**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.util.List;

**import** java.util.ArrayList;

**public** **class** AddC {

**private** **static** **final** String ***URL*** = "jdbc:derby:C:\\Users\\Denis.Long\\MyDB";

**private** **static** **final** String ***USERNAME*** = "";

**private** **static** **final** String ***PASSWORD*** = "";

**private** Connection connection = **null**; // manages connection

**private** PreparedStatement insertNewPerson = **null**;

// constructor

**public** AddC()

{

**try**

{

connection =

DriverManager.*getConnection*( ***URL***, ***USERNAME***, ***PASSWORD*** );

insertNewPerson = connection.prepareStatement(

"INSERT INTO Addresses " +

"( FirstName, LastName, Email, PhoneNumber ) " +

"VALUES ( ?, ?, ?, ? )" );

} // end try

**catch** ( SQLException sqlException )

{

sqlException.printStackTrace();

System.*exit*( 1 );

} // end catch

} // end PersonQueries constructor

// select all of the addresses in the database

// add an entry

**public** **int** addPerson(

String fname, String lname, String email, String num )

{

**int** result = 0;

// set parameters, then execute insertNewPerson

**try**

{

insertNewPerson.setString( 1, fname );

insertNewPerson.setString( 2, lname );

insertNewPerson.setString( 3, email );

insertNewPerson.setString( 4, num );

// insert the new entry; returns # of rows updated

result = insertNewPerson.executeUpdate();

} // end try

**catch** ( SQLException sqlException )

{

sqlException.printStackTrace();

close();

} // end catch

**return** result;

} // end method addPerson

// close the database connection

**public** **void** close()

{

**try**

{

connection.close();

} // end try

**catch** ( SQLException sqlException )

{

sqlException.printStackTrace();

} // end catch

} // end method close

} // end class PersonQueries

**public** **class** DBNoGui {

**private** AddC addC;

**public** DBNoGui (){

addC = **new** AddC();

**int** result = addC.addPerson("FG1","FF2","denis.long@cit.ie","5455573" );

**if** ( result == 1 )

System.***out***.println( "Person added!" );

**else**

System.***out***.println( "Person not added!" );

}

**public** **static** **void** main( String args[] )

{

**new** DBNoGui();

} // end method main

}

Swing code easy to convert to fx:

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.awt.FlowLayout;

import java.awt.GridLayout;

import java.util.List;

import javax.swing.JButton;

import javax.swing.Box;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextField;

import javax.swing.WindowConstants;

import javax.swing.BoxLayout;

import javax.swing.BorderFactory;

import javax.swing.JOptionPane;

public class UseAddC extends JFrame{

private Person currentEntry;

private AddC addC;

private List< Person > results;

JButton browseButton;

private JLabel emailLabel;

private JTextField emailTextField;

private JLabel firstNameLabel;

private JTextField firstNameTextField;

private JLabel idLabel;

private JTextField idTextField;

private JTextField indexTextField;

private JLabel lastNameLabel;

private JTextField lastNameTextField;

private JPanel displayPanel;

private JLabel phoneLabel;

private JTextField phoneTextField;

private JButton insertButton;

// no-argument constructor

public UseAddC()

{

super( "Address Book" );

// establish database connection and set up PreparedStatements

addC = new AddC();

// create GUI

idLabel = new JLabel();

idTextField = new JTextField( 10 );

firstNameLabel = new JLabel();

firstNameTextField = new JTextField( 10 );

lastNameLabel = new JLabel();

lastNameTextField = new JTextField( 10 );

emailLabel = new JLabel();

emailTextField = new JTextField( 10 );

phoneLabel = new JLabel();

phoneTextField = new JTextField( 10 );

browseButton = new JButton();

insertButton = new JButton();

setLayout( new FlowLayout( FlowLayout.CENTER, 10, 10 ) );

setSize( 400, 300 );

setResizable( false );

displayPanel = new JPanel();

displayPanel.setLayout( new GridLayout( 5, 2, 4, 4 ) );

idLabel.setText( "Address ID:" );

displayPanel.add( idLabel );

idTextField.setEditable( false );

displayPanel.add( idTextField );

firstNameLabel.setText( "First Name:" );

displayPanel.add( firstNameLabel );

displayPanel.add( firstNameTextField );

lastNameLabel.setText( "Last Name:" );

displayPanel.add( lastNameLabel );

displayPanel.add( lastNameTextField );

emailLabel.setText( "Email:" );

displayPanel.add( emailLabel );

displayPanel.add( emailTextField );

phoneLabel.setText( "Phone Number:" );

displayPanel.add( phoneLabel );

displayPanel.add( phoneTextField );

add( displayPanel );

insertButton.setText( "Insert New Entry" );

insertButton.addActionListener(

new ActionListener()

{

public void actionPerformed( ActionEvent evt )

{

insertButtonActionPerformed( evt );

} // end method actionPerformed

} // end anonymous inner class

); // end call to addActionListener

add( insertButton );

addWindowListener(

new WindowAdapter()

{

public void windowClosing( WindowEvent evt )

{

addC.close(); // close database connection

System.exit( 0 );

} // end method windowClosing

} // end anonymous inner class

); // end call to addWindowListener

setVisible( true );

} // end no-argument constructor

// handles call when previousButton is clicked

// handles call when insertButton is clicked

private void insertButtonActionPerformed( ActionEvent evt )

{

int result = addC.addPerson( firstNameTextField.getText(),

lastNameTextField.getText(), emailTextField.getText(),

phoneTextField.getText() );

if ( result == 1 )

JOptionPane.showMessageDialog( this, "Person added!",

"Person added", JOptionPane.PLAIN\_MESSAGE );

else

JOptionPane.showMessageDialog( this, "Person not added!",

"Error", JOptionPane.PLAIN\_MESSAGE );

} // end method insertButtonActionPerformed

// main method

public static void main( String args[] )

{

new UseAddC();

} // end method main

} // end class AddressBookDisplay

**public** **class** Person

{

**private** **int** addressID;

**private** String firstName;

**private** String lastName;

**private** String email;

**private** String phoneNumber;

// no-argument constructor

**public** Person()

{

} // end no-argument Person constructor

// constructor

**public** Person( **int** id, String first, String last,

String emailAddress, String phone )

{

setAddressID( id );

setFirstName( first );

setLastName( last );

setEmail( emailAddress );

setPhoneNumber( phone );

} // end five-argument Person constructor

// sets the addressID

**public** **void** setAddressID( **int** id )

{

addressID = id;

} // end method setAddressID

// returns the addressID

**public** **int** getAddressID()

{

**return** addressID;

} // end method getAddressID

// sets the firstName

**public** **void** setFirstName( String first )

{

firstName = first;

} // end method setFirstName

// returns the first name

**public** String getFirstName()

{

**return** firstName;

} // end method getFirstName

// sets the lastName

**public** **void** setLastName( String last )

{

lastName = last;

} // end method setLastName

// returns the last name

**public** String getLastName()

{

**return** lastName;

} // end method getLastName

// sets the email address

**public** **void** setEmail( String emailAddress )

{

email = emailAddress;

} // end method setEmail

// returns the email address

**public** String getEmail()

{

**return** email;

} // end method getEmail

// sets the phone number

**public** **void** setPhoneNumber( String phone )

{

phoneNumber = phone;

} // end method setPhoneNumber

// returns the phone number

**public** String getPhoneNumber()

{

**return** phoneNumber;

} // end method getPhoneNumber

} // end class Person

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* (C) Copyright 1992-2010 by Deitel & Associates, Inc. and \*

\* Pearson Education, Inc. All Rights Reserved. \*

\* \*

\* DISCLAIMER: The authors and publisher of this book have used their \*

\* best efforts in preparing the book. These efforts include the \*

\* development, research, and testing of the theories and programs \*

\* to determine their effectiveness. The authors and publisher make \*

\* no warranty of any kind, expressed or implied, with regard to these \*

\* programs or to the documentation contained in these books. The authors \*

\* and publisher shall not be liable in any event for incidental or \*

\* consequential damages in connection with, or arising out of, the \*

\* furnishing, performance, or use of these programs. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Fig. 28.32: AddressBookDisplay.java

// A simple address book

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.awt.FlowLayout;

import java.awt.GridLayout;

import java.util.List;

import javax.swing.JButton;

import javax.swing.Box;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextField;

import javax.swing.WindowConstants;

import javax.swing.BoxLayout;

import javax.swing.BorderFactory;

import javax.swing.JOptionPane;

public class AddressBookDisplay extends JFrame

{

private Person currentEntry;

private PersonQueries personQueries;

private List< Person > results;

private int numberOfEntries = 0;

private int currentEntryIndex;

private JButton browseButton;

private JLabel emailLabel;

private JTextField emailTextField;

private JLabel firstNameLabel;

private JTextField firstNameTextField;

private JLabel idLabel;

private JTextField idTextField;

private JTextField indexTextField;

private JLabel lastNameLabel;

private JTextField lastNameTextField;

private JTextField maxTextField;

private JButton nextButton;

private JLabel ofLabel;

private JLabel phoneLabel;

private JTextField phoneTextField;

private JButton previousButton;

private JButton queryButton;

private JLabel queryLabel;

private JPanel queryPanel;

private JPanel navigatePanel;

private JPanel displayPanel;

private JTextField queryTextField;

private JButton insertButton;

// no-argument constructor

public AddressBookDisplay()

{

super( "Address Book" );

// establish database connection and set up PreparedStatements

personQueries = new PersonQueries();

// create GUI

navigatePanel = new JPanel();

previousButton = new JButton();

indexTextField = new JTextField( 2 );

ofLabel = new JLabel();

maxTextField = new JTextField( 2 );

nextButton = new JButton();

displayPanel = new JPanel();

idLabel = new JLabel();

idTextField = new JTextField( 10 );

firstNameLabel = new JLabel();

firstNameTextField = new JTextField( 10 );

lastNameLabel = new JLabel();

lastNameTextField = new JTextField( 10 );

emailLabel = new JLabel();

emailTextField = new JTextField( 10 );

phoneLabel = new JLabel();

phoneTextField = new JTextField( 10 );

queryPanel = new JPanel();

queryLabel = new JLabel();

queryTextField = new JTextField( 10 );

queryButton = new JButton();

browseButton = new JButton();

insertButton = new JButton();

setLayout( new FlowLayout( FlowLayout.CENTER, 10, 10 ) );

setSize( 400, 300 );

setResizable( false );

navigatePanel.setLayout(

new BoxLayout( navigatePanel, BoxLayout.X\_AXIS ) );

previousButton.setText( "Previous" );

previousButton.setEnabled( false );

previousButton.addActionListener(

new ActionListener()

{

public void actionPerformed( ActionEvent evt )

{

previousButtonActionPerformed( evt );

} // end method actionPerformed

} // end anonymous inner class

); // end call to addActionListener

navigatePanel.add( previousButton );

navigatePanel.add( Box.createHorizontalStrut( 10 ) );

indexTextField.setHorizontalAlignment(

JTextField.CENTER );

indexTextField.addActionListener(

new ActionListener()

{

public void actionPerformed( ActionEvent evt )

{

indexTextFieldActionPerformed( evt );

} // end method actionPerformed

} // end anonymous inner class

); // end call to addActionListener

navigatePanel.add( indexTextField );

navigatePanel.add( Box.createHorizontalStrut( 10 ) );

ofLabel.setText( "of" );

navigatePanel.add( ofLabel );

navigatePanel.add( Box.createHorizontalStrut( 10 ) );

maxTextField.setHorizontalAlignment(

JTextField.CENTER );

maxTextField.setEditable( false );

navigatePanel.add( maxTextField );

navigatePanel.add( Box.createHorizontalStrut( 10 ) );

nextButton.setText( "Next" );

nextButton.setEnabled( false );

nextButton.addActionListener(

new ActionListener()

{

public void actionPerformed( ActionEvent evt )

{

nextButtonActionPerformed( evt );

} // end method actionPerformed

} // end anonymous inner class

); // end call to addActionListener

navigatePanel.add( nextButton );

add( navigatePanel );

displayPanel.setLayout( new GridLayout( 5, 2, 4, 4 ) );

idLabel.setText( "Address ID:" );

displayPanel.add( idLabel );

idTextField.setEditable( false );

displayPanel.add( idTextField );

firstNameLabel.setText( "First Name:" );

displayPanel.add( firstNameLabel );

displayPanel.add( firstNameTextField );

lastNameLabel.setText( "Last Name:" );

displayPanel.add( lastNameLabel );

displayPanel.add( lastNameTextField );

emailLabel.setText( "Email:" );

displayPanel.add( emailLabel );

displayPanel.add( emailTextField );

phoneLabel.setText( "Phone Number:" );

displayPanel.add( phoneLabel );

displayPanel.add( phoneTextField );

add( displayPanel );

queryPanel.setLayout(

new BoxLayout( queryPanel, BoxLayout.X\_AXIS) );

queryPanel.setBorder( BorderFactory.createTitledBorder(

"Find an entry by last name" ) );

queryLabel.setText( "Last Name:" );

queryPanel.add( Box.createHorizontalStrut( 5 ) );

queryPanel.add( queryLabel );

queryPanel.add( Box.createHorizontalStrut( 10 ) );

queryPanel.add( queryTextField );

queryPanel.add( Box.createHorizontalStrut( 10 ) );

queryButton.setText( "Find" );

queryButton.addActionListener(

new ActionListener()

{

public void actionPerformed( ActionEvent evt )

{

queryButtonActionPerformed( evt );

} // end method actionPerformed

} // end anonymous inner class

); // end call to addActionListener

queryPanel.add( queryButton );

queryPanel.add( Box.createHorizontalStrut( 5 ) );

add( queryPanel );

browseButton.setText( "Browse All Entries" );

browseButton.addActionListener(

new ActionListener()

{

public void actionPerformed( ActionEvent evt )

{

browseButtonActionPerformed( evt );

} // end method actionPerformed

} // end anonymous inner class

); // end call to addActionListener

add( browseButton );

insertButton.setText( "Insert New Entry" );

insertButton.addActionListener(

new ActionListener()

{

public void actionPerformed( ActionEvent evt )

{

insertButtonActionPerformed( evt );

} // end method actionPerformed

} // end anonymous inner class

); // end call to addActionListener

add( insertButton );

addWindowListener(

new WindowAdapter()

{

public void windowClosing( WindowEvent evt )

{

personQueries.close(); // close database connection

System.exit( 0 );

} // end method windowClosing

} // end anonymous inner class

); // end call to addWindowListener

setVisible( true );

} // end no-argument constructor

// handles call when previousButton is clicked

private void previousButtonActionPerformed( ActionEvent evt )

{

currentEntryIndex--;

if ( currentEntryIndex < 0 )

currentEntryIndex = numberOfEntries - 1;

indexTextField.setText( "" + ( currentEntryIndex + 1 ) );

indexTextFieldActionPerformed( evt );

} // end method previousButtonActionPerformed

// handles call when nextButton is clicked

private void nextButtonActionPerformed( ActionEvent evt )

{

currentEntryIndex++;

if ( currentEntryIndex >= numberOfEntries )

currentEntryIndex = 0;

indexTextField.setText( "" + ( currentEntryIndex + 1 ) );

indexTextFieldActionPerformed( evt );

} // end method nextButtonActionPerformed

// handles call when queryButton is clicked

private void queryButtonActionPerformed( ActionEvent evt )

{

results =

personQueries.getPeopleByLastName( queryTextField.getText() );

numberOfEntries = results.size();

if ( numberOfEntries != 0 )

{

currentEntryIndex = 0;

currentEntry = results.get( currentEntryIndex );

idTextField.setText("" + currentEntry.getAddressID() );

firstNameTextField.setText( currentEntry.getFirstName() );

lastNameTextField.setText( currentEntry.getLastName() );

emailTextField.setText( currentEntry.getEmail() );

phoneTextField.setText( currentEntry.getPhoneNumber() );

maxTextField.setText( "" + numberOfEntries );

indexTextField.setText( "" + ( currentEntryIndex + 1 ) );

nextButton.setEnabled( true );

previousButton.setEnabled( true );

} // end if

else

browseButtonActionPerformed( evt );

} // end method queryButtonActionPerformed

// handles call when a new value is entered in indexTextField

private void indexTextFieldActionPerformed( ActionEvent evt )

{

currentEntryIndex =

( Integer.parseInt( indexTextField.getText() ) - 1 );

if ( numberOfEntries != 0 && currentEntryIndex < numberOfEntries )

{

currentEntry = results.get( currentEntryIndex );

idTextField.setText("" + currentEntry.getAddressID() );

firstNameTextField.setText( currentEntry.getFirstName() );

lastNameTextField.setText( currentEntry.getLastName() );

emailTextField.setText( currentEntry.getEmail() );

phoneTextField.setText( currentEntry.getPhoneNumber() );

maxTextField.setText( "" + numberOfEntries );

indexTextField.setText( "" + ( currentEntryIndex + 1 ) );

} // end if

} // end method indexTextFieldActionPerformed

// handles call when browseButton is clicked

private void browseButtonActionPerformed( ActionEvent evt )

{

try

{

results = personQueries.getAllPeople();

numberOfEntries = results.size();

if ( numberOfEntries != 0 )

{

currentEntryIndex = 0;

currentEntry = results.get( currentEntryIndex );

idTextField.setText("" + currentEntry.getAddressID() );

firstNameTextField.setText( currentEntry.getFirstName() );

lastNameTextField.setText( currentEntry.getLastName() );

emailTextField.setText( currentEntry.getEmail() );

phoneTextField.setText( currentEntry.getPhoneNumber() );

maxTextField.setText( "" + numberOfEntries );

indexTextField.setText( "" + ( currentEntryIndex + 1 ) );

nextButton.setEnabled( true );

previousButton.setEnabled( true );

} // end if

} // end try

catch ( Exception e )

{

e.printStackTrace();

} // end catch

} // end method browseButtonActionPerformed

// handles call when insertButton is clicked

private void insertButtonActionPerformed( ActionEvent evt )

{

int result = personQueries.addPerson( firstNameTextField.getText(),

lastNameTextField.getText(), emailTextField.getText(),

phoneTextField.getText() );

if ( result == 1 )

JOptionPane.showMessageDialog( this, "Person added!",

"Person added", JOptionPane.PLAIN\_MESSAGE );

else

JOptionPane.showMessageDialog( this, "Person not added!",

"Error", JOptionPane.PLAIN\_MESSAGE );

browseButtonActionPerformed( evt );

} // end method insertButtonActionPerformed

// main method

public static void main( String args[] )

{

new AddressBookDisplay();

} // end method main

} // end class AddressBookDisplay

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* (C) Copyright 1992-2010 by Deitel & Associates, Inc. and \*

\* Pearson Education, Inc. All Rights Reserved. \*

\* \*

\* DISCLAIMER: The authors and publisher of this book have used their \*

\* best efforts in preparing the book. These efforts include the \*

\* development, research, and testing of the theories and programs \*

\* to determine their effectiveness. The authors and publisher make \*

\* no warranty of any kind, expressed or implied, with regard to these \*

\* programs or to the documentation contained in these books. The authors \*

\* and publisher shall not be liable in any event for incidental or \*

\* consequential damages in connection with, or arising out of, the \*

\* furnishing, performance, or use of these programs. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Fig. 28.31: PersonQueries.java

// PreparedStatements used by the Address Book application

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.List;

import java.util.ArrayList;

public class PersonQueries

{

private static final String URL = "jdbc:derby:C:\\Users\\Denis.Long\\MyDB";

private static final String USERNAME = "";

private static final String PASSWORD = "";

private Connection connection = null; // manages connection

private PreparedStatement selectAllPeople = null;

private PreparedStatement selectPeopleByLastName = null;

private PreparedStatement insertNewPerson = null;

// constructor

public PersonQueries()

{

try

{

connection =

DriverManager.getConnection( URL, USERNAME, PASSWORD );

// create query that selects all entries in the AddressBook

selectAllPeople =

connection.prepareStatement( "SELECT \* FROM Addresses" );

// create query that selects entries with a specific last name

selectPeopleByLastName = connection.prepareStatement(

"SELECT \* FROM Addresses WHERE LastName = ?" );

// create insert that adds a new entry into the database

insertNewPerson = connection.prepareStatement(

"INSERT INTO Addresses " +

"( FirstName, LastName, Email, PhoneNumber ) " +

"VALUES ( ?, ?, ?, ? )" );

} // end try

catch ( SQLException sqlException )

{

sqlException.printStackTrace();

System.exit( 1 );

} // end catch

} // end PersonQueries constructor

// select all of the addresses in the database

public List< Person > getAllPeople()

{

List< Person > results = null;

ResultSet resultSet = null;

try

{

// executeQuery returns ResultSet containing matching entries

resultSet = selectAllPeople.executeQuery();

results = new ArrayList< Person >();

while ( resultSet.next() )

{

results.add( new Person(

resultSet.getInt( "addressID" ),

resultSet.getString( "firstName" ),

resultSet.getString( "lastName" ),

resultSet.getString( "email" ),

resultSet.getString( "phoneNumber" ) ) );

} // end while

} // end try

catch ( SQLException sqlException )

{

sqlException.printStackTrace();

} // end catch

finally

{

try

{

resultSet.close();

} // end try

catch ( SQLException sqlException )

{

sqlException.printStackTrace();

close();

} // end catch

} // end finally

return results;

} // end method getAllPeople

// select person by last name

public List< Person > getPeopleByLastName( String name )

{

List< Person > results = null;

ResultSet resultSet = null;

try

{

selectPeopleByLastName.setString( 1, name ); // specify last name

// executeQuery returns ResultSet containing matching entries

resultSet = selectPeopleByLastName.executeQuery();

results = new ArrayList< Person >();

int i=0;

while ( resultSet.next() )

{

results.add( new Person( i++,

resultSet.getString( "firstName" ),

resultSet.getString( "lastName" ),

resultSet.getString( "email" ),

resultSet.getString( "phoneNumber" ) ) );

} // end while

} // end try

catch ( SQLException sqlException )

{

sqlException.printStackTrace();

} // end catch

finally

{

try

{

resultSet.close();

} // end try

catch ( SQLException sqlException )

{

sqlException.printStackTrace();

close();

} // end catch

} // end finally

return results;

} // end method getPeopleByName

// add an entry

public int addPerson(

String fname, String lname, String email, String num )

{

int result = 0;

// set parameters, then execute insertNewPerson

try

{

insertNewPerson.setString( 1, fname );

insertNewPerson.setString( 2, lname );

insertNewPerson.setString( 3, email );

insertNewPerson.setString( 4, num );

// insert the new entry; returns # of rows updated

result = insertNewPerson.executeUpdate();

} // end try

catch ( SQLException sqlException )

{

sqlException.printStackTrace();

close();

} // end catch

return result;

} // end method addPerson

// close the database connection

public void close()

{

try

{

connection.close();

} // end try

catch ( SQLException sqlException )

{

sqlException.printStackTrace();

} // end catch

} // end method close

} // end class PersonQueries

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* (C) Copyright 1992-2010 by Deitel & Associates, Inc. and \*

\* Pearson Education, Inc. All Rights Reserved. \*

\* \*

\* DISCLAIMER: The authors and publisher of this book have used their \*

\* best efforts in preparing the book. These efforts include the \*

\* development, research, and testing of the theories and programs \*

\* to determine their effectiveness. The authors and publisher make \*

\* no warranty of any kind, expressed or implied, with regard to these \*

\* programs or to the documentation contained in these books. The authors \*

\* and publisher shall not be liable in any event for incidental or \*

\* consequential damages in connection with, or arising out of, the \*

\* furnishing, performance, or use of these programs. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

package com.tutorialspoint.eclipselink.entity;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private int eid;

private String ename;

private double salary;

private String deg;

public Employee(int eid, String ename, double salary, String deg) {

super( );

this.eid = eid;

this.ename = ename;

this.salary = salary;

this.deg = deg;

}

public Employee( ) {

super();

}

public int getEid( ) {

return eid;

}

public void setEid(int eid) {

this.eid = eid;

}

public String getEname( ) {

return ename;

}

public void setEname(String ename) {

this.ename = ename;

}

public double getSalary( ) {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public String getDeg( ) {

return deg;

}

public void setDeg(String deg) {

this.deg = deg;

}

@Override

public String toString() {

return "Employee [eid=" + eid + ", ename=" + ename + ", salary=" + salary + ", deg=" + deg + "]";

}

}

package com.tutorialspoint.eclipselink.service;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.Persistence;

import com.tutorialspoint.eclipselink.entity.Employee;

public class CreateEmployee {

public static void main( String[ ] args ) {

EntityManagerFactory emfactory = Persistence.createEntityManagerFactory( "Eclipselink\_JPA" );

EntityManager entitymanager = emfactory.createEntityManager( );

entitymanager.getTransaction( ).begin( );

Employee employee = new Employee( );

employee.setEid( 125 );

employee.setEname( "Gopa" );

employee.setSalary( 400 );

employee.setDeg( "Technical Manager" );

entitymanager.persist( employee );

entitymanager.getTransaction( ).commit( );

entitymanager.close( );

emfactory.close( );

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<persistence version=*"2.1"* xmlns=*"http://xmlns.jcp.org/xml/ns/persistence"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd"*>

<persistence-unit name=*"Eclipselink\_JPA"* transaction-type=*"RESOURCE\_LOCAL"*>

<class>com.tutorialspoint.eclipselink.entity.Employee</class>

<properties>

<property name=*"javax.persistence.jdbc.url"* value=*"jdbc:mysql://localhost:3306/database"*/>

<property name=*"javax.persistence.jdbc.user"* value=*"root"*/>

<property name=*"javax.persistence.jdbc.password"* value=*"root"*/>

<property name=*"javax.persistence.jdbc.driver"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"eclipselink.logging.level"* value=*"FINE"*/>

<property name=*"eclipselink.ddl-generation"* value=*"create-tables"*/>

</properties>

</persistence-unit>

</persistence>