowrd: "+pwd);

}

}

Output



Conclusion:

We have created login form using swing framework and extract the textfield data using event handler.

LAB 3

# Write a program to connect to MYSQL database

Objective:

To connect to MYSQL database using JDBC

Source code:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DBConnection {

public DBConnection(){

try {

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

Connection c = DriverManager.getConnection("jdbc:mysql://localhost:3307/mysql","root","");

System.out.println("Connection established");

} catch (Exception ex) {

System.out.println(ex);

}

}

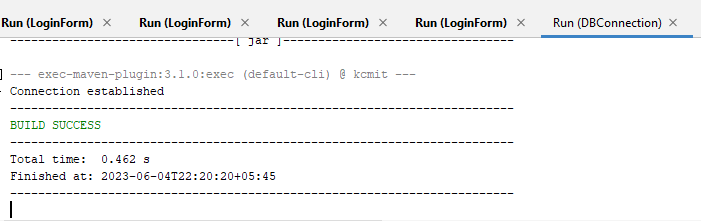
public static void main(String[] args) {

new DBConnection();

}

}

Output:



Conclusion:

We have connect to database using JDBC.

# Write a program to perform all the CRUD operation with JDBC

Objective:

To perform all crud operation with JDBC

Source code:

import java.sql.\*;

public class CRUDOperations {

private static final String URL = "jdbc:mysql://localhost:3307/advancejava";

private static final String USERNAME = "root";

private static final String PASSWORD = "";

public static void main(String[] args) {

// Create a connection

try (Connection connection = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {

// Create a table

createTable(connection);

// Insert a record

int id = insertRecord(connection, "John Doe", "john@example.com");

// Read records

readRecords(connection);

// Update a record

updateRecord(connection, id, "John Smith", "john.smith@example.com");

// Read records again

readRecords(connection);

// Delete a record

deleteRecord(connection, id);

// Read records again

readRecords(connection);

} catch (SQLException e) {

e.printStackTrace();

}

}

private static void createTable(Connection connection) throws SQLException {

String query = "CREATE TABLE IF NOT EXISTS users (id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(100), email VARCHAR(100))";

try (Statement statement = connection.createStatement()) {

statement.execute(query);

System.out.println("Table created successfully.");

}

}

private static int insertRecord(Connection connection, String name, String email) throws SQLException {

String query = "INSERT INTO users (name, email) VALUES (?, ?)";

try (PreparedStatement statement = connection.prepareStatement(query, Statement.RETURN\_GENERATED\_KEYS)) {

statement.setString(1, name);

statement.setString(2, email);

statement.executeUpdate();

ResultSet rs = statement.getGeneratedKeys();

if (rs.next()) {

int id = rs.getInt(1);

System.out.println("Record inserted with ID: " + id);

return id;

}

}

return -1;

}

private static void readRecords(Connection connection) throws SQLException {

String query = "SELECT \* FROM users";

try (Statement statement = connection.createStatement()) {

ResultSet resultSet = statement.executeQuery(query);

while (resultSet.next()) {

int id = resultSet.getInt("id");

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

String email = resultSet.getString("email");

System.out.println("ID: " + id + ", Name: " + name + ", Email: " + email);

}

}

}

private static void updateRecord(Connection connection, int id, String name, String email) throws SQLException {

String query = "UPDATE users SET name = ?, email = ? WHERE id = ?";

try (PreparedStatement statement = connection.prepareStatement(query)) {

statement.setString(1, name);

statement.setString(2, email);

statement.setInt(3, id);

int rowsUpdated = statement.executeUpdate();

System.out.println(rowsUpdated + " record(s) updated.");

}

}

private static void deleteRecord(Connection connection, int id) throws SQLException {

String query = "DELETE FROM users WHERE id = ?";

try (PreparedStatement statement = connection.prepareStatement(query)) {

statement.setInt(1, id);

int rowsDeleted = statement.executeUpdate();

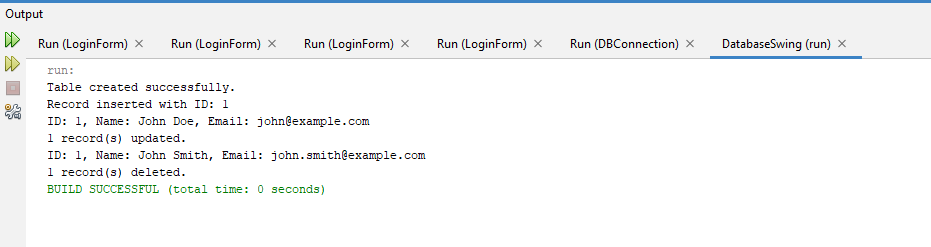
System.out.println(rowsDeleted + " record(s) deleted.");

}

}

}

Output:



Conclusion:

We have performed CRUD operation in database using JDBC.

LAB 4

# Write a program to illustrate application of JavaBeans.

Objective:

To illustrate the application of JavaBeans.

Source code:

public class Person {

private String name;

private int age;

private String address;

public Person() {

// Default constructor

}

// Getters and Setters

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

// toString() method

@Override

public String toString() {

return "Person{" +

"name='" + name + '\'' +

", age=" + age +

", address='" + address + '\'' +

'}';

}

}

public class Main {

public static void main(String[] args) {

// Create a new Person object

Person person = new Person();

// Set the values using the setter methods

person.setName("John Doe");

person.setAge(30);

person.setAddress("123 Main Street");

// Access the values using the getter methods

String name = person.getName();

int age = person.getAge();

String address = person.getAddress();

// Print the values

System.out.println("Name: " + name);

System.out.println("Age: " + age);

System.out.println("Address: " + address);

// Update the values using the setter methods

person.setName("Jane Smith");

person.setAge(25);

person.setAddress("456 Elm Street");

// Access the updated values

name = person.getName();

age = person.getAge();

address = person.getAddress();

// Print the updated values

System.out.println("Name: " + name);

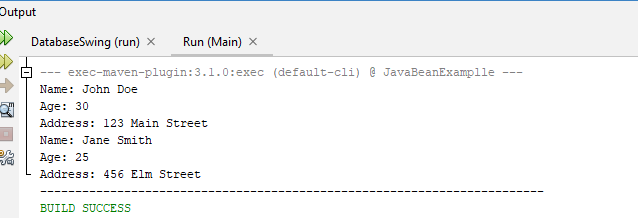
System.out.println("Age: " + age);

System.out.println("Address: " + address);

}

}

Output:



Conclusion:

We have illustrate the example of JavaBean.

LAB 5

# Write a program to initialize servlet and perform basic reading and writing using JSP.

Objective:

To initialize servlet and perform basic reading and writing operation.

Source code:

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\*

\* @author prajwal

\*/

@WebServlet(urlPatterns = {"/MyServlet"})

public class MyServlet extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

System.out.println(request.getParameter("num"));

int x =Integer.parseInt(request.getParameter("num"));

int y=Integer.parseInt(request.getParameter("num2"));

response.getWriter().println(x+y);

}

@Override

public void service(HttpServletRequest request, HttpServletResponse response)

throws IOException, ServletException {

processRequest(request, response);

}

}

<%@page contentType="text/html" pageEncoding="windows-1252"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Login Page</title>

</head>

<body>

<form action="MyServlet">

<div><input type="text" placeholder="enter 1st number" name="num"/></div>

<div><input type="text" placeholder="enter 2nd number" name="num2"/></div>

<div><input type="submit" value="login" name="submit"/></div>

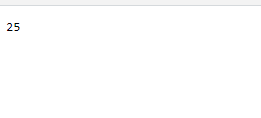
</form>

</body>

</html>

Output:





Conclusion:

We have perform basic read and write operation using JSP from servlet.

LAB 6

# Write a program to handle session and cookie in JSP.

Objective:

To handle to session and cookie in JSP.

Source code:

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%

// Create a new cookie

Cookie newCookie = new Cookie("myCookie", "what");

newCookie.setMaxAge(3600); // Set the cookie's maximum age in seconds (e.g., 1 hour)

newCookie.setPath("/"); // Set the cookie's path ("/" means it is accessible across the entire website)

response.addCookie(newCookie);

//retrieve

out.println("Cookie:"+newCookie.getValue());

//deleteCookie

newCookie.setMaxAge(0);

response.addCookie(newCookie);

//create

session.setAttribute("username","prajwal");

//retrieve

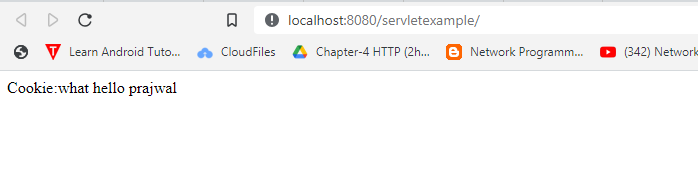
out.println("hello "+session.getAttribute("username"));

//delete

session.removeAttribute("username");

%>

Output:



Conclusion:

We have handle session and cookie.

LAB 7

# Write a program to demonstrate RMI

Objective:

To demonstrate RMI in java

Source code:

package com.mycompany.rmiexample;

import java.rmi.RemoteException;

import java.rmi.registry.\*;

import java.util.Scanner;

import java.rmi.server.UnicastRemoteObject;

import java.rmi.\*;

public interface MessageServices extends Remote {

public void sendMessage(String msg) throws RemoteException;

public String recieveMessage()throws RemoteException;

}

public class RemoteAdder extends UnicastRemoteObject implements MessageServices{

private String message;

public RemoteAdder() throws RemoteException{

}

@Override

public void sendMessage(String msg) throws RemoteException {

this.message=msg;

}

@Override

public String recieveMessage() throws RemoteException {

return message;

}

}

public class Server {

public static void main(String[] args) {

try {

Registry reg = LocateRegistry.createRegistry(9999);

MessageServices stub = new RemoteAdder();

reg.rebind("msg",stub);

Scanner sc = new Scanner(System.in);

System.out.println("enter a message to send: ");

String msg = sc.nextLine();

stub.sendMessage(msg);

} catch (RemoteException ex) {

System.out.println(ex.toString());

}

}

}

public class Client {

public static void main(String[] args) {

try {

Registry reg =LocateRegistry.getRegistry("127.0.0.1",9999);

MessageServices skeleton=(MessageServices) reg.lookup("msg");

System.out.println("msg= "+ skeleton.recieveMessage());

} catch (NotBoundException | RemoteException ex) {

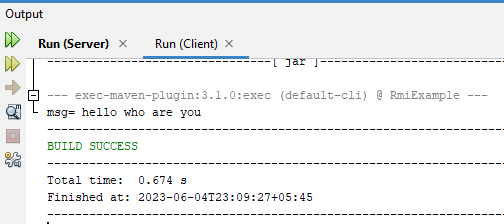
System.out.println(ex.toString());

}

}

}

Output:



Conclusion:

We have demonstrate RMI in java.