**PROGRAM CODING:**

**CollegeManagementApp.java**

package collegemanagement;

import org.jdesktop.application.Application;

import org.jdesktop.application.SingleFrameApplication;

public class CollegeManagementApp extends SingleFrameApplication {

@Override protected void startup() {

show(new CollegeManagementView(this));

}

@Override protected void configureWindow(java.awt.Window root) {

}

public static CollegeManagementApp getApplication() {

return Application.getInstance(CollegeManagementApp.class);

}

public static void main(String[] args) {

launch(CollegeManagementApp.class, args);

}

}

**CollegeManagementView.java**

package collegemanagement;

import org.jdesktop.application.Action;

import org.jdesktop.application.ResourceMap;

import org.jdesktop.application.SingleFrameApplication;

import org.jdesktop.application.FrameView;

import org.jdesktop.application.TaskMonitor;

import org.jdesktop.application.Task;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.RollbackException;

import javax.swing.Timer;

import javax.swing.Icon;

public class CollegeManagementView extends FrameView {

public CollegeManagementView(SingleFrameApplication app) {

super(app);

initComponents();

// status bar initialization - message timeout, idle icon and busy animation, etc

ResourceMap resourceMap = getResourceMap();

int messageTimeout = resourceMap.getInteger("StatusBar.messageTimeout");

messageTimer = new Timer(messageTimeout, new ActionListener() {

public void actionPerformed(ActionEvent e) {

statusMessageLabel.setText("");

}

});

messageTimer.setRepeats(false);

int busyAnimationRate = resourceMap.getInteger("StatusBar.busyAnimationRate");

for (int i = 0; i < busyIcons.length; i++) {

busyIcons[i] = resourceMap.getIcon("StatusBar.busyIcons[" + i + "]");

}

busyIconTimer = new Timer(busyAnimationRate, new ActionListener() {

public void actionPerformed(ActionEvent e) {

busyIconIndex = (busyIconIndex + 1) % busyIcons.length;

statusAnimationLabel.setIcon(busyIcons[busyIconIndex]);

}

});

idleIcon = resourceMap.getIcon("StatusBar.idleIcon");

statusAnimationLabel.setIcon(idleIcon);

progressBar.setVisible(false);

// connecting action tasks to status bar via TaskMonitor

TaskMonitor taskMonitor = new TaskMonitor(getApplication().getContext());

taskMonitor.addPropertyChangeListener(new java.beans.PropertyChangeListener() {

public void propertyChange(java.beans.PropertyChangeEvent evt) {

String propertyName = evt.getPropertyName();

if ("started".equals(propertyName)) {

if (!busyIconTimer.isRunning()) {

statusAnimationLabel.setIcon(busyIcons[0]);

busyIconIndex = 0;

busyIconTimer.start();

}

progressBar.setVisible(true);

progressBar.setIndeterminate(true);

} else if ("done".equals(propertyName)) {

busyIconTimer.stop();

statusAnimationLabel.setIcon(idleIcon);

progressBar.setVisible(false);

progressBar.setValue(0);

} else if ("message".equals(propertyName)) {

String text = (String)(evt.getNewValue());

statusMessageLabel.setText((text == null) ? "" : text);

messageTimer.restart();

} else if ("progress".equals(propertyName)) {

int value = (Integer)(evt.getNewValue());

progressBar.setVisible(true);

progressBar.setIndeterminate(false);

progressBar.setValue(value);

}

}

});

// tracking table selection

masterTable.getSelectionModel().addListSelectionListener(

new ListSelectionListener() {

public void valueChanged(ListSelectionEvent e) {

firePropertyChange("recordSelected", !isRecordSelected(), isRecordSelected());

}

});

// tracking changes to save

bindingGroup.addBindingListener(new AbstractBindingListener() {

@Override

public void targetChanged(Binding binding, PropertyStateEvent event) {

// save action observes saveNeeded property

setSaveNeeded(true);

}

});

// have a transaction started

entityManager.getTransaction().begin();

}

public boolean isSaveNeeded() {

return saveNeeded;

}

private void setSaveNeeded(boolean saveNeeded) {

if (saveNeeded != this.saveNeeded) {

this.saveNeeded = saveNeeded;

firePropertyChange("saveNeeded", !saveNeeded, saveNeeded);

}

}

public boolean isRecordSelected() {

return masterTable.getSelectedRow() != -1;

}

@Action

public void newRecord() {

collegemanagement.College c = new collegemanagement.College();

entityManager.persist(c);

list.add(c);

int row = list.size()-1;

masterTable.setRowSelectionInterval(row, row);

masterTable.scrollRectToVisible(masterTable.getCellRect(row, 0, true));

setSaveNeeded(true);

}

@Action(enabledProperty = "recordSelected")

public void deleteRecord() {

int[] selected = masterTable.getSelectedRows();

List<collegemanagement.College> toRemove = new ArrayList<collegemanagement.College>(selected.length);

for (int idx=0; idx<selected.length; idx++) {

collegemanagement.College c = list.get(masterTable.convertRowIndexToModel(selected[idx]));

toRemove.add(c);

entityManager.remove(c);

}

list.removeAll(toRemove);

setSaveNeeded(true);

}

@Action(enabledProperty = "saveNeeded")

public Task save() {

return new SaveTask(getApplication());

}

private class SaveTask extends Task {

SaveTask(org.jdesktop.application.Application app) {

super(app);

}

@Override protected Void doInBackground() {

try {

entityManager.getTransaction().commit();

entityManager.getTransaction().begin();

} catch (RollbackException rex) {

rex.printStackTrace();

entityManager.getTransaction().begin();

List<collegemanagement.College> merged = new ArrayList<collegemanagement.College>(list.size());

for (collegemanagement.College c : list) {

merged.add(entityManager.merge(c));

}

list.clear();

list.addAll(merged);

}

return null;

}

@Override protected void finished() {

setSaveNeeded(false);

}

}

@Action

public Task refresh() {

return new RefreshTask(getApplication());

}

private class RefreshTask extends Task {

RefreshTask(org.jdesktop.application.Application app) {

super(app);

}

@SuppressWarnings("unchecked")

@Override protected Void doInBackground() {

try {

setProgress(0, 0, 4);

setMessage("Rolling back the current changes...");

setProgress(1, 0, 4);

entityManager.getTransaction().rollback();

Thread.sleep(1000L); // remove for real app

setProgress(2, 0, 4);

setMessage("Starting a new transaction...");

entityManager.getTransaction().begin();

Thread.sleep(500L); // remove for real app

setProgress(3, 0, 4);

setMessage("Fetching new data...");

java.util.Collection data = query.getResultList();

for (Object entity : data) {

entityManager.refresh(entity);

}

Thread.sleep(1300L); // remove for real app

setProgress(4, 0, 4);

Thread.sleep(150L); // remove for real app

list.clear();

list.addAll(data);

} catch(InterruptedException ignore) { }

return null;

}

@Override protected void finished() {

setMessage("Done.");

setSaveNeeded(false);

}

}

@Action

public void showAboutBox() {

if (aboutBox == null) {

JFrame mainFrame = CollegeManagementApp.getApplication().getMainFrame();

aboutBox = new CollegeManagementAboutBox(mainFrame);

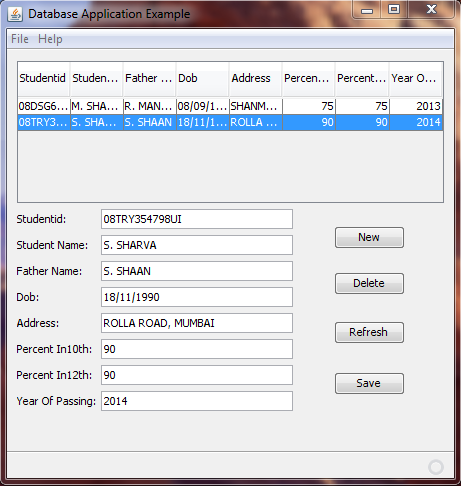
aboutBox.setLocationRelativeTo(mainFrame);

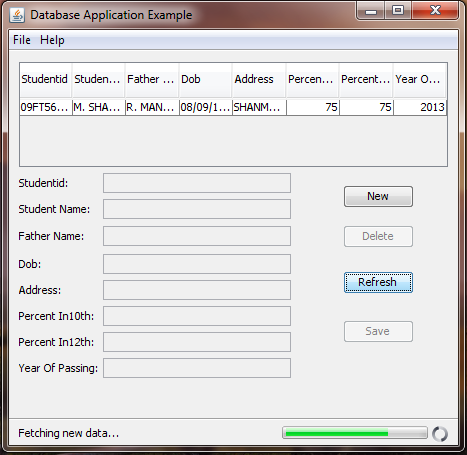
}

CollegeManagementApp.getApplication().show(aboutBox);

}

**OUTPUT:**

****

****

**RESULT:**

Thus, the college management form has been designed using SWING with JDBC and implemented successfully and output is verified.