Java Programming Open Ended

A Open Ended Experiment Submitted to

Amity University, Uttar Pradesh

in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in

Computer Science & Engineering

STUDENT INFORMATION SYSTEM

Student Name : Ayush Agarwal
Enrollment No: A2305216324

Faculty Name: Mr. Ankur Chaudhary
Designation : Asst. Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY UTTAR PRADESH, NOIDA (U.P.)

APRIL, 2018

/home/lucifer/NetBeansProjects/sis/src/sis/Sis.java

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package sis;
7
8 import sis.gui.MainWindow;
9
10 /**
11  *
12  * @author lucifer
13  */
14 public class Sis {
15     public MainWindow mainAppWindow;
16     /**
17      * @param args the command line arguments
18      */
19     public static void main(String[] args) {
20         // TODO code application logic here
21         Sis s = new Sis();
22         s.init();
23     }
24     public void init(){
25         mainAppWindow = new MainWindow();
26     }
27 }
```

/home/lucifer/NetBeansProjects/sis/src/sis/config.java

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package sis;
7
8 /**
9  *
10  * @author lucifer
11  */
12 public class config {
13     final public String applicationTitle = "Student Information System";
14
15     final public int WindowHeight = 400;
16     final public int WindowWidth = 400;
17
18     final public boolean isResizable = false;
19
20     final public String DatabaseName = "sis";
21     final public String DatabaseUser = "postgres";
22     final public String DatabasePass = "password";
23     final public String DatabaseHost = "localhost";
24     final public String DatabasePort = "5432";
25
26 }
```

```
 1 /*
 2  * To change this license header, choose License Headers in Project Properties.
 3  * To change this template file, choose Tools | Templates
 4  * and open the template in the editor.
 5  */
 6 package sis.gui;
 7
 8 import java.awt.*;
 9 import javax.swing.*;
10 import java.awt.event.*;
11 import java.util.ArrayList;
12 import java.util.Dictionary;
13 import java.util.Hashtable;
14 import java.util.logging.Level;
15 import java.util.logging.Logger;
16 import sis.config;
17 import sis.utils.Exceptions.*;
18 import sis.utils.studentUtils;
19 import sis.utils.validations;
20 /**
21  *
22  * @author lucifer
23  */
24 class StudentView extends JFrame{
25     config c = new config();
26     SpringLayout layout = new SpringLayout();
27
28     final protected String nameFieldText = "Enter Student Name";
29     final protected String enrollFieldText = "Enter Enrollment";
30     final protected String dobFieldText = "DD/MM/YYYY";
31
32     public StudentView(){
33         setTitle("Add Student");
34         setSize(400, 200);
35         setResizable(c.isResizable);
36         setLayout(layout);
37         setJMenuBar(new exitOnlyMenuBar(this).generateMenu());
38         initializeStudentFields();
39         setVisible(true);
40     }
41
42     protected void initializeStudentFields(){
43         Frame parent = this;
44         JPanel p = new JPanel(new SpringLayout());
45         p.setBackground(Color.white);
46         JLabel name = new JLabel("Student Name: ");
47         JLabel enroll =  new JLabel("Enrollment: ");
48         JLabel dob = new JLabel("Date of Birth");
49
50         JTextField nameField = new JTextField(nameFieldText);
51         nameField.setName("name");
52         JTextField enrollField = new JTextField(enrollFieldText);
53         enrollField.setName("enroll");
54         JTextField dobField = new JTextField(dobFieldText);
55         dobField.setName("dob");
56
57         JButton submit = new JButton("Submit");
```

```
58          JButton reset = new JButton("Reset");
59
60          name.setLabelFor(nameField);
61          enroll.setLabelFor(enrollField);
62          dob.setLabelFor(dobField);
63
64          submit.addActionListener(new ActionListener(){
65              @Override
66              public void actionPerformed(ActionEvent e){
67                  validations v = new validations();
68                  ErrorBox eb = new ErrorBox();
69                  if(!v.validate(nameField.getText(), "name")){
70                      eb.addMsg(v.getMsg());
71                  }
72                  if(!v.validate(enrollField.getText(), "enroll")){
73                      eb.addMsg(v.getMsg());
74                  }
75                  if(!v.validate(dobField.getText(), "dob")){
76                      eb.addMsg(v.getMsg());
77                  }
78                  if(eb.is_set){
79                      eb.showBox();
80                  }else{
81                      eb.clear();
82                      studentUtils s = new studentUtils();
83
84                      try{
85                          s.createNewStudent(
86                              nameField.getText(),
87                              enrollField.getText(),
88                              dobField.getText()
89                          );
90                          eb.setType("success");
91                          eb.addMsg("Added Successfully!");
92                          eb.showBox();
93                          parent.dispose();
94                      }catch(EnrollmentExistsException error){
95                          eb.addMsg("Enrollment Exists in Database!");
96                          eb.showBox();
97                      }catch(DateParsingException error){
98                          eb.addMsg("Unable to Process Date!");
99                          eb.showBox();
100                     }catch(DatabaseException error){
101                         eb.addMsg("Database Error Occured : " + error.msg);
102                         eb.showBox();
103                     }
104                 }
105             }
106         });
107
108         reset.addActionListener(new ActionListener(){
109             @Override
110             public void actionPerformed(ActionEvent e){
111                 nameField.setText(nameFieldText);
112                 enrollField.setText(enrollFieldText);
113                 dobField.setText(dobFieldText);
114             }
```

```
115         });
116 
117         p.add(name);
118         p.add(nameField);
119         p.add(enroll);
120         p.add(enrollField);
121         p.add(dob);
122         p.add(dobField);
123         p.add(reset);
124         p.add(submit);
125 
126         SpringUtilities.makeCompactGrid(p, 4, 2, 6, 6, 5, 5);
127 
128         setContentPane(p);
129     }
130 }
131 
132 class StudentsListView extends JFrame{
133     config c = new config();
134     JPanel mainPanel = new JPanel();
135     JPanel headerPanel;
136     JScrollPane studentListPanel = new JScrollPane();
137 
138     final String[] tableColumns = {
139             "id",
140             "Name",
141             "Enrollment Number",
142             "Date of Birth"
143         };
144 
145     public StudentsListView(){
146         setTitle("Students List");
147         setSize(c.WindowHeight, c.WindowWidth);
148         setResizable(c.isResizable);
149         setJMenuBar(new studentListMenuBar(this).generateMenu());
150         setVisible(true);
151         mainPanel.setLayout(new GridLayout(2,1));
152         initializeLayout();
153         studentUtils s = new studentUtils();
154         try {
155             generateStudentList(s.getStudents());
156         } catch (DatabaseException ex) {
157             System.out.println(ex.getMessage());
158         }
159     }
160 
161     private void initializeLayout(){
162         headerPanel = generateHeaderPanel();
163         mainPanel.add(headerPanel);
164         mainPanel.add(studentListPanel);
165         add(mainPanel);
166     }
167 
168     private JPanel generateHeaderPanel(){
169         JPanel j = new JPanel();
170 
171         JLabel searchNameText = new JLabel("Search By Name:");
```

```
172         searchNameText.setPreferredSize(new Dimension(c.WindowHeight, 20));
173         JTextField searchNameField = new JTextField("");
174         searchNameField.setPreferredSize(new Dimension(c.WindowHeight, 20));
175         searchNameText.setLabelFor(searchNameField);
176 
177 
178         JLabel searchEnrollText = new JLabel("Search By Enrollment:");
179         searchEnrollText.setPreferredSize(new Dimension(c.WindowHeight, 20));
180         JTextField searchEnrollField = new JTextField("");
181         searchEnrollField.setPreferredSize(new Dimension(c.WindowHeight, 20));
182         searchEnrollText.setLabelFor(searchEnrollField);
183 
184         JButton searchBtn = new JButton("Search");
185 
186         searchBtn.addActionListener(new ActionListener(){
187            @Override
188            public void actionPerformed(ActionEvent e){
189                ErrorBox eb = new ErrorBox();
190                if(searchNameField.getText().length() == 0 &&
searchEnrollField.getText().length() == 0){
191                    eb.addMsg("Please provide either name or enrollment!");
192                }else{
193                    if(searchEnrollField.getText().length() != 0){
194                        validations v = new validations();
195                        if(!v.validate(searchEnrollField.getText(),"enroll")){
196                            eb.addMsg("Enter a valid enrollment number!" +
v.getMsg());
197                        }else{
198                            studentUtils s = new studentUtils();
199                            try {
200                                Dictionary res =
s.searchStudentByEnrollment(searchEnrollField.getText());
201                                generateStudentList(res);
202                            } catch (DatabaseException ex) {
203                                eb.addMsg("Database Error: " + ex.msg);
204                            } catch (EnrollmentNotFoundException ex) {
205                                eb.addMsg("Enrollment Not Found!");
206                            }
207                        }
208                    }else{
209                        validations v = new validations();
210 
if(!v.validate(searchNameField.getText(),"name_not_full")){
211                            eb.addMsg("Enter a valid name!" + v.getMsg());
212                        }else{
213                            studentUtils s = new studentUtils();
214                            try {
215                                ArrayList<Dictionary> res =
s.searchStudentsByName(searchNameField.getText());
216                                generateStudentList(res);
217                            } catch (DatabaseException ex) {
218                                eb.addMsg("Database Error: " + ex.msg);
219                            }
220                        }
221                    }
222                }
223                if(eb.is_set)
```

```
224                     eb.showBox();
225                }
226            });
227
228            j.add(searchNameText);
229            j.add(searchNameField);
230            j.add(searchEnrollText);
231            j.add(searchEnrollField);
232            j.add(searchBtn);
233            j.setSize(c.WindowWidth, 150);
234            j.setLayout(new GridLayout(0,1));
235            return j;
236        }
237
238        private void generateStudentList(Dictionary student){
239
240            String[][] data = {
241                {
242                    student.get("id").toString(),
243                    student.get("name").toString(),
244                    student.get("enroll").toString(),
245                    student.get("dob").toString(),
246                },
247            };
248
249            JTable studentTable = new JTable(data, tableColumns);
250
251            JScrollPane jp = new JScrollPane(studentTable);
252            updateStudentListPanel(jp);
253
254        }
255
256        private void generateStudentList(ArrayList<Dictionary> students){
257
258            String[][] data = new String[students.size()][4];
259
260            int i=0;
261            for(Dictionary student:students){
262                data[i][0] = student.get("id").toString();
263                data[i][1] = student.get("name").toString();
264                data[i][2] = student.get("enroll").toString();
265                data[i][3] = student.get("dob").toString();
266                i++;
267            }
268
269            JTable studentTable = new JTable(data, tableColumns);
270
271            JScrollPane jp = new JScrollPane(studentTable);
272            updateStudentListPanel(jp);
273
274        }
275
276        private void updateStudentListPanel(JScrollPane jp){
277            mainPanel.remove(studentListPanel);
278            studentListPanel = jp;
279            studentListPanel.setBounds(0, 170, c.WindowWidth, c.WindowHeight-170);
280            mainPanel.add(studentListPanel);
```

```
281         //mainPanel.revalidate();
282 
283     }
284 
285 }
286 
287 class ErrorBox{
288     public boolean is_set;
289     String[] msg;
290     JPanel jp;
291     JDialog jd;
292     private int currentHeight = 50;
293     private Color textColor = Color.white;
294     ErrorBox(){
295         init();
296         jp.setBackground(Color.red);
297         textColor = Color.white;
298     }
299     ErrorBox(String type){
300         init();
301         setType(type);
302     }
303     private void init(){
304         is_set = false;
305         jp = new JPanel();
306         jp.setLayout(new FlowLayout());
307     }
308     public void setType(String type){
309         if(type == "success"){
310             jp.setBackground(Color.GREEN);
311             textColor = Color.BLACK;
312         }
313     }
314     public void showBox(){
315         jd = new JDialog();
316         jd.add(jp);
317         jd.setVisible(true);
318         jd.setSize(400,currentHeight);
319     }
320     public void addMsg(String msg){
321         is_set = true;
322         JLabel j = new JLabel(msg);
323         j.setForeground(textColor);
324         currentHeight += 50;
325         jp.add(j);
326     }
327     public void clear(){
328        jp.removeAll();
329     }
330 }
331 
332 class DeleteStudentView extends JFrame{
333 
334 
335     DeleteStudentView(){
336         setTitle("Delete a Student");
337         setSize(300, 100);
```

```
338         setVisible(true);
339 
340         setLayout(new GridLayout(1,1));
341 
342         JPanel j = new JPanel();
343         j.setBackground(Color.white);
344         j.setLayout(new GridLayout(3,1));
345         j.setSize(300, 100);
346 
347         JLabel title = new JLabel("Enter Enrollment Number:");
348         JTextField enrollField = new JTextField("");
349         JButton submit = new JButton("Search");
350 
351         submit.addActionListener(new ActionListener(){
352 
353             @Override
354             public void actionPerformed(ActionEvent e){
355                 validations v = new validations();
356                 if(!v.validate(enrollField.getText(), "enroll")){
357                     ErrorBox eb = new ErrorBox();
358                     eb.addMsg("Please enter a valid enrollment number!");
359                     eb.showBox();
360 
361                 }else{
362                     studentUtils s = new studentUtils();
363                     try{
364                         Dictionary student =
s.searchStudentByEnrollment(enrollField.getText());
365 
366                         remove(j);
367                         add(getDeleteStudentPanel(student));
368                         setSize(300, 300);
369 
370                     }catch(EnrollmentNotFoundException ex){
371                         ErrorBox eb = new ErrorBox();
372                         eb.addMsg("Enrollment Not Found! Please Recheck!");
373                         eb.showBox();
374                     } catch (DatabaseException ex) {
375                         ErrorBox eb = new ErrorBox();
376                         eb.addMsg("Database Error Occured!");
377                         eb.showBox();
378                     }
379                 }
380             }
381 
382         });
383 
384         j.add(title);
385         j.add(enrollField);
386         j.add(submit);
387 
388         add(j);
389 
390     }
391 
392     JPanel getDeleteStudentPanel(Dictionary student){
393         JPanel jp = new JPanel();
```

```
394
395         jp.setLayout(new GridLayout(5,1));
396         jp.setSize(300, 300);
397
398         jp.add(new JLabel("Enrollment: " + student.get("enroll").toString()));
399         jp.add(new JLabel("Name: " + student.get("name").toString()));
400         jp.add(new JLabel("Date Of Birth: " + student.get("dob").toString()));
401
402         JButton deleteBtn = new JButton("Delete");
403         JButton cancelBtn = new JButton("Cancel");
404         jp.add(deleteBtn);
405         jp.add(cancelBtn);
406         deleteBtn.addActionListener(new ActionListener(){
407             @Override
408             public void actionPerformed(ActionEvent e){
409                 studentUtils s = new studentUtils();
410                 try{
411
412 if(s.deleteStudentByEnrollment(student.get("enroll").toString())){
413                         ErrorBox eb = new ErrorBox("success");
414                         eb.addMsg("Deleted Successfully!");
415                         eb.showBox();
416                         dispose();
417                     }
418                 }catch(EnrollmentNotFoundException ex){
419                     ErrorBox eb = new ErrorBox();
420                     eb.addMsg("Enrollment Number Doesn't exist in database!");
421                     eb.showBox();
422                 }catch(DatabaseException ex){
423                     ErrorBox eb = new ErrorBox();
424                     eb.addMsg("Database Error Occured: " + ex.getMessage());
425                     eb.showBox();
426                 }
427             }
428         });
429         cancelBtn.addActionListener(new ActionListener(){
430             @Override
431             public void actionPerformed(ActionEvent e){
432                 dispose();
433             }
434         });
435         jp.setVisible(true);
436         return jp;
437     }
438
439 }
440
441 class EditStudentView extends JFrame{
442
443
444     EditStudentView(){
445         setTitle("Edit a Student");
446
447         setSize(300, 100);
448         setVisible(true);
449
450         setLayout(new GridLayout(1,1));
```

```
450
451         JPanel j = new JPanel();
452         j.setBackground(Color.white);
453         j.setLayout(new GridLayout(3,1));
454         j.setSize(300, 100);
455
456         JLabel title = new JLabel("Enter Enrollment Number:");
457         JTextField enrollField = new JTextField("");
458         JButton submit = new JButton("Search");
459
460         submit.addActionListener(new ActionListener(){
461
462             @Override
463             public void actionPerformed(ActionEvent e){
464                 validations v = new validations();
465                 if(!v.validate(enrollField.getText(), "enroll")){
466                     ErrorBox eb = new ErrorBox();
467                     eb.addMsg("Please enter a valid enrollment number!");
468                     eb.showBox();
469
470                 }else{
471                     studentUtils s = new studentUtils();
472                     try{
473                         Dictionary student =
s.searchStudentByEnrollment(enrollField.getText());
474                         remove(j);
475                         add(getEditStudentPanel(student));
476                         setSize(400, 200);
477
478                     }catch(EnrollmentNotFoundException ex){
479                         ErrorBox eb = new ErrorBox();
480                         eb.addMsg("Enrollment Not Found! Please Recheck!");
481                         eb.showBox();
482                     } catch (DatabaseException ex) {
483                         ErrorBox eb = new ErrorBox();
484                         eb.addMsg("Database Error Occured!");
485                         eb.showBox();
486                     }
487                 }
488             }
489
490         });
491
492         j.add(title);
493         j.add(enrollField);
494         j.add(submit);
495
496         add(j);
497
498     }
499
500     JPanel getEditStudentPanel(Dictionary student){
501         JPanel jp = new JPanel();
502         jp.setLayout(new GridLayout(4,2));
503         jp.setSize(400, 200);
504
505         jp.add(new JLabel("Enrollment"));
```

```
506         JTextField enrollField = new
JTextField(student.get("enroll").toString());
507         jp.add(enrollField);
508 
509         jp.add(new JLabel("Name"));
510         JTextField nameField = new JTextField(student.get("name").toString());
511         jp.add(nameField);
512 
513         jp.add(new JLabel("Date Of Birth"));
514         JTextField dobField = new JTextField(student.get("dob").toString());
515         jp.add(dobField);
516 
517         JButton submitBtn = new JButton("Submit");
518         JButton cancelBtn = new JButton("Cancel");
519 
520         jp.add(submitBtn);
521         jp.add(cancelBtn);
522 
523         submitBtn.addActionListener(new ActionListener(){
524             @Override
525             public void actionPerformed(ActionEvent e){
526                 validations v = new validations();
527                 ErrorBox eb = new ErrorBox();
528                 if(!v.validate(nameField.getText(), "name")){
529                     eb.addMsg(v.getMsg());
530                 }
531                 if(!v.validate(enrollField.getText(), "enroll")){
532                     eb.addMsg(v.getMsg());
533                 }
534                 if(!v.validate(dobField.getText(), "dob-db")){
535                     eb.addMsg(v.getMsg());
536                 }
537                 if(eb.is_set){
538                     eb.showBox();
539                 }else{
540                     eb.clear();
541                     studentUtils s = new studentUtils();
542 
543                     Dictionary studentNew = new Hashtable();
544                     studentNew.put("name", nameField.getText());
545                     studentNew.put("enroll", enrollField.getText());
546                     studentNew.put("dob", dobField.getText());
547                     studentNew.put("id", student.get("id"));
548                     try{
549                         if(s.editStudent(student, studentNew)){
550                             ErrorBox ebS = new ErrorBox("success");
551                             ebS.addMsg("Updated Successfully!");
552                             ebS.showBox();
553                             dispose();
554                         }
555                     }catch(DatabaseException ex){
556                         eb.addMsg("Database Error Occured: " + ex.getMessage());
557                         eb.showBox();
558                     } catch (EnrollmentExistsException ex) {
559                         eb.addMsg("Enrollment Already in Database!");
560                         eb.showBox();
561                     }
```

```
562                 }
563             }
564         });
565 
566         cancelBtn.addActionListener(new ActionListener(){
567             @Override
568             public void actionPerformed(ActionEvent e){
569                 dispose();
570             }
571         });
572 
573         return jp;
574     }
575 
576 }
```

```
 1 /*
 2  * Copyright (c) 1995, 2008, Oracle and/or its affiliates. All rights reserved.
 3  *
 4  * Redistribution and use in source and binary forms, with or without
 5  * modification, are permitted provided that the following conditions
 6  * are met:
 7  *
 8  *   - Redistributions of source code must retain the above copyright
 9  *     notice, this list of conditions and the following disclaimer.
10  *
11  *   - Redistributions in binary form must reproduce the above copyright
12  *     notice, this list of conditions and the following disclaimer in the
13  *     documentation and/or other materials provided with the distribution.
14  *
15  *   - Neither the name of Oracle or the names of its
16  *     contributors may be used to endorse or promote products derived
17  *     from this software without specific prior written permission.
18  *
19  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
20  * IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
21  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
22  * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
23  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
24  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
25  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
26  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
27  * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
28  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
29  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
30  */
31
32 package sis.gui;
33
34 import javax.swing.*;
35 import javax.swing.SpringLayout;
36 import java.awt.*;
37
38 /**
39  * A 1.4 file that provides utility methods for
40  * creating form- or grid-style layouts with SpringLayout.
41  * These utilities are used by several programs, such as
42  * SpringBox and SpringCompactGrid.
43  */
44 public class SpringUtilities {
45     /**
46      * A debugging utility that prints to stdout the component's
47      * minimum, preferred, and maximum sizes.
48      */
49     public static void printSizes(Component c) {
50         System.out.println("minimumSize = " + c.getMinimumSize());
51         System.out.println("preferredSize = " + c.getPreferredSize());
52         System.out.println("maximumSize = " + c.getMaximumSize());
53     }
54
55     /**
56      * Aligns the first <code>rows</code> * <code>cols</code>
57      * components of <code>parent</code> in
```

```
 58      * a grid. Each component is as big as the maximum
 59      * preferred width and height of the components.
 60      * The parent is made just big enough to fit them all.
 61      *
 62      * @param rows number of rows
 63      * @param cols number of columns
 64      * @param initialX x location to start the grid at
 65      * @param initialY y location to start the grid at
 66      * @param xPad x padding between cells
 67      * @param yPad y padding between cells
 68      */
 69     public static void makeGrid(Container parent,
 70                                 int rows, int cols,
 71                                 int initialX, int initialY,
 72                                 int xPad, int yPad) {
 73         SpringLayout layout;
 74         try {
 75             layout = (SpringLayout)parent.getLayout();
 76         } catch (ClassCastException exc) {
 77             System.err.println("The first argument to makeGrid must use
SpringLayout.");
 78             return;
 79         }
 80 
 81         Spring xPadSpring = Spring.constant(xPad);
 82         Spring yPadSpring = Spring.constant(yPad);
 83         Spring initialXSpring = Spring.constant(initialX);
 84         Spring initialYSpring = Spring.constant(initialY);
 85         int max = rows * cols;
 86 
 87         //Calculate Springs that are the max of the width/height so that all
 88         //cells have the same size.
 89         Spring maxWidthSpring = layout.getConstraints(parent.getComponent(0)).
 90                                     getWidth();
 91         Spring maxHeightSpring = layout.getConstraints(parent.getComponent(0)).
 92                                     getWidth();
 93         for (int i = 1; i < max; i++) {
 94             SpringLayout.Constraints cons = layout.getConstraints(
 95                                             parent.getComponent(i));
 96 
 97             maxWidthSpring = Spring.max(maxWidthSpring, cons.getWidth());
 98             maxHeightSpring = Spring.max(maxHeightSpring, cons.getHeight());
 99         }
100 
101         //Apply the new width/height Spring. This forces all the
102         //components to have the same size.
103         for (int i = 0; i < max; i++) {
104             SpringLayout.Constraints cons = layout.getConstraints(
105                                             parent.getComponent(i));
106 
107             cons.setWidth(maxWidthSpring);
108             cons.setHeight(maxHeightSpring);
109         }
110 
111         //Then adjust the x/y constraints of all the cells so that they
112         //are aligned in a grid.
113         SpringLayout.Constraints lastCons = null;
```

```
114         SpringLayout.Constraints lastRowCons = null;
115         for (int i = 0; i < max; i++) {
116             SpringLayout.Constraints cons = layout.getConstraints(
117                                                 parent.getComponent(i));
118             if (i % cols == 0) { //start of new row
119                 lastRowCons = lastCons;
120                 cons.setX(initialXSpring);
121             } else { //x position depends on previous component
122                 cons.setX(Spring.sum(lastCons.getConstraint(SpringLayout.EAST),
123                                      xPadSpring));
124             }
125 
126             if (i / cols == 0) { //first row
127                 cons.setY(initialYSpring);
128             } else { //y position depends on previous row
129 
cons.setY(Spring.sum(lastRowCons.getConstraint(SpringLayout.SOUTH),
130                                      yPadSpring));
131             }
132             lastCons = cons;
133         }
134 
135         //Set the parent's size.
136         SpringLayout.Constraints pCons = layout.getConstraints(parent);
137         pCons.setConstraint(SpringLayout.SOUTH,
138                             Spring.sum(
139                                 Spring.constant(yPad),
140                                 lastCons.getConstraint(SpringLayout.SOUTH)));
141         pCons.setConstraint(SpringLayout.EAST,
142                             Spring.sum(
143                                 Spring.constant(xPad),
144                                 lastCons.getConstraint(SpringLayout.EAST)));
145     }
146 
147     /* Used by makeCompactGrid. */
148     private static SpringLayout.Constraints getConstraintsForCell(
149                                                 int row, int col,
150                                                 Container parent,
151                                                 int cols) {
152         SpringLayout layout = (SpringLayout) parent.getLayout();
153         Component c = parent.getComponent(row * cols + col);
154         return layout.getConstraints(c);
155     }
156 
157     /**
158      * Aligns the first <code>rows</code> * <code>cols</code>
159      * components of <code>parent</code> in
160      * a grid. Each component in a column is as wide as the maximum
161      * preferred width of the components in that column;
162      * height is similarly determined for each row.
163      * The parent is made just big enough to fit them all.
164      *
165      * @param rows number of rows
166      * @param cols number of columns
167      * @param initialX x location to start the grid at
168      * @param initialY y location to start the grid at
169      * @param xPad x padding between cells
```

```
170      * @param yPad y padding between cells
171      */
172     public static void makeCompactGrid(Container parent,
173                                        int rows, int cols,
174                                        int initialX, int initialY,
175                                        int xPad, int yPad) {
176         SpringLayout layout;
177         try {
178             layout = (SpringLayout)parent.getLayout();
179         } catch (ClassCastException exc) {
180             System.err.println("The first argument to makeCompactGrid must use
SpringLayout.");
181             return;
182         }
183 
184         //Align all cells in each column and make them the same width.
185         Spring x = Spring.constant(initialX);
186         for (int c = 0; c < cols; c++) {
187             Spring width = Spring.constant(0);
188             for (int r = 0; r < rows; r++) {
189                 width = Spring.max(width,
190                                    getConstraintsForCell(r, c, parent, cols).
191                                        getWidth());
192             }
193             for (int r = 0; r < rows; r++) {
194                 SpringLayout.Constraints constraints =
195                         getConstraintsForCell(r, c, parent, cols);
196                 constraints.setX(x);
197                 constraints.setWidth(width);
198             }
199             x = Spring.sum(x, Spring.sum(width, Spring.constant(xPad)));
200         }
201 
202         //Align all cells in each row and make them the same height.
203         Spring y = Spring.constant(initialY);
204         for (int r = 0; r < rows; r++) {
205             Spring height = Spring.constant(0);
206             for (int c = 0; c < cols; c++) {
207                 height = Spring.max(height,
208                                     getConstraintsForCell(r, c, parent, cols).
209                                         getHeight());
210             }
211             for (int c = 0; c < cols; c++) {
212                 SpringLayout.Constraints constraints =
213                         getConstraintsForCell(r, c, parent, cols);
214                 constraints.setY(y);
215                 constraints.setHeight(height);
216             }
217             y = Spring.sum(y, Spring.sum(height, Spring.constant(yPad)));
218         }
219 
220         //Set the parent's size.
221         SpringLayout.Constraints pCons = layout.getConstraints(parent);
222         pCons.setConstraint(SpringLayout.SOUTH, y);
223         pCons.setConstraint(SpringLayout.EAST, x);
224     }
225 }
```

/home/lucifer/NetBeansProjects/sis/src/sis/gui/listeners.java

```
 1 /*
 2  * To change this license header, choose License Headers in Project Properties.
 3  * To change this template file, choose Tools | Templates
 4  * and open the template in the editor.
 5  */
 6 package sis.gui;
 7 import java.awt.*;
 8 import java.awt.event.*;
 9 /**
10  *
11  * @author lucifer
12  */
13 class ExitListener implements ActionListener{
14     private Frame parentFrame = null;
15     ExitListener(Frame parent){
16         parentFrame = parent;
17     }
18     ExitListener(){}
19     @Override
20     public void actionPerformed(ActionEvent e){
21         if (parentFrame == null)
22             System.exit(0);
23         else
24             parentFrame.dispose();
25     }
26 }
27
28 class addStudentListener implements ActionListener{
29
30     @Override
31     public void actionPerformed(ActionEvent e){
32         StudentView sv = new StudentView();
33     }
34 }
35
36 class studentListListener implements ActionListener{
37
38     @Override
39     public void actionPerformed(ActionEvent e){
40         StudentsListView sv = new StudentsListView();
41     }
42 }
43
44 class EditStudentListener implements ActionListener{
45
46     @Override
47     public void actionPerformed(ActionEvent e){
48         EditStudentView ev = new EditStudentView();
49     }
50 }
51
52 class DeleteStudentListener implements ActionListener{
53     @Override
54     public void actionPerformed(ActionEvent e){
55         DeleteStudentView dv = new DeleteStudentView();
56     }
57 }
```

/home/lucifer/NetBeansProjects/sis/src/sis/config.java

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package sis;
7
8 /**
9  *
10  * @author lucifer
11  */
12 public class config {
13     final public String applicationTitle = "Student Information System";
14
15     final public int WindowHeight = 400;
16     final public int WindowWidth = 400;
17
18     final public boolean isResizable = false;
19
20     final public String DatabaseName = "sis";
21     final public String DatabaseUser = "postgres";
22     final public String DatabasePass = "@yush7592";
23     final public String DatabaseHost = "localhost";
24     final public String DatabasePort = "5432";
25
26 }
```

/home/lucifer/NetBeansProjects/sis/src/sis/gui/listeners.java

```
 1 /*
 2  * To change this license header, choose License Headers in Project Properties.
 3  * To change this template file, choose Tools | Templates
 4  * and open the template in the editor.
 5  */
 6 package sis.gui;
 7 import java.awt.*;
 8 import java.awt.event.*;
 9 /**
10  *
11  * @author lucifer
12  */
13 class ExitListener implements ActionListener{
14     private Frame parentFrame = null;
15     ExitListener(Frame parent){
16         parentFrame = parent;
17     }
18     ExitListener(){}
19     @Override
20     public void actionPerformed(ActionEvent e){
21         if (parentFrame == null)
22             System.exit(0);
23         else
24             parentFrame.dispose();
25     }
26 }
27
28 class addStudentListener implements ActionListener{
29
30     @Override
31     public void actionPerformed(ActionEvent e){
32         StudentView sv = new StudentView();
33     }
34 }
35
36 class studentListListener implements ActionListener{
37
38     @Override
39     public void actionPerformed(ActionEvent e){
40         StudentsListView sv = new StudentsListView();
41     }
42 }
43
44 class EditStudentListener implements ActionListener{
45
46     @Override
47     public void actionPerformed(ActionEvent e){
48         EditStudentView ev = new EditStudentView();
49     }
50 }
51
52 class DeleteStudentListener implements ActionListener{
53     @Override
54     public void actionPerformed(ActionEvent e){
55         DeleteStudentView dv = new DeleteStudentView();
56     }
57 }
```

/home/lucifer/NetBeansProjects/sis/src/sis/utils/validations.java

```
 1 /*
 2  * To change this license header, choose License Headers in Project Properties.
 3  * To change this template file, choose Tools | Templates
 4  * and open the template in the editor.
 5  */
 6 package sis.utils;
 7
 8 import java.util.regex.*;
 9 /**
10  *
11  * @author lucifer
12  */
13
14 public class validations {
15     String msg = new String("Unknown");
16
17     Pattern namePattern = Pattern.compile("[a-zA-Z]{1,}\\s[a-zA-Z]{1,}");
18     Pattern nameNotFullPattern =
Pattern.compile("^\\s*[a-zA-Z]{1,}(\\s[a-zA-Z]{0,}){0,1}\\s*$");
19     Pattern enrollPattern = Pattern.compile("[A-Z][0-9]{10,11}");
20     Pattern datePattern =
Pattern.compile("^\\s*(3[01]|[12][0-9]|0?[1-9])/(1[012]|0?[1-9])/((?:19|20)\\d{2})\\
s*$");
21     Pattern dateDbPattern =
Pattern.compile("^\\s*((?:19|20)\\d{2})-(1[012]|0?[1-9])-(3[01]|[12][0-9]|0?[1-9])\\
s*$");
22     public boolean validate(String str, String type){
23         if(type == "name"){
24             if(str.length() == 0){
25                 msg = new String("Name is Empty");
26                 return false;
27             }
28             if(!namePattern.matcher(str).matches()){
29                 msg = new String("Only Alphabets and a single space is
allowed!");
30                 return false;
31             }
32         }
33         if(type == "name_not_full"){
34             if(str.length() == 0){
35                 msg = new String("Name is Empty");
36                 return false;
37             }
38             if(!nameNotFullPattern.matcher(str).matches()){
39                 msg = new String("Only Alphabets and a space is allowed!");
40                 return false;
41             }
42         }
43         if(type == "enroll"){
44             if(str.length() == 0){
45                 msg = new String("Enrollment is Empty");
46                 return false;
47             }
48             if(!enrollPattern.matcher(str).matches()){
49                 msg = new String("Please enter a valid enrollment number");
50                 return false;
51             }
```

```
52          }
53          if(type == "dob"){
54              if(str.length() == 0){
55                  msg = new String("DOB is Empty");
56                  return false;
57              }
58              if(!datePattern.matcher(str).matches()){
59                  msg = new String("Please enter a valid date!");
60                  return false;
61              }
62          }
63          if(type == "dob-db"){
64              if(str.length() == 0){
65                  msg = new String("DOB is Empty");
66                  return false;
67              }
68              if(!dateDbPattern.matcher(str).matches()){
69                  msg = new String("Please enter a valid date!");
70                  return false;
71              }
72          }
73
74          return true;
75      }
76
77      public String getMsg(){
78          return msg;
79      }
80  }
```

/home/lucifer/NetBeansProjects/sis/src/sis/utils/studentUtils.java

```
 1 /*
 2  * To change this license header, choose License Headers in Project Properties.
 3  * To change this template file, choose Tools | Templates
 4  * and open the template in the editor.
 5  */
 6 package sis.utils;
 7
 8 import java.text.ParseException;
 9 import java.util.Date;
10 import java.util.Dictionary;
11 import java.text.SimpleDateFormat;
12 import java.util.ArrayList;
13 import java.util.Hashtable;
14
15 import sis.utils.Exceptions.*;
16 /**
17  *
18  * @author lucifer
19  */
20 public class studentUtils {
21
22     public boolean createNewStudent(String name, String enroll, String dob)
throws EnrollmentExistsException, DateParsingException,
DatabaseInitializationException, DatabaseException{
23         StudentDb database = new StudentDb();
24         try{
25             database.connect();
26             if(!database.enrollmentExists(enroll)){
27                 Date dateofbirth = new
SimpleDateFormat("dd/MM/yyyy").parse(dob);
28                 database.addNewStudent(name, enroll, dateofbirth);
29                 return true;
30             }else{
31                     throw new EnrollmentExistsException();
32             }
33         }catch(ParseException e){
34             throw new DateParsingException();
35         }catch(DatabaseInitializationException e){
36             throw e;
37         }catch(DatabaseException e){
38             throw e;
39         }
40     }
41
42     public Dictionary searchStudentByEnrollment(String enroll) throws
DatabaseException, EnrollmentNotFoundException{
43         Dictionary student;
44         StudentDb s = new StudentDb();
45         try{
46             s.connect();
47             student = s.getStudentByEnrollment(enroll);
48         }catch(EnrollmentNotFoundException e){
49             throw e;
50         }catch(DatabaseException e){
51             throw e;
52         }
53         return student;
```

```
 54     }
 55     public ArrayList<Dictionary> getStudents() throws DatabaseException{
 56         ArrayList<Dictionary> students;
 57         StudentDb s = new StudentDb();
 58         try{
 59             s.connect();
 60             students = s.getStudents();
 61             return students;
 62         }catch(DatabaseException e){
 63             throw e;
 64         }
 65     }
 66     public ArrayList<Dictionary> searchStudentsByName(String name) throws
DatabaseException{
 67         ArrayList<Dictionary> students;
 68         StudentDb s = new StudentDb();
 69         try{
 70             s.connect();
 71             students = s.getStudentsByName(name);
 72             return students;
 73         }catch(DatabaseException e){
 74             throw e;
 75         }
 76     }
 77
 78     public boolean deleteStudentByEnrollment(String enroll) throws
DatabaseException, EnrollmentNotFoundException{
 79         try{
 80             searchStudentByEnrollment(enroll);
 81
 82             StudentDb s = new StudentDb();
 83
 84             try{
 85                 s.connect();
 86                 if(s.deleteStudentByEnrollment(enroll)){
 87                     return true;
 88                 }else{
 89                     return false;
 90                 }
 91             }catch(DatabaseException ex){
 92                 throw ex;
 93             }
 94         }catch(EnrollmentNotFoundException ex){
 95             throw ex;
 96         }
 97     }
 98     public boolean editStudent(Dictionary student, Dictionary studentNew) throws
DatabaseException, EnrollmentExistsException{
 99         try{
100             StudentDb s = new StudentDb();
101             s.connect();
102             s.editStudent(student, studentNew);
103             return true;
104         }catch(DatabaseException ex){
105                 throw ex;
106         }
107     }
```

```
108 }
```

/home/lucifer/NetBeansProjects/sis/src/sis/utils/db.java

```
 1 /*
 2  * To change this license header, choose License Headers in Project Properties.
 3  * To change this template file, choose Tools | Templates
 4  * and open the template in the editor.
 5  */
 6 package sis.utils;
 7
 8 import java.sql.*;
 9 import java.text.ParseException;
10 import java.text.SimpleDateFormat;
11 import java.util.ArrayList;
12 import java.util.Dictionary;
13 import java.util.Hashtable;
14 import java.util.logging.Level;
15 import java.util.logging.Logger;
16
17 import sis.config;
18 import sis.utils.Exceptions.*;
19 /**
20  *
21  * @author lucifer
22  */
23 public class db {
24     protected Connection con = null;
25     protected config c = new config();
26
27     private String CreateStudentTableSQL = "CREATE TABLE IF NOT EXISTS
students("
28             + "id SERIAL PRIMARY KEY,"
29             + "name VARCHAR(255) NOT NULL,"
30             + "enrollment VARCHAR(12) NOT NULL UNIQUE,"
31             + "dob DATE NOT NULL"
32             + ");";
33
34     protected String AddNewStudentSQL = "INSERT INTO students(name, enrollment,
dob) VALUES(?, ?, ?) RETURNING id;";
35
36     public boolean connect() throws DatabaseException{
37         if(isConnected()){
38             return true;
39         }
40         try{
41             con = DriverManager.getConnection(getConnectionString(),
c.DatabaseUser, c.DatabasePass);
42             System.out.println("Connected to Database!");
43             try{
44                 initialize();
45                 return true;
46             }catch(DatabaseInitializationException e){
47                 con.close();
48                 con =  null;
49                 throw e;
50             }
51         }catch(SQLException e){
52             System.out.println(e.getMessage());
53             throw new DatabaseException(e.getMessage());
54         }
```

```
 55     }
 56
 57     private String getConnectionString(){
 58         return
"jdbc:postgresql://"+c.DatabaseHost+":"+c.DatabasePort+"/"+c.DatabaseName;
 59     }
 60     private boolean initialize() throws DatabaseInitializationException{
 61         if(isConnected()){
 62             try{
 63                 PreparedStatement st =
con.prepareStatement(CreateStudentTableSQL);
 64                 st.execute();
 65                 System.out.println("Database Initialized");
 66                 return true;
 67             }catch(SQLException e){
 68                 System.out.println("Unable to Initialize the database!");
 69                 System.out.println(e.getMessage());
 70                 throw new DatabaseInitializationException();
 71             }
 72         }
 73         return false;
 74     }
 75
 76     public boolean isConnected(){
 77         if(con!=null)
 78             return true;
 79         else
 80             return false;
 81     }
 82 }
 83
 84
 85 class StudentDb extends db{
 86
 87     public int addNewStudent(String name, String enroll, java.util.Date dob)
throws DatabaseException{
 88         try{
 89             PreparedStatement st = con.prepareStatement(AddNewStudentSQL);
 90             st.setString(1, name);
 91             st.setString(2, enroll);
 92             st.setDate(3, new java.sql.Date(dob.getTime()));
 93             ResultSet res = st.executeQuery();
 94             res.next();
 95             int id = res.getInt("id");
 96             System.out.println("Inserted Student with id " + id);
 97             return id;
 98         }catch(SQLException e){
 99             System.out.println(e.getMessage());
100             throw new DatabaseException(e.getMessage());
101         }
102     }
103
104     public boolean enrollmentExists(String enroll) throws DatabaseException{
105         try{
106             PreparedStatement st = con.prepareStatement("SELECT COUNT(id) AS
total FROM students WHERE enrollment=?");
107             st.setString(1, enroll);
```

```
108             ResultSet res = st.executeQuery();
109             res.next();
110             if(res.getInt(1) == 1)
111                 return true;
112             else
113                 return false;
114         }catch(SQLException e){
115             throw new DatabaseException(e.getMessage());
116         }
117     }
118 
119     public Dictionary getStudentByEnrollment(String enroll) throws
EnrollmentNotFoundException, DatabaseException{
120         Dictionary student = new Hashtable();
121         try{
122             PreparedStatement st = con.prepareStatement("SELECT id, name,
enrollment, dob FROM students WHERE enrollment = ? LIMIT 1;");
123             st.setString(1, enroll);
124 
125             ResultSet res = st.executeQuery();
126             if(res.next()){
127                 student.put("id", res.getString("id"));
128                 student.put("name", res.getString("name"));
129                 student.put("enroll", res.getString("enrollment"));
130                 student.put("dob", res.getString("dob"));
131             }else{
132                 throw new EnrollmentNotFoundException();
133             }
134         }catch(SQLException e){
135             throw new DatabaseException(e.getMessage());
136         }
137         return student;
138     }
139 
140     public ArrayList<Dictionary> getStudents() throws DatabaseException{
141         ArrayList<Dictionary> students = new ArrayList<Dictionary>();
142         try{
143             PreparedStatement st = con.prepareStatement("SELECT id, name,
enrollment, dob FROM students;");
144 
145             ResultSet res = st.executeQuery();
146             while(res.next()){
147                 Dictionary student = new Hashtable();
148                 student.put("id", res.getString("id"));
149                 student.put("name", res.getString("name"));
150                 student.put("enroll", res.getString("enrollment"));
151                 student.put("dob", res.getString("dob"));
152                 students.add(student);
153             }
154         }catch(SQLException e){
155             throw new DatabaseException(e.getMessage());
156         }
157         return students;
158     }
159     public ArrayList<Dictionary> getStudentsByName(String name) throws
DatabaseException{
160         ArrayList<Dictionary> students = new ArrayList<Dictionary>();
```

```
161         try{
162             PreparedStatement st = con.prepareStatement("SELECT id, name,
enrollment, dob FROM students WHERE LOWER(name) LIKE ?;");
163             st.setString(1, "%"+name.toLowerCase()+"%");
164             ResultSet res = st.executeQuery();
165             while(res.next()){
166                 Dictionary student = new Hashtable();
167                 student.put("id", res.getString("id"));
168                 student.put("name", res.getString("name"));
169                 student.put("enroll", res.getString("enrollment"));
170                 student.put("dob", res.getString("dob"));
171                 students.add(student);
172             }
173         }catch(SQLException e){
174             throw new DatabaseException(e.getMessage());
175         }
176         return students;
177     }
178 
179     public boolean deleteStudentByEnrollment(String enroll) throws
DatabaseException{
180         try{
181             PreparedStatement st = con.prepareStatement("DELETE FROM students
WHERE enrollment=?;");
182             st.setString(1, enroll);
183             st.execute();
184             try{
185                 getStudentByEnrollment(enroll);
186                 return false;
187             } catch (EnrollmentNotFoundException ex) {
188                 return true;
189             }
190         }catch(SQLException e){
191             throw new DatabaseException(e.getMessage());
192         }
193     }
194     public boolean editStudent(Dictionary student, Dictionary studentNew) throws
DatabaseException, EnrollmentExistsException{
195 
196         try{
197             Dictionary currentData =
getStudentByEnrollment(studentNew.get("enroll").toString());
198             if(currentData.get("id").toString().equals(student.get("id"))){
199                 throw new EnrollmentNotFoundException();
200             }else{
201                 throw new EnrollmentExistsException();
202             }
203 
204         }catch(EnrollmentNotFoundException ex){
205             try{
206                 PreparedStatement st = con.prepareStatement("UPDATE students SET
name=?, enrollment=?, dob=? WHERE id=?;");
207 
208                 SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
209 
210                 st.setString(1, studentNew.get("name").toString());
211                 st.setString(2, studentNew.get("enroll").toString());
```

```
212                 st.setDate(3, new
java.sql.Date(sdf.parse(studentNew.get("dob").toString()).getTime()));
213                 st.setInt(4, Integer.parseInt(studentNew.get("id").toString()));
214 
215                 st.execute();
216                 return true;
217             }catch(SQLException e){
218                 e.printStackTrace();
219                 throw new DatabaseException(e.getMessage());
220             } catch (ParseException e) {
221                 return false;
222             }
223         }
224 
225     }
226 }
```

/home/lucifer/NetBeansProjects/sis/src/sis/utils/Exceptions/EnrollmentExistsException.java

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package sis.utils.Exceptions;
7 import java.lang.Exception;
8 /**
9  *
10  * @author lucifer
11  */
12 public class EnrollmentExistsException extends Exception {
13
14 }
```

/home/lucifer/NetBeansProjects/sis/src/sis/utils/Exceptions/DateParsingException.java

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package sis.utils.Exceptions;
7
8 import java.lang.Exception;
9 /**
10  *
11  * @author lucifer
12  */
13 public class DateParsingException extends Exception{
14
15 }
```

/home/lucifer/NetBeansProjects/sis/src/sis/utils/Exceptions/DatabaseInitializationException.java

```
 1 /*
 2  * To change this license header, choose License Headers in Project Properties.
 3  * To change this template file, choose Tools | Templates
 4  * and open the template in the editor.
 5  */
 6 package sis.utils.Exceptions;
 7
 8 /**
 9  *
10  * @author lucifer
11  */
12 public class DatabaseInitializationException extends DatabaseException{
13     public DatabaseInitializationException(){msg = "Unable to Initialize
Database";}
14     public DatabaseInitializationException(String msg){this.msg = msg;}
15 }
```

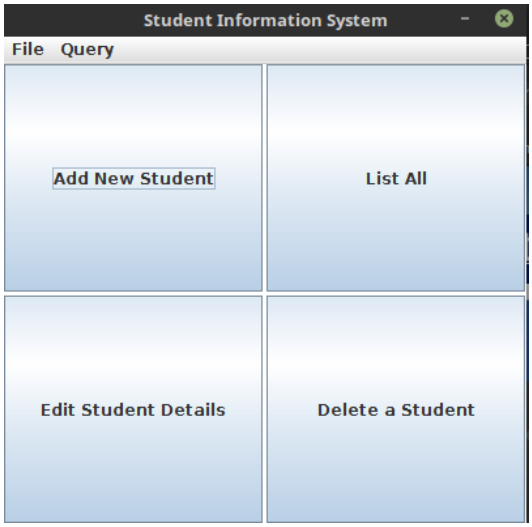
/home/lucifer/NetBeansProjects/sis/src/sis/utils/Exceptions/DatabaseException.java

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package sis.utils.Exceptions;
7 import java.lang.Exception;
8
9 /**
10  *
11  * @author lucifer
12  */
13 public class DatabaseException extends Exception{
14     public String msg;
15
16     DatabaseException(){}
17     public DatabaseException(String msg){this.msg = msg;}
18 }
```

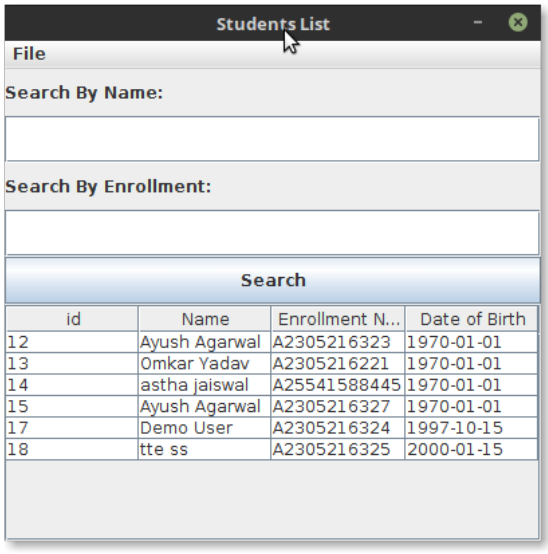
/home/lucifer/NetBeansProjects/sis/src/sis/utils/Exceptions/EnrollmentNotFoundException.java

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package sis.utils.Exceptions;
7
8 import java.lang.Exception;
9
10 /**
11  *
12  * @author lucifer
13  */
14 public class EnrollmentNotFoundException extends Exception{
15
16 }
```

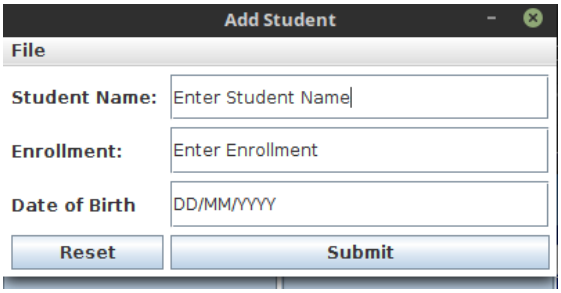
SCREENSHOTS



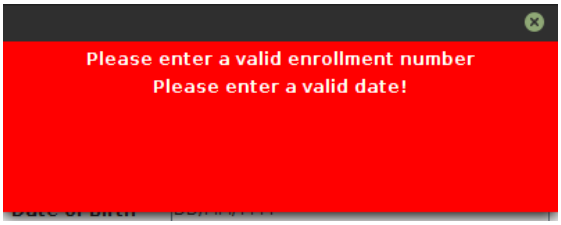
Main App Window



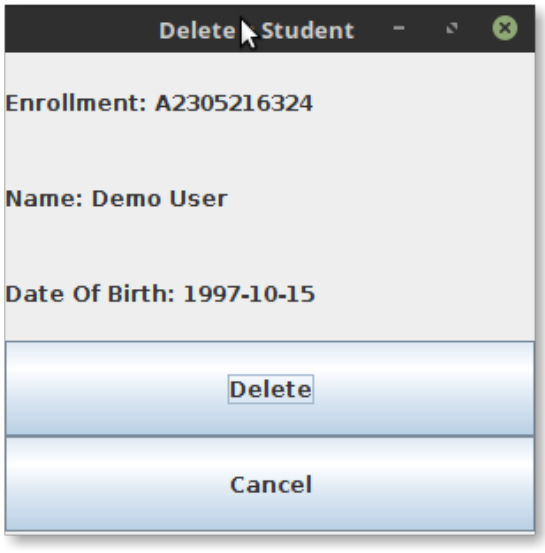
List All Student View

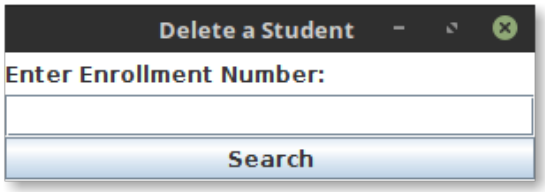


Add New Student View

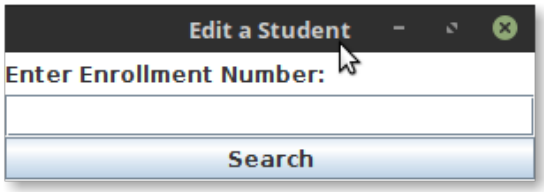


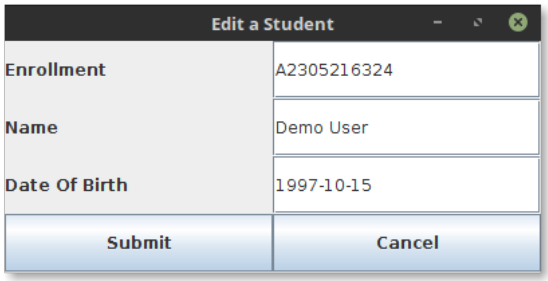
Error Box





Delete Student View





Edit Student Screen
1: Search 2: Edit Details