CLIFFHOUSE DATABASE WITH WEB AND LAN INTERFACE

Group Number: 10

BY

Dan Rosenthal
Satish Singhal
Walter Valicente
Rosemary Alvarez – Helped in the early phase of project

Class
CS56 Advanced Java
9 A. M. – 12:05 P. M.
Santa Monica College

2. Introduction:

In California and other coastal communities there are many resorts and facilities that are ideal for holding specialized seminars and trainings for professionals. Some of them are called Cliffhouse as they are located on ocean cliffs overlooking beautiful scenic Pacific and Atlantic Oceans. In fact the scenic photograph of one such actual facility is given below:

Fig.1 Cliffhouse Photograph

The purpose of this project is to proto-type the database of a seminar/lecture facility along with the java interface to it. We made the following assumptions about the business of the Cliffhouse.

1. It only holds seminars or lectures with one lecture delivered by one Lecturer.
2. The facility has 4 rooms with 2 having the ocean view and 2 on the opposite side of them.
3. All attendees pay by credit card and Cliffhouse pays half of the amount collected from the attendees to the Lecturer as his/her earning by check.
4. Cliffhouse does not have food preparation facility, and caterers supply food.
5. Attendees can register for the events/seminars either through the Cliffhouse web site or they cal call in or fax the necessary information. Lecturers can do the same.

The purposes of the database and associated software are:

1. To enter the business data in the database tables.
2. To update the change in Attendee or Lecturer information.
3. To display the data stored in various tables.
4. To run queries on the database to facilitate the decision making process.

We wish to emphasize here that this product is proto-type, which can be, with extended efforts, converted into a business level software product. The database used in this application is Microsoft Access 2000, but extension to any other relational database can be done easily.

We decided on this project because it blends best aspects of the education as well as business. The business of adult education to improve the performance of individuals is quite popular in America and many adult education industries have sprung to fulfill such needs. Often a geographically well situated facility such as ocean side Cliffhouse is appreciated by both the seminar leaders and participants.

Some of the major hurdles encountered by us in this project were such as the incompatible software tools among the participants. Most surprising was that programs that ran incessantly on Windows 98 machine kept on giving "The Memory Can not Be Read" message on many SMC Windows 2000 computers, and we had to fish for one that will not give a cryptic message like that. One participant eventually dropped out leaving the number of participants to rather small number (3 people) for the project as large as this.

Class Diagrams

The Class tree for this single package project as generated by the javadoc tool is given below. Our classes are shown in blue typeset. Inheritance relationship starting from the highest level (java.lang.Object) to the classes that we use in this project can be easily deduced from this tree diagram. Following the diagram we give the documentation for fields and methods for all classes created by us. Java classes are shown in black typeset.

Class Diagram

- **class java.lang.Object**
  - **class** Accounts_Receivable **(implements BusinessConstants)**
  - **class** Accts_Pay.ButtonHandler **(implements java.awt.event.ActionListener)**
  - **class** AddRecord **(implements java.awt.event.ActionListener, BusinessConstants)**
  - **class** Attendees
  - **class** ClearFields **(implements java.awt.event.ActionListener)**
  - **class java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)**
    - **class java.awt.Container**
      - **class javax.swing.JComponent (implements java.io.Serializable)**
        - **class javax.swing.JPanel (implements javax.accessibility.Accessible)**
          - **class** CliffLogo
          - **class** ControlPanel

- **class** DataEntryForm1
- **class** DataEntryForm2
- **class** MyColorChooser
- **class** PicRoom
- **class** Rooms
- **class** ScrollingPanel **(implements BusinessConstants)**

- **class java.awt.Panel (implements javax.accessibility.Accessible)**
  - **class java.applet.Applet**
    - **class javax.swing.JApplet (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer)**
      - **class** ConfList
      - **class** EntryForm1Applet
      - **class** EntryForm2Applet
      - **class** RoomSelector **(implements java.awt.event.ActionListener)**
- **class java.awt.Window (implements javax.accessibility.Accessible)**
  - **class java.awt.Frame (implements java.awt.MenuContainer)**
    - **class javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)**
      - **class** Accts_Pay
      - **class** AttendeeDataFrm
      - **class** AttendeeGUI **(implements ManageShutDown)**
      - **class** DataDisplayGUI **(implements java.awt.event.ActionListener, javax.swing.SwingConstants)**
      - **class** DataEntryGUI **(implements java.awt.event.ActionListener, BusinessConstants)**
      - **class** DataQueryGUI **(implements java.awt.event.ActionListener, javax.swing.SwingConstants)**
      - **class** DisplayAccountsPayable **(implements ManageShutDown)**
      - **class** DisplayAccountsReceivable **(implements ManageShutDown)**
      - **class** DisplayAttendee **(implements ManageShutDown)**

- **class** DisplayEvents **(implements ManageShutDown)**
- **class** DisplayLecturer **(implements ManageShutDown)**
- **class** DisplayQueryResults **(implements ManageShutDown)**
- **class** DisplayRegistration **(implements ManageShutDown)**
- **class** Main_Menu **(implements java.awt.event.ActionListener, javax.swing.SwingConstants)**
- **class** RandomRead
- **class** RandomWrite

- **class** ConfList.Handler **(implements java.awt.event.ActionListener)**
- **class** ControlPanel.Temporary **(implements java.awt.event.ActionListener)**
- **class** DBConnect
- **class** EntryForm1Applet.Handler **(implements java.awt.event.ActionListener)**
- **class** FindRecord **(implements java.awt.event.ActionListener)**
- **class** Help **(implements java.awt.event.ActionListener)**
- **class** LoadDB
- **class** ManageString
- **class** Registration
- **class** UpdateRecord **(implements java.awt.event.ActionListener)**

- **class** AttendServ
- **class** DispServ2
- **class** LectServ
- **class** RoomServ

Interface Hierarchy

- **interface** ManageAttendee
- **interface** ManageShutDown
- **interface javax.swing.SwingConstants**
  - **interface** BusinessConstants

Fig. 2. Class Hierarchy Diagram for Cliffhouse Project

The details of various classes are shown below in alphabetical order:

Class Accounts_Receivable

```
java.lang.Object
  |
  +--Accounts_Receivable
```

Fields

private java.lang.String	CredtCard_Num
private java.lang.String	Social_Security_Num
private int	Transaction_Amount
private java.lang.String	Transaction_Date
private java.lang.String	Transaction_Num

Constructor	
Accounts_Receivable()	

Methods	
static void	main(java.lang.String[] args)
void	setCredtCard_Num(java.lang.String m_CredtCard_Num)
void	setSocial_Security_Num(java.lang.String m_Social_Security_Num)
void	setTransaction_Amount()
void	setTransaction_Date()
void	setTransaction_Num(java.lang.String m_Transaction_Num)

Class Accts_Pay.ButtonHandler

```
java.lang.Object
  |
  +--Accts_Pay.ButtonHandler
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.util.EventListener

Enclosing class:

Accts_Pay

Class AddRecord

public class AddRecord
extends java.lang.Object

implements java.awt.event.ActionListener, BusinessConstants

Fields	
private Accounts_Receivable	Acts_Rcv_Data
private java.lang.String	ActsInsert
private java.sql.Connection	AddRec_Connection
private Attendees	Attendee_Data
private java.lang.String	comma
private java.lang.Object[]	EventListArray **Object array for holding multiple events selected**
private ScrollingPanel	fields
(package private) int	numEvents
private javax.swing.JTextArea	Output
private java.lang.String	QueryString
private java.lang.String	QueryString1
private java.lang.String	QueryString2
private java.lang.String	quote
private java.sql.ResultSet	rs
private java.sql.Statement	Smt
(package private) java.lang.String	tempSSN

Constructors	
AddRecord() **No argument default constructor**	
AddRecord(java.sql.Connection C,	

ScrollingPanel f, javax.swing.JTextArea O) **Explicit Argument Constructor**	

Method Summary	
void	actionPerformed(java.awt.event.ActionEvent e)
private void	fillAccountsReceivable(java.lang.String tempCreditCard)
private void	fillAttendee()
private void	fillRegistration()
static void	main(java.lang.String[] args)
private void	processExistingAttendee()
private void	processNewAttendee()

Class Attendees

```
java.lang.Object
  |
  +--Attendees
```

Fields	
private java.lang.String	Address
private java.lang.String	City **City is a String limited to maximum 30 characters**
private java.lang.String	comma
private static java.sql.Connection	connection
(package private) java.lang.Object[]	data
private java.lang.String	EMail **E Mail is a String maximum limited to 40 characters**

private java.lang.String	Final_String
private java.lang.String	FirstName **FirstName is the Attendees FirstName, Maximum Length is 20 characters.**
private java.lang.String	HomePhone **HomePhone is a String fixed to 10 characters.**
static java.lang.String	Insert **The field numColumns is equal to the number of columns in the database table**
private java.lang.String	LastName **LastName is the Attendees LastName, Maximum Length is 20 characters.**
private static int	numColumns
static java.lang.String	quote
private java.lang.String	SocialNum **SocialNum is the Social Security Number of the Attendee which will be restricted to a length of 9 character.**
private static int	SSNLength
private java.lang.String	State **State is a String but having fixed 2 characters**
private static java.lang.String	TableName
static java.lang.String	Values
private java.lang.String	WorkPhone **WorkPhone is a String fixed to 10 characters**
private java.lang.String	ZipCode **ZipCode is a String fixed to 5 characters length**

Constructors

Attendees() **Default Constructor takes no arguments and initializes the field values to java defaults**	
Attendees(java.lang.String M_SocialNum, java.lang.String M_LastName,	

java.lang.String M_FirstName, java.lang.String M_Address, java.lang.String M_City, java.lang.String M_State, java.lang.String M_ZipCode, java.lang.String M_HomePhone, java.lang.String M_WorkPhone, java.lang.String M_EMail) **Explicit Constructor sets the Attendees object to some pre-defined values.**	

Methods	
void	checkApostrophe(**java.lang.String input**)
static Attendees	constructAttendee()
java.lang.String	constructSqlObject()
java.lang.String	constructSqlUpdate()
java.lang.String	getAddress()
java.lang.String	getCity()
java.lang.String	getEmail()
java.lang.String	getFinal_String()
java.lang.String	getFName()
java.lang.String	getHomePhone()
java.lang.String	getLName()
java.lang.String	getSocialNum()
java.lang.String	getState()
java.lang.String	getWorkPhone()
java.lang.String	getZip()
static void	main(java.lang.String[] args)

void	setAddress(**java.lang.String Addrs**)
void	setCity(**java.lang.String m_City**)
void	setEMail(**java.lang.String m_EMail**) **Method setEMail will set the e mail address in the object.**
void	setFirstName(**java.lang.String Fname**)
void	setHomePhone(**java.lang.String hPhone**)
void	setLastName(**java.lang.String Lname**)
void	setSocialNum(**java.lang.String SSN**)
void	setState(**java.lang.String m_State**) **This Method accepts the state input and makes sure that no numerics are entered.**
void	setWorkPhone(**java.lang.String wPhone**)
void	setZipCode(**java.lang.String m_Zip**)

Class ClearFields

```
java.lang.Object
  |
  +--ClearFields
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.util.EventListener

Fields	
private ScrollingPanel	fields

Constructors	
ClearFields()	
ClearFields(**ScrollingPanel f**)	

Methods

void	actionPerformed(java.awt.event.ActionEvent e)
static void	main(java.lang.String[] args)

Class CliffLogo

public class CliffLogo
extends javax.swing.JPanel

Fields	
private javax.swing.ImageIcon	cliffhouseLogo
private java.awt.Image	i

Constructors	
CliffLogo()	

Methods	
void	paint(java.awt.Graphics g)

Class ControlPanel

public class ControlPanel
extends javax.swing.JPanel

Fields		
private	javax.swing.JButton	addName
private	javax.swing.JButton	clear
private	javax.swing.JButton	findName
private	javax.swing.JButton	help
private	javax.swing.JButton	previous
private	javax.swing.JButton	updateName

Constructors	
ControlPanel()	
ControlPanel(java.sql.Connection c, ScrollingPanel s, javax.swing.JTextArea t)	

Methods	
static void	main(java.lang.String[] args)

Class DataEntryForm1

public class DataEntryForm1
extends javax.swing.JPanel

Fields	
private javax.swing.JLabel[]	directions
private javax.swing.JTextField[]	entries
private javax.swing.JPanel	form1
private java.lang.String[]	legends
private Attendees	New_Attendee
private javax.swing.JLabel[]	spacers

Constructors	
DataEntryForm1()	

Methods	
void	draw(java.awt.Color z)
java.awt.Dimension	getMinimumSize()
java.awt.Dimension	getPreferreSize()
static void	main(java.lang.String[] args)

Class DataEntryForm2

public class DataEntryForm2
extends javax.swing.JPanel

Field Summary	
private javax.swing.JLabel[]	directions
private javax.swing.JTextField[]	entries
private javax.swing.JPanel	form1
private java.lang.String[]	legends
private javax.swing.JLabel[]	spacers

Constructors	
DataEntryForm2()	

Methods	
void	draw(java.awt.Color z)
java.awt.Dimension	getMinimumSize()
java.awt.Dimension	getPreferreSize()

Class MyColorChooser

public class MyColorChooser
extends javax.swing.JPanel

Field Summary	
private int	b
private javax.swing.Box	blueBox
private javax.swing.JPanel	colorGui
private int	g

private javax.swing.Box	greenBox
private javax.swing.Box	myBox
private int	r
private javax.swing.Box	redBox
private javax.swing.JSlider	sBlue
private javax.swing.JSlider	sGreen
private javax.swing.JTextField	showBlue
private javax.swing.JTextField	showGreen
private javax.swing.JTextField	showRed
private javax.swing.JSlider	sRed

Constructors	
MyColorChooser()	

Methods	
private void	doBackGround()
java.awt.Color	getTheColor()
void	paintComponent(java.awt.Graphics g)

Class PicRoom

class PicRoom
extends javax.swing.JPanel

Fields	
private java.awt.Image	i

private javax.swing.ImageIcon	roomPhoto

Constructors	
PicRoom(java.lang.String photoName)	

Methods	
void	paint(java.awt.Graphics g)

Class Rooms

class Rooms
extends javax.swing.JPanel

Fields	
private PicRoom	roomPhoto
private javax.swing.JTextArea	text1

Constructors	
Rooms(java.lang.String photoName, java.lang.String textAreaText)	

Class ScrollingPanel

public class ScrollingPanel
extends javax.swing.JPanel
implements BusinessConstants

Fields	
(package private) javax.swing.JTextField	address
(package private) javax.swing.JTextField	city
(package private) javax.swing.JTextField	creditcard
(package private) javax.swing.JTextField	email
private javax.swing.JList	EventList

private java.lang.Object[]	EventsSelected
private java.util.Vector	EventVector
private javax.swing.JPanel	fieldsPanel
(package private) javax.swing.JTextField	first
(package private) javax.swing.JTextField	home
(package private) javax.swing.JTextField	id
private javax.swing.JPanel	labelPanel
private java.lang.String[]	labels
(package private) javax.swing.JTextField	last
(package private) javax.swing.JLabel	LogoLabel
private java.lang.String[]	StateArray
private javax.swing.JList	StateList
private java.lang.String	StateSelected
private java.lang.String[]	ToolText
(package private) javax.swing.JTextField	work
(package private) javax.swing.JTextField	zip

Constructors

ScrollingPanel()	

Methods

private void	fillEventList()

private java.lang.String[]	fillStateArray()
java.lang.Object[]	getEvents()
java.lang.String	getState()
static void	main(java.lang.String[] args)

Class ConfList

public class ConfList
extends javax.swing.JApplet

Inner Class Summary	
private class	ConfList.Handler

Fields	
private java.awt.Container	c
private javax.swing.Box	controls
private ConfList.Handler	handler
private javax.swing.JTextArea	output
private javax.swing.JButton	refresh
private javax.swing.JButton	register

Constructors	
ConfList()	

Methods	
void	init()

Class EntryForm1Applet

public class EntryForm1Applet
extends javax.swing.JApplet

Inner Class Summary

private class	EntryForm1Applet.Handler

Fields

private javax.swing.Box	auxBox1
private javax.swing.Box	auxBox2
private javax.swing.Box	bottomBox
private javax.swing.JButton	button1
private javax.swing.JButton	button2
private javax.swing.JButton	button3
private javax.swing.Box	buttonBox
private javax.swing.JPanel	buttons
private DataEntryForm1	drawingPanel
private EntryForm1Applet.Handler	handler
private javax.swing.JLabel	headline
private CliffLogo	logo
private MyColorChooser	sliderPanel

Constructors

EntryForm1Applet()	

Methods

void	init()

Class EntryForm2Applet

public class EntryForm2Applet
extends javax.swing.JApplet

Fields	
private javax.swing.Box	auxBox1
private javax.swing.Box	auxBox2
private javax.swing.Box	bottomBox
private javax.swing.JButton	button1
private javax.swing.JButton	button2
private javax.swing.JButton	button3
private javax.swing.Box	buttonBox
private javax.swing.JPanel	buttons
private DataEntryForm2	drawingPanel
private javax.swing.JLabel	headline
private CliffLogo	logo
private MyColorChooser	sliderPanel

Constructors	
EntryForm2Applet()	

Methods	
void	init()

Class RoomSelector

public class RoomSelector
extends javax.swing.JApplet
implements java.awt.event.ActionListener

Fields	
private Rooms	card1
private Rooms	card2
private Rooms	card3
private Rooms	card4
private java.awt.CardLayout	cardManager
private javax.swing.JButton[]	controls
private javax.swing.JPanel	deck
private java.lang.String[]	names
private java.lang.String	s1
private java.lang.String	s2
private java.lang.String	s3
private java.lang.String	s4

Constructors	
RoomSelector()	

Methods	
void	actionPerformed(java.awt.event.ActionEvent e)
void	init()

Class Accts_Pay

public class Accts_Pay
extends javax.swing.JFrame

Inner Class Summary	
private class	Accts_Pay.ButtonHandler

Fields	
private javax.swing.JButton	accept
protected static int	amount
private static javax.swing.JTextField	amt_box
private javax.swing.JLabel	amt_lbl
private static java.lang.String	amt_str
private javax.swing.JButton	cancel
protected static java.lang.String	check_num
private static javax.swing.JTextField	check_num_box
private javax.swing.JLabel	check_num_lbl
protected static java.lang.String	comma
private java.awt.Container	container
private java.awt.GridBagConstraints	gbConstraints
private java.awt.GridBagLayout	gbLayout
protected static java.lang.String	insert_into_str
protected static java.lang.String	odbc_filename

protected static java.lang.String	quote
private javax.swing.JButton	reject
protected static java.lang.String	social_security_num
private static javax.swing.JTextField	ssn_box
private javax.swing.JLabel	ssn_lbl

Constructors	
Accts_Pay()	

Methods	
private static void	clear_text()
private static void	get_text()
static void	main(java.lang.String[] args)
protected void	processWindowEvent(java.awt.event.WindowEvent e)
private static void	write_to_database()

Class AttendeeDataFrm

public class AttendeeDataFrm
extends javax.swing.JFrame

Fields	
private javax.swing.Box	auxBox1
private javax.swing.Box	auxBox2
private javax.swing.Box	bottomBox
private javax.swing.JButton	button1

private javax.swing.JButton	button2
private javax.swing.JButton	button3
private javax.swing.Box	buttonBox
private javax.swing.JPanel	buttons
private DataEntryForm1	drawingPanel
private javax.swing.JLabel	headline
private int	height
private CliffLogo	logo
private MyColorChooser	sliderPanel
private int	width

Constructors	
AttendeeDataFrm()	

Methods	
static void	main(java.lang.String[] args)

Class AttendeeGUI

public class AttendeeGUI
extends javax.swing.JFrame
implements ManageShutDown

Fields	
private java.sql.Connection	Connect
private ControlPanel	Controls
private javax.swing.JTextArea	Output

private **ScrollingPanel**	ScrollArea
private javax.swing.JScrollPane	TextPane

Constructors	
AttendeeGUI()	

Methods	
static void	main(java.lang.String[] args)
protected void	processWindowEvent(java.awt.event.WindowEvent e)
void	shutDown()

Class DataDisplayGUI

public class DataDisplayGUI
extends javax.swing.JFrame
implements java.awt.event.ActionListener, javax.swing.SwingConstants

Fields	
private javax.swing.JButton[]	b
private java.awt.Container	c
private javax.swing.JLabel	ImageLabel
private javax.swing.JLabel	label
private java.lang.String[]	names
private java.lang.String[]	tools

Constructors	
DataDisplayGUI()	

Methods	
void	actionPerformed(java.awt.event.ActionEvent e)

static void	main(java.lang.String[] args)
protected void	processWindowEvent(java.awt.event.WindowEvent e)

Class DataEntryGUI

public class DataEntryGUI
extends javax.swing.JFrame
implements java.awt.event.ActionListener, BusinessConstants

Fields	
private javax.swing.JButton[]	b
private java.awt.Container	c
private javax.swing.JLabel	ImageLabel
private javax.swing.JLabel	label
private java.lang.String[]	names
private java.lang.String[]	tools
Constructors	
DataEntryGUI()	
Methods	
void	actionPerformed(java.awt.event.ActionEvent e)
static void	main(java.lang.String[] args)
protected void	processWindowEvent(java.awt.event.WindowEvent e)

Class DataQueryGUI

public class DataQueryGUI
extends javax.swing.JFrame

implements java.awt.event.ActionListener, javax.swing.SwingConstants

Fields	
private javax.swing.JButton[]	b
private java.awt.Container	c
private javax.swing.JLabel	ImageLabel
private javax.swing.JLabel	label
private java.lang.String[]	names
private java.lang.String[]	tools

Constructors	
DataQueryGUI()	

Methods	
void	actionPerformed(java.awt.event.ActionEvent e)
static void	main(java.lang.String[] args)
protected void	processWindowEvent(java.awt.event.WindowEvent e)

Class DisplayAccountsPayable

public class DisplayAccountsPayable
extends javax.swing.JFrame
implements ManageShutDown

Fields	
private java.sql.Connection	connection
private java.sql.ResultSet	resultSet
private java.sql.ResultSetMetaData	rsMetaData
private java.sql.Statement	statement

private javax.swing.JTable	table

Constructors	
DisplayAccountsPayable()	

Methods	
private void	displayResultSet(java.sql.ResultSet rs)
private java.util.Vector	getNextRow(java.sql.ResultSet rs, java.sql.ResultSetMetaData rsmd)
private void	getTable()
static void	main(java.lang.String[] args)
void	shutDown()

Class DisplayAccountsReceivable

public class DisplayAccountsReceivable
extends javax.swing.JFrame
implements ManageShutDown

Fields	
private java.sql.Connection	connection
private java.sql.ResultSet	resultSet
private java.sql.ResultSetMetaData	rsMetaData
private java.sql.Statement	statement
private javax.swing.JTable	table

Constructors	
DisplayAccountsReceivable()	

Methods

private void	displayResultSet(java.sql.ResultSet rs)
private java.util.Vector	getNextRow(java.sql.ResultSet rs, java.sql.ResultSetMetaData rsmd)
private void	getTable()
static void	main(java.lang.String[] args)
void	shutDown()

Class DisplayAttendee

public class DisplayAttendee
extends javax.swing.JFrame
implements ManageShutDown

Fields	
private java.sql.Connection	connection
private java.sql.ResultSet	resultSet
private java.sql.ResultSetMetaData	rsMetaData
private java.sql.Statement	statement
private javax.swing.JTable	table

Constructors	
DisplayAttendee()	

Methods	
private void	displayResultSet(java.sql.ResultSet rs)
private java.util.Vector	getNextRow(java.sql.ResultSet rs, java.sql.ResultSetMetaData rsmd)
private void	getTable()
static void	main(java.lang.String[] args)

void	shutDown()

Class DisplayEvents

public class DisplayEvents
extends javax.swing.JFrame
implements ManageShutDown

Fields	
private java.sql.Connection	connection
private java.sql.ResultSet	resultSet
private java.sql.ResultSetMetaData	rsMetaData
private java.sql.Statement	statement
private javax.swing.JTable	table

Constructors	
DisplayEvents()	

Methods	
private void	displayResultSet(java.sql.ResultSet rs)
private java.util.Vector	getNextRow(java.sql.ResultSet rs, java.sql.ResultSetMetaData rsmd)
private void	getTable()
static void	main(java.lang.String[] args)
void	shutDown()

Class DisplayLecturer

public class DisplayLecturer

extends javax.swing.JFrame
implements ManageShutDown

Fields	
private java.sql.Connection	connection
private java.sql.ResultSet	resultSet
private java.sql.ResultSetMetaData	rsMetaData
private java.sql.Statement	statement
private javax.swing.JTable	table

Constructors	
DisplayLecturer()	

Methods	
private void	displayResultSet(java.sql.ResultSet rs)
private java.util.Vector	getNextRow(java.sql.ResultSet rs, java.sql.ResultSetMetaData rsmd)
private void	getTable()
static void	main(java.lang.String[] args)
void	shutDown()

Class DisplayQueryResults

public class DisplayQueryResults
extends javax.swing.JFrame
implements ManageShutDown

Fields	
private java.sql.Connection	connection
private javax.swing.JTextArea	inputQuery

private java.sql.ResultSet	resultSet
private java.sql.ResultSetMetaData	rsMetaData
private java.sql.Statement	statement
private javax.swing.JButton	submitQuery
private javax.swing.JTable	table

Constructors	
DisplayQueryResults()	

Methods	
private void	displayResultSet(java.sql.ResultSet rs)
private java.util.Vector	getNextRow(java.sql.ResultSet rs, java.sql.ResultSetMetaData rsmd)
private void	getTable()
static void	main(java.lang.String[] args)
void	shutDown()

Class DisplayRegistration

public class DisplayRegistration
extends javax.swing.JFrame
implements ManageShutDown

Fields	
private java.sql.Connection	connection
private java.sql.ResultSet	resultSet
private java.sql.ResultSetMetaData	rsMetaData

private java.sql.Statement	statement
private javax.swing.JTable	table

Constructors	
DisplayRegistration()	

Methods	
private void	displayResultSet(java.sql.ResultSet rs)
private java.util.Vector	getNextRow(java.sql.ResultSet rs, java.sql.ResultSetMetaData rsmd)
private void	getTable()
static void	main(java.lang.String[] args)
void	shutDown()

Class Main_Menu

public class Main_Menu
extends javax.swing.JFrame
implements java.awt.event.ActionListener, javax.swing.SwingConstants

Fields	
private javax.swing.JButton[]	b
private java.awt.Container	c
private javax.swing.JLabel	ImageLabel
private javax.swing.JLabel	label
private java.lang.String[]	names
private java.lang.String[]	tools

Constructors	
Main_Menu()	

Methods	
void	actionPerformed(java.awt.event.ActionEvent e)
static void	main(java.lang.String[] args)
protected void	processWindowEvent(java.awt.event.WindowEvent e)

Class RandomRead

public class RandomRead
extends javax.swing.JFrame

Fields	
static long	amount
static long	check_num
static long	position
long	record_length
static long	record_number

Constructors	
RandomRead()	

Methods	
static void	main(java.lang.String[] args)

Class RandomWrite

public class RandomWrite
extends javax.swing.JFrame

Fields	
static java.lang.String	amt_str

static java.lang.String	check_num
static java.lang.String	output_str

Constructors	
RandomWrite()	

Methods	
static void	main(java.lang.String[] args)

Class ConfList.Handler

Enclosing class:

ConfList

private class ConfList.Handler
extends java.lang.Object
implements java.awt.event.ActionListener

Class ControlPanel.Temporary

Enclosing class:

ControlPanel

Constructors	
ControlPanel.Temporary()	

Methods	
void	actionPerformed(java.awt.event.ActionEvent e)

Class DBConnect

public class DBConnect
extends java.lang.Object

Fields	
static java.lang.String	DBURL
static java.lang.String	Driver
static java.lang.String	Password

static java.lang.String	User

Constructor Summary

DBConnect()	

Class EntryForm1Applet.Handler

Enclosing class:

EntryForm1Applet

private class EntryForm1Applet.Handler
extends java.lang.Object
implements java.awt.event.ActionListener

Class FindRecord

public class FindRecord
extends java.lang.Object
implements java.awt.event.ActionListener

Fields

private java.sql.Connection	AddRec_Connection
private Attendees	Attendee_Data
private ScrollingPanel	fields
private javax.swing.JTextArea	Output
private java.lang.String	QueryString
private java.sql.ResultSet	rs
private java.sql.Statement	Smt
private java.lang.String	tempSSN

Constructors

FindRecord()	
FindRecord(java.sql.Connection c, ScrollingPanel f, javax.swing.JTextArea o)	

Methods	
void	actionPerformed(java.awt.event.ActionEvent e)
void	display(java.sql.ResultSet rs)
static void	main(java.lang.String[] args)

Class Help

public class Help
extends java.lang.Object
implements java.awt.event.ActionListener

Fields	
private javax.swing.JTextArea	output

Constructors
Help()
Help(javax.swing.JTextArea o)

Methods	
void	actionPerformed(java.awt.event.ActionEvent e)
static void	main(java.lang.String[] args)

Class LoadDB

public class LoadDB
extends java.lang.Object

Field Summary	
(package private) java.lang.Object[]	data
(package private) java.sql.Connection	MyConnection
(package private) java.sql.Statement	MyStatement

(package private) Attendees	Set1

Constructors	
LoadDB()	

Methods	
void	cleanUp() Method cleanUp() will close the database connection cleanly.
void	executeInsert(java.lang.Object[] data) THE METHOD executeInseret can insert data into any Table.
void	load() The method load will actually call the executeInsert to actually enter the data into the database
static void	main(java.lang.String[] args)

Class ManageString

public class ManageString
extends java.lang.Object

Constructors	
ManageString()	

Methods	
static java.lang.String	addApostrophe(java.lang.String Input)
static void	main(java.lang.String[] args)
static java.lang.String	removeApostrophe(java.lang.String Output)

Class Registration

public class Registration
extends java.lang.Object

Fields	
private java.lang.String	AttendeeSSN

(package private) java.lang.Object[]	data
private java.lang.String	Event_ID
static java.lang.String	Insert
private static int	numColumns
static java.lang.String	quote
private static java.lang.String	TableName
static java.lang.String	Values

Constructors	
Registration()	

Methods	
java.lang.String	constructSqlObject()
java.lang.String	getAttendeeSSN()
java.lang.String	getEventID()
static void	main(java.lang.String[] args)
void	setAttendeeSSN(java.lang.String m_AttendeeSSN)
void	setEvent_ID(java.lang.String m_Event_ID)

Class UpdateRecord

public class UpdateRecord
extends java.lang.Object
implements java.awt.event.ActionListener

Fields	
private java.sql.Connection	AddRec_Connection

private Attendees	Attendee_Data
private java.lang.String	comma
private ScrollingPanel	fields
private javax.swing.JTextArea	Output
private java.lang.String	QueryString
private java.lang.String	quote
private java.sql.ResultSet	rs
private java.sql.Statement	Smt
private java.lang.String	tempSSN

Constructors

UpdateRecord()
UpdateRecord(java.sql.Connection c, ScrollingPanel f, javax.swing.JTextArea o)

Methods

void	actionPerformed(java.awt.event.ActionEvent e)
private void	fillAttendee()
static void	main(java.lang.String[] args)
private void	processUpdate()

Class AttendServ

public class AttendServ

Fields

private java.sql.Connection	connection
private java.lang.String	password
private java.sql.Statement	statement
private java.lang.String	URL
private java.lang.String	username

Constructors	
AttendServ()	

Methods	
void	destroy()
void	doPost(HttpServletRequest req, HttpServletResponse res)
void	init(ServletConfig config)
private boolean	insertIntoDB(java.lang.String stringtoinsert)

Class DispServ2

DispServ2

public class DispServ2

Fields	
private java.sql.Connection	connection
private java.lang.String	password
private java.sql.ResultSet	resultSet
private java.sql.Statement	statement
private java.lang.String	URL

private java.lang.String	username

Constructors	
DispServ2()	

Methods	
void	destroy()
void	doGet(HttpServletRequest req, HttpServletResponse res)
void	init(ServletConfig config)
private boolean	queryDB(java.lang.String query)

Class LectServ

public class LectServ

Fields	
private java.sql.Connection	connection
private java.lang.String	password
private java.sql.Statement	statement
private java.lang.String	URL
private java.lang.String	username

Constructors	
LectServ()	

Methods	
void	destroy()
void	doPost(HttpServletRequest req, HttpServletResponse res)
void	init(ServletConfig config)

private boolean	insertIntoDB(java.lang.String stringtoinsert)

Class RoomServ

public class RoomServ

Fields	
private java.sql.Connection	connection
private java.lang.String	password
private java.sql.ResultSet	resultSet
private java.sql.Statement	statement
private java.lang.String	URL
private java.lang.String	username

Constructors	
RoomServ()	

Methods	
void	destroy()
void	doGet(HttpServletRequest req, HttpServletResponse res)
void	init(ServletConfig config)
private boolean	queryDB(java.lang.String query)

Interface ManageAttendee

public interface ManageAttendee

Methods	
void	fillAttendee()

Interface ManageShutDown

public interface ManageShutDown

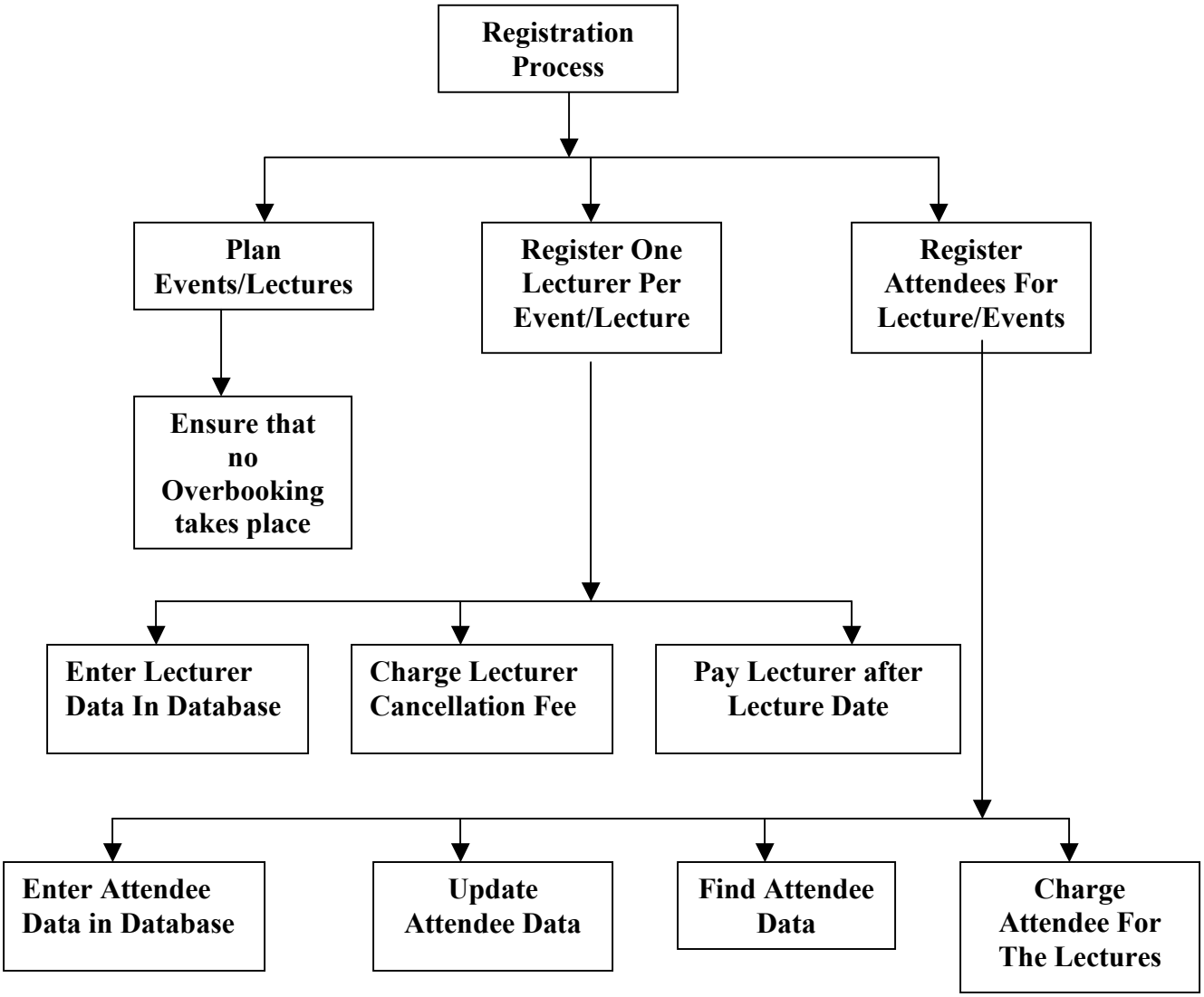
Methods	
void	shutDown()

Interface BusinessConstants

public interface BusinessConstants
extends javax.swing.SwingConstants

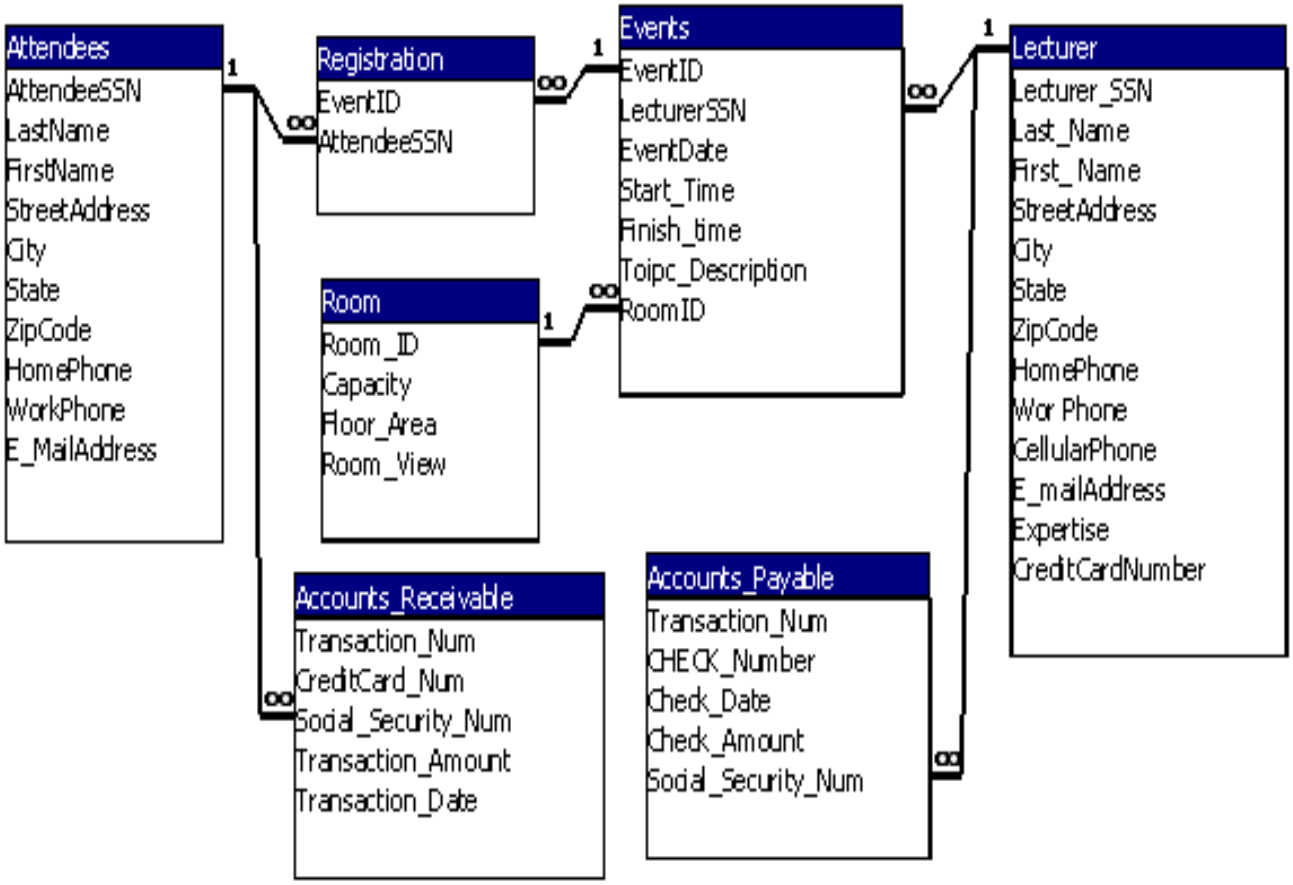
Fields	
static double	LecturerPortion **the factor by which lecturer will get paid.**
static int	NumRooms **Number of conference or seminar rooms**
static int	NumStates **Number of States from where the Attendees and Lecturer are accepted.**
static int	SeminarFee **The seminar fee per attendee.**

Business Function Diagrams



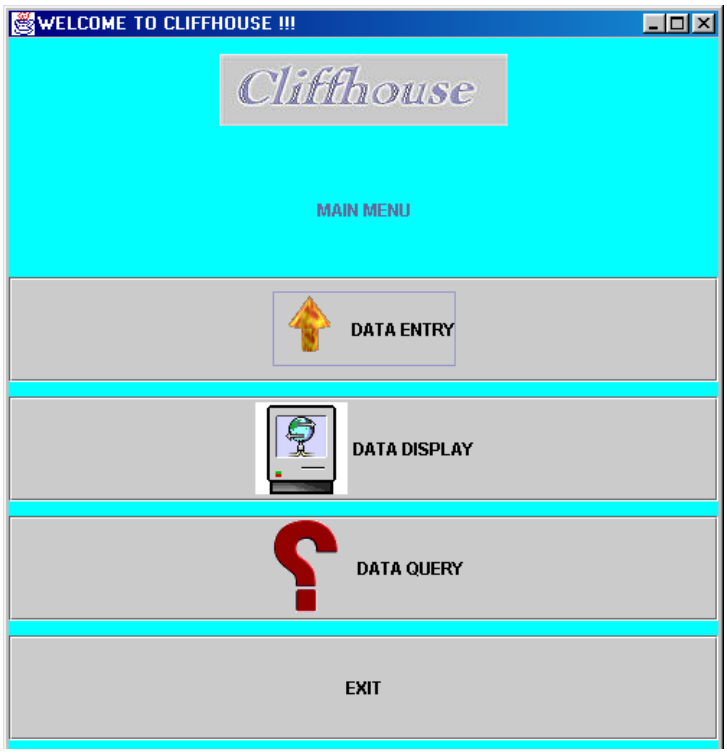
Database ERD To Accomplish Above Business Functions

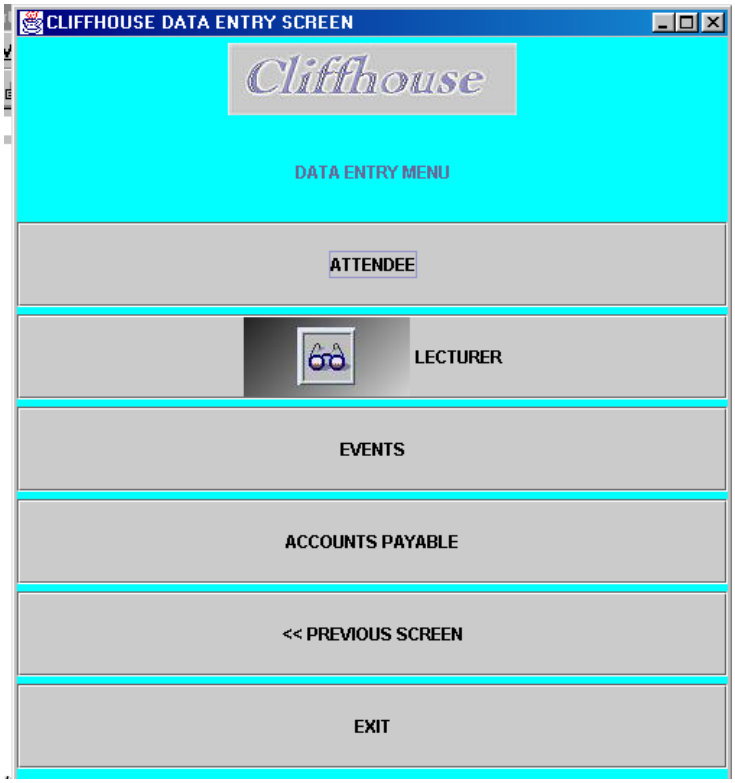
ERD to accomplish the above business function is shown on next page.

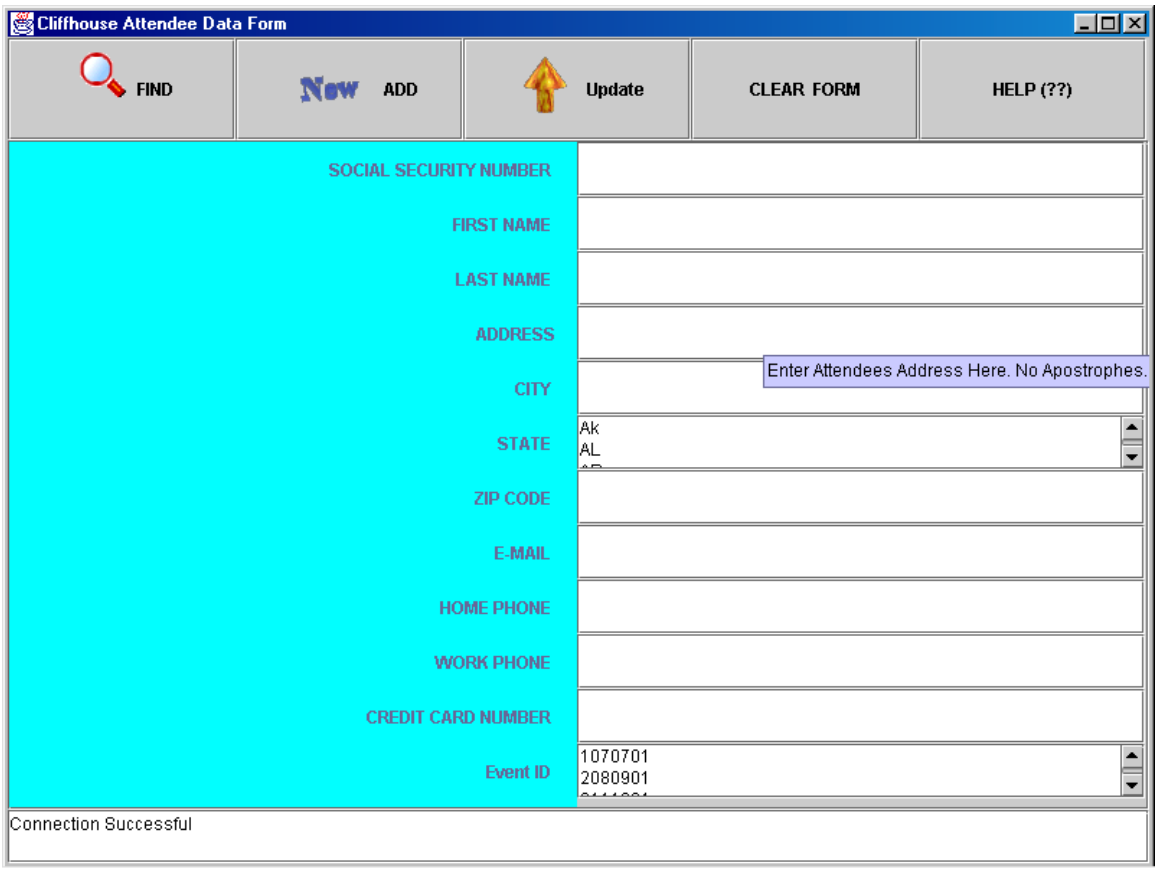


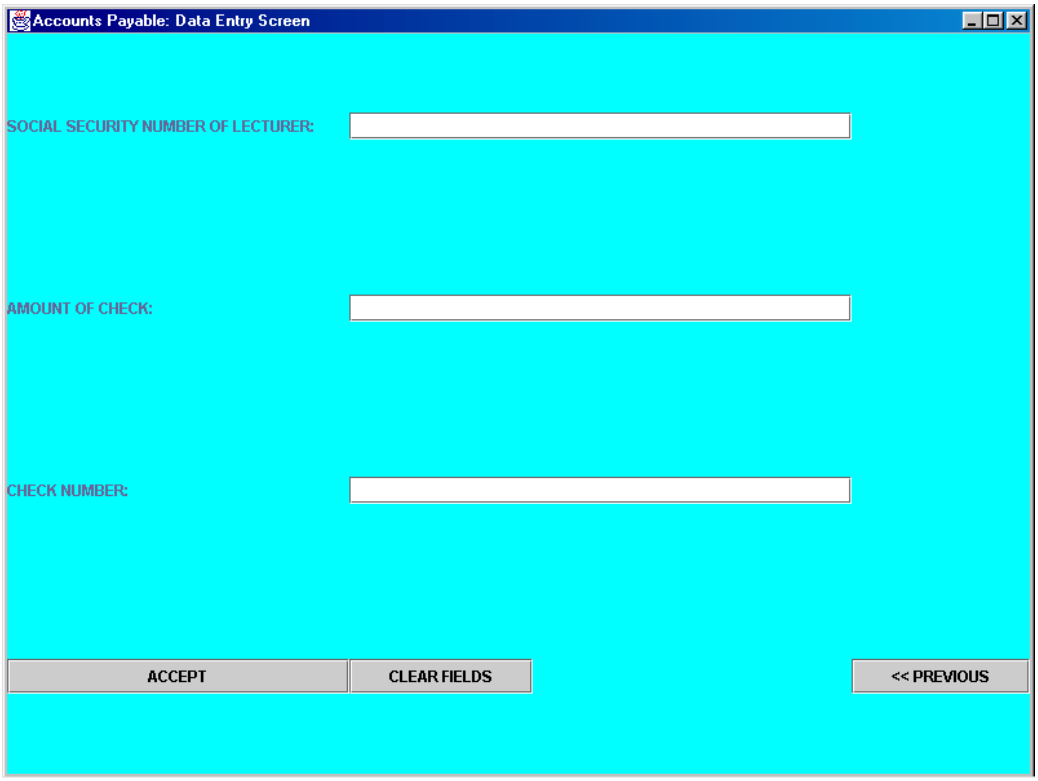
Interface Screen in the Database and Related Software that Accomplish Business Functions

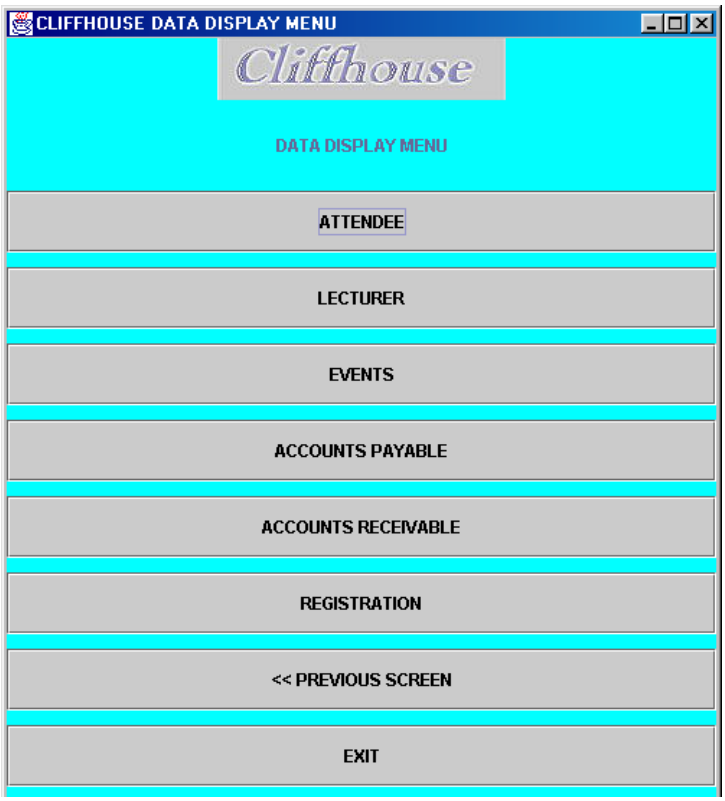
Related Screens are shown on next page and onwards.

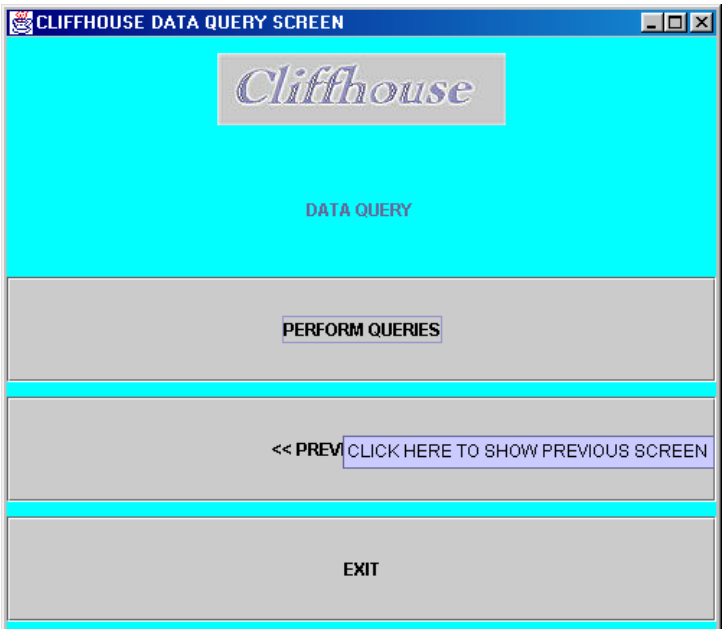

WELCOME TO CLIFFHOUSE !!!
Cliffhouse
MAIN MENU
DATA ENTRY
DATA DISPLAY
DATA QUERY
EXIT


CLIFFHOUSE DATA ENTRY SCREEN
Cliffhouse
DATA ENTRY MENU
ATTENDEE
LECTURER
EVENTS
ACCOUNTS PAYABLE
<< PREVIOUS SCREEN
EXIT


Cliffhouse Attendee Data Form
FIND
New
ADD
Update
CLEAR FORM
HELP (??)
SOCIAL SECURITY NUMBER
FIRST NAME
LAST NAME
ADDRESS
CITY
Enter Attendees Address Here. No Apostrophes.
STATE
Ak
AL
ZIP CODE
E-MAIL
HOME PHONE
WORK PHONE
CREDIT CARD NUMBER
Event ID
1070701
2080901
Connection Successful

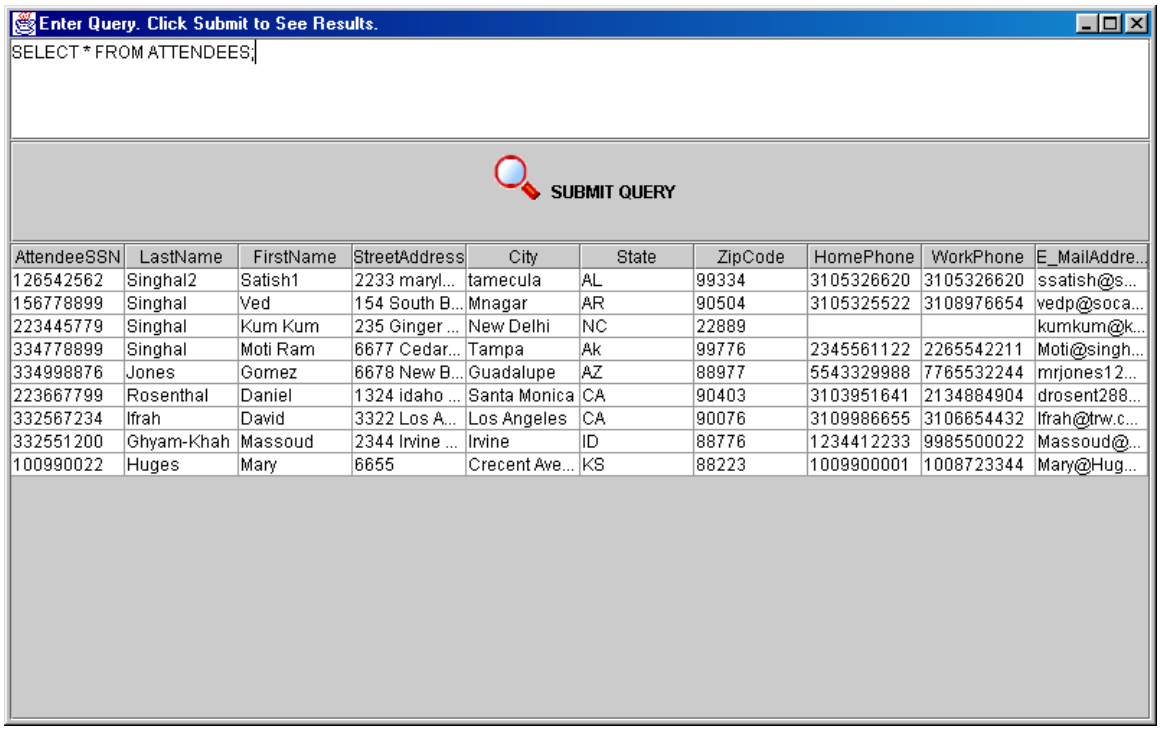

Accounts Payable: Data Entry Screen
SOCIAL SECURITY NUMBER OF LECTURER:
AMOUNT OF CHECK:
CHECK NUMBER:
ACCEPT
CLEAR FIELDS
<< PREVIOUS





THIS IS THE DISPLAY OF LECTURER TABLE.

Lecturer_...	Last_Na...	First_ Na...	StreetAd...	City	State	ZipCode	HomePh...	Wor Pho...	CellularP...	E_mailA...	Expertise	CreditCa...
1235225...	Singhal	Uma	7 Del Ca...	Irvine	CA	92666	3105326...	310532...	3106662...	ssatish...	Comput...	
5524622...	Valicente	Walter	7 Choch...	Redond...	PA	19174	2134453...	213990...	3107765...	valicente...	Network ...	
6672266...	Rosenthal	Daniel	55 Santa...	New York	SD	22334	6098875...	609344...	6095674...	rosenta...	Visual B...	



AttendeeSSN	LastName	FirstName	StreetAddress	City	State	ZipCode	HomePhone	WorkPhone	E_MailAddre...
126542562	Singhal2	Satish1	2233 maryl...	tamecula	AL	99334	3105326620	3105326620	ssatish@s...
156778899	Singhal	Ved	154 South B...	Mnagar	AR	90504	3105325522	3108976654	vedp@soca...
223445779	Singhal	Kum Kum	235 Ginger ...	New Delhi	NC	22889			kumkum@k...
334778899	Singhal	Moti Ram	6677 Cedar...	Tampa	Ak	99776	2345561122	2265542211	Moti@singh...
334998876	Jones	Gomez	6678 New B...	Guadalupe	AZ	88977	5543329988	7765532244	mrjones12...
223667799	Rosenthal	Daniel	1324 idaho ...	Santa Monica	CA	90403	3103951641	2134884904	drosent288...
332567234	Ifrah	David	3322 Los A...	Los Angeles	CA	90076	3109986655	3106654432	Ifrah@trw.c...
332551200	Ghyam-Khah	Massoud	2344 Irvine ...	Irvine	ID	88776	1234412233	9985500022	Massoud@...
100990022	Huges	Mary	6655	Crecent Ave...	KS	88223	1009900001	1008723344	Mary@Hug...

Application File Diagram

The Text Box below shows the list of all *.java files

Attendees.java ,LoadDB.java, AttendeeDataFrm.java , MyColorChooser.java , CliffLogo.java ,DataEntryForm1.java, AttendeeGUI.java , ScrollingPanel.java ,ControlPanel.java ,BusinessConstants.java , AddRecord.java ,Accounts_Receivable.java, Registration.java, DBConnect.java, ManageString.java, Help.java, UpdateRecord.java, FindRecord.java, ClearFields.java, ManageAttendee.java, DisplayQueryResults.java, ManageShutDown.java, Accts_Pay.java, DataEntryGUI.java, Main_Menu.java, DataDisplayGUI.java, DataQueryGUI.java, DisplayAttendee.java, DisplayLecturer.java, DisplayEvents.java, DisplayAccountsPayable.java, DisplayAccountsReceivable.java, DisplayRegistration.java, AttendServ.java, ConfList.java. DataEntryForm2.java, DispServ2.java, EntryForm1Applet.java, EntryForm2Applet.java, LectServ.java, RoomSelector.java, RoomServ.java, RandomWrite.java, RandomRead.java

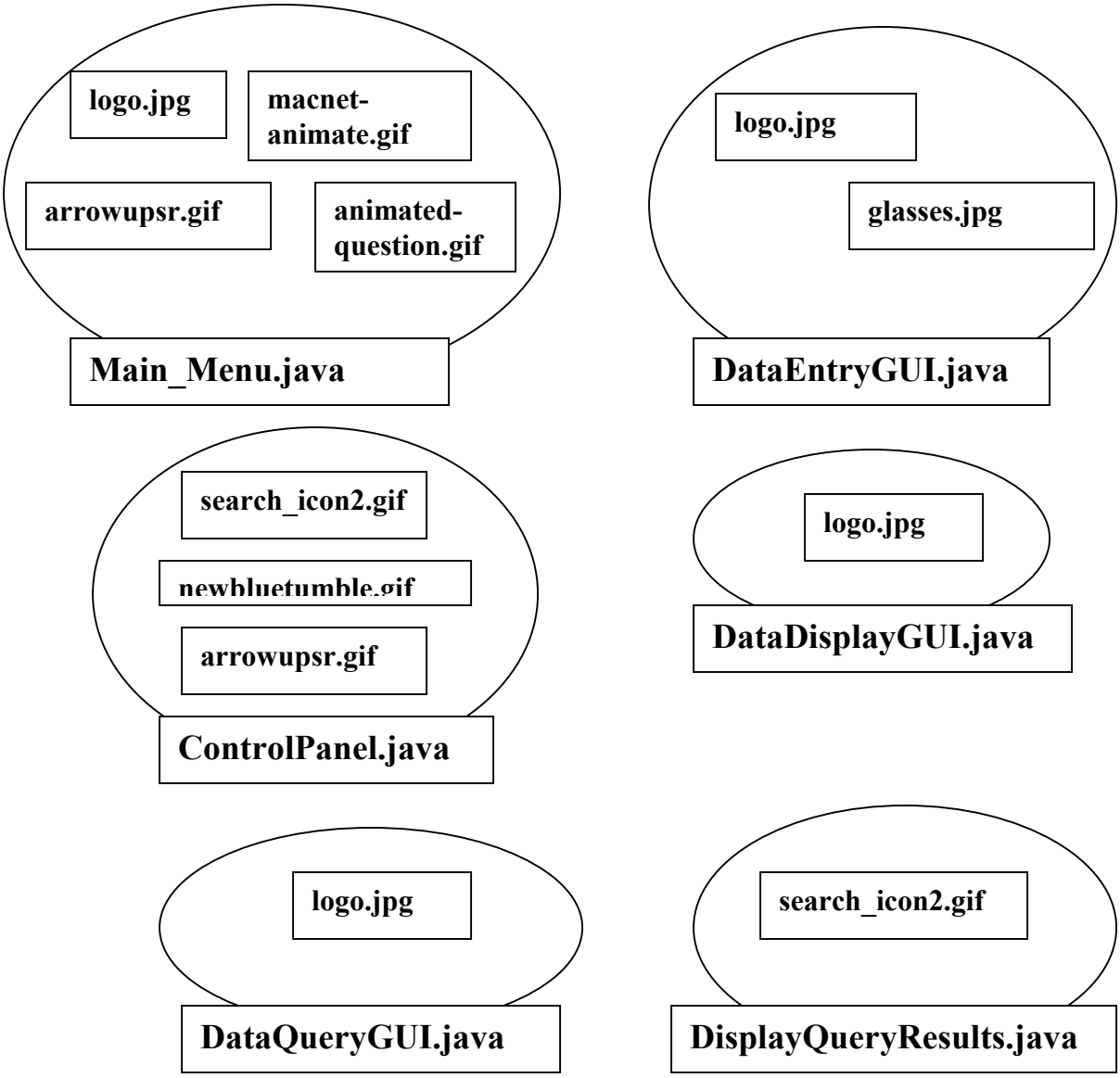
The text box below lists all the *.jpg files

logo.jpg, glasses.jpg, auditorium.jpg, mtgroom2.jpg, mtgroom3.jpg, mtgroom4.jpg, mtgroom5.jpg

The text box below shows all the *.gif files

arrowupsr.gif, search_icon2.gif, newbluetumble.gif, arrowrightsr.gif, arrow2rightsy.gif ch2brn.gif, animated-question.gif, macnet-animate.gif

The relationships between *.java files and *.jpg and *.gif files are shown below.

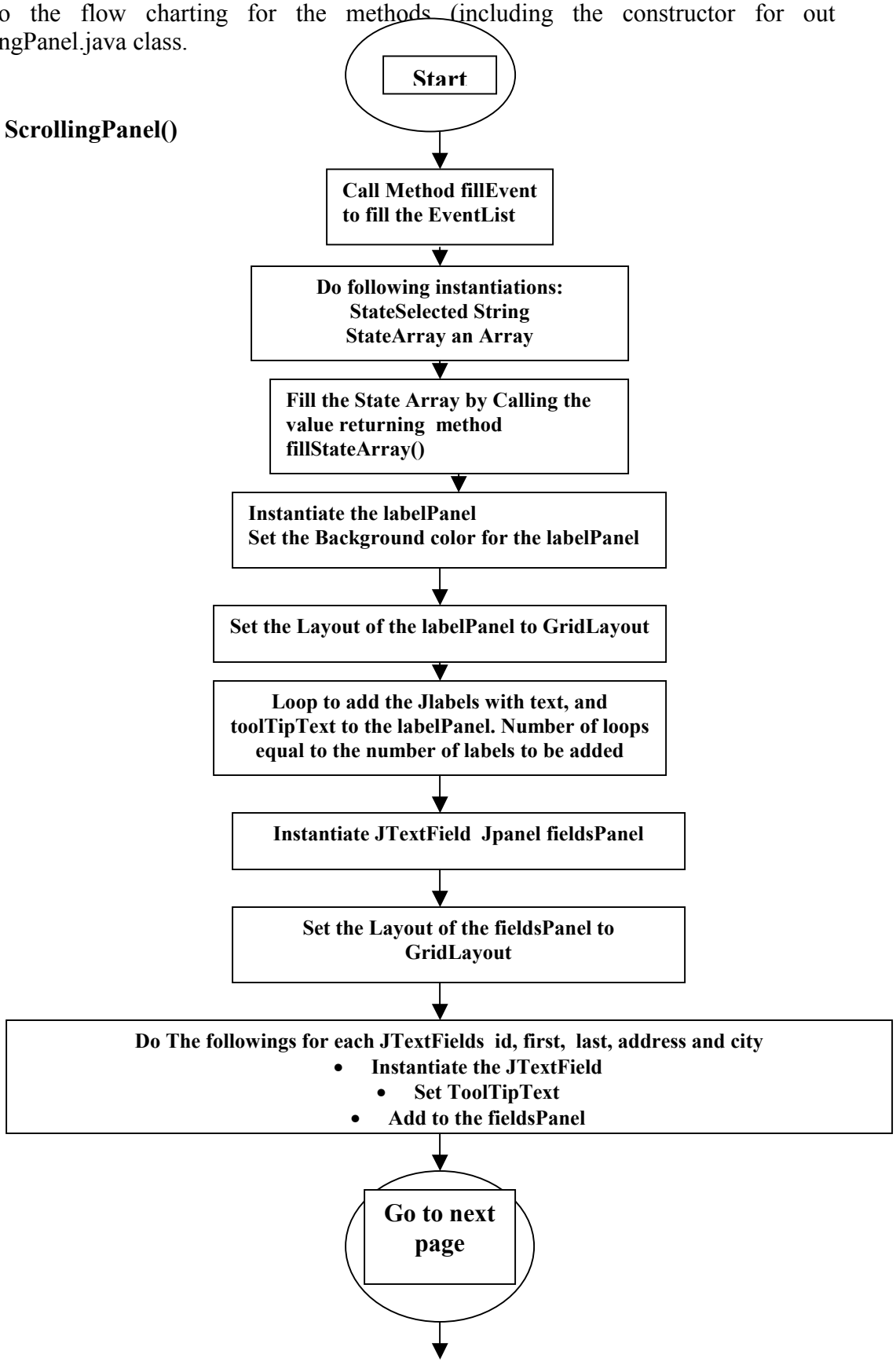


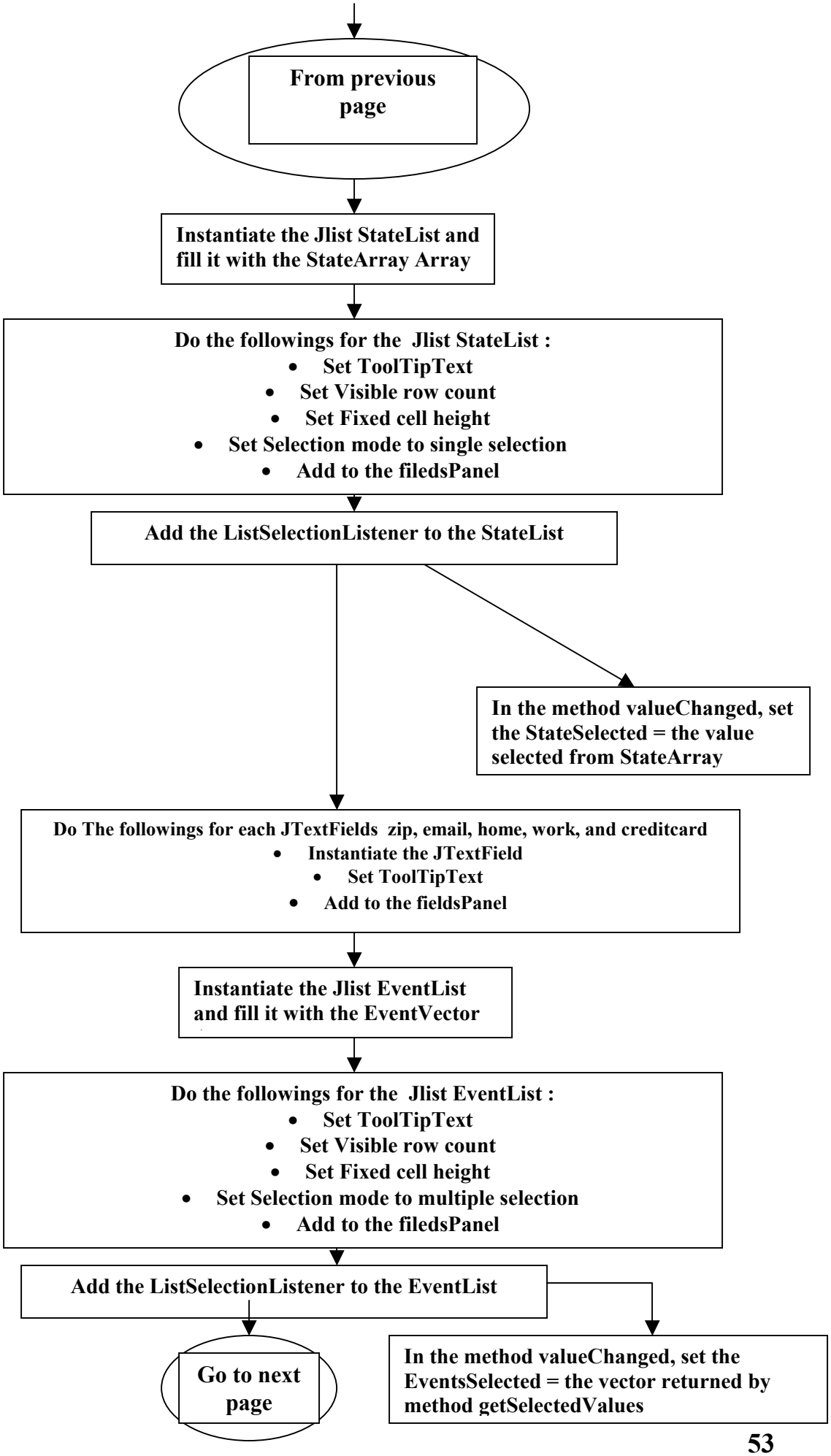
Other files are unrelated to each other.

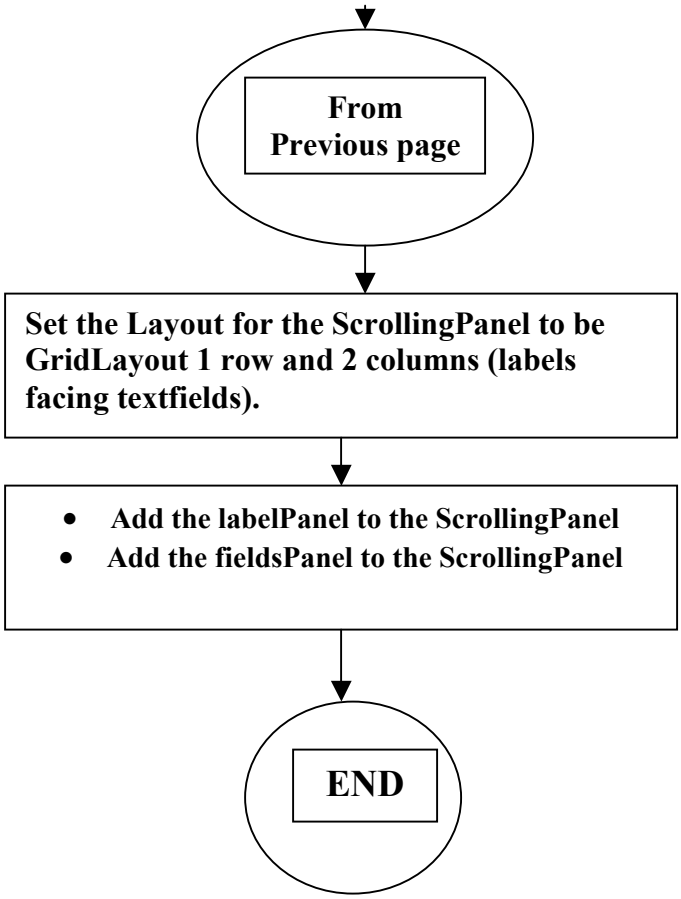
Flow Charts

We do the flow charting for the methods (including the constructor for out ScrollingPanel.java class.

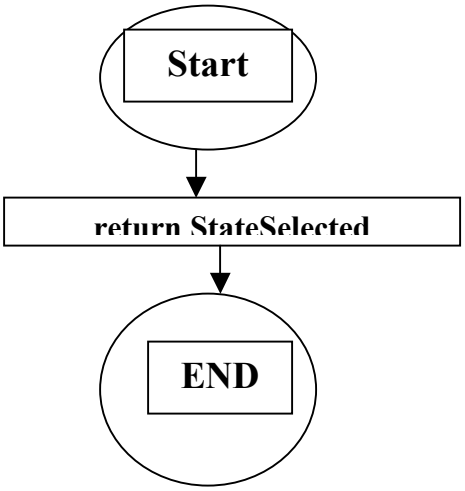
public ScrollingPanel()



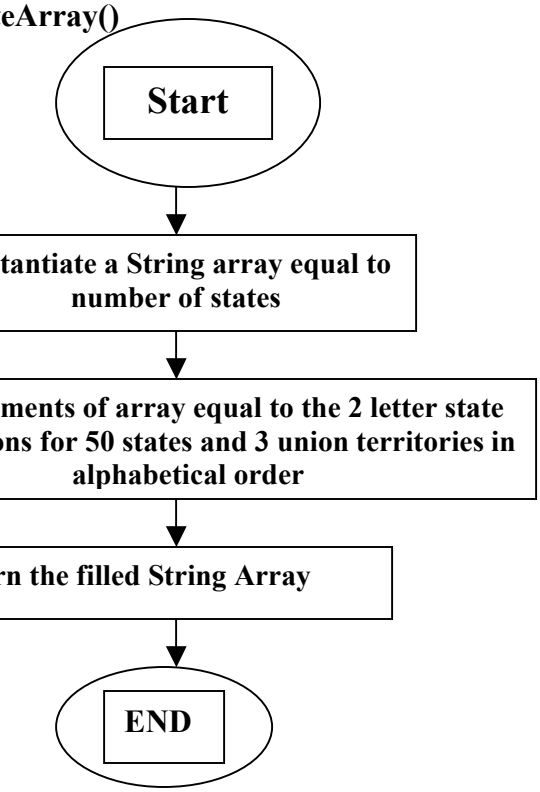

From previous page
Instantiate the Jlist StateList and fill it with the StateArray Array
Do the followings for the Jlist StateList :
Set ToolTipText
Set Visible row count
Set Fixed cell height
Set Selection mode to single selection
Add to the filedsPanel
Add the ListSelectionListener to the StateList
In the method valueChanged, set the StateSelected = the value selected from StateArray
Do The followings for each JTextFields zip, email, home, work, and creditcard
Instantiate the JTextField
Set ToolTipText
Add to the fieldsPanel
Instantiate the Jlist EventList and fill it with the EventVector
Do the followings for the Jlist EventList :
Set ToolTipText
Set Visible row count
Set Fixed cell height
Set Selection mode to multiple selection
Add to the filedsPanel
Add the ListSelectionListener to the EventList
Go to next page
In the method valueChanged, set the EventsSelected = the vector returned by method getSelectedValues



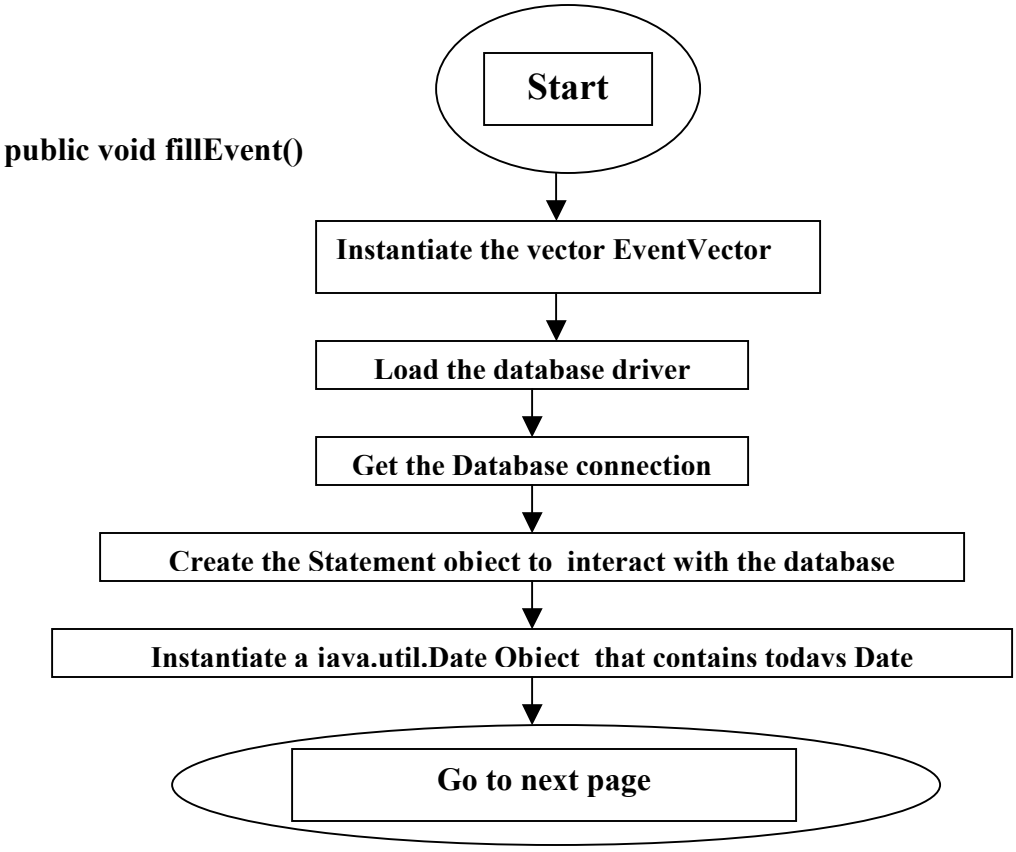
Public Object [] getEvents()

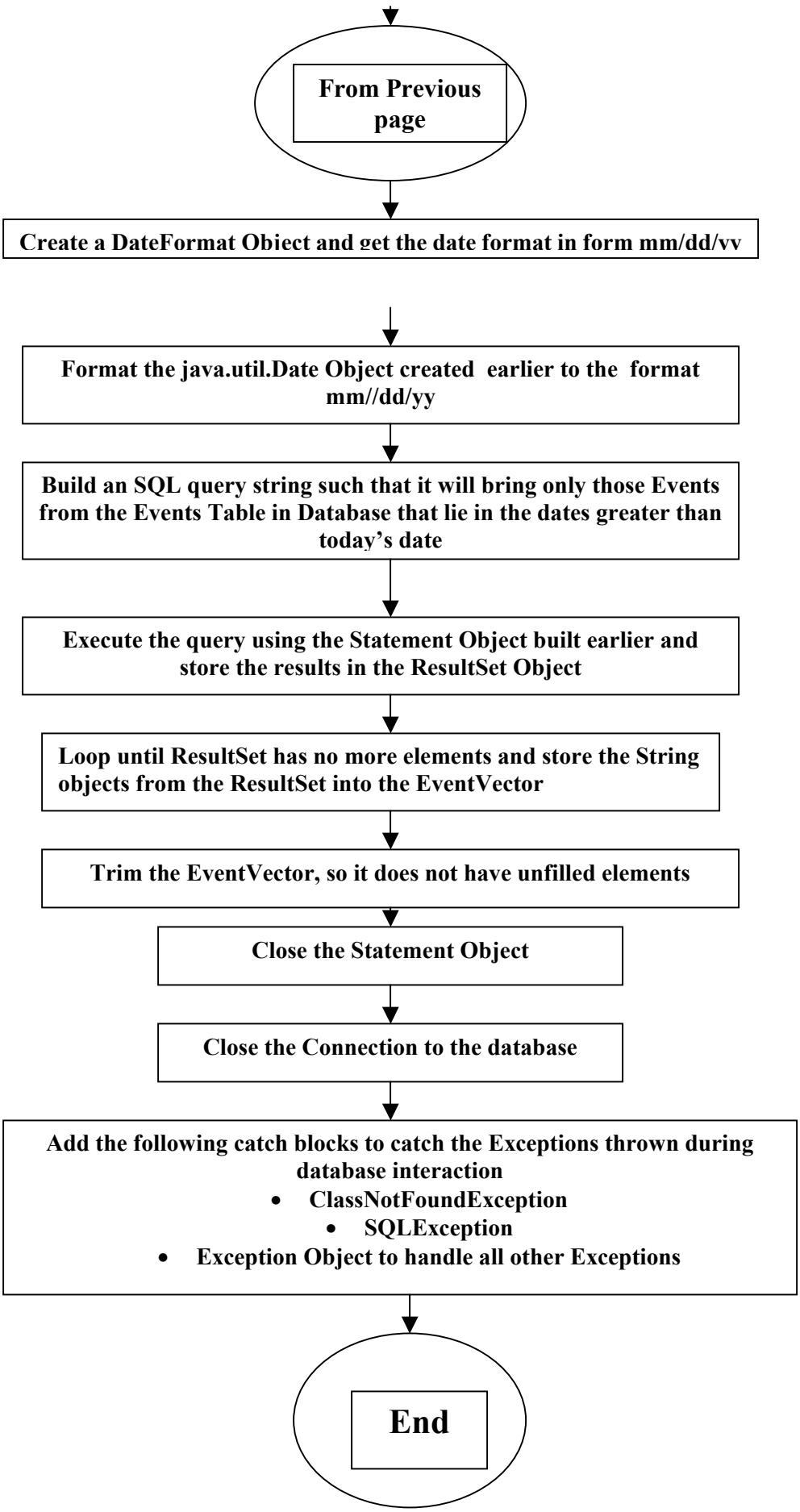


private String [] fillStateArray()



public void fillEvent()




From Previous page
Create a DateFormat Object and get the date format in form mm/dd/yy
Format the java.util.Date Object created earlier to the format mm//dd/yy
Build an SQL query string such that it will bring only those Events from the Events Table in Database that lie in the dates greater than today's date
Execute the query using the Statement Object built earlier and store the results in the ResultSet Object
Loop until ResultSet has no more elements and store the String objects from the ResultSet into the EventVector
Trim the EventVector, so it does not have unfilled elements
Close the Statement Object
Close the Connection to the database
Add the following catch blocks to catch the Exceptions thrown during database interaction
ClassNotFoundException
SQLException
Exception Object to handle all other Exceptions
End

Special Topics

In This section we describe special topics such as Networking, Java Beans, RMI etc. We do not discuss servlets because we actually implemented servlets in our project.

Networking

Sometimes several computers need to share information with one another. In thisparticular case, for example, several operators at different computers may be entering data all at once; the data base resides on the *server* computer; the computers that connect up to it are the *client* computers. The client computers perform some action; e.g. reading in data entered from the keyboard, and then the server computer perform some operation that the client computers ask it to do, such as writing data to a formal database. One server computer may have many client computers making requests of it.

There are a number of *protocols* that are used for communication between computers; one of the most common is HTTP or *Hyper Text Transfer Protocol.* This uses two main methods; *GET* and *POST.* GET retrieves informatin from the server; POST sends information to the server.

Sockets* are used in communication between computers. These simulate (in terms of syntax) reading to or writing from a file. There are two kinds of sockets; *stream sockets*, and *datagram sockets*. In a stream socket, data flows continuously from one process to another. This is sometimes referred to as a *connection oriented service.* A binary -1 (end-of-file mark) is transmitted at the end of a data stream. An error (EOF exception) is generated when a program tries to read past the end-of-file mark--just as it would if the program tried to read past the end of a file. After the data is transmitted, the socket is *closed.

Datagram sockets are like light quanta--the data is broken up into discrete packets. of information. Only, instead of wave packets (photons), the packets in this case are byte arrays. There are some hazards associated with Datagram sockets. Packets can be lost, duplicated, or their order may be switched around. Extra code must be written to insure data integrity.

A server always requires a *port number* and a *queue length*. A client also requires a port number--and a *server address* as well. The connections are part of a *socket* object. The port number is the physical i/o port where the computer looks for the data. The server address is where the client computers send their data. The connection point between a client computer and a server computer is called the *handshake point.*
The *queue length* is the number of requests from clients that a server can handle. This is applicable for all socket connections. Datagram sockets require some

additional information: the name of the byte array and its length--it has to know how much information to expect per packet. Failure to establish the i/o port, the server address will result in an IOException error. Similarly, an error in sending a Datagram packet will also result in an IOException error.

JavaBeans

Programmers in Java, Visual Basic, C++ and other object-oriented languages are used to the idea of re-usable software objects such as buttons, labels, text fields, and text areas. Java has a procedure whereby a programmer can create and archive his or her own reusable objects. Archived re-usable objects created by a Java Programmer are called Java Beans. (In Visual Basic, they are called Active-X controls.) A testing/previewing facility called the Bean Box is also provided.

Suppose a user wants to create an object--say an animated picture with start and stop buttons, for example.

The user can drop the three components--the animated icon, and its start and stop buttons, into the bean box for testing purposes. The bean box contains menus--each with a number of tools--for the user to do this.

It contains 4 windows: Toolbox, Bean Box, Properties, and Method Tracer. The bean box also has its own directory, BDK1.1. A subdirectory contains the JAR (JAva ARchive) files; BDK1.1\jars.

When the user is ready to make a class a Java Bean he/she does the following:
First, the user adds a package statement to the very start of the program.
Second: after the "implements ActionListener" statement, the user must add ", Serializable".
The program must be compiled using a special option, -d. E.g. javac -d UserClass.java.

Java beans are then put into an archive file, the JAR file. A special text file, manifest.tmp, contains descriptions of objects in the JAR file. To create the JAR file, we type "jar.cfm UserClass.jar manifest.tmp jhtp3beans *.*". To check if the Java Bean is in the archive file, we type "jar tvf UserClass.jar", UserClass being the name of the file we are using in this example. If the Java Bean contains its own "main", we can run it by typing "java -jar UserClass.jar", if UserClass is the name of the Java Bean. (Of course, it can have any name.). If there is no "manifest" text file, we can run the program like this:
"java -cp UserClass.jar jhtp3beans.UserClass".

Remote Method Invocation

Remote Method Invocation (RMI) is the object equivalent of Remote Procedure Calls (RPC). While RPC allows you to call procedures over a network, RMI invokes an object's methods over a network.

In the RMI model, the server defines objects that the client can use remotely. The clients can now invoke methods of this remote object as if it were a local object running in the same virtual machine as the client. RMI hides the underlying mechanism of transporting method arguments and return values across the network. In Java-RMI, an argument or return value can be of any primitive Java type or any other **Serializable** Java object.

What is Java-RMI and how does it compare with other Middleware Specifications?

1. **Java-RMI is a Java-specific middleware spec that allows client Java programs to invoke server Java objects as if they were local.**
2. **Java-RMI is tightly coupled with the Java language. Hence there are no separate IDL mappings that are required to invoke remote object methods.**
3. **Since Java-RMI is tightly coupled with The Java Language, Java-RMI can work with true sub-classes.**
4. **Because of this, parameters passed during method calls between machines can be true Java Objects.**
5. **If a process in an RMI system receives an object of a class that it has never seen before, it can request that its class information be sent over the network.**
6. **Over and above all this, Java-RMI supports Distributed Garbage Collection that ties into the local Garbage Collectors in each JVM.**

What makes Java-RMI tick

Since both the client and the server may reside on different machines/processes, there needs to be a mechanism that can establish a relationship between the two. Java-RMI uses a network-based registry program called RMIRegistry to keep track of the distributed objects.

The RMI Registry

The server object makes methods available for remote invocation by binding it to a name in the RMI Registry. The client object, can thus check for the availability of a certain server object by looking up its name in the

registry. The RMI Registry thus acts as a central management point for Java-RMI. The RMI Registry is thus a simple name repository. It does not address the problem of actually invoking remote methods.

Stubs and Skeletons

Since the two objects may physically reside on different machines, a mechanism is needed to transmit the client's request to invoke a method on the server object to the server object and provide a response. Java-RMI uses an approach similar to RPC in this regard. The code for the server object must be processed by an RMI compiler called **rmic**, which is part of the JDK.

The **rmic** compiler generates two files: a stub and a skeleton. The stub resides on the client machine and the skeleton resides on the server machine. When a client invokes a server method, the JVM looks at the stub to do type checking. The request is then routed to the skeleton on the server, which in turn calls the appropriate method on the server object. In other words, the stub acts as a proxy to the skeleton and the skeleton is a proxy to the actual remote method.

Test Results:

We are showing test results for main data entry and data display classes only because they were the most complex ones. We monitor the items in the database before and after the data entry. Following screen shots give the database tables Attendees, Registration and Accounts_Receivable before entering a new Attendee Cindy Poppins. We can clearly see that she is not in this table. Let us also assume that Cindy poppins has the following information.

Social Security Number: 234567890

Address : 3344 Poppins Lane, E Mail : cindy@cindy.com

City: Chicago, State : IL, Zip: 67834, Home Phone: 1003004000

Work Phone: 1002334455 Credit Card Number: 2344443255666655 and she chooses to register for 3 seminars (total fee $300). The seminar events she chooses are: event numbers :1070701,2080901,31111201.

Table Attendees Before Data Entry for Cindy Poppins

THIS IS THE DISPLAY OF ATTENDEE TABLE.

AttendeeSSN	LastName	FirstName	StreetAddress	City	State	ZipCode	HomePhone	WorkPhone	E_MailAddre...
126542562	Singhal2	Satish1	2233 maryl...	tamecula	AL	99334	3105326620	3105326620	ssatish@s...
156778899	Singhal	Ved	154 South B...	Mnagar	AR	90504	3105325522	3108976654	vedp@soca...
223445779	Singhal	Kum Kum	235 Ginger ...	New Delhi	NC	22889			kumkum@k...
334778899	Singhal	Moti Ram	6677 Cedar...	Tampa	Ak	99776	2345561122	2265542211	Moti@singh...
334998876	Jones	Gomez	6678 New B...	Guadalupe	AZ	88977	5543329988	7765532244	mrjones12...
223667799	Rosenthal	Daniel	1324 idaho ...	Santa Monica	CA	90403	3103951641	2134884904	drosent288...
332567234	Ifrah	David	3322 Los A...	Los Angeles	CA	90076	3109986655	3106654432	Ifrah@trw.c...
332551200	Ghyam-Khah	Massoud	2344 Irvine ...	Irvine	ID	88776	1234412233	9985500022	Massoud@...
100990022	Huges	Mary	6655	Crecent Ave...	KS	88223	1009900001	1008723344	Mary@Hug...

Table Events (not affected by Attendee Data Entry)

THIS IS THE DISPLAY OF EVENTS TABLE.

EventID	LecturerSSN	EventDate	Start_Time	Finish_time	Toipc_Description	RoomID
1070701	123522562	2001-07-07 00:00:...	1899-12-30 08:00:...	1899-12-30 17:00...	C++ Programming	1
2080901	552462233	2001-08-09 00:00:...	1899-12-30 09:00:...	1899-12-30 18:00...	Smashing Java Pr...	2
3111201	667226699	2001-11-12 00:00:...	1899-12-30 10:00:...	1899-12-30 19:00...	Eat all you can Vis...	3
4040401	667226699	2001-04-04 00:00:...	1899-12-30 07:00:...	1899-12-30 17:00...	All You wish to kn...	4

Table Registration Before Data Entry for Cindy Poppins

THIS IS THE DISPLAY OF REGISTRATION TABLE.

EventID	AttendeeSSN
1070701	126542562
3111201	126542562
1070701	334998876
2080901	334998876
1070701	223667799
2080901	223667799
3111201	223667799
1070701	332567234
2080901	332567234
3111201	332567234
1070701	332551200
2080901	332551200
1070701	100990022
2080901	100990022
3111201	100990022

Now we enter the information for Cindy Poppins as described above in the Attendee data entry form:

Next page shows the data entry form for Cindy Poppins

Cliffhouse Attendee Data Form

FIND | ADD | Update | CLEAR FORM | HELP (??)

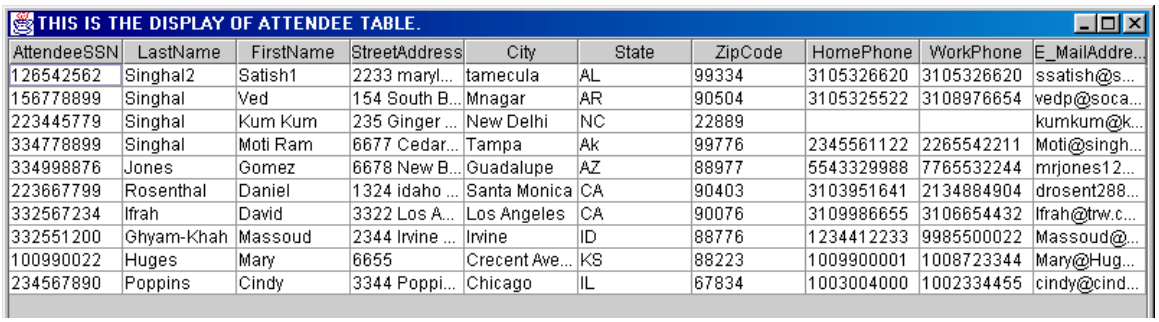
SOCIAL SECURITY NUMBER	234567890
FIRST NAME	Cindy
LAST NAME	Poppins
ADDRESS	3344 Poppins Lane
CITY	Chicago
STATE	IL IN
ZIP CODE	67834
E-MAIL	cindy@cindy.com
HOME PHONE	1003004000
WORK PHONE	Zip: 67834, Home Phone: 1003004000 Work Phone
CREDIT CARD NUMBER	2344443255666655
Event ID	2080901 3111201

Connection Successful

We click on Add after filling the form and a JoptionPane will ask to confirm all the information. Clicking Ok on that will add the

information into the database. We see all the tables now after insertion of Cindy Poppins

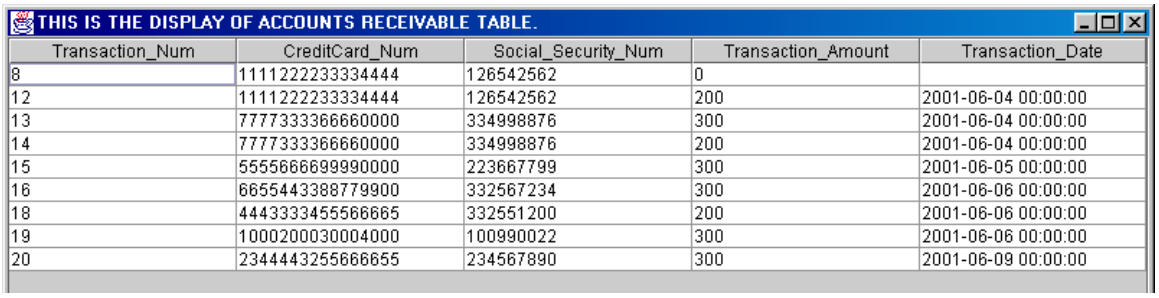
Table Attendees after Insertion of Cindy Poppins



THIS IS THE DISPLAY OF ATTENDEE TABLE.

AttendeeSSN	LastName	FirstName	StreetAddress	City	State	ZipCode	HomePhone	WorkPhone	E_MailAddre...
126542562	Singhal2	Satish1	2233 maryl...	tamecula	AL	99334	3105326620	3105326620	ssatish@s...
156778899	Singhal	Ved	154 South B...	Mnagar	AR	90504	3105325522	3108976654	vedp@soca...
223445779	Singhal	Kum Kum	235 Ginger ...	New Delhi	NC	22889			kumkum@k...
334778899	Singhal	Moti Ram	6677 Cedar...	Tampa	Ak	99776	2345561122	2265542211	Moti@singh...
334998876	Jones	Gomez	6678 New B...	Guadalupe	AZ	88977	5543329988	7765532244	mrjones12...
223667799	Rosenthal	Daniel	1324 idaho ...	Santa Monica	CA	90403	3103951641	2134884904	drosent288...
332567234	Ifrah	David	3322 Los A...	Los Angeles	CA	90076	3109986655	3106654432	Ifrah@trw.c...
332551200	Ghyam-Khah	Massoud	2344 Irvine ...	Irvine	ID	88776	1234412233	9985500022	Massoud@...
100990022	Huges	Mary	6655	Crecent Ave...	KS	88223	1009900001	1008723344	Mary@Hug...
234567890	Poppins	Cindy	3344 Poppi...	Chicago	IL	67834	1003004000	1002334455	cindy@cind...

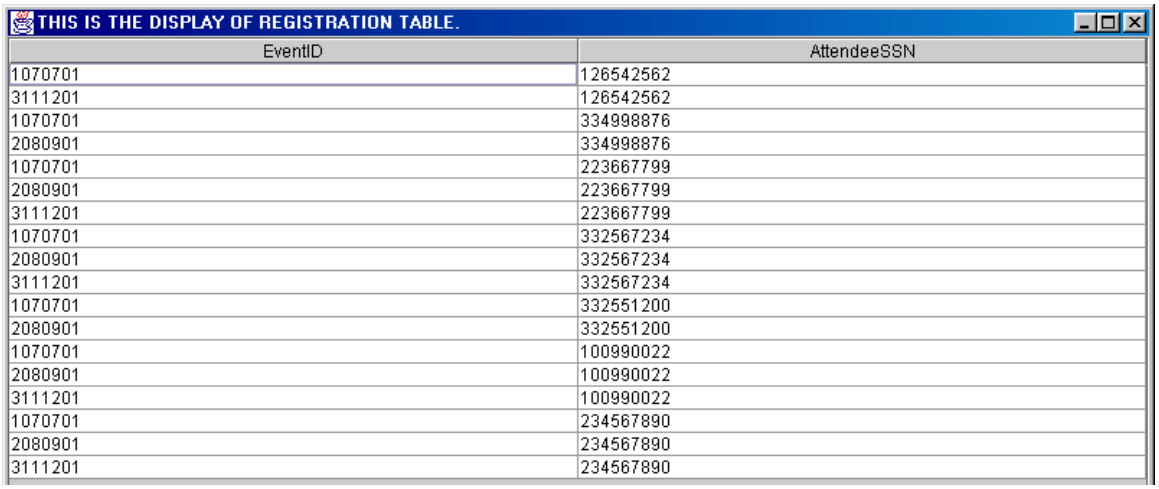
Table Accounts_Receivable after Insertion of Cindy Poppins



THIS IS THE DISPLAY OF ACCOUNTS RECEIVABLE TABLE.

Transaction_Num	CreditCard_Num	Social_Security_Num	Transaction_Amount	Transaction_Date
8	1111222233334444	126542562	0	
12	1111222233334444	126542562	200	2001-06-04 00:00:00
13	7777333366660000	334998876	300	2001-06-04 00:00:00
14	7777333366660000	334998876	200	2001-06-04 00:00:00
15	5555666699990000	223667799	300	2001-06-05 00:00:00
16	6655443388779900	332567234	300	2001-06-06 00:00:00
18	4443333455566665	332551200	200	2001-06-06 00:00:00
19	1000200030004000	100990022	300	2001-06-06 00:00:00
20	2344443255666655	234567890	300	2001-06-09 00:00:00

Table Registration After Data Entry for Cindy Poppins



THIS IS THE DISPLAY OF REGISTRATION TABLE.

EventID	AttendeeSSN
1070701	126542562
3111201	126542562
1070701	334998876
2080901	334998876
1070701	223667799
2080901	223667799
3111201	223667799
1070701	332567234
2080901	332567234
3111201	332567234
1070701	332551200
2080901	332551200
1070701	100990022
2080901	100990022
3111201	100990022
1070701	234567890
2080901	234567890
3111201	234567890

We can see that Cindy poppins have been added as an Attendee in the attendee table and her credit card has been charged for $300 for 3 events she registered for and the events numbers and her social security numbers are entered in the Registration Table.

We carried out numerous such tests on all classes and their smooth functioning. Unfortunately, the limitation of space, time, and funds does not allow us to present all the results in this report at this time.

More results from other tests are shown on subsequent pages.

Source Code

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class Accounts_Receivable implements BusinessConstants
{
      private String Transaction_Num;
      private String CredtCard_Num;
      private String Social_Security_Num;
      private int Transaction_Amount;
      private String Transaction_Date;

      public Accounts_Receivable()
      {


      }


      public void setTransaction_Num(String m_Transaction_Num)
      {
            this.Transaction_Num = m_Transaction_Num;
      }

      public void setCredtCard_Num(String m_CredtCard_Num)
      {
            this.CredtCard_Num = m_CredtCard_Num;
      }

      public void setSocial_Security_Num(String m_Social_Security_Num)
      {
            this.Social_Security_Num = m_Social_Security_Num;
      }

      public void setTransaction_Date()
      {
            //Find the system date. Convert to string and set
      }

      public void setTransaction_Amount()
      {
            this.Transaction_Amount = SeminarFee;
      }

      public static void main(String[] args)
      {
      }
}
// Daniel Rosenthal 268519 CS56 Advanced Java
// Exercise 14.21 pp732

import javax.swing.*;
import javax.swing.JOptionPane.*;
import java.awt.*;
```

```
import java.awt.event.*;
import java.sql.*;

public class Accts_Pay extends JFrame
{
     protected static String insert_into_str = "";
     protected static String odbc_filename;
     protected static String social_security_num;
     protected static String check_num;
     protected static int amount;
     protected static String quote = "'";
     protected static String comma = ", ";
     private JLabel ssn_lbl;
     private static JTextField ssn_box;
     private JLabel amt_lbl;
     private static String amt_str;
     private static JTextField amt_box;
     private JLabel check_num_lbl;
     private static JTextField check_num_box;
     private JButton accept;
     private JButton reject;
     private JButton cancel;
     private Container container;
     private GridBagLayout gbLayout;
     private GridBagConstraints gbConstraints;

     // Initialization
     public Accts_Pay()
     {
          super("Accounts Payable: Data Entry Screen");

          gbLayout = new GridBagLayout();
          gbConstraints = new GridBagConstraints();

          Container container = getContentPane();
              container.setBackground(new Color(0,200,200).brighter());
          container.setLayout(gbLayout);

          ssn_lbl = new JLabel("SOCIAL SECURITY NUMBER OF LECTURER:");
          ssn_box = new JTextField(20);
              ssn_box.setToolTipText("ENTER LECTURERS SOCIAL SECURITY"
+
                  " NUMBER HERE.");
          amt_lbl = new JLabel("AMOUNT OF CHECK:");
          amt_box = new JTextField(8);
              amt_box.setToolTipText("ENTER THE AMOUNT TO BE PAID
HERE.");
              check_num_lbl = new JLabel("CHECK NUMBER:");
          check_num_box = new JTextField(20);
              check_num_box.setToolTipText("ENTER THE CHECK NUMBER
HERE.");
          accept = new JButton("ACCEPT");
              accept.setToolTipText("CLICK HERE TO ENTER THE DATA IN
THE" +
              " DATABASE.");
          reject = new JButton("CLEAR FIELDS");
              reject.setToolTipText("CLICK HERE TO CLEAR ALL FIELDS.");
```

```
            cancel = new JButton("<< PREVIOUS");
                cancel.setToolTipText("CLICK HERE TO GO BACK TO PREVIOUS
MENU.");
//          Create the text boxes to enter thee various items.

            gbConstraints.fill = GridBagConstraints.HORIZONTAL;
            gbConstraints.weightx = 1;
            gbConstraints.weighty = 1;
            gbConstraints.anchor = GridBagConstraints.WEST;
            gbConstraints.gridheight = 1;

            gbConstraints.gridx = 1;
            gbConstraints.gridy = 10;
            gbConstraints.gridwidth = 40;
            gbLayout.setConstraints(ssn_lbl, gbConstraints);
            container.add(ssn_lbl);

            gbConstraints.gridx = 41;
            gbConstraints.gridy = 10;
            gbConstraints.gridwidth = 10;
            gbLayout.setConstraints(ssn_box, gbConstraints);
            container.add(ssn_box);

            gbConstraints.gridx = 1;
            gbConstraints.gridy = 20;
            gbConstraints.gridwidth = 40;
            gbLayout.setConstraints(amt_lbl, gbConstraints);
            container.add(amt_lbl);

            gbConstraints.gridx = 41;
            gbConstraints.gridy = 20;
            gbConstraints.gridwidth = 10;
            gbLayout.setConstraints(amt_box, gbConstraints);
            container.add(amt_box);

            gbConstraints.gridx = 1;
            gbConstraints.gridy = 30;
            gbConstraints.gridwidth = 40;
            gbLayout.setConstraints(check_num_lbl, gbConstraints);
            container.add(check_num_lbl);

            gbConstraints.gridx = 41;
            gbConstraints.gridy = 30;
            gbConstraints.gridwidth = 20;
            gbLayout.setConstraints(check_num_box, gbConstraints);
            container.add(check_num_box);

//          Now do the buttons.

            gbConstraints.gridx = 20;
            gbConstraints.gridy = 40;
            gbConstraints.gridwidth = 10;
            gbLayout.setConstraints(accept, gbConstraints);
            container.add(accept);

            gbConstraints.gridx = 40;
            gbConstraints.gridy = 40;
```

```
        gbConstraints.gridwidth = 10;
        gbLayout.setConstraints(reject, gbConstraints);
        container.add(reject);

        gbConstraints.gridx = 60;
        gbConstraints.gridy = 40;
        gbConstraints.gridwidth = 10;
        gbLayout.setConstraints(cancel, gbConstraints);
        container.add(cancel);

        ButtonHandler handler = new ButtonHandler();
        accept.addActionListener(handler);
        reject.addActionListener(handler);
        cancel.addActionListener(handler);

        setSize(800,600);
        show();
   }
     // Handle window events
protected void processWindowEvent(WindowEvent e)
{
  if (e.getID() == WindowEvent.WINDOW_CLOSING)
  {
    dispose();                    // Release resources
    System.exit(0);               // Exit the program
  }
  super.processWindowEvent(e);    // Pass on the event
}


   private class ButtonHandler implements ActionListener
   {
       // Process GUI Events
       public void actionPerformed(ActionEvent e)
       {
           String command;

           command = (String) (e.getActionCommand());
           // System.out.println("Command is: " + command);

           if (command.equalsIgnoreCase("ACCEPT"))
           {
               get_text();
               try
               {
                    write_to_database();
               }
               catch(SQLException sqlex)
               {
               }
               catch(ClassNotFoundException ex)
               {
               }
               clear_text();  // Clear for next text entry.
           }

           if (command.equalsIgnoreCase("CLEAR FIELDS"))
```

```
        {
            clear_text();
        }

        if (command.equalsIgnoreCase("<< PREVIOUS"))
        {
            clear_text();
            new DataEntryGUI();
                   dispose();
        }
    }
}


private static void get_text()  // Read text fields
{
     social_security_num = ssn_box.getText();
     amt_str = amt_box.getText();
     amount = Integer.parseInt(amt_str);
     check_num = check_num_box.getText();
}

private static void clear_text()  // Clear text fields
{
     ssn_box.setText("");
     amt_box.setText("");
     check_num_box.setText("");
}

private static void write_to_database() throws SQLException,
ClassNotFoundException
{
     // Create the SQL string
     insert_into_str =
         "Insert into" +
       " Accounts_Payable ( "
       + " CHECK_Number,Check_Amount, Social_Security_Num"
       +") values (" +
          quote + check_num + quote + comma
           + amt_str + comma +
          quote + social_security_num + quote + ");";
     System.out.println(insert_into_str);
     Class.forName(DBConnect.Driver);
     Connection c = DriverManager.getConnection(
     DBConnect.DBURL,DBConnect.User,DBConnect.Password);
     Statement s = c.createStatement();
     // Execute the SQL command.
     try
     {
          s.executeUpdate(insert_into_str);
     }
     catch(SQLException sqlex)
     {
     }
     s.close();
     c.close();
}
```

```
    public static void main(String args[])
    {
        Accts_Pay accts_pay = new Accts_Pay();/*
            accts_pay.addWindowListener(
            new WindowAdapter()
            {
                public void windowClosing(WindowEvent e)
                {
                     e.getWindow().dispose();
                     System.exit(0);
                }
            }
        );*/
    }
}

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.*;
import javax.swing.event.*;
import java.text.*;

public class AddRecord implements ActionListener,
BusinessConstants
{
      //***************************************
      //Fields
      //***************************************
      private final String ActsInsert = "Insert into" +
            " Accounts_Receivable ( "
            + " CreditCard_Num, Social_Security_Num,"
            +" Transaction_Amount, Transaction_Date) values (";
      private JTextArea Output;
      private Connection AddRec_Connection;
      private ScrollingPanel fields;
      private Attendees Attendee_Data;
      private Accounts_Receivable Acts_Rcv_Data;
      private ResultSet  rs;
      private Statement Smt;
      private String QueryString;
      private String QueryString1;
      private String QueryString2;
      private String quote = "'";
      private String comma = ",";
      /**Object array for holding multiple events selected
       */
      private Object [] EventListArray;
      String tempSSN;
      int numEvents;
      /**No argument default constructor
       */
      //***************************************
      //Constructors
      //***************************************
```

```
	public AddRecord()
	{
		this.Attendee_Data = new Attendees();
		String tempSSN = new String("");
	}
	/**Explicit Argument Constructor
	 */
	public AddRecord(Connection C, ScrollingPanel f,JTextArea O)
	{
		this.QueryString = new String("");
		this.Attendee_Data = new Attendees();
		this.Acts_Rcv_Data = new Accounts_Receivable();
		this.AddRec_Connection = C;
		this.fields = f;
		this.Output = O;
		String tempSSN = new String("");
		numEvents=0;
	}
	//**************************************
	//Methods
	//**************************************
	public void actionPerformed(ActionEvent e)
	{
		try
		{
			Smt = this.AddRec_Connection.createStatement();
			if((this.fields.id.getText()).equals(""))
			{
				JOptionPane.showMessageDialog(null,
				"A New Record for Attendee"
				+ " can not be Added with out"+
				" Social Security Number!");
			}
			else
			{
				tempSSN = (this.fields.id.getText()).trim();
System.out.println(tempSSN);
				/**check to see if this social security
				 * number is already in the Attendee
				 * table or not.
				 */
				QueryString = "Select * From Attendees Where "
+
				" AttendeeSSN = " + "'" + tempSSN + "';";
System.out.println(QueryString);
				rs = Smt.executeQuery(QueryString);
				boolean moreRecords = rs.next();//Will return
true if record set

	//has data in it.
System.out.println("The moreRecords boolean = " + moreRecords);

				if(!moreRecords)
				{
					/**
					If no records are returned then that
means that
```

```
                              person with the social security number
tempSSN
                              is not in the database, so it is OK to
proceed
                              and add all the information.
                              */
                              processNewAttendee();
                        }
                        else
                        {
                              processExistingAttendee();
                        }
                  }//end of outer else
                  this.Smt.close();
            }//end of try block
            catch(SQLException sqlex)
            {
                  sqlex.printStackTrace();
                  this.Output.append(sqlex.toString());
            }//end of catch block
            //Close the database connection now
            /*
            try
            {
                  this.AddRec_Connection.close();
            }
            catch(SQLException sqlex2)
            {
                  System.err.println("Unable to Close Connection.");
                  sqlex2.printStackTrace();
            }
            */
      }//end of public void actionPerformed
//*******************************************************
      private void processNewAttendee() throws SQLException

      {
            fillAttendee();
            String InputDataConfirm = Attendee_Data.getFinal_String();
            InputDataConfirm+="\nIf all of this information is correct
" +
                  "then press enter or OK and add Attendee Record.";
            int userResponse =0;
            Object[] options = { "OK", "CANCEL" };
            userResponse = JOptionPane.showOptionDialog(null,
            InputDataConfirm, "Warning", JOptionPane.DEFAULT_OPTION,
            JOptionPane.WARNING_MESSAGE,null, options, options[0]);
            if(userResponse==1)
                  return;
            //Insert into database
            QueryString = Attendee_Data.constructSqlObject();
System.out.println(QueryString);

            this.Output.append("\nSending the data to" +
            " database: " +
            this.AddRec_Connection.nativeSQL(
            this.QueryString) +     "\n");
```

```
            int result = this.Smt.executeUpdate(
            this.QueryString);
            if(result == 1)
                  this.Output.append("\nInsertion Successful\n");
            else
            {
                  this.Output.append(
                  "\nInsertion Failed. Contact" +
                  " System Administrator.\n");
            }
            /*
            this.Acts_Rcv_Data.setSocial_Security_Num(
            this.fields.id.getText());
            this.Acts_Rcv_Data.setCredtCard_Num(
            this.fields.creditcard.getText());*/
//System.out.println(this.fields.creditcard.getText());

            fillRegistration();
            fillAccountsReceivable(
            (this.fields.creditcard.getText()).trim());
      }
      //************************************************************
      private void processExistingAttendee() throws SQLException
      {
            JOptionPane.showMessageDialog(null,
            "Attendee is already in "+
            "our database. Please choose EVENTID's and"+
            " press ADD !");
            fillRegistration();
            String tempCreditCard = new String("");
            ResultSet RS =  Smt.executeQuery(
            "Select distinct CreditCard_Num from" +
            " Accounts_Receivable where  Social_Security_Num = "
            + "'" + tempSSN + "';");
                        while(RS.next())
                        {
                              tempCreditCard =
                              RS.getString("CreditCard_Num");
                        }//end of while loop
                        fillAccountsReceivable(tempCreditCard);

      }
      //************************************************************
      private void fillRegistration() throws SQLException
      {

            /**Fill out the registration object
            * 1. find out number of events IDs selected
            * 2. create that many Registration object,
            * with each object having the same Attendee
            * Social Security number but different
            * Event ID.
            */
            this.EventListArray = this.fields.getEvents();
            //That many Registration Objects needed
            numEvents = this.EventListArray.length;
            Registration [] Reg_Data = new Registration [numEvents];
```

```
            for(int i= 0; i<numEvents; i++)
            {
                  Reg_Data[i] = new Registration();
                  Reg_Data[i].setAttendeeSSN(tempSSN);
                  Reg_Data[i].setEvent_ID(
                  (String)(this.EventListArray[i]));
                  this.QueryString2 =
                  Reg_Data[i].constructSqlObject();
System.out.println(QueryString2);
                  int result = this.Smt.executeUpdate(
                  this.QueryString2);
                  if(result == 1)
                        this.Output.append(
                        "\nInsertion Successful.\n");
                  else
                        break;
            }//for loop

      }
      //*****************************************************************
*******
      private void fillAccountsReceivable(String tempCreditCard)
      throws SQLException
      {

            /**Get the credit card number from
            * Accounts_Receivable table and charge for
            * SeminarFee*numEvents.*/
            java.util.Date Today = new java.util.Date();
            DateFormat DF = DateFormat.getDateInstance(3);
            String DateStr = DF.format(Today);
            Integer Amount = new Integer(
            numEvents*SeminarFee);
            String AmountStr = Amount.toString();
            QueryString2 = ActsInsert
            + quote + tempCreditCard + quote + comma +
            quote + tempSSN + quote + comma + quote +
            AmountStr + quote + comma +
            "DateValue(" + "'"+ DateStr + "'" + "));";
System.out.println(QueryString2);
            int result = Smt.executeUpdate(QueryString2);
            if(result == 1)
                  System.out.println("Inserstion Successful");
      }

      //*****************************************************************
********
      private void fillAttendee()
      {
            Attendee_Data.setSocialNum((fields.id.getText()).trim());
System.out.println(this.fields.id.getText());
            this.Attendee_Data.setFirstName(
            (this.fields.first.getText()).trim());
System.out.println(this.fields.first.getText());

            this.Attendee_Data.setLastName(
            (this.fields.last.getText()).trim());
```

```
System.out.println(this.fields.last.getText());
            this.Attendee_Data.setAddress(
            (this.fields.address.getText()).trim());
System.out.println(this.fields.address.getText());

            this.Attendee_Data.setCity(
            (this.fields.city.getText()).trim());
System.out.println(this.fields.city.getText());
            this.Attendee_Data.setZipCode(
            (this.fields.zip.getText()).trim());
System.out.println(this.fields.zip.getText());
            this.Attendee_Data.setEMail(
            (this.fields.email.getText()).trim());
System.out.println(this.fields.email.getText());

            this.Attendee_Data.setHomePhone(
            (this.fields.home.getText()).trim());
System.out.println(this.fields.home.getText());
            this.Attendee_Data.setWorkPhone(
            (this.fields.work.getText()).trim());
System.out.println(this.fields.work.getText());
            this.Attendee_Data.setState(this.fields.getState());
System.out.println(this.fields.getState());
System.out.println("Finished filling Attendee Object");
      }

//*********************************************************************
*
      public static void main(String[] args)
      {
            AddRecord temp = new AddRecord();
      }
}//end of class UpdateRecord
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class AttendeeDataFrm extends JFrame
{
      private int width = 700;
      private int height = 400;
      private MyColorChooser  sliderPanel;
      private DataEntryForm1  drawingPanel;
      private JLabel          headline;
      private Box bottomBox;
      private JPanel buttons;
      private CliffLogo logo;
      private JButton button1;
      private JButton button2;
      private JButton button3;
      private Box auxBox1;
      private Box auxBox2;
      private Box buttonBox;

      public AttendeeDataFrm()
      {
```

```
        super("Attendee Data Entry Form");

        sliderPanel =  new MyColorChooser();
        drawingPanel=  new DataEntryForm1();

        headline    =  new JLabel("Attendee Registration FORM");
        headline.setHorizontalTextPosition(SwingConstants.CENTER);
        headline.setBackground(Color.pink);

        bottomBox   = Box.createHorizontalBox();
        buttons     = new JPanel();


        logo        = new CliffLogo();

        button1     = new JButton("****Submit****");
        button2     = new JButton("Cancel");
        button3     = new JButton("Reset");

        auxBox1     = Box.createHorizontalBox();
        auxBox2     = Box.createHorizontalBox();
        buttonBox   = Box.createVerticalBox();

        auxBox1.add(button1);
        auxBox2.add(button2); auxBox2.add(button3);
        buttonBox.add(auxBox1); buttonBox.add(auxBox2);

        sliderPanel.addMouseMotionListener(
          new MouseMotionAdapter() {
            public void mouseMoved( MouseEvent e)
              {
                Color x = sliderPanel.getTheColor();
                drawingPanel.draw(x);
              }
            }
          );

        bottomBox.add(logo);
        bottomBox.add(sliderPanel);
        bottomBox.add(buttonBox);

        Container c = getContentPane();
        c.setLayout( new BorderLayout() );
        c.add(headline, BorderLayout.NORTH);
        c.add(drawingPanel, BorderLayout.CENTER);
        c.add(bottomBox, BorderLayout.SOUTH);
        setSize(width, height);
        show();

}

public static void main(String[] args)
{
        AttendeeDataFrm temp = new AttendeeDataFrm();

        temp.addWindowListener( new WindowAdapter(){
        public void windowClosing(WindowEvent e)
```

```
            {
                  System.exit(0);
            }
            }
            );

      }
}

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import DBConnect;

public class AttendeeGUI extends JFrame implements ManageShutDown
{
      private ScrollingPanel ScrollArea;
      private JTextArea Output;
      private JScrollPane TextPane;
      private Connection Connect;
      private ControlPanel Controls;
      public AttendeeGUI()
      {
            super("Cliffhouse Attendee Data Form");
            Container C = getContentPane();
            C.setBackground(new Color(200,0,200));
            this.ScrollArea = new ScrollingPanel();
            this.Output = new JTextArea(2,30);
            C.setLayout(new BorderLayout());
            C.add(new JScrollPane(this.ScrollArea),
BorderLayout.CENTER);
            this.TextPane = new JScrollPane(this.Output);
            C.add(this.TextPane, BorderLayout.SOUTH);
            try
            {
                  Class.forName (DBConnect.Driver);
                  this.Connect = DriverManager.getConnection (
                  DBConnect.DBURL,DBConnect.User,DBConnect.Password);
                  this.Output.append("Connection Successful");
            }
            catch ( ClassNotFoundException cnfex )
            {
                  // process ClassNotFoundExceptions here
                  cnfex.printStackTrace();
                  Output.append( "Connection unsuccessful\n" +
                                 cnfex.toString() );
            }
            catch ( SQLException sqlex )
            {
               // process SQLExceptions here
               sqlex.printStackTrace();
               Output.append( "Connection unsuccessful\n" +
                              sqlex.toString() );
            }
            catch ( Exception ex )
            {
```

```
               // process remaining Exceptions here
               ex.printStackTrace();
               Output.append( ex.toString() );
            }
            this.Controls = new ControlPanel(this.Connect,
this.ScrollArea,

this.Output);
            C.add(this.Controls, BorderLayout.NORTH);
            setSize(800,600);
            show();
      }

      public void shutDown()
      {
            try
            {
System.out.println("from Method shut down of AttendeeGUI");
                  this.Connect.close();
            }
            catch(SQLException sqlex)
            {
                  System.err.println("Unable to disconnect.");
                  sqlex.printStackTrace();
            }
      }
      // Handle window events
  protected void processWindowEvent(WindowEvent e)
  {
    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
            shutDown();
            new DataEntryGUI();
      dispose();                    // Release resources
      //System.exit(0);               // Exit the program
    }
    super.processWindowEvent(e);    // Pass on the event
  }
      public static void main(String[] args)
      {
            AttendeeGUI temp = new AttendeeGUI();

      }
}
/**This class acts as a template to form an Attendees object
 * that will interact with the Attendees Table in Cliffhouse database
 * in order to (variuos methods)
 * 1. Insert a row in the table
 * 2. Delete a row in the table
 * 3. Modify a row in the table
 * 4. select rows from tabel where all Attendees are from a given
zipcode
 * 5. Select rows from table where all Attendees are from a given state
 * 6. Select Attendees with certain last name
 * 7. select Attendees with certain first name
 * 8. Search an attendee given the social security number
 * 9.
```

```
 *
 */
import java.sql.*;
import java.io.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Attendees extends Object
{
      /**Fields
       */
      /**The field numColumns is equal to the number of columns in the
       * database table
       */
      public static final String Insert = "INSERT INTO ";
      public static final String Values = " VALUES(";
      public static final String quote = "'";
      private static final int numColumns = 10;
      private static final String TableName = "Attendees";
      Object [] data;
      private static final int SSNLength = 9;
      /**SocialNum is the Social Security Number of the Attendee which
will
       * be restricted to a length of 9 character.
       */
      private String SocialNum;
      /**LastName is the Attendees LastName, Maximum Length is 20
characters.
       */
      private String LastName;
      /**FirstName is the Attendees FirstName, Maximum Length is 20
characters.
       */
      private String FirstName;
      /**Initials is the 1 character Initial of Attendee.
       */

      private String Address;
      /**City is a String limited to maximum 30 characters
       */
      private String City;
      /**State is a String but having fixed 2 characters
       */
      private String State;
      /**ZipCode is a String fixed to 5 characters length
       */
      private String ZipCode;
      /**HomePhone is a String fixed to 10 characters.
       */
      private String HomePhone;
      /**WorkPhone is a String fixed to 10 characters
       */
      private String WorkPhone;
      /**E Mail is a String maximum limited to 40 characters
       */
      private String EMail;
```

```
      private static Connection connection;
      private String comma = ",";
      private String Final_String = new String("You are about to enter
" +
      "the following information in the database. Please verify that it
" +
      " is correct and then do add or update.\n");
      //*************************************************************
      //**Constructors
       //*************************************************************
      /**Default Constructor takes no arguments and initializes the
field
       * values to java defaults
       */
      public Attendees()
      {
            this.data = new Object [this.numColumns + 1];
      }
      /**Explicit Constructor sets the Attendees object to some pre-
defined
       * values. These may be values either to be entered to the
database
       * or values retrieved from the database.
       */
      public Attendees(String M_SocialNum, String M_LastName, String
M_FirstName
      ,String M_Address, String M_City, String M_State,
       String M_ZipCode,String M_HomePhone, String M_WorkPhone,
       String M_EMail)
      {
            this.SocialNum = M_SocialNum;
            this.LastName = M_LastName;
            this.FirstName = M_FirstName;
            this.Address = M_Address;
            this.City = M_City;
            this.State = M_State;
            this.ZipCode = M_ZipCode;
            this.HomePhone = M_HomePhone;
            this.WorkPhone = M_WorkPhone;
            this.EMail = M_EMail;
            this.data = new Object [this.numColumns + 1];
      }
      //*************************************************************
      /**Instance Methods: These methods will need an object to be
called
       * upon
       */
      /**Set Methods: These methods will take the data from GUI and set
       * the variuos fields of the Attendees object to the values
       * entered by the database user.
       */
      /**Methot setSocialNum() must take the user input from a GUI
       * JTextField and check:
       * 1. String has exact 9 characters,
       * 2. String has only numeric characters.
       * 3. Issue a warning if above is not correct.
```

```
       * For the number 1, get the length of the String. If less than 9
or
       * more than 9, then issue a warning. (GUI can control maximum
number
       * of characters but not minimum)
       * For #2 we try and convert the string to a long and if number
format
       * exception occurs then we show a JOptionPane indicating that
non
       * numeric characters have been entered.
       */
      public void setSocialNum(String SSN) throws NumberFormatException
      {
            long tempNum = 0;
            if(SSN.length() < this.SSNLength)
            {
                  JOptionPane.showMessageDialog(null,"Not Enough
Characters " +
                  "Have Been Entered for Social Security Number!");
            }

            else
            {
                  try
                  {
                        tempNum = Long.parseLong(SSN);
                  }
                  catch(NumberFormatException e)
                  {
                        JOptionPane.showMessageDialog(null,"Non Numeric
Characters " +
                        "Have Been Entered for Social Security
Number!");
                  }
            }
            {
                  this.SocialNum = SSN;
                  this.Final_String+= "\nSocial Security Number = " +
SSN;
            }
      }

      public void setFirstName(String Fname)
      {
            this.checkApostrophe(Fname);
            if(Fname.length()==0)
            {
                  JOptionPane.showMessageDialog(null,"You have not
Entered " +
                  "a First Name");
            }
            else
            {
                  this.FirstName = Fname;
                  this.Final_String+= "\nFirst Name = " + Fname;
            }
      }
```

```
	public void setLastName(String Lname)
	{
		this.checkApostrophe(Lname);
		if(Lname.length()==0)
		{
			JOptionPane.showMessageDialog(null,"You have not
Entered " +
			"a Last Name");
		}
		else
		{
			this.LastName = Lname;
			this.Final_String+= "\nLast Name = " + Lname;
		}
	}


	public void setAddress(String Addrs)
	{
		this.checkApostrophe(Addrs);
		if(Addrs.length()==0)
		{
			JOptionPane.showMessageDialog(null,"You have not
Entered " +
			"an Address");
		}
		else
		{
			this.Address = Addrs;
			this.Final_String+= "\nAddress: " + Addrs;
		}
	}

	public void setCity(String m_City)
	{
		this.checkApostrophe(m_City);
		if(m_City.length()==0)
		{
			JOptionPane.showMessageDialog(null,"You have not
Entered " +
			"a City");
		}
		else
		{
			this.City = m_City;
			this.Final_String+= "\nCity: " + m_City;
		}
	}
	/**This Method accepts the state input and makes sure that no
numerics
	 * are entered.
	 */
	public void setState(String m_State)
	{
		if(m_State.length()<2)
		{
```

```
                  JOptionPane.showMessageDialog(null,"Not enough state
information"
                  + " entered");
            }

            else
            {
                  char [] temp = m_State.toCharArray();

                  if(Character.isDigit(temp[0]) ||
Character.isDigit(temp[1]))
               {
                        JOptionPane.showMessageDialog(null,"Invalid
State Entered");
               }
                  else
                  {
                        this.State = m_State;
                        this.Final_String+= "\nState : " + m_State;
                  }
            }
      }

      /**Method setZipCode must do the followings:
       * 1. Check whether only numerical data is entered
       * 2. No less than 5 characters are entered.
       */
      public void setZipCode(String m_Zip)
      {
            if(m_Zip.length()<5)
            {
                  JOptionPane.showMessageDialog(null,"Not enough Zip
Code information"
                  + " entered");
            }
            else
            {
                  char [] temp = m_Zip.toCharArray();
                  boolean myBoolean = (!Character.isDigit(temp[0])||

      (!Character.isDigit(temp[1]))||(!Character.isDigit(temp[2]))
                  ||
(!Character.isDigit(temp[3]))||(!Character.isDigit(temp[4])));
                  if(myBoolean)
                  {
                        JOptionPane.showMessageDialog(null,"Invalid Zip
Code information"
                        + " entered");
                  }
                  else
                  {
                        this.ZipCode = m_Zip;
                        this.Final_String+= "\nZip Code; " + m_Zip;
                  }
            }
      }
```

```
	/**Method setHomePhone will check whether:
	 * 1. 10 digits have been entered for the phone
	 * 2. Whether all characters are numerics (no alphabets)
	 * 3. area code authentication?
	 */
	public void setHomePhone(String hPhone)
	{
		if(hPhone.length()<10)
		{
			JOptionPane.showMessageDialog(null,"Not enough Home
Phone information"
			+ " entered");
		}
		else
		{
			char [] temp = hPhone.toCharArray();
			boolean myBoolean = (!Character.isDigit(temp[0])||

	(!Character.isDigit(temp[1]))||(!Character.isDigit(temp[2]))
			||
(!Character.isDigit(temp[3]))||(!Character.isDigit(temp[4]))

	||(!Character.isDigit(temp[5]))||(!Character.isDigit(temp[6]))

	||(!Character.isDigit(temp[7]))||(!Character.isDigit(temp[8]))
			||(!Character.isDigit(temp[9])));
			if(myBoolean)
			{
				JOptionPane.showMessageDialog(null,"Invalid
Phone Number"
				+ " entered");
			}
			else
			{
				this.HomePhone = hPhone;
				this.Final_String+="\nHome Phone: " + hPhone;
			}
		}
	}
	/**Method setWorkPhone is exactly similar to the method
setHomePhone
	 */
	public void setWorkPhone(String wPhone)
	{
		if(wPhone.length()<10)
		{
			JOptionPane.showMessageDialog(null,"Not enough Work
Phone information"
			+ " entered");
		}
		else
		{
			char [] temp = wPhone.toCharArray();
			boolean myBoolean = (!Character.isDigit(temp[0])||

	(!Character.isDigit(temp[1]))||(!Character.isDigit(temp[2]))
```

```
                  ||
(!Character.isDigit(temp[3]))||(!Character.isDigit(temp[4]))

      ||(!Character.isDigit(temp[5]))||(!Character.isDigit(temp[6]))

      ||(!Character.isDigit(temp[7]))||(!Character.isDigit(temp[8]))
                  ||(!Character.isDigit(temp[9])));
                  if(myBoolean)
                  {
                        JOptionPane.showMessageDialog(null,"Invalid
Phone Number"
                        + " entered");
                  }
                  else
                  {
                        this.WorkPhone = wPhone;
                        this.Final_String+="\nWork Phone: " + wPhone;
                  }
            }
      }
      /**Method setEMail will set the e mail address in the object. It
will
       * just check for the @ character in the string entered.
       * Short algorithm for this method is as follows:
       * 1. Get the string length.
       * 2. Run a loop that looks at each character
       * 3. if character is @ then break and set a boolean flag to true
       * 4. After the loop is over, if Boolean flag is true then store
the
       * string into field EMail
       * else display the message that no valid e mail address has been
entered.
       */
      public void setEMail(String m_EMail)
      {
            boolean isValid1 = false;
            boolean isValid2 = false;
            //Look for @ character in the e mail address
            for(int i=0; i<m_EMail.length(); i++)
            {
                  if(m_EMail.charAt(i)=='@')
                  {
                        isValid1 = true;
                        break;
                  }
            }
            //Look for dot in the e mail address
            for(int i=0; i<m_EMail.length(); i++)
            {
                  if(m_EMail.charAt(i)=='.')
                  {
                        isValid2 = true;
                        break;
                  }
            }
            if(isValid1&&isValid2)
            {
```

```
                this.EMail = m_EMail;
                this.Final_String+="\nE Mail ; " + m_EMail;
            }
            else
            {
                JOptionPane.showMessageDialog(null,"Invalid E Mail
Address"
                    + " entered");
            }
      }/**Method constructAttendee is required because we are doing
      data validation and we can not just make a simple constructor
      call for that.
        */

      public String constructSqlObject()
      {
            this.data[0] = this.TableName;
            this.data[1] = this.getSocialNum();
            this.data[2] = this.getLName();
            this.data[3] = this.getFName();
            this.data[4] = this.getAddress();
            this.data[5] = this.getCity();
            this.data[6] = this.getState();
            this.data[7] = this.getZip();
            this.data[8] = this.getHomePhone();
            this.data[9] = this.getWorkPhone();
            this.data[10] = this.getEmail();
            String sql = this.Insert + this.data[0] + this.Values;
            for(int i=1; i<data.length; i++)
            {
                if(data[i] instanceof String)
                    sql += this.quote + data[i] + this.quote;
                else if(data[i] instanceof Character)
                {
                    String temp =
((Character)(data[i])).toString();
                    sql +=this.quote + temp + this.quote;
                }
                else
                    sql += data[i];
                /**put commas every where except after last value
                 */
                if(i<this.data.length-1)
                    sql += ", ";
            }
            sql += " ); ";
System.out.println(sql);
            return sql;
      }

      public String getFinal_String()
      {
            return this.Final_String;
      }

      public String constructSqlUpdate()
      {
```

```
            this.Final_String+="\nIf all of this information is correct
" +
            "then press enter or OK and update Attendee Record.";
            this.data[0] = this.TableName;
            this.data[1] = this.getSocialNum();
            this.data[2] = this.getLName();
            this.data[3] = this.getFName();
            this.data[4] = this.getAddress();
            this.data[5] = this.getCity();
            this.data[6] = this.getState();
            this.data[7] = this.getZip();
            this.data[8] = this.getHomePhone();
            this.data[9] = this.getWorkPhone();
            this.data[10] = this.getEmail();
            String sql = "Update " + this.data[0] + " SET ";
            sql+= " LastName = " + this.quote+this.getLName()+
            quote + comma + " FirstName = " + quote +
            getFName()+quote+ comma + " StreetAddress = "
            + quote + getAddress()+quote + comma + " City = " +
            quote + getCity() + quote + comma + " ZipCode = " +
            quote + getZip() + quote + comma  + " State = "
            + quote + getState()+ quote + comma + " HomePhone = "
            + quote + getHomePhone() + quote + comma +
            " WorkPhone = " + quote + getWorkPhone() + quote +
            comma + " E_MailAddress = " + quote + getEmail() +
            quote + " Where AttendeeSSN = " + quote + getSocialNum()
            +quote;
            sql += ";";
      System.out.println(sql);
            return sql;
      }

/**Now all the get functions
 */
      public String getSocialNum()
      {
            return this.SocialNum;
      }
      public String getFName()
      {
            return this.FirstName;
      }
      public String getLName()
      {
            return this.LastName;
      }

      public String getAddress()
      {
            return this.Address;
      }
      public String getCity()
      {
            return this.City;
      }
      public String getZip()
      {
```

```
            return this.ZipCode;
      }
      public String getState()
      {
            return this.State;
      }
      public String getHomePhone()
      {
            return this.HomePhone;
      }

      public String getWorkPhone()
      {
            return this.WorkPhone;
      }
      public String getEmail()
      {
            return this.EMail;
      }

      public  void checkApostrophe(String input)
      {
            String Apos = new String("'");
            char [] ChArray;
            ChArray = input.toCharArray();
            for(int i=0; i<ChArray.length; i++)
            {
                  Character Input = new Character(ChArray[i]);
                  String StrInput = Input.toString();
                  if(StrInput.equals(Apos))
                  {
                        JOptionPane.showMessageDialog(null,"At this
time "
                        + " please enter your data with out Apostrophe.
"
                        + " We will fix this problem soon. ");

                        break;
                  }
            }
      }

      public static void main(String[] args)
      {
            Attendees temp = new Attendees();
      }
      public static Attendees constructAttendee()
      {
            /**Later on each hard coded data will be replaced by a
             * value obtained from a JTextField from the GUI screen.
             * This can also be a staic method.
             */
            Attendees temp = new Attendees();
            temp.setSocialNum("126542562");
            temp.setLastName("Singhal");
            temp.setFirstName("Satish");
            temp.setAddress("16625 Crenshaw Blvd 32");
```

```
/**/  temp.setCity("Torrance");
/**/  temp.setState("CA");
/**/  temp.setZipCode("90504");
/**/  temp.setHomePhone("3105326620");
/**/  temp.setWorkPhone("3105326620");
/**/  temp.setEMail("ssatish@socal.rr.com");
            return temp;
      }
}
import javax.swing.*;
public interface BusinessConstants extends SwingConstants
{
      /**Number of States from where the Attendees and Lecturer are
       * accepted. = 53.
       */
      public final int NumStates = 53;
      /**Number of conference or seminar rooms
       */
      public final int NumRooms = 4;
      /**The seminar fee per attendee.
       */
      public final int SeminarFee = 100;
      /**the factor by which lecturer will get paid.
       * Number Of attendees*99.99*0.5 will be the payment to
       * lecturer
       */
      public final double LecturerPortion = 0.5;
      /**The credit cards accepted from lecturer and attendees
       */

}
import java.awt.*;
import java.awt.event.*;

public class ClearFields implements ActionListener
{
   private ScrollingPanel fields;


   public ClearFields()
   {

   }
   public ClearFields( ScrollingPanel f )
   {
      fields = f;
   }

   public void actionPerformed( ActionEvent e )
   {
      fields.id.setText( "" );
      fields.first.setText( "" );
      fields.last.setText( "" );
      fields.address.setText( "" );
      fields.city.setText( "" );
      fields.zip.setText( "" );
      fields.email.setText( "" );
```

```
      fields.home.setText( "" );
      fields.work.setText( "" );
        fields.creditcard.setText("");
   }

   public static void main(String[] args)
      {
         ClearFields temp = new ClearFields();
      }
}
import java.awt.*;
import javax.swing.*;


public class CliffLogo extends JPanel
{

   private ImageIcon cliffhouseLogo;
   private Image i;

   public CliffLogo()
      {
      super();

        cliffhouseLogo = new ImageIcon("logo.jpg");

        i = cliffhouseLogo.getImage();



        }


   public void paint(Graphics g)
   {
       g.drawImage(i, 0, 0, getWidth(), getHeight(), this);


//     cliffhouseLogo.paintIcon( this, g, 0, 0 );
   }

}

// Class ControlPanel definition
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.*;

public class ControlPanel extends JPanel
{
   private JButton findName, addName,
                   updateName, clear, help, previous;
   public ControlPanel()
   {
```

```
   }

   public ControlPanel( Connection c, ScrollingPanel s,
                        JTextArea t )
   {
      setLayout( new GridLayout( 1, 6 ) );
      Icon find = new ImageIcon("search_icon2.gif");
      findName = new JButton( "FIND",find);
        this.findName.setToolTipText("Click To Find Certain
Attendee.");
      findName.addActionListener( new FindRecord( c, s, t ) );
      add( findName );
        Icon Add = new ImageIcon("newbluetumble.gif");
      addName = new JButton( "ADD",Add);
        this.addName.setToolTipText("Click To Add Attendee.");
      addName.addActionListener( new AddRecord( c, s, t ) );
      add( addName );
        Icon UpArrow = new ImageIcon("arrowupsr.gif");
      updateName = new JButton( "Update", UpArrow);
        this.updateName.setToolTipText("Click To Update Attendee.");
      updateName.addActionListener(
       new UpdateRecord( c, s, t ) );
      add( updateName );
      clear = new JButton( "CLEAR  FORM" );
        this.clear.setToolTipText("Click To Clear All Text Fields in
The Form.");
      clear.addActionListener( new ClearFields( s ) );
      add( clear );
      help = new JButton( "HELP (??)" );
        this.help.setToolTipText("Click To See Help Menu.");
      help.addActionListener( new Help( t ) );
      add( help ); /*
        this.previous = new JButton("<< PREVIOUS");
        this.previous.setToolTipText("CLICK HERE TO GO TO " +
        "PREVIOUS SCREEN.");
        this.previous.addActionListener(new Temporary());
        add(previous);*/
   }

  class Temporary  implements ActionListener
  {
        public Temporary()
        {

        }
         public void actionPerformed( ActionEvent e )
            {
               new AttendeeGUI().dispose();
         }
  }


   public static void main (String [] args)
   {
         ControlPanel temp = new ControlPanel();
   }
}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;


public class DataDisplayGUI extends JFrame implements ActionListener,
SwingConstants
{
         private JButton b[];

         private String names [] =
         {"ATTENDEE", "LECTURER", "EVENTS",
          "ACCOUNTS PAYABLE", "ACCOUNTS RECEIVABLE",
          "REGISTRATION", "<< PREVIOUS SCREEN", "EXIT"};
         private String tools [] = {"CLICK HERE TO DISPLAY ATTENDEES",
                                    "CLICK HERE TO DISPLAY LECTURERS",
                                    "CLICK HERE TO DISPLAY EVENTS",
                                    "CLICK HERE TO DISPLAY ACCOUNTS
PAYABLE",
                                    "CLICK HERE TO DISPLAY ACCOUNTS
RECEIVABLE",
                                    "CLICK HERE TO DISPLAY
REGISTRATION",
                                    "CLICK HERE TO SHOW PREVIOUS
SCREEN",
                                    "CLICK HERE TO EXIT APPLICATION"};
         private Container c;
         private JLabel label;
             private JLabel ImageLabel;
         public DataDisplayGUI()
         {
             super("CLIFFHOUSE DATA DISPLAY MENU");
             this.c = getContentPane();
             c.setBackground(new Color(0,200,200).brighter());
             c.setLayout(new GridLayout(11,1,0,10));
                   Icon CliffLogo = new ImageIcon("logo.jpg");
                   ImageLabel = new JLabel(CliffLogo,
SwingConstants.CENTER);
                   c.add(ImageLabel);
             this.label = new JLabel("DATA DISPLAY MENU",
                 SwingConstants.CENTER);
                   c.add(label);
                   /*
             this.label = new JLabel("DATA DISPLAY MENU", CENTER);
             c.add(label);
                   */
             this.b = new JButton [names.length];
             for (int i = 0; i < names.length; i++ )
             {
                 b[ i ] = new JButton( names[ i ] );
                 b[i].setToolTipText(tools[i]);
                 b[ i ].addActionListener( this );
                 c.add( b[ i ] );
             }
             //  Display
```

```
            setSize( 500, 600 );
            show();
        }

        // Process GUI Events
        public void actionPerformed(ActionEvent e)
        {
            String command;

            command = (String) (e.getActionCommand());

            if (command.equalsIgnoreCase("ATTENDEE"))
            {
               new DisplayAttendee();
            }

            if (command.equalsIgnoreCase("LECTURER"))
            {
               new DisplayLecturer();
            }

            if (command.equalsIgnoreCase("EVENTS"))
            {
               new DisplayEvents();
            }

            if (command.equalsIgnoreCase("ACCOUNTS RECEIVABLE"))
            {
               new DisplayAccountsReceivable();
            }

            if (command.equalsIgnoreCase("ACCOUNTS PAYABLE"))
            {
               new DisplayAccountsPayable();
            }

            if (command.equalsIgnoreCase("REGISTRATION"))
            {
              new DisplayRegistration();
            }

            if (command.equalsIgnoreCase("<< PREVIOUS SCREEN"))
            {
               new Main_Menu();
                       setVisible(false);
            }

            if (command.equalsIgnoreCase("EXIT"))
            {
               System.exit(0);
            }
        }
protected void processWindowEvent(WindowEvent e)
{
  if (e.getID() == WindowEvent.WINDOW_CLOSING)
  {
    dispose();                    // Release resources
```

```
      System.exit(0);               // Exit the program
    }
    super.processWindowEvent(e);    // Pass on the event
  }

    public static void main(String args[])
    {
        DataDisplayGUI datadisplayGUI = new DataDisplayGUI();

        datadisplayGUI.addWindowListener(
            new WindowAdapter()
            {
                public void windowClosing(WindowEvent e)
                {
                     System.exit(0);
                }
            }
        );
    }
}
//SMC CS56
//Class project
//Team: Satish, Dan, Rosemary, Walter
/* This is a data entry form set up in a panel that can be included in
 an applet or application */



import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;


public class DataEntryForm1 extends JPanel
{

// ================ declarations ======================

private JPanel form1;
private JLabel [] directions;
private JLabel [] spacers;
private JTextField [] entries;
private Attendees New_Attendee;
//===========make sure %2=0

private String [] legends = {
" First Name", " E-mail", " Middle Initial",
" Expertise", " Last Name", " Conference date",
" Address", " Starting time", " City",
" Finishing time",
" State",  " Room name", " Zip Code",
" Conference description", " Home telephone",
" Credit card number", " Work telephone",
" Expiration date", " Cellular telephone",
" Social Security Nr." };
```

```
// ============== constructor =========================

public DataEntryForm1()
      {
      super();
      New_Attendee = new Attendees();
      form1      = new JPanel();
      directions = new JLabel[legends.length];
      spacers    = new JLabel[legends.length];
      entries    = new JTextField[legends.length];

      form1.setLayout( new GridLayout(legends.length/2, 5) );
      // make sure %2=0

      for (int i=0; i<legends.length; i++)
      {
            directions[i] = new JLabel(legends[i]);
            form1.add(directions[i]);

            entries[i] = new JTextField(5);
            form1.add(entries[i]);

            if (i%2 == 0)
                  {
                  spacers[i]    = new JLabel(" ");
                  form1.add(spacers[i]);
                  }
            }

      add(form1);

      setSize(700,400);
      show();


      } // end constructor




// ==================== cutie added things ====================
/*
    public void paintComponent( Graphics g )
        {
          super.paintComponent(g);

          g.fillRect( 15, 15, 300, 30);

        }
*/
    public void draw(Color z)

      {
          setBackground(z);
        form1.setBackground(z);
```

```
/*          for (int j=0; j<legends.length; j++)
            {
            directions[j].setBackground(z);
            spacers[j].setBackground(z);
            }
*/

           repaint();
      }

    public Dimension getPreferreSize()
        {
          return new Dimension(700, 400);
        }

    public Dimension getMinimumSize()
        {
          return getPreferredSize();
        }
      public static void main(String [] args)
      {


      }

} // end class
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class DataEntryGUI extends JFrame
implements ActionListener, BusinessConstants
{
      private JButton b[];
      private String names [] =
      {"ATTENDEE", "LECTURER", "EVENTS", "ACCOUNTS PAYABLE",
       "<< PREVIOUS SCREEN", "EXIT"};
      private String tools [] =
      { "CLICK HERE TO ENTER ATTENDEE DATA", "CLICK HERE TO ENTER "
        + " LECTURER DATA", "CLICK HERE TO ENTER EVENTS DATA",
        "CLICK HERE TO ENTER ACCOUNTS PAYABLE DATA",
        "CLICK HERE TO GO TO PREVIOUS MENU",
        "CLICK HERE TO EXIT THE APPLICATION"};
      private Container c;
      private JLabel label;
      private JLabel ImageLabel;
      public DataEntryGUI()
      {
            super("CLIFFHOUSE DATA ENTRY SCREEN");
            this.c = getContentPane();
            c.setBackground(new Color(0,200,200).brighter());
            c.setLayout(new GridLayout(9,1,0,5));
            Icon CliffLogo = new ImageIcon("logo.jpg");
            ImageLabel = new JLabel(CliffLogo, SwingConstants.CENTER);
            c.add(ImageLabel);
```

```
      this.label = new JLabel("DATA ENTRY MENU",
          SwingConstants.CENTER);
          c.add(label);
          /*
          this.label = new JLabel("DATA ENTRY", CENTER);
          c.add(label);
          */
          this.b = new JButton [names.length];
          b[0] = new JButton("ATTENDEE");
          b[0].setToolTipText("CLICK HERE TO ENTER ATTENDEE DATA");
          b[0].addActionListener(this);
          c.add(b[0]);
          Icon myGlasses = new ImageIcon("glasses.jpg");
          b[1] = new JButton("LECTURER",myGlasses);
          b[1].setToolTipText("CLICK HERE TO ENTER LECTURER DATA");
          b[1].addActionListener(this);
          c.add(b[1]);
          for (int i = 2; i < names.length; i++ )
          {
       b[ i ] = new JButton( names[ i ] );
           b[i].setToolTipText(tools[i]);
       b[ i ].addActionListener( this );
       c.add( b[ i ] );
    }

    setSize( 500, 600 );
    show();

    }
     public void actionPerformed( ActionEvent e )
{
           Object source = e.getActionCommand();
           if(source.equals("EXIT"))
                 System.exit(0);
           if(source.equals("ATTENDEE"))
          {
                 AttendeeGUI temp = new AttendeeGUI();
                 dispose();

           }

           if(source.equals("ACCOUNTS PAYABLE"))
          {
                 Accts_Pay MyAcctsPay = new Accts_Pay();
                dispose();
           }
            if(source.equals("LECTURER"))
          {

            }
           if(source.equals("EVENTS"))
          {

            }
          if(source.equals("<< PREVIOUS SCREEN"))
          {
                new Main_Menu();
```

```
                dispose();
            }


        }
        // Handle window events
  protected void processWindowEvent(WindowEvent e)
  {
    if (e.getID() == WindowEvent.WINDOW_CLOSING)
    {
      dispose();                    // Release resources
      System.exit(0);               // Exit the program
    }
    super.processWindowEvent(e);    // Pass on the event
  }

      public static void main(String[] args)
      {
            DataEntryGUI app = new DataEntryGUI();
             app.addWindowListener(
         new WindowAdapter() {
            public void windowClosing( WindowEvent e )
            {
               System.exit( 0 );
            }
         }
      );
      }
}
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class DataQueryGUI extends JFrame
implements ActionListener, SwingConstants
{
      private JButton b[];
      private String names [] =
        {"PERFORM QUERIES", "<< PREVIOUS SCREEN", "EXIT"};
        private String tools [] = {"CLICK HERE TO QUERY THE DATABASE",
                                   "CLICK HERE TO SHOW PREVIOUS
SCREEN",
                                   "CLICK HERE TO EXIT APPLICATION"};

      private Container c;
      private JLabel label;
      private JLabel ImageLabel;
    public DataQueryGUI()
      {
            super("CLIFFHOUSE DATA QUERY SCREEN");
            this.c = getContentPane();
            c.setBackground(new Color(0,200,200).brighter());
            c.setLayout(new GridLayout(7,1,0,10));
                  Icon CliffLogo = new ImageIcon("logo.jpg");
                  ImageLabel = new JLabel(CliffLogo,
SwingConstants.CENTER);
                  c.add(ImageLabel);
```

```
            this.label = new JLabel("DATA QUERY ",
                SwingConstants.CENTER);
                  c.add(label);

            //this.label = new JLabel("DATA QUERY", CENTER);
            //c.add(label);
            this.b = new JButton [names.length];
            for (int i = 0; i < names.length; i++ )
            {
                b[ i ] = new JButton( names[ i ] );
                b[i].setToolTipText(tools[i]);
                b[ i ].addActionListener( this );
                c.add( b[ i ] );
            }

            setSize( 500, 600 );
            show();
        }

          public void actionPerformed( ActionEvent e )
          {
               String command;

               command = (String) (e.getActionCommand());

               if (command.equalsIgnoreCase("PERFORM QUERIES"))
               {
                  new DisplayQueryResults();
               }

               if (command.equalsIgnoreCase("<< PREVIOUS SCREEN"))
               {
                  new Main_Menu();
                          setVisible(false);
               }

               if (command.equalsIgnoreCase("EXIT"))
               {
                  System.exit(0);
               }

          }
        protected void processWindowEvent(WindowEvent e)
    {
      if (e.getID() == WindowEvent.WINDOW_CLOSING)
      {
        dispose();                    // Release resources
        System.exit(0);               // Exit the program
      }
      super.processWindowEvent(e);    // Pass on the event
    }

        public static void main(String[] args)
        {
              DataQueryGUI app = new DataQueryGUI();
              app.addWindowListener(
              new WindowAdapter()
```

```
            {
                public void windowClosing( WindowEvent e )
                {
                   System.exit( 0 );
                }
             }
          );
      }
}
public class DBConnect
{
      public static String Driver ="sun.jdbc.odbc.JdbcOdbcDriver";
      public static String DBURL = "jdbc:odbc:Cliffhouse";
      public static String User =  "Admin";
      public static String Password = "ilovegod";
}
import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class DisplayAccountsPayable extends JFrame implements
ManageShutDown
{
   // java.sql types needed for database processing
   private Connection connection;
   private Statement statement;
   private ResultSet resultSet;
   private ResultSetMetaData rsMetaData;

   // javax.swing types needed for GUI
   private JTable table;


   public DisplayAccountsPayable()
   {
      super( "THIS IS THE DISPLAY OF ACCOUNTS PAYABLE TABLE." );


      try
        {
         Class.forName (DBConnect.Driver);
             this.connection = DriverManager.getConnection (
                  DBConnect.DBURL,DBConnect.User,DBConnect.Password);
      }
      catch ( ClassNotFoundException cnfex ) {
         System.err.println(
            "Failed to load JDBC/ODBC driver." );
         cnfex.printStackTrace();
         System.exit( 1 );  // terminate program
      }
      catch ( SQLException sqlex ) {
         System.err.println( "Unable to connect" );
         sqlex.printStackTrace();
         System.exit( 1 );  // terminate program
      }
```

```
      JPanel topPanel = new JPanel();
      topPanel.setBackground(new Color(200,0,200));//added
   topPanel.setLayout( new BorderLayout() );


   table = new JTable( 4, 4 );

   Container c = getContentPane();
     c.setBackground(new Color(200,0,200));
   c.setLayout( new BorderLayout() );
   c.add( topPanel, BorderLayout.NORTH );
   c.add( table, BorderLayout.CENTER );

   getTable();

   setSize( 800, 500 );
   show();
}

private void getTable()
{
   try
     {
      String query = "SELECT * FROM ACCOUNTS_PAYABLE;";
      statement = connection.createStatement();
      resultSet = statement.executeQuery( query );
      displayResultSet( resultSet );
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private void displayResultSet( ResultSet rs )
   throws SQLException
{
   // position to first record
   boolean moreRecords = rs.next();

   // If there are no records, display a message
   if ( ! moreRecords )
     {
      JOptionPane.showMessageDialog( this,
         "ResultSet contained no records" );
      setTitle( "No records to display" );
      return;
   }

   Vector columnHeads = new Vector();
   Vector rows = new Vector();

   try
     {
      // get column heads
      ResultSetMetaData rsmd = rs.getMetaData();
```

```
      for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
         columnHeads.addElement( rsmd.getColumnName( i ) );

      // get row data
      do
          {
         rows.addElement( getNextRow( rs, rsmd ) );
      } while ( rs.next() );

      // display table with ResultSet contents
      table = new JTable( rows, columnHeads );
      JScrollPane scroller = new JScrollPane( table );
      Container c = getContentPane();
      c.remove( 1 );
      c.add( scroller, BorderLayout.CENTER );
      c.validate();
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private Vector getNextRow( ResultSet rs,
                           ResultSetMetaData rsmd )
    throws SQLException
{
   Vector currentRow = new Vector();

   for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
      switch( rsmd.getColumnType( i ) ) {
         case Types.VARCHAR:
         case Types.LONGVARCHAR:
               currentRow.addElement( rs.getString( i ) );
            break;
         case Types.DOUBLE:
               currentRow.addElement(
                  new Double( rs.getDouble( i ) ) );
            break;
          case Types.INTEGER:
                  currentRow.addElement(
                     new Long(rs.getLong(i)));
                  break;
          case Types.DATE:
                  currentRow.addElement(rs.getDate(i));
                  break;
         default:
                  String temp = rsmd.getColumnTypeName(i);
                  if(temp.equals("DATETIME"))
                 {
                        currentRow.addElement(
                        rs.getString(i));
                  }
            System.out.println( "Type was: " +
               rsmd.getColumnTypeName( i ) );
      }
```

```
      return currentRow;
   }

   public void shutDown()
   {
      try
        {
         connection.close();
      }
      catch ( SQLException sqlex )
        {
         System.err.println( "Unable to disconnect" );
         sqlex.printStackTrace();
      }
   }

   public static void main( String args[] )
   {
      final DisplayAccountsPayable app =
         new DisplayAccountsPayable();

      app.addWindowListener(
         new WindowAdapter() {
            public void windowClosing( WindowEvent e )
            {
               app.shutDown();
               System.exit( 0 );
            }
         }
      );
   }
}
import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class DisplayAccountsReceivable extends JFrame implements
ManageShutDown
{
   // java.sql types needed for database processing
   private Connection connection;
   private Statement statement;
   private ResultSet resultSet;
   private ResultSetMetaData rsMetaData;

   // javax.swing types needed for GUI
   private JTable table;


   public DisplayAccountsReceivable()
   {
      super( "THIS IS THE DISPLAY OF ACCOUNTS RECEIVABLE TABLE." );
```

```
   try
     {
      Class.forName (DBConnect.Driver);
          this.connection = DriverManager.getConnection (
               DBConnect.DBURL,DBConnect.User,DBConnect.Password);
   }
   catch ( ClassNotFoundException cnfex ) {
      System.err.println(
         "Failed to load JDBC/ODBC driver." );
      cnfex.printStackTrace();
      System.exit( 1 );  // terminate program
   }
   catch ( SQLException sqlex ) {
      System.err.println( "Unable to connect" );
      sqlex.printStackTrace();
      System.exit( 1 );  // terminate program
   }

     JPanel topPanel = new JPanel();
     topPanel.setBackground(new Color(200,0,200));//added
   topPanel.setLayout( new BorderLayout() );


   table = new JTable( 4, 4 );

   Container c = getContentPane();
     c.setBackground(new Color(200,0,200));
   c.setLayout( new BorderLayout() );
   c.add( topPanel, BorderLayout.NORTH );
   c.add( table, BorderLayout.CENTER );

   getTable();

   setSize( 800, 500 );
   show();
}

private void getTable()
{
   try
     {
      String query = "SELECT * FROM ACCOUNTS_RECEIVABLE;";
      statement = connection.createStatement();
      resultSet = statement.executeQuery( query );
      displayResultSet( resultSet );
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private void displayResultSet( ResultSet rs )
   throws SQLException
{
   // position to first record
   boolean moreRecords = rs.next();
```

```
   // If there are no records, display a message
   if ( ! moreRecords )
     {
      JOptionPane.showMessageDialog( this,
         "ResultSet contained no records" );
      setTitle( "No records to display" );
      return;
   }

   Vector columnHeads = new Vector();
   Vector rows = new Vector();

   try
     {
      // get column heads
      ResultSetMetaData rsmd = rs.getMetaData();

      for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
         columnHeads.addElement( rsmd.getColumnName( i ) );

      // get row data
      do
          {
         rows.addElement( getNextRow( rs, rsmd ) );
      } while ( rs.next() );

      // display table with ResultSet contents
      table = new JTable( rows, columnHeads );
      JScrollPane scroller = new JScrollPane( table );
      Container c = getContentPane();
      c.remove( 1 );
      c.add( scroller, BorderLayout.CENTER );
      c.validate();
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private Vector getNextRow( ResultSet rs,
                           ResultSetMetaData rsmd )
    throws SQLException
{
   Vector currentRow = new Vector();

   for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
      switch( rsmd.getColumnType( i ) ) {
         case Types.VARCHAR:
         case Types.LONGVARCHAR:
               currentRow.addElement( rs.getString( i ) );
            break;
         case Types.DOUBLE:
               currentRow.addElement(
                  new Double( rs.getDouble( i ) ) );
            break;
```

```
            case Types.INTEGER:
                    currentRow.addElement(
                       new Long(rs.getLong(i)));
                    break;
            case Types.DATE:
                    currentRow.addElement(rs.getDate(i));
                    break;
           default:
                    String temp = rsmd.getColumnTypeName(i);
                    if(temp.equals("DATETIME"))
                   {
                          currentRow.addElement(
                          rs.getString(i));
                    }
              System.out.println( "Type was: " +
                 rsmd.getColumnTypeName( i ) );
         }

      return currentRow;
   }

   public void shutDown()
   {
      try
        {
         connection.close();
      }
      catch ( SQLException sqlex )
        {
         System.err.println( "Unable to disconnect" );
         sqlex.printStackTrace();
      }
   }

   public static void main( String args[] )
   {
      final DisplayAccountsReceivable app =
         new DisplayAccountsReceivable();

      app.addWindowListener(
         new WindowAdapter() {
            public void windowClosing( WindowEvent e )
            {
               app.shutDown();
               System.exit( 0 );
            }
         }
      );
   }
}
import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class DisplayAttendee extends JFrame implements
```

```
ManageShutDown
{
   // java.sql types needed for database processing
   private Connection connection;
   private Statement statement;
   private ResultSet resultSet;
   private ResultSetMetaData rsMetaData;

   // javax.swing types needed for GUI
   private JTable table;


   public DisplayAttendee()
   {
      super( "THIS IS THE DISPLAY OF ATTENDEE TABLE." );


      try
        {
         Class.forName (DBConnect.Driver);
             this.connection = DriverManager.getConnection (
                  DBConnect.DBURL,DBConnect.User,DBConnect.Password);
      }
      catch ( ClassNotFoundException cnfex ) {
         System.err.println(
            "Failed to load JDBC/ODBC driver." );
         cnfex.printStackTrace();
         System.exit( 1 );  // terminate program
      }
      catch ( SQLException sqlex ) {
         System.err.println( "Unable to connect" );
         sqlex.printStackTrace();
         System.exit( 1 );  // terminate program
      }

        JPanel topPanel = new JPanel();
        topPanel.setBackground(new Color(200,0,200));//added
      topPanel.setLayout( new BorderLayout() );


      table = new JTable( 4, 4 );

      Container c = getContentPane();
        c.setBackground(new Color(200,0,200));
      c.setLayout( new BorderLayout() );
      c.add( topPanel, BorderLayout.NORTH );
      c.add( table, BorderLayout.CENTER );

      getTable();

      setSize( 800, 600 );
      show();
   }

   private void getTable()
   {
      try
```

```
      {
       String query = "SELECT * FROM ATTENDEES;";
       statement = connection.createStatement();
       resultSet = statement.executeQuery( query );
       displayResultSet( resultSet );
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private void displayResultSet( ResultSet rs )
   throws SQLException
{
   // position to first record
   boolean moreRecords = rs.next();

   // If there are no records, display a message
   if ( ! moreRecords )
     {
      JOptionPane.showMessageDialog( this,
         "ResultSet contained no records" );
      setTitle( "No records to display" );
      return;
   }

   Vector columnHeads = new Vector();
   Vector rows = new Vector();

   try
     {
      // get column heads
      ResultSetMetaData rsmd = rs.getMetaData();

      for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
         columnHeads.addElement( rsmd.getColumnName( i ) );

      // get row data
      do
          {
         rows.addElement( getNextRow( rs, rsmd ) );
      } while ( rs.next() );

      // display table with ResultSet contents
      table = new JTable( rows, columnHeads );
      JScrollPane scroller = new JScrollPane( table );
      Container c = getContentPane();
      c.remove( 1 );
      c.add( scroller, BorderLayout.CENTER );
      c.validate();
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}
```

```
private Vector getNextRow( ResultSet rs,
                           ResultSetMetaData rsmd )
    throws SQLException
{
   Vector currentRow = new Vector();

   for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
      switch( rsmd.getColumnType( i ) ) {
         case Types.VARCHAR:
         case Types.LONGVARCHAR:
               currentRow.addElement( rs.getString( i ) );
            break;
         case Types.DOUBLE:
               currentRow.addElement(
                  new Double( rs.getDouble( i ) ) );
            break;
          case Types.INTEGER:
                  currentRow.addElement(
                     new Long(rs.getLong(i)));
                  break;
          case Types.DATE:
                  currentRow.addElement(rs.getDate(i));
                  break;
         default:
                  String temp = rsmd.getColumnTypeName(i);
                  if(temp.equals("DATETIME"))
                 {
                        currentRow.addElement(
                        rs.getString(i));
                  }
            System.out.println( "Type was: " +
               rsmd.getColumnTypeName( i ) );
      }

   return currentRow;
}

public void shutDown()
{
   try
     {
      connection.close();
   }
   catch ( SQLException sqlex )
     {
      System.err.println( "Unable to disconnect" );
      sqlex.printStackTrace();
   }
}

public static void main( String args[] )
{
   final DisplayAttendee app =
      new DisplayAttendee();

   app.addWindowListener(
```

```
         new WindowAdapter() {
            public void windowClosing( WindowEvent e )
            {
               app.shutDown();
               System.exit( 0 );
            }
         }
      );
   }
}
import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class DisplayEvents extends JFrame implements
ManageShutDown
{
   // java.sql types needed for database processing
   private Connection connection;
   private Statement statement;
   private ResultSet resultSet;
   private ResultSetMetaData rsMetaData;

   // javax.swing types needed for GUI
   private JTable table;


   public DisplayEvents()
   {
      super( "THIS IS THE DISPLAY OF EVENTS TABLE." );


      try
        {
         Class.forName (DBConnect.Driver);
             this.connection = DriverManager.getConnection (
                  DBConnect.DBURL,DBConnect.User,DBConnect.Password);
      }
      catch ( ClassNotFoundException cnfex ) {
         System.err.println(
            "Failed to load JDBC/ODBC driver." );
         cnfex.printStackTrace();
         System.exit( 1 );  // terminate program
      }
      catch ( SQLException sqlex ) {
         System.err.println( "Unable to connect" );
         sqlex.printStackTrace();
         System.exit( 1 );  // terminate program
      }

        JPanel topPanel = new JPanel();
        topPanel.setBackground(new Color(200,0,200));//added
      topPanel.setLayout( new BorderLayout() );
```

```
   table = new JTable( 4, 4 );

   Container c = getContentPane();
     c.setBackground(new Color(200,0,200));
   c.setLayout( new BorderLayout() );
   c.add( topPanel, BorderLayout.NORTH );
   c.add( table, BorderLayout.CENTER );

   getTable();

   setSize( 800, 500 );
   show();
}

private void getTable()
{
   try
     {
      String query = "SELECT * FROM EVENTS;";
      statement = connection.createStatement();
      resultSet = statement.executeQuery( query );
      displayResultSet( resultSet );
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private void displayResultSet( ResultSet rs )
   throws SQLException
{
   // position to first record
   boolean moreRecords = rs.next();

   // If there are no records, display a message
   if ( ! moreRecords )
     {
      JOptionPane.showMessageDialog( this,
         "ResultSet contained no records" );
      setTitle( "No records to display" );
      return;
   }

   Vector columnHeads = new Vector();
   Vector rows = new Vector();

   try
     {
      // get column heads
      ResultSetMetaData rsmd = rs.getMetaData();

      for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
         columnHeads.addElement( rsmd.getColumnName( i ) );

      // get row data
      do
```

```
          {
         rows.addElement( getNextRow( rs, rsmd ) );
      } while ( rs.next() );

      // display table with ResultSet contents
      table = new JTable( rows, columnHeads );
      JScrollPane scroller = new JScrollPane( table );
      Container c = getContentPane();
      c.remove( 1 );
      c.add( scroller, BorderLayout.CENTER );
      c.validate();
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private Vector getNextRow( ResultSet rs,
                           ResultSetMetaData rsmd )
    throws SQLException
{
   Vector currentRow = new Vector();

   for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
      switch( rsmd.getColumnType( i ) ) {
         case Types.VARCHAR:
         case Types.LONGVARCHAR:
               currentRow.addElement( rs.getString( i ) );
            break;
         case Types.DOUBLE:
               currentRow.addElement(
                  new Double( rs.getDouble( i ) ) );
            break;
          case Types.INTEGER:
                  currentRow.addElement(
                     new Long(rs.getLong(i)));
                  break;
          case Types.DATE:
                  currentRow.addElement(rs.getDate(i));
                  break;
         default:
                  String temp = rsmd.getColumnTypeName(i);
                  if(temp.equals("DATETIME"))
                 {
                        currentRow.addElement(
                        rs.getString(i));
                  }
            System.out.println( "Type was: " +
               rsmd.getColumnTypeName( i ) );
      }

   return currentRow;
}

public void shutDown()
{
```

```
      try
        {
         connection.close();
      }
      catch ( SQLException sqlex )
        {
         System.err.println( "Unable to disconnect" );
         sqlex.printStackTrace();
      }
   }

   public static void main( String args[] )
   {
      final DisplayEvents app =
         new DisplayEvents();

      app.addWindowListener(
         new WindowAdapter() {
            public void windowClosing( WindowEvent e )
            {
               app.shutDown();
               System.exit( 0 );
            }
         }
      );
   }
}
import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class DisplayLecturer extends JFrame implements
ManageShutDown
{
   // java.sql types needed for database processing
   private Connection connection;
   private Statement statement;
   private ResultSet resultSet;
   private ResultSetMetaData rsMetaData;

   // javax.swing types needed for GUI
   private JTable table;


   public DisplayLecturer()
   {
      super( "THIS IS THE DISPLAY OF LECTURER TABLE." );


      try
        {
         Class.forName (DBConnect.Driver);
             this.connection = DriverManager.getConnection (
                  DBConnect.DBURL,DBConnect.User,DBConnect.Password);
      }
```

```
   catch ( ClassNotFoundException cnfex ) {
      System.err.println(
         "Failed to load JDBC/ODBC driver." );
      cnfex.printStackTrace();
      System.exit( 1 );  // terminate program
   }
   catch ( SQLException sqlex ) {
      System.err.println( "Unable to connect" );
      sqlex.printStackTrace();
      System.exit( 1 );  // terminate program
   }

     JPanel topPanel = new JPanel();
     topPanel.setBackground(new Color(200,0,200));//added
   topPanel.setLayout( new BorderLayout() );


   table = new JTable( 4, 4 );

   Container c = getContentPane();
     c.setBackground(new Color(200,0,200));
   c.setLayout( new BorderLayout() );
   c.add( topPanel, BorderLayout.NORTH );
   c.add( table, BorderLayout.CENTER );

   getTable();

   setSize( 800, 500 );
   show();
}

private void getTable()
{
   try
     {
      String query = "SELECT * FROM LECTURER;";
      statement = connection.createStatement();
      resultSet = statement.executeQuery( query );
      displayResultSet( resultSet );
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private void displayResultSet( ResultSet rs )
   throws SQLException
{
   // position to first record
   boolean moreRecords = rs.next();

   // If there are no records, display a message
   if ( ! moreRecords )
     {
      JOptionPane.showMessageDialog( this,
         "ResultSet contained no records" );
```

```
      setTitle( "No records to display" );
      return;
   }

   Vector columnHeads = new Vector();
   Vector rows = new Vector();

   try
     {
      // get column heads
      ResultSetMetaData rsmd = rs.getMetaData();

      for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
         columnHeads.addElement( rsmd.getColumnName( i ) );

      // get row data
      do
          {
         rows.addElement( getNextRow( rs, rsmd ) );
      } while ( rs.next() );

      // display table with ResultSet contents
      table = new JTable( rows, columnHeads );
      JScrollPane scroller = new JScrollPane( table );
      Container c = getContentPane();
      c.remove( 1 );
      c.add( scroller, BorderLayout.CENTER );
      c.validate();
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private Vector getNextRow( ResultSet rs,
                           ResultSetMetaData rsmd )
    throws SQLException
{
   Vector currentRow = new Vector();

   for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
      switch( rsmd.getColumnType( i ) ) {
         case Types.VARCHAR:
         case Types.LONGVARCHAR:
               currentRow.addElement( rs.getString( i ) );
            break;
         case Types.DOUBLE:
               currentRow.addElement(
                  new Double( rs.getDouble( i ) ) );
            break;
          case Types.INTEGER:
                  currentRow.addElement(
                     new Long(rs.getLong(i)));
                  break;
          case Types.DATE:
                  currentRow.addElement(rs.getDate(i));
```

```
                    break;
            default:
                    String temp = rsmd.getColumnTypeName(i);
                    if(temp.equals("DATETIME"))
                   {
                          currentRow.addElement(
                          rs.getString(i));
                    }
               System.out.println( "Type was: " +
                  rsmd.getColumnTypeName( i ) );
         }

      return currentRow;
   }

   public void shutDown()
   {
      try
        {
         connection.close();
      }
      catch ( SQLException sqlex )
        {
         System.err.println( "Unable to disconnect" );
         sqlex.printStackTrace();
      }
   }

   public static void main( String args[] )
   {
      final DisplayLecturer app =
         new DisplayLecturer();

      app.addWindowListener(
         new WindowAdapter() {
            public void windowClosing( WindowEvent e )
            {
               app.shutDown();
               System.exit( 0 );
            }
         }
      );
   }
}
import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class DisplayQueryResults extends JFrame implements
ManageShutDown
{
   // java.sql types needed for database processing
   private Connection connection;
   private Statement statement;
   private ResultSet resultSet;
```

```
private ResultSetMetaData rsMetaData;

// javax.swing types needed for GUI
private JTable table;
private JTextArea inputQuery;
private JButton submitQuery;

public DisplayQueryResults()
{
   super( "Enter Query. Click Submit to See Results." );


   try
     {
      Class.forName (DBConnect.Driver);
          this.connection = DriverManager.getConnection (
               DBConnect.DBURL,DBConnect.User,DBConnect.Password);
   }
   catch ( ClassNotFoundException cnfex ) {
      System.err.println(
         "Failed to load JDBC/ODBC driver." );
      cnfex.printStackTrace();
      System.exit( 1 );  // terminate program
   }
   catch ( SQLException sqlex ) {
      System.err.println( "Unable to connect" );
      sqlex.printStackTrace();
      System.exit( 1 );  // terminate program
   }

   // If connected to database, set up GUI
   inputQuery =
      new JTextArea( "SELECT * FROM ATTENDEES;", 4, 30 );
     this.inputQuery.setToolTipText("ENTER SQL QUERIES HERE!");
     Icon find = new ImageIcon("search_icon2.gif");
   submitQuery = new JButton( "SUBMIT QUERY",find );
     this.submitQuery.setToolTipText("CLICK ON THE LENSE " +
     "TO RUN YOUR QUERY!");
   submitQuery.addActionListener(
      new ActionListener() {
         public void actionPerformed( ActionEvent e )
         {
            getTable();
         }
      }
   );

   JPanel topPanel = new JPanel();
     topPanel.setBackground(new Color(200,0,200));//added
   topPanel.setLayout( new BorderLayout() );
   topPanel.add( new JScrollPane( inputQuery),
                 BorderLayout.CENTER );
   topPanel.add( submitQuery, BorderLayout.SOUTH );

   table = new JTable( 4, 4 );

   Container c = getContentPane();
```

```
      c.setBackground(new Color(200,0,200));
   c.setLayout( new BorderLayout() );
   c.add( topPanel, BorderLayout.NORTH );
   c.add( table, BorderLayout.CENTER );

   getTable();

   setSize( 800, 500 );
   show();
}

private void getTable()
{
   try
     {
      String query = inputQuery.getText();
      statement = connection.createStatement();
      resultSet = statement.executeQuery( query );
      displayResultSet( resultSet );
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private void displayResultSet( ResultSet rs )
   throws SQLException
{
   // position to first record
   boolean moreRecords = rs.next();

   // If there are no records, display a message
   if ( ! moreRecords )
     {
      JOptionPane.showMessageDialog( this,
         "ResultSet contained no records" );
      setTitle( "No records to display" );
      return;
   }

   Vector columnHeads = new Vector();
   Vector rows = new Vector();

   try
     {
      // get column heads
      ResultSetMetaData rsmd = rs.getMetaData();

      for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
         columnHeads.addElement( rsmd.getColumnName( i ) );

      // get row data
      do
          {
         rows.addElement( getNextRow( rs, rsmd ) );
      } while ( rs.next() );
```

```
      // display table with ResultSet contents
      table = new JTable( rows, columnHeads );
      JScrollPane scroller = new JScrollPane( table );
      Container c = getContentPane();
      c.remove( 1 );
      c.add( scroller, BorderLayout.CENTER );
      c.validate();
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private Vector getNextRow( ResultSet rs,
                           ResultSetMetaData rsmd )
    throws SQLException
{
   Vector currentRow = new Vector();

   for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
      switch( rsmd.getColumnType( i ) ) {
         case Types.VARCHAR:
         case Types.LONGVARCHAR:
               currentRow.addElement( rs.getString( i ) );
            break;
         case Types.DOUBLE:
               currentRow.addElement(
                  new Double( rs.getDouble( i ) ) );
            break;
          case Types.INTEGER:
                  currentRow.addElement(
                     new Long(rs.getLong(i)));
                  break;
          case Types.DATE:
                  currentRow.addElement(rs.getDate(i));
                  break;
         default:
                  String temp = rsmd.getColumnTypeName(i);
                  if(temp.equals("DATETIME"))
                 {
                        currentRow.addElement(
                        rs.getString(i));
                  }
            System.out.println( "Type was: " +
               rsmd.getColumnTypeName( i ) );
      }

   return currentRow;
}

public void shutDown()
{
   try
     {
      connection.close();
```

```
      }
      catch ( SQLException sqlex )
        {
         System.err.println( "Unable to disconnect" );
         sqlex.printStackTrace();
      }
   }

   public static void main( String args[] )
   {
      final DisplayQueryResults app =
         new DisplayQueryResults();

      app.addWindowListener(
         new WindowAdapter() {
            public void windowClosing( WindowEvent e )
            {
               app.shutDown();
               System.exit( 0 );
            }
         }
      );
   }
}
import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

public class DisplayRegistration extends JFrame implements
ManageShutDown
{
   // java.sql types needed for database processing
   private Connection connection;
   private Statement statement;
   private ResultSet resultSet;
   private ResultSetMetaData rsMetaData;

   // javax.swing types needed for GUI
   private JTable table;


   public DisplayRegistration()
   {
      super( "THIS IS THE DISPLAY OF REGISTRATION TABLE." );


      try
        {
         Class.forName (DBConnect.Driver);
             this.connection = DriverManager.getConnection (
                  DBConnect.DBURL,DBConnect.User,DBConnect.Password);
      }
      catch ( ClassNotFoundException cnfex ) {
         System.err.println(
            "Failed to load JDBC/ODBC driver." );
```

```
      cnfex.printStackTrace();
      System.exit( 1 );  // terminate program
   }
   catch ( SQLException sqlex ) {
      System.err.println( "Unable to connect" );
      sqlex.printStackTrace();
      System.exit( 1 );  // terminate program
   }

     JPanel topPanel = new JPanel();
     topPanel.setBackground(new Color(200,0,200));//added
   topPanel.setLayout( new BorderLayout() );


   table = new JTable( 4, 4 );

   Container c = getContentPane();
     c.setBackground(new Color(200,0,200));
   c.setLayout( new BorderLayout() );
   c.add( topPanel, BorderLayout.NORTH );
   c.add( table, BorderLayout.CENTER );

   getTable();

   setSize( 800, 500 );
   show();
}

private void getTable()
{
   try
     {
      String query = "SELECT * FROM REGISTRATION;";
      statement = connection.createStatement();
      resultSet = statement.executeQuery( query );
      displayResultSet( resultSet );
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private void displayResultSet( ResultSet rs )
   throws SQLException
{
   // position to first record
   boolean moreRecords = rs.next();

   // If there are no records, display a message
   if ( ! moreRecords )
     {
      JOptionPane.showMessageDialog( this,
         "ResultSet contained no records" );
      setTitle( "No records to display" );
      return;
   }
```

```
   Vector columnHeads = new Vector();
   Vector rows = new Vector();

   try
     {
      // get column heads
      ResultSetMetaData rsmd = rs.getMetaData();

      for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
         columnHeads.addElement( rsmd.getColumnName( i ) );

      // get row data
      do
          {
         rows.addElement( getNextRow( rs, rsmd ) );
      } while ( rs.next() );

      // display table with ResultSet contents
      table = new JTable( rows, columnHeads );
      JScrollPane scroller = new JScrollPane( table );
      Container c = getContentPane();
      c.remove( 1 );
      c.add( scroller, BorderLayout.CENTER );
      c.validate();
   }
   catch ( SQLException sqlex )
     {
      sqlex.printStackTrace();
   }
}

private Vector getNextRow( ResultSet rs,
                           ResultSetMetaData rsmd )
    throws SQLException
{
   Vector currentRow = new Vector();

   for ( int i = 1; i <= rsmd.getColumnCount(); ++i )
      switch( rsmd.getColumnType( i ) ) {
         case Types.VARCHAR:
         case Types.LONGVARCHAR:
               currentRow.addElement( rs.getString( i ) );
            break;
         case Types.DOUBLE:
               currentRow.addElement(
                  new Double( rs.getDouble( i ) ) );
            break;
          case Types.INTEGER:
                  currentRow.addElement(
                     new Long(rs.getLong(i)));
                  break;
          case Types.DATE:
                  currentRow.addElement(rs.getDate(i));
                  break;
         default:
                  String temp = rsmd.getColumnTypeName(i);
```

```
                    if(temp.equals("DATETIME"))
                   {
                          currentRow.addElement(
                          rs.getString(i));
                    }
               System.out.println( "Type was: " +
                  rsmd.getColumnTypeName( i ) );
         }

      return currentRow;
   }

   public void shutDown()
   {
      try
        {
         connection.close();
      }
      catch ( SQLException sqlex )
        {
         System.err.println( "Unable to disconnect" );
         sqlex.printStackTrace();
      }
   }

   public static void main( String args[] )
   {
      final DisplayRegistration app =
         new DisplayRegistration();

      app.addWindowListener(
         new WindowAdapter() {
            public void windowClosing( WindowEvent e )
            {
               app.shutDown();
               System.exit( 0 );
            }
         }
      );
   }
}
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.event.*;
public class FindRecord implements ActionListener
{
      //*********************************
      //Fields
      //*********************************
      private JTextArea Output;
      private Connection AddRec_Connection;
      private ScrollingPanel fields;
      private Attendees Attendee_Data;
      private String QueryString = new String("");
      private Statement Smt;
```

```
   private String tempSSN;
   private ResultSet rs;
   //*****************************************
   //Constructors
   //*****************************************
   public FindRecord()
   {

   }

public FindRecord( Connection c, ScrollingPanel f,
JTextArea o )
{
   AddRec_Connection = c;
   fields = f;
   Output = o;
     this.Attendee_Data = new Attendees();
}
   //*****************************************
   //Methods
   //*****************************************
public void actionPerformed( ActionEvent e )
{
   try
     {
      Smt = this.AddRec_Connection.createStatement();
                if((this.fields.id.getText()).equals(""))
                {
                      JOptionPane.showMessageDialog(null,
                      "An Attendee can not be searched "
                      + "with out"+
                      " Social Security Number!");
                }
                else
                {
                      tempSSN = this.fields.id.getText();
                      /**check to see if this social security
                       * number is already in the Attendee
                       * table or not. If not then can not
                       * update. Must be a new record then.
                       */
                      QueryString = "Select * From Attendees Where "
+
                      " AttendeeSSN = " + "'" + tempSSN + "';";
                      rs = Smt.executeQuery(QueryString);
                      display(rs);
                      this.Output.append("\nQuery Successful.\n");
                      this.Smt.close();
                }//end else
                this.Smt.close();
     }//end of try block
   catch ( SQLException sqlex )
   {
      sqlex.printStackTrace();
      Output.append( sqlex.toString() );
 }//end of catch block
}//end of public void actionPerformed
```

```
   //*********************************************
   public void display(ResultSet rs)
   {
         try
         {
               if(rs.next())
               {
                     this.fields.id.setText(rs.getString(1));
                     this.fields.first.setText(rs.getString(3));
                     this.fields.last.setText(rs.getString(2));
                     this.fields.address.setText(rs.getString(4));
                     this.fields.city.setText(rs.getString(5));
                     this.fields.zip.setText(rs.getString(7));
                     this.fields.home.setText(rs.getString(8));
                     this.fields.work.setText(rs.getString(9));
                     this.fields.email.setText(rs.getString(10));
               }
               else
                        Output.append("\nNo record found.\n");
         }
         catch(SQLException sqlex )
         {
                  sqlex.printStackTrace();
                  Output.append( sqlex.toString() );
    }//end of catch block
   }//end of method display
//*********************************************
         public static void main(String [] args)
         {
                  FindRecord temp = new FindRecord();

         }
}//end of class FindRecord
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class Help implements ActionListener
{
   private JTextArea output;


   public Help()
   {

   }
   public Help( JTextArea o )
   {
      output = o;
   }

   public void actionPerformed( ActionEvent e )
   {
      JOptionPane.showMessageDialog(null,
        "\nClick Find to locate a record.\n" +
      "Click Add to insert a new record.\n" +
```

```
      "Click Update to update " +
      "the information in a record.\n" +
      "Click Clear to empty" +
      " the textfields.\n" );
   }

      public static void main(String[] args)
      {
            Help temp = new Help();
      }
}
import java.sql.*;
public class LoadDB
{
      Statement MyStatement;
      Connection MyConnection;
      Attendees Set1;
      Object [] data;

      public LoadDB() throws SQLException
      {
            /**Set the ATTENDEES Object equal to the inputted value
             */
            this.Set1 = Attendees.constructAttendee();
            this.data = new Object [11];
            this.data[0] = "Attendees";
            this.data[1] = Set1.getSocialNum();
            this.data[2] = this.Set1.getLName();
            this.data[3] = this.Set1.getFName();
            this.data[4] = this.Set1.getAddress();
            this.data[5] = this.Set1.getCity();
            this.data[6] = this.Set1.getState();
            this.data[7] = this.Set1.getZip();
            this.data[8] = this.Set1.getHomePhone();
            this.data[9] = this.Set1.getWorkPhone();
            this.data[10] = this.Set1.getEmail();
            /**Simple check here with out database call.
             */
            for(int i=0; i<data.length; i++)
                  System.out.println((String)(data[i]));
            try
            {
                  //Load the database driver
                  Class.forName(MyConnect.DbDriver);
            }
            catch(java.lang.ClassNotFoundException e)
            {
                  e.printStackTrace(System.err);
            }
            this.MyConnection = DriverManager.getConnection(
            MyConnect.DbURL,MyConnect.UserName,MyConnect.Password);
            this.MyStatement = this.MyConnection.createStatement();
      }
      /**Method cleanUp() will close the database connection cleanly.
       */
      public void cleanUp() throws SQLException
      {
```

```
            this.MyStatement.close();
            this.MyConnection.close();
      }
      /**tHE METHOD executeInseret can insert data into any Table.
       * The Objects must be inserted in way to maintain refrential
       * integrity.
       */
      public void executeInsert(Object[] data)
      {
            String sql = "INSERT INTO "
              + data[0] + " VALUES(";
            for(int i = 1; i < data.length; i++)
            {
              if(data[i] instanceof String)
                sql += "'" + data[i] + "'";
              else
                sql += data[i];
              if(i < data.length - 1)
                sql += ", ";
            }
            sql += ')';
            System.out.println(sql);
            try
            {
             this.MyStatement.executeUpdate(sql);
            }
            catch(SQLException sqlEx)
            {
              System.err.println("Insert failed.");
              while (sqlEx != null)
              {
                System.err.println(sqlEx.toString());
                sqlEx = sqlEx.getNextException();
              }
            }
    }//End of Method executeInsert

      /**The method load will actually call the executeInsert to
       * actually enter the data into the database
       */
      public void load()
      {
            //for(int i=0; i<data.length; i++)
                  //executeInsert(data[i]);
      }




      public static void main(String[] args) throws SQLException
      {
            LoadDB temp = new LoadDB();

      }
}
import javax.swing.*;
```

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;


public class Main_Menu extends JFrame implements ActionListener,
SwingConstants
{
         private JButton b[];

         private String names [] =
         {"DATA ENTRY", "DATA DISPLAY", "DATA QUERY", "EXIT"};
         private String tools [] = {"CLICK HERE TO ENTER DATA",
                                    "CLICK HERE TO DISPLAY DATA",
                                    "CLICK HERE TO TO DO QUERIES",
                                    "CLICK HERE TO EXIT APPLICATION"};
         private Container c;
         private JLabel label;
             private JLabel ImageLabel;
         public Main_Menu()
         {
             super("WELCOME TO CLIFFHOUSE !!!");
             this.c = getContentPane();
             c.setBackground(new Color(0,200,200).brighter());
             c.setLayout(new GridLayout(7,1,0,10));
                   Icon CliffLogo = new ImageIcon("logo.jpg");
                   ImageLabel = new JLabel(CliffLogo,
SwingConstants.CENTER);
             this.label = new JLabel("MAIN MENU",
                        SwingConstants.CENTER);

                   c.add(ImageLabel);
             c.add(label);
             this.b = new JButton [names.length];
                   Icon UpArrow = new ImageIcon("arrowupsr.gif");
                   b[0] = new JButton("DATA ENTRY",UpArrow);
                   b[0].setToolTipText("CLICK HERE TO ENTER DATA");
                   b[0].addActionListener(this);
                   c.add(b[0]);
                   Icon Mac = new ImageIcon("macnet-animate.gif");
                    for (int i = 1; i < 2; i++ )
             {
                 b[ i ] = new JButton( names[ i ],Mac );
                 b[i].setToolTipText(tools[i]);
                 b[ i ].addActionListener( this );
                 c.add( b[ i ] );
             }
                   Icon question = new ImageIcon("animated-
question.gif");
                   b[2] = new JButton ("DATA QUERY",
                              question);
                   b[2].setToolTipText("CLICK HERE TO TO DO QUERIES");
                   b[2].addActionListener(this);
                   c.add(b[2]);
             for (int i = 3; i < 4; i++ )
             {
                 b[ i ] = new JButton( names[ i ] );
```

```
                b[i].setToolTipText(tools[i]);
                b[ i ].addActionListener( this );
                c.add( b[ i ] );
            }
            //  Display
            setSize( 500, 600 );
            show();
        }

        // Process GUI Events
        public void actionPerformed(ActionEvent e)
        {
            String command;

            command = (String) (e.getActionCommand());

            if (command.equalsIgnoreCase("DATA ENTRY"))
            {
               DataEntryGUI dataentryGUI = new DataEntryGUI();
                       //setVisible(false);
                       dispose();
            }

            if (command.equalsIgnoreCase("DATA DISPLAY"))
            {
               DataDisplayGUI datadisplayGUI = new DataDisplayGUI();
                       setVisible(false);
            }

            if (command.equalsIgnoreCase("DATA QUERY"))
            {
               DataQueryGUI dataqueryGUI = new DataQueryGUI();
                       setVisible(false);
            }

            if (command.equalsIgnoreCase("EXIT"))
            {
                        dispose();
               System.exit(0);
            }

        }
  // Handle window events
protected void processWindowEvent(WindowEvent e)
{
  if (e.getID() == WindowEvent.WINDOW_CLOSING)
  {
    dispose();                    // Release resources
    System.exit(0);               // Exit the program
  }
  super.processWindowEvent(e);    // Pass on the event
}


   public static void main(String args[])
   {
       Main_Menu main_menu = new Main_Menu();
```

```
         main_menu.addWindowListener(
             new WindowAdapter()
             {
                 public void windowClosing(WindowEvent e)
                 {
                      System.exit(0);
                 }
             }
         );
     }
}
public interface ManageShutDown
{
      public void shutDown();
}
public class ManageString
{

      public static String addApostrophe(String Input)
      {
            //Add Apostrophe before the String
            String Input2 = Input.trim();
            Input2 = "'" + Input2;
            //Add Apostrophe at the end of the string
            int length = Input.length();
            String Input3 = Input2.concat("'");
            return Input3;
      }

      public static String removeApostrophe(String Output)
      {
            Output = Output.trim();
            //Get the char array for the string
            char [] CharArray = Output.toCharArray();
            //Copy into a new char array ignoring the first
            //and last cahracter
            char [] ModifiedArray = new char [CharArray.length -2];
            for(int i=1;i<CharArray.length-1; i++)
                  ModifiedArray[i-1] = CharArray[i];
            String Output2 = new String(ModifiedArray);
            return Output2;
      }



      public static void main(String[] args)
      {
            String temp = "  Satish  ";
System.out.println(temp);
            String temp1 = ManageString.addApostrophe(temp);
System.out.println(temp1);
            String temp2 = ManageString.removeApostrophe(temp1);
System.out.println(temp2);
      }
}
//CS56
```

```
// Walter H Valicente
// Pg 696 13.13


import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class MyColorChooser extends JPanel {


   private int r=204;
   private int g=204;
   private int b=204;

   private JPanel colorGui;

   private Box myBox;
   private Box redBox;
   private Box greenBox;
   private Box blueBox;


   private JTextField showRed;
   private JTextField showGreen;
   private JTextField showBlue;

   private JSlider sRed;
   private JSlider sGreen;
   private JSlider sBlue;



   public MyColorChooser()
   {
      super();

      colorGui  = new JPanel();

      myBox     = Box.createVerticalBox();
      redBox    = Box.createHorizontalBox();
      greenBox  = Box.createHorizontalBox();
      blueBox   = Box.createHorizontalBox();

      showRed   = new JTextField("Red.........204",10);
        showRed.setEditable(false);
      showGreen = new JTextField("Green.....204",10);
        showGreen.setEditable(false);
      showBlue  = new JTextField("Blue........204",10);
        showBlue.setEditable(false);

      sRed   = new JSlider(SwingConstants.HORIZONTAL, 0, 255, 10);
      sRed.setMajorTickSpacing(10);
      sRed.setPaintTicks(true);
      sRed.addChangeListener(
         new ChangeListener() {
```

```
            public void stateChanged( ChangeEvent e )
              {
                r = sRed.getValue();
                showRed.setText("Red........" + new
Integer(r).toString() );
                doBackGround();
              }
             }
            );

      sGreen = new JSlider(SwingConstants.HORIZONTAL, 0, 255, 10);
      sGreen.setMajorTickSpacing(10);
      sGreen.setPaintTicks(true);
      sGreen.addChangeListener(
         new ChangeListener() {
            public void stateChanged( ChangeEvent e )
              {
                g = sGreen.getValue();
                showGreen.setText("Green...." + new
Integer(g).toString());
                doBackGround();
              }
             }
            );

      sBlue  = new JSlider(SwingConstants.HORIZONTAL, 0, 255, 10);
      sBlue.setMajorTickSpacing(10);
      sBlue.setPaintTicks(true);
      sBlue.addChangeListener(
         new ChangeListener() {
            public void stateChanged( ChangeEvent e )
              {
                b = sBlue.getValue();
                showBlue.setText("Blue......." + new
Integer(b).toString() );
                doBackGround();
              }
             }
            );

      redBox.add(sRed); redBox.add(showRed);
      greenBox.add(sGreen); greenBox.add(showGreen);
      blueBox.add(sBlue); blueBox.add(showBlue);

      myBox.add(redBox); myBox.add(greenBox); myBox.add(blueBox);

      colorGui.setLayout( new BorderLayout(5,5) );
      colorGui.add(myBox, BorderLayout.CENTER);

      add(colorGui);

      setBackground(new Color(r,g,b) );

      setSize( 200, 150 );
      show();

    }
```

```
    private void doBackGround()
    {
      setBackground(new Color(r,g,b) );
      repaint();
    }


    public void paintComponent( Graphics g )
    {
        super.paintComponent(g);
    }


    public Color getTheColor()
    {
      Color theColor = new Color(r,g,b);
      return theColor;
    }

}
public interface ManageAttendee
{
      public void fillAttendee();
}
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class Registration
{
      private String Event_ID;
      private String AttendeeSSN;
      public static final String Insert = "INSERT INTO ";
      public static final String Values = " VALUES(";
      public static final String quote = "'";
      private static final int numColumns = 2;
      private static final String TableName = "Registration";
      Object [] data;

      public Registration()
      {
            this.data = new Object [this.numColumns + 1];
            this.AttendeeSSN = new String("");
            this.Event_ID = new String("");
      }

      public void setEvent_ID(String m_Event_ID)
      {
            this.Event_ID = m_Event_ID;
      }

      public void setAttendeeSSN(String m_AttendeeSSN)
      {
System.out.println(" I am in setAttendeeSSN of Registration class");
```

```
            this.AttendeeSSN = m_AttendeeSSN;
      }

      public String constructSqlObject()
      {
            this.data[0] = this.TableName;
            this.data[1] = this.Event_ID;
            this.data[2] = this.AttendeeSSN;
            String sql = this.Insert + this.data[0] + this.Values;
            for(int i=1; i<data.length; i++)
            {
                  if(data[i] instanceof String)
                        sql += this.quote + data[i] + this.quote;
                  else if(data[i] instanceof Character)
                  {
                        String temp =
((Character)(data[i])).toString();
                        sql +=this.quote + temp + this.quote;
                  }
                  else
                        sql += data[i];
                  /**put commas every where except after last value
                   */
                  if(i<this.data.length-1)
                        sql += ", ";
            }
            sql += " ); ";
            //System.out.println(sql);
            return sql;
      }

      public String getEventID()
      {
            return this.Event_ID;
      }

      public String getAttendeeSSN()
      {
            return this.AttendeeSSN;
      }

      public static void main(String[] args)
      {
            Registration temp = new Registration();
      }
}
/**When the add button is pressed, first only the social
 * security number must be submitted and queried against the database.
 * if the social security number does not exist in the database, then
 * entire EventVector must populate the EventList, ListBox. But if
 * social security number already exists in the attendee database
 * then the remaining fields must be filled from the database and
 * the EventList List box should only have the eventid's that fulfill
 * the following conditions
 * 1. Populate the event IDs that do not lie on the same date on
 * which attendee is already registered.
 * 2. event Id and date are shown together, so that by mistake
```

```
 * attendee is not registered for 2 events lying on the same date.
 * EventId is assigned as follows:
 * room Number || date in format DD-MM-YY. So for example an event
 * in room number 1 on 27 July 2001 will be shown as:
 * 1270701. Since business rule allows only 1 event per day per room
 * this system will work ok.
 */
// Class ScrollingPanel
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import javax.swing.event.*;
import java.sql.*;
import java.text.*;

public class ScrollingPanel extends JPanel implements BusinessConstants
{
   private JList EventList;
   private JList StateList;
   private String [] StateArray;
   private Vector EventVector;//Gets the eventID from the database
   private JPanel labelPanel;
   private JPanel fieldsPanel;
   private String labels[] =
                { "SOCIAL SECURITY NUMBER      ", "FIRST NAME      ",
                          "LAST NAME      ",
                  "ADDRESS     ", "CITY      ", "STATE      ",
                  "ZIP CODE      ", "E-MAIL      ",
                  "HOME PHONE      ", "WORK PHONE      ",
                          "CREDIT CARD NUMBER      ",
                             "Event ID      "};
   private String ToolText [] =
   {"Enter in Format 111224444 with no space and hyphens",
      "Enter First Name upto 20 Characters",
      "Enter Last Name upto 20 Characters",
      "Enter Address upto 30 Characters",
      "Enter City upto 30 Characters",
      "Choose the state from the list",
      "Enter 5 digit Zip Code",
      "Enter E Mail upto 20 characters",
      "Enter Home Phone like 3105326620. No spaces or Hyphens",
      "Enter Work Phone like 3105326620. No spaces or Hyphens",
      "Enter 16 digit credit card number with no spaces or hyphens",
      "Choose From Event ID List Box. You can choose more than one
Event ID."
   };
   JTextField id, first, last, address,     // package access
   city,  zip,email, home, work, creditcard;
   JLabel LogoLabel;
   private String StateSelected;
   private Object [] EventsSelected;
   //Constructor
   public ScrollingPanel()
   {
         /**EventVector will get the event numbers assigned after
          * the sysdate and store them in the vector.
```

```
         */
         fillEventList();
         StateSelected = new String("");
         this.StateArray = new String [this.NumStates];
         this.StateArray = this.fillStateArray();
      // Label panel
      labelPanel = new JPanel();
        this.labelPanel.setBackground(new Color(0,200,200).brighter());
      labelPanel.setLayout(
                      new GridLayout( labels.length, 1 ) );

      ImageIcon ii = new ImageIcon( "logo.jpg" );
        JLabel temp;
      for ( int i = 0; i<this.labels.length; i++ )
        {
             temp = new JLabel(labels[i],this.RIGHT);
             temp.setToolTipText(this.ToolText[i]);
         labelPanel.add(temp);
        }

        // TextField panel
      fieldsPanel = new JPanel();
        fieldsPanel.setLayout(
      new GridLayout( labels.length, 1 ) );
      id = new JTextField( 20 );
        id.setToolTipText("Enter Attendees Social Security Number
Here.");
      fieldsPanel.add( id );
      first = new JTextField( 20 );
        first.setToolTipText("Enter Attendees First Name Here. No
Apostrophes.");
      fieldsPanel.add( first );
      last = new JTextField( 20 );
        last.setToolTipText("Enter Attendees Last Name Here. No
Apostrophes.");
      fieldsPanel.add( last );
      address = new JTextField( 20 );
        this.address.setToolTipText("Enter Attendees Address Here. No
Apostrophes.");
      fieldsPanel.add( address );
      city = new JTextField( 20 );
        this.city.setToolTipText("Enter Attendees City Here. No
Apostrophes.");
      fieldsPanel.add( city );
        this.StateList = new JList(this.StateArray);
        this.StateList.setToolTipText(
        "Select Attendees State From the list");
        this.StateList.setVisibleRowCount(1);
        this.StateList.setFixedCellHeight(15);
        this.StateList.setSelectionMode(
        ListSelectionModel.SINGLE_SELECTION);
        this.fieldsPanel.add(new JScrollPane(this.StateList));
        //Add the action listener for the state list
        StateList.addListSelectionListener
       (
            new ListSelectionListener()
              {
```

```
                    public void valueChanged(ListSelectionEvent e)
                    {
                          StateSelected =
StateArray[StateList.getSelectedIndex()];
                    }
               }
          );
        zip = new JTextField( 20 );
          this.zip.setToolTipText("Enter Attendees Zip Code Here.");
        fieldsPanel.add( zip );
        email = new JTextField( 20 );
          this.email.setToolTipText("Enter Attendees E Mail Address Here.
No Apostrophes.");
        fieldsPanel.add( email );
        home = new JTextField( 20 );
          this.home.setToolTipText("Enter Attendees Home Phone Number
Here.");
        fieldsPanel.add( home );
        work = new JTextField( 20 );
          this.work.setToolTipText("Enter Attendees Work Phone Number
Here.");
        fieldsPanel.add( work );
          this.creditcard = new JTextField(20);
          this.creditcard.setToolTipText("Enter Attendees Credit card
Number Here.");
          this.fieldsPanel.add(this.creditcard);
          this.EventList = new JList(this.EventVector);
          this.EventList.setToolTipText("Choose the event ID from the
List.");
          this.EventList.setVisibleRowCount(1);
          this.EventList.setFixedCellHeight(15);
          this.EventList.setSelectionMode(
          ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
          this.fieldsPanel.add(new JScrollPane(this.EventList));

          EventList.addListSelectionListener
          (
              new ListSelectionListener()
                {
                    public void valueChanged(ListSelectionEvent e)
                    {
                          EventsSelected =
EventList.getSelectedValues();
                    }
               }
          );
          setLayout( new GridLayout( 1, 2 ) );
        add( labelPanel );
        add( fieldsPanel );
   }


   public Object [] getEvents()
   {
         return EventsSelected;
   }
```

```
public String getState()
{
      return StateSelected;
}

private String [] fillStateArray()
{
      String [] temp = new String [this.NumStates];
      int i = 0;
      temp[i++] = "Ak";
      temp[i++] = "AL";
      temp[i++] ="AR";
      temp[i++] ="AZ";
      temp[i++] ="CA";
      temp[i++] ="CO";
      temp[i++] ="CT";
      temp[i++] ="DC";
      temp[i++] ="DE";
      temp[i++] ="FL";
      temp[i++] ="GA";
      temp[i++] ="HI";
      temp[i++] ="IA";
      temp[i++] ="ID";
      temp[i++] ="IL";
    temp[i++] ="IN";
      temp[i++] ="KS";
      temp[i++] ="KY";
      temp[i++] ="LA";
      temp[i++] ="MA";
      temp[i++] ="MD";
      temp[i++] ="ME";
      temp[i++] ="MI";
      temp[i++] ="MN";
      temp[i++] ="MO";
      temp[i++] ="MS";
      temp[i++] ="MT";
      temp[i++] ="NC";
      temp[i++] ="ND";
      temp[i++] ="NE";
      temp[i++] ="NH";
      temp[i++] ="NJ";
      temp[i++] ="NM";
      temp[i++] ="NV";
      temp[i++] ="NY";
      temp[i++] ="OH";
      temp[i++] ="OK";
      temp[i++] ="OR";
      temp[i++] ="PA";
      temp[i++] ="PR";
      temp[i++] ="RI";
      temp[i++] ="SC";
      temp[i++] ="SD";
      temp[i++] ="TN";
      temp[i++] ="TX";
      temp[i++] ="UT";
      temp[i++] ="VA";
      temp[i++] ="VI";
```

```
         temp[i++] ="VT";
         temp[i++] ="WA";
         temp[i++] ="WI";
         temp[i++] ="WV";
         temp[i] ="WY";
         return temp;
   }
   private void fillEventList()
   {
         EventVector = new Vector();
         try
            {
                  Class.forName (DBConnect.Driver);
                  Connection Connect = DriverManager.getConnection (
                  DBConnect.DBURL,DBConnect.User,DBConnect.Password);
                  Statement Smt = Connect.createStatement();
                  //Do the query on events table and get the EventID
vector
                  //Create proper Date String
                  java.util.Date Today = new java.util.Date();
                  DateFormat DF = DateFormat.getDateInstance(3);
                  String DateStr = DF.format(Today);
//System.out.println(DateStr);
                  String QueryStr = "Select EventID From Events where"
                  + " EventDate > DateValue(" + "'"+ DateStr + "'" +
");";
//System.out.println(QueryStr);
                  ResultSet RS =  Smt.executeQuery(QueryStr);
                  while(RS.next())
                  {

      EventVector.addElement(RS.getString("EventID"));
//System.out.println(RS.getString("EventID"));
                  }
                  EventVector.trimToSize();
//System.out.println(EventVector.toString());
                  Smt.close();
                  Connect.close();
            }//end of try block
            catch ( ClassNotFoundException cnfex )
            {
                  // process ClassNotFoundExceptions here
                  cnfex.printStackTrace();
            }
            catch ( SQLException sqlex )
            {
               // process SQLExceptions here
               sqlex.printStackTrace();
            }
            catch ( Exception ex )
            {
               // process remaining Exceptions here
               ex.printStackTrace();
            }
   }

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

```
   {
         ScrollingPanel temp = new ScrollingPanel();



   }


}//end of ScrollingPanel Class

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.event.*;
public class UpdateRecord implements ActionListener
{
      //*********************************
      //Fields
      //*********************************
      private JTextArea Output;
      private Connection AddRec_Connection;
      private ScrollingPanel fields;
      private Attendees Attendee_Data;
      private String QueryString = new String("");
      private Statement Smt;
      private String quote = "'";
      private String comma = ",";
      private String tempSSN;
      private ResultSet rs;
      //*****************************************
      //Constructors
      //*****************************************
      public UpdateRecord()
      {

      }

   public UpdateRecord( Connection c, ScrollingPanel f,
   JTextArea o )
   {
      AddRec_Connection = c;
      fields = f;
      Output = o;
        this.Attendee_Data = new Attendees();
   }
      //*****************************************
      //Methods
      //*****************************************
   public void actionPerformed( ActionEvent e )
   {
      try
        {
         Smt = this.AddRec_Connection.createStatement();
                  if((this.fields.id.getText()).equals(""))
                  {
                        JOptionPane.showMessageDialog(null,
```

```
                        "An Update of Record for Attendee"
                        + " can not be done with out"+
                        " Social Security Number!");
                  }
                  else
                  {
                        tempSSN = this.fields.id.getText();
                        /**check to see if this social security
                         * number is already in the Attendee
                         * table or not. If not then can not
                         * update. Must be a new record then.
                         */
                        QueryString = "Select * From Attendees Where "
+
                        " AttendeeSSN = " + "'" + tempSSN + "';";
//System.out.println(QueryString);
                        rs = Smt.executeQuery(QueryString);
                        //Will return true if record set
                        //has data in it.
                        boolean moreRecords = rs.next();
                        if(moreRecords)
                        {
                              processUpdate();
                              this.Smt.close();
                        }
                        else
                        {
System.out.println("Update can not be done because the attendee"
                           + "Does not exist in our database.");
                        }
                  }//end of outer else
                  this.Smt.close();
        }//end of try block
      catch ( SQLException sqlex )
      {
         sqlex.printStackTrace();
         Output.append( sqlex.toString() );
    }//end of catch block
   }//end of public void actionPerformed
   //*****************************************
   private void fillAttendee()
      {
            this.Attendee_Data.setSocialNum(tempSSN);
System.out.println(tempSSN);

      this.Attendee_Data.setFirstName(this.fields.first.getText());
System.out.println(this.fields.first.getText());

            this.Attendee_Data.setLastName(this.fields.last.getText());
System.out.println(this.fields.last.getText());

      this.Attendee_Data.setAddress(this.fields.address.getText());
System.out.println(this.fields.address.getText());

            this.Attendee_Data.setCity(this.fields.city.getText());
System.out.println(this.fields.city.getText());
            this.Attendee_Data.setZipCode(this.fields.zip.getText());
```

```
System.out.println(this.fields.zip.getText());
            this.Attendee_Data.setEMail(this.fields.email.getText());
System.out.println(this.fields.email.getText());


      this.Attendee_Data.setHomePhone(this.fields.home.getText());
System.out.println(this.fields.home.getText());

      this.Attendee_Data.setWorkPhone(this.fields.work.getText());
System.out.println(this.fields.work.getText());
            this.Attendee_Data.setState(this.fields.getState());
System.out.println(this.fields.getState());
System.out.println("Finished filling Attendee Object");
      }
//*************************************************************
   private void processUpdate() throws SQLException
   {
         fillAttendee();
         //Insert into database
            QueryString =
            Attendee_Data.constructSqlUpdate();
            this.Output.append("\nSending the data to" +
            " database: " +
            this.AddRec_Connection.nativeSQL(
            this.QueryString) +     "\n");
            int result = this.Smt.executeUpdate(
            this.QueryString);
            if(result == 1)
            this.Output.append("\nInsertion Successful\n");
            else
            {
                  this.Output.append(
                  "\nInsertion Failed. Contact" +
                  " System Administrator.\n");
            }
   }
   //*************************************************************
      public static void main(String[] args)
      {
      }
}
```

Conclusions

This project helped me tremendously in learning as how to connect and manipulate a database through a Java Program Interface. Though we did not use the Oracle database for this project, I(Satish Singhal) learned and established to manipulate the Oracle 8i personal edition through a java program, which no book authors discuss in their advanced java books. We also learned a great deal about designing and coding Java Graphical user interfaces and data validation. Walter worked very diligently on learning how to design and implement servlets and deliver web pages from servelets, and Dan made extensive use of GridBagLayout class, which is perhaps the most difficult to use Layout class in Java. Overall our knowledge of Relational Databases and their relationship to java programming has been advanced significantly.