# Task MVC:

## Controller:

package MVC;

public class c {

m objm;

v objv;

c(m mo, v vo)

{

objm=mo;

objv=vo;

}

void runner()

{

while(true)

{

int ch =objv.getop();

if(ch==5)

{

objv.exdis();

}

int a=objv.getv1();

int b=objv.getv2();

switch(ch)

{

case 1:

{

int result = objm.add(a,b);

objv.disre(result);

break;

}

case 2:

{

int result = objm.sub(a,b);

objv.disre(result);

break;

}

case 3:

{

int result = objm.mul(a,b);

objv.disre(result);

break;

}

case 4:

{

int result = objm.div(a,b);

objv.disre(result);

break;

}

}

}

}

}

## Model

package MVC;

public class m {

public int add(int a, int b )

{

return a+b;

}

public int sub(int a, int b )

{

return a-b;

}

public int mul(int a, int b )

{

return a\*b;

}

public int div(int a, int b )

{

return a/b;

}

}

## View

package MVC;

import java.util.Scanner;

public class v {

Scanner scanner = new Scanner(System.***in***);

public int getop()

{

System.***out***.println("Enter operation\n");

int ch=scanner.nextInt();

return ch;

}

public int getv1()

{

System.***out***.println("Enter variable 1\n");

int ch=scanner.nextInt();

return ch;

}

public int getv2()

{

System.***out***.println("Enter v2\n");

int ch=scanner.nextInt();

return ch;

}

public void disre(int r)

{

System.***out***.println("Result : "+ r);

}

public void exdis()

{

System.***out***.println("Exit\n");

}

}

## Main

package MVC;

public class m {

public int add(int a, int b )

{

return a+b;

}

public int sub(int a, int b )

{

return a-b;

}

public int mul(int a, int b )

{

return a\*b;

}

public int div(int a, int b )

{

return a/b;

}

}

# Task MVP

## Model

package MVP;

public class M {

public int add(int a, int b )

{

return a+b;

}

public int sub(int a, int b )

{

return a-b;

}

public int mul(int a, int b )

{

return a\*b;

}

public int div(int a, int b )

{

return a/b;

}

}

## View

package MVP;

import java.util.Scanner;

public class V {

Scanner scanner = new Scanner(System.***in***);

public int getop(String p)

{

System.***out***.println(p);

int ch=scanner.nextInt();

return ch;

}

public int getv1(String p)

{

System.***out***.println(p);

int ch=scanner.nextInt();

return ch;

}

public int getv2(String p)

{

System.***out***.println(p);

int ch=scanner.nextInt();

return ch;

}

public void disre(String r)

{

System.***out***.println( r);

}

public void exdis(String p)

{

System.***out***.println(p);

}

}

## Presenter

package MVP;

import MVC.m;

import MVC.v;

public class p {

M objm;

V objv;

p(M mo, V vo)

{

objm=mo;

objv=vo;

}

void runner()

{

while(true)

{

int ch =objv.getop("Enter operation\n");

if(ch==5)

{

objv.exdis("Exit\n");

}

int a=objv.getv1("Enter v1\n");

int b=objv.getv2("Enter v2\n");

switch(ch)

{

case 1:

{

int result = objm.add(a,b);

objv.disre("Result is : "+result);

break;

}

case 2:

{

int result = objm.sub(a,b);

objv.disre("Result is : "+result);

break;

}

case 3:

{

int result = objm.mul(a,b);

objv.disre("Result is : "+result);

break;

}

case 4:

{

int result = objm.div(a,b);

objv.disre("Result is : "+result);

break;

}

}

}

}

}

## Main

package MVP;

public class ma {

public static void main(String [] args)

{

V objv = new V();

M objm = new M();

p objc = new p(objm,objv);

objc.runner();

}

}

# Task observer:

oid notifyObservers(String operation, int a, int b, int result);

}

// ObserverView interface

interface ObserverView {

void displayResult(String operation, int a, int b, int result);

void displayError(String message);

}

// Subject class (Model)

class CalculatorModel implements ObserverModel {

private final List<Observer> observers = new ArrayList<>();

@Override

public void addObserver(Observer observer) {

observers.add(observer);

}

@Override

public void removeObserver(Observer observer) {

observers.remove(observer);

}

@Override

public void notifyObservers(String operation, int a, int b, int result) {

for (Observer observer : observers) {

observer.update(operation, a, b, result);

}

}

// Operations

public int add(int a, int b) {

int result = a + b;

notifyObservers("Addition", a, b, result);

return result;

}

public int sub(int a, int b) {

int result = a - b;

notifyObservers("Subtraction", a, b, result);

return result;

}

public int mul(int a, int b) {

int result = a \* b;

notifyObservers("Multiplication", a, b, result);

return result;

}

public int div(int a, int b) {

if (b == 0) {

throw new ArithmeticException("Division by zero is not allowed");

}

int result = a / b;

notifyObservers("Division", a, b, result);

return result;

}

}

// View class

class CalculatorView implements ObserverView {

@Override

public void displayResult(String operation, int a, int b, int result) {

System.out.println("Operation: " + operation + ", A: " + a + ", B: " + b + ", Result: " + result);

}

@Override

public void displayError(String message) {

System.out.println("Error: " + message);

}

}

// Controller class

class CalculatorController implements Observer {

private final CalculatorModel model;

private final CalculatorView view;

public CalculatorController(CalculatorModel model, CalculatorView view) {

this.model = model;

this.view = view;

this.model.addObserver(this);

}

public void performOperation(int choice, int a, int b) {

try {

switch (choice) {

case 1 -> model.add(a, b);

case 2 -> model.sub(a, b);

case 3 -> model.mul(a, b);

case 4 -> model.div(a, b);

default -> view.displayError("Invalid choice");

}

} catch (ArithmeticException e) {

view.displayError(e.getMessage());

}

}

@Override

public void update(String operation, int a, int b, int result) {

view.displayResult(operation, a, b, result);

}

}

// Main class

public class ObserverPatternCalculatorMVC {

public static void main(String[] args) {

CalculatorModel model = new CalculatorModel();

CalculatorView view = new CalculatorView();

CalculatorController controller = new CalculatorController(model, view);

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("Enter Choice:\n1. Add\n2. Sub\n3. Mul\n4. Div\n5. Exit\n");

int choice = scanner.nextInt();

if (choice == 5) {

scanner.close();

break;

}

System.out.println("Enter A:");

int a = scanner.nextInt();

System.out.println("Enter B:");

int b = scanner.nextInt();

controller.performOperation(choice, a, b);

}

}

}

# Task Jtable

package jtab;

import java.awt.BorderLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.table.DefaultTableModel;

public class jt {

// Calculator operations

private static int add(int a, int b) {

return a + b;

}

private static int sub(int a, int b) {

return a - b;

}

private static int mul(int a, int b) {

return a \* b;

}

private static int div(int a, int b) {

if (b == 0) {

throw new ArithmeticException("Division by zero is not allowed");

}

return a / b;

}

public static void main(String[] args) {

// Create JFrame

JFrame frame = new JFrame("Calculator with JTable");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(600, 400);

frame.setLayout(new BorderLayout());

// Create table model and JTable

String[] columns = { "Operation", "A", "B", "Result" };

DefaultTableModel tableModel = new DefaultTableModel(columns, 0);

JTable table = new JTable(tableModel);

// Add JScrollPane for the table

JScrollPane scrollPane = new JScrollPane(table);

frame.add(scrollPane, BorderLayout.CENTER);

// Create panel for inputs and buttons

JPanel inputPanel = new JPanel(new GridLayout(2, 2, 10, 10));

// Dropdown for operations

String[] operations = { "Add", "Sub", "Mul", "Div" };

JComboBox<String> operationDropdown = new JComboBox<>(operations);

inputPanel.add(new JLabel("Select Operation:"));

inputPanel.add(operationDropdown);

// Text fields for input A and B

JTextField inputA = new JTextField();

JTextField inputB = new JTextField();

inputPanel.add(new JLabel("Enter A:"));

inputPanel.add(inputA);

inputPanel.add(new JLabel("Enter B:"));

inputPanel.add(inputB);

frame.add(inputPanel, BorderLayout.NORTH);

// Add Calculate Button

JButton calculateButton = new JButton("Calculate");

frame.add(calculateButton, BorderLayout.SOUTH);

// Add action listener to the button

calculateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

// Get inputs and operation

String operation = (String) operationDropdown.getSelectedItem();

int a = Integer.parseInt(inputA.getText());

int b = Integer.parseInt(inputB.getText());

int result = 0;

// Perform operation

switch (operation) {

case "Add":

result = add(a, b);

break;

case "Sub":

result = sub(a, b);

break;

case "Mul":

result = mul(a, b);

break;

case "Div":

result = div(a, b);

break;

}

// Add result to the table

tableModel.addRow(new Object[] { operation, a, b, result });

} catch (ArithmeticException ex) {

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

JOptionPane.ERROR\_MESSAGE);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(frame, "Please enter valid numbers for A and B", "Input Error",

JOptionPane.WARNING\_MESSAGE);

}

}

});

// Set frame visible

frame.setVisible(true);

}

}

# Task Layered

## Bl

package cal\_Layered;

public class cal\_bl {

public int add(int a, int b )

{

return a+b;

}

public int sub(int a, int b )

{

return a-b;

}

public int mul(int a, int b )

{

return a\*b;

}

public int div(int a, int b )

{

return a/b;

}

}

## Pre

package cal\_Layered;

import java.util.Scanner;

public class cal\_pre {

public static void main(String[] args)

{

cal\_bl obj=new cal\_bl();

Scanner scanner=new Scanner(System.***in***);

while(true)

{

System.***out***.println("Enter Choise\n1. Add\n2. Sub\n3. Mul\n4. Div\n");

int ch=scanner.nextInt();

System.***out***.println("Enter A\n");

int a=scanner.nextInt();

System.***out***.println("Enter B\n");

int b=scanner.nextInt();

switch(ch)

{

case 1:

{

int result = obj.add(a,b);

System.***out***.println("Result is : "+result);

break;

}

case 2:

{

int result = obj.sub(a,b);

System.***out***.println("Result is : "+result);

break;

}

case 3:

{

int result = obj.mul(a,b);

System.***out***.println("Result is : "+result);

break;

}

case 4:

{

int result = obj.div(a,b);

System.***out***.println("Result is : "+result);

break;

}

case 5:

{

scanner.close();

return;

}

default:

System.***out***.println("Invalid\n");

}

}

}

}

# Task threading

package threading\_cal;

import java.util.Scanner;

import cal\_Layered.cal\_bl;

public class pre {

public static void main(String[] args)

{

cal\_bl obj=new cal\_bl();

Scanner scanner=new Scanner(System.in);

while(true)

{

System.out.println("Enter Choise\n1. Add\n2. Sub\n3. Mul\n4. Div\n");

int ch=scanner.nextInt();

System.out.println("Enter A\n");

int a=scanner.nextInt();

System.out.println("Enter B\n");

int b=scanner.nextInt();

if(ch==5)

{

scanner.close();

return;

}

Runnable task = ()->

{

switch(ch)

{

case 1:

{

int result = obj.add(a,b);

System.out.println("Result is : "+result);

break;

}

case 2:

{

int result = obj.sub(a,b);

System.out.println("Result is : "+result);

break;

}

case 3:

{

int result = obj.mul(a,b);

System.out.println("Result is : "+result);

break;

}

case 4:

{

int result = obj.div(a,b);

System.out.println("Result is : "+result);

break;

}

default:

System.out.println("Invalid\n");

}

};

Thread thread=new Thread();

thread.start();

}

}

}

## Task2 threading

class SharedResource {

private int counter = 0;

// Synchronized method to increment the counter

public synchronized void increment() {

try {

// Simulating some delay

Thread.sleep(100);

} catch (InterruptedException e) {

System.out.println("Thread interrupted during sleep: " + e.getMessage());

}

counter++;

System.out.println(Thread.currentThread().getName() + " incremented counter to: " + counter);

}

}

class IncrementThread extends Thread {

private final SharedResource resource;

public IncrementThread(SharedResource resource) {

this.resource = resource;

}

@Override

public void run() {

for (int i = 0; i < 5; i++) { // Each thread increments 5 times

resource.increment();

try {

Thread.sleep(50); // Simulating a small delay between operations

} catch (InterruptedException e) {

System.out.println("Thread interrupted: " + e.getMessage());

}

}

}

}

public class SynchronizedExample {

public static void main(String[] args) {

SharedResource sharedResource = new SharedResource();

// Create multiple threads sharing the same resource

Thread thread1 = new IncrementThread(sharedResource);

Thread thread2 = new IncrementThread(sharedResource);

thread1.setName("Thread-1");

thread2.setName("Thread-2");

thread1.start();

thread2.start();

}

}

## Task 3 thread

class SharedResource {

private int counter = 0;

// Synchronized method to increment the counter

public synchronized void increment() {

counter++;

System.out.println(Thread.currentThread().getName() + " incremented counter to: " + counter);

}

}

class IncrementThread extends Thread {

private final SharedResource resource;

public IncrementThread(SharedResource resource) {

this.resource = resource;

}

@Override

public void run() {

for (int i = 0; i < 5; i++) { // Each thread increments 5 times

resource.increment();

try {

Thread.sleep(100); // Simulate some delay

} catch (InterruptedException e) {

System.out.println("Thread interrupted: " + e.getMessage());

}

}

}

}

public class SynchronizedExample {

public static void main(String[] args) {

SharedResource sharedResource = new SharedResource();

// Create multiple threads sharing the same resource

Thread thread1 = new IncrementThread(sharedResource);

Thread thread2 = new IncrementThread(sharedResource);

thread1.setName("Thread-1");

thread2.setName("Thread-2");

thread1.start();

thread2.start();

}

}

# Task logger

Log4j2.xml file:

&lt;Configuration status=&quot;WARN&quot;&gt;

&lt;Appenders&gt;

&lt;Console name=&quot;Console&quot; target=&quot;SYSTEM\_OUT&quot;&gt;

&lt;PatternLayout pattern=&quot;%d{yyyy-MM-dd HH:mm:ss} %p %c{1} - %m%n&quot;/&gt;

&lt;/Console&gt;

&lt;File name=&quot;FileLogger&quot; fileName=&quot;logs\_folder/Bugs\_Tracker.log&quot; append=&quot;true&quot;&gt;

&lt;PatternLayout pattern=&quot;%d{yyyy-MM-dd HH:mm:ss} %p %c{1} - %m%n&quot;/&gt;

&lt;/File&gt;

&lt;/Appenders&gt;

&lt;Loggers&gt;

&lt;Root level=&quot;info&quot;&gt;

&lt;AppenderRef ref=&quot;Console&quot;/&gt;

&lt;AppenderRef ref=&quot;FileLogger&quot;/&gt;

&lt;/Root&gt;

&lt;/Loggers&gt;

&lt;/Configuration&gt;

Bugc class:

public class Bugc {

private int id;

private String title;

private String description;

private String severity;

private String status;

public Bugc(int id, String title, String description, String severity, String status) {

this.id = id;

this.title = title;

this.description = description;

this.severity = severity;

this.status = status;

}

public int getId() {

return id;

}

public String getTitle() {

return title;

}

public String getSeverity() {

return severity;

}

public String getStatus() {

return status;

}

@Override

public String toString() {

return &quot;Bug ID: &quot; + id + &quot;, Title: &quot; + title + &quot;, Severity: &quot; + severity + &quot;, Status: &quot; + status;

}

}

Logger\_Bug class:

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

import java.util.ArrayList;

import java.util.List;

public class Logger\_Bug {

private static final Logger logger = LogManager.getLogger(Logger\_Bug.class);

private int bugIdCounter = 1;

private List&lt;Bugc&gt; bugs = new ArrayList&lt;&gt;();

public void logBug(String title, String description, String severity, String status) {

if (title == null || description == null || severity == null || status == null) {

logger.warn(&quot;Missing required bug information.&quot;);

return;

}

Bugc bug = new Bugc(bugIdCounter++, title, description, severity, status);

bugs.add(bug);

logger.info(&quot;New bug created: [Bug ID: {}, Title: {}, Severity: {}]&quot;, bug.getId(), title, severity);

System.out.println(&quot;Logged Bug is : &quot; + bug);

}

public void getAllBugs() {

logger.info(&quot;Fetching all bugs...&quot;);

long startTime = System.currentTimeMillis();

if (bugs.isEmpty()) {

logger.warn(&quot;No bugs found.&quot;);

return;

}

for (Bugc bug : bugs) {

System.out.println(bug);

}

long executionTime = System.currentTimeMillis() - startTime;

logger.info(&quot;Fetched {} bugs in {} ms.&quot;, bugs.size(), executionTime);

}

public void searchBugs(String criteria) {

logger.debug(&quot;Search bugs with amazing criteria {}&quot;, criteria);

List&lt;Bugc&gt; results = new ArrayList&lt;&gt;();

for (Bugc bug : bugs) {

if (bug.getTitle().equalsIgnoreCase(criteria) ||

bug.getStatus().equalsIgnoreCase(criteria) ||

bug.getSeverity().equalsIgnoreCase(criteria)) {

results.add(bug);

}

}

if (results.isEmpty()) {

logger.warn(&quot;No bugs found matching criteria: {}&quot;, criteria);

} else {

logger.info(&quot;Found {} bugs matching criteria: {}&quot;, results.size(), criteria);

for (Bugc bug : results) {

System.out.println(bug);

}

}

}

}

Main class:

import java.util.Scanner;

public class MainC {

public static void main(String[] args) {

Logger\_Bug bugLogger = new Logger\_Bug();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println(&quot;\nBug Tracking System&quot;);

System.out.println(&quot;1. Create a new bug&quot;);

System.out.println(&quot;2. Display all bugs&quot;);

System.out.println(&quot;3. Search for bugs&quot;);

System.out.println(&quot;4. End Program :(&quot;);

System.out.print(&quot;Choose : &quot;);

int choice = scanner.nextInt();

scanner.nextLine(); // Consume newline

switch (choice) {

case 1:

System.out.print(&quot;Enter Title for bug: &quot;);

String title = scanner.nextLine();

System.out.print(&quot;Enter description: &quot;);

String description = scanner.nextLine();

System.out.print(&quot;Enter severity (Low, Medium, High): &quot;);

String severity = scanner.nextLine();

System.out.print(&quot;Enter status (Open, In Progress, Resolved):) &quot;);

String status = scanner.nextLine();

bugLogger.logBug(title, description, severity, status);

break;

case 2:

bugLogger.getAllBugs();

break;

case 3:

System.out.print(&quot;Search With (Title, Status, Severity): &quot;);

String criteria = scanner.nextLine();

bugLogger.searchBugs(criteria);

break;

case 4:

System.out.println(&quot;Exiting...&quot;);

scanner.close();

return;

default:

System.out.println(&quot;Invalid&quot;);

}

}

}

}

# Task db

Database Connection class:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class Database\_Connect {

// MariaDB connection URL format

private static final String URL = &quot;jdbc:mariadb://localhost:3306/student\_management&quot;;

private static final String USER = &quot;root&quot;;

private static final String PASSWORD = &quot;3718&quot;;

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(URL, USER, PASSWORD);

}

public static void closeConnection(Connection connection) {

if (connection != null) {

try {

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

Crud operation Class

import java.sql.\*;

public class CRUD\_operation {

private static final String URL = &quot;jdbc:mariadb://localhost:3306/student\_management&quot;;

private static final String USER = &quot;root&quot;;

private static final String PASSWORD = &quot;3718&quot;;

// Establish a connection to the database

private Connection getConnection() throws SQLException {

return DriverManager.getConnection(URL, USER, PASSWORD);

}

// Add a new student

public void addStudent(String name, String email, String department) {

String query = &quot;INSERT INTO fast\_student (name, email, department) VALUES (?, ?, ?)&quot;;

try (Connection connection = getConnection();

PreparedStatement stmt = connection.prepareStatement(query)) {

stmt.setString(1, name);

stmt.setString(2, email);

stmt.setString(3, department);

int rowsInserted = stmt.executeUpdate();

System.out.println(&quot;fast\_student added successfully. Rows affected: &quot; + rowsInserted);

} catch (SQLException e) {

e.printStackTrace();

}

}

// Display all students

public void displayStudents() {

String query = &quot;SELECT \* FROM fast\_student&quot;;

try (Connection connection = getConnection();

Statement stmt = connection.createStatement();

ResultSet resultSet = stmt.executeQuery(query)) {

while (resultSet.next()) {

System.out.println(&quot;ID: &quot; + resultSet.getInt(&quot;student\_id&quot;));

System.out.println(&quot;Name: &quot; + resultSet.getString(&quot;name&quot;));

System.out.println(&quot;Email: &quot; + resultSet.getString(&quot;email&quot;));

System.out.println(&quot;Department: &quot; + resultSet.getString(&quot;department&quot;));

System.out.println(&quot;-----------------------&quot;);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

// Update an existing student

public void updateStudent(int studentId, String newEmail, String newDepartment) {

String query = &quot;UPDATE fast\_student SET email = ?, department = ? WHERE student\_id = ?&quot;;

try (Connection connection = getConnection();

PreparedStatement stmt = connection.prepareStatement(query)) {

stmt.setString(1, newEmail);

stmt.setString(2, newDepartment);

stmt.setInt(3, studentId);

int rowsUpdated = stmt.executeUpdate();

System.out.println(&quot;fast\_student updated successfully. Rows affected: &quot; + rowsUpdated);

} catch (SQLException e) {

e.printStackTrace();

}

}

// Delete a student by ID

public void deleteStudent(int studentId) {

String query = &quot;DELETE FROM fast\_student WHERE student\_id = ?&quot;;

try (Connection connection = getConnection();

PreparedStatement stmt = connection.prepareStatement(query)) {

stmt.setInt(1, studentId);

int rowsDeleted = stmt.executeUpdate();

System.out.println(&quot;fast\_student deleted successfully. Rows affected: &quot; + rowsDeleted);

} catch (SQLException e) {

e.printStackTrace();

}

}

}

Main Class:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class Main extends JFrame {

// Instantiate StudentDAO to handle CRUD operations

private CRUD\_operation studentDAO = new CRUD\_operation();

// Text fields for input

private JTextField nameField;

private JTextField emailField;

private JTextField departmentField;

private JTextField studentIdField;

public Main() {

setTitle(&quot;Student Management System&quot;);

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new GridLayout(6, 2)); // 6 rows, 2 columns

// Create input fields for student details

nameField = new JTextField(15);

emailField = new JTextField(15);

departmentField = new JTextField(15);

studentIdField = new JTextField(5);

// Add labels and text fields to the GUI in a grid layout

add(new JLabel(&quot;Name:&quot;));

add(nameField);

add(new JLabel(&quot;Student ID:&quot;));

add(studentIdField);

add(new JLabel(&quot;Email:&quot;));

add(emailField);

add(new JLabel(&quot;Department:&quot;));

add(departmentField);

// Button to display all students

JButton displayButton = new JButton(&quot;Display All Students&quot;);

displayButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

studentDAO.displayStudents();

}

});

add(displayButton); // Add to grid layout

// Button to add a student

JButton addButton = new JButton(&quot;Add Student&quot;);

addButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String name = nameField.getText();

String email = emailField.getText();

String department = departmentField.getText();

studentDAO.addStudent(name, email, department);

// Clear fields after addition

nameField.setText(&quot;&quot;);

emailField.setText(&quot;&quot;);

departmentField.setText(&quot;&quot;);

}

});

add(addButton); // Add to grid layout

// Button to update a student

JButton updateButton = new JButton(&quot;Update Student&quot;);

updateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int studentId = Integer.parseInt(studentIdField.getText());

String newEmail = emailField.getText();

String newDepartment = departmentField.getText();

studentDAO.updateStudent(studentId, newEmail, newDepartment);

// Clear fields after update

studentIdField.setText(&quot;&quot;);

emailField.setText(&quot;&quot;);

departmentField.setText(&quot;&quot;);

}

});

add(updateButton); // Add to grid layout

// Button to delete a student

JButton deleteButton = new JButton(&quot;Delete Student&quot;);

deleteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int studentId = Integer.parseInt(studentIdField.getText());

studentDAO.deleteStudent(studentId); // Change this ID to delete a specific student

// Clear student ID field after deletion

studentIdField.setText(&quot;&quot;);

}

});

add(deleteButton); // Add to grid layout

setVisible(true);

}

public static void main(String[] args) {

// Instantiate the GUI

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

new Main(); // Start the GUI

}

});

}

}