MainClass.java

```java
package Driver;

import AbstractDataType.LinkedList;

public class MainClass{
    public static void main (String args[]){

        //Declaration of all the classes so that they are defined (other compile error).
        RegisterScreen r = new RegisterScreen();
        SelectionMenu s = new SelectionMenu();
        Hasher hash = new Hasher();
        LinkedList l = new LinkedList();
        MCProblem prob = new MCProblem();
        SAProblem prob2 = new SAProblem();
        LoginScreen ls = new LoginScreen();
        TeacherMode tm = new TeacherMode();
        StudentMode sm = new StudentMode();
        QuestionsetMenu qs = new QuestionsetMenu();

        //Setting the starting screen to visible and to appear centered.
        ls.setVisible(true);
        ls.setLocationRelativeTo(null);
        ls.setResizable(false);
    }
}
```

```
 1 package GUIMenus;
 2
 3 import java.awt.*;
10 public class LoginScreen extends JFrame {
11
12     // Variables declaration
13     private JButton loginButton;
14     private JButton regButton;
15     private JLabel jLabel1;
16     private JLabel jLabel2;
17     private JPasswordField passField;
18     private JTextField userField;
19     private JOptionPane displayError;
20     private Image header;
21     // End of variables declaration
22
23     /** Creates new form loginScreen */
24     public LoginScreen() {
25         initComponents();
26     }
27
28     /**
29      * Method to construct the GUI layout of jFrame
30      */
31     private void initComponents() {
32
33         loginButton = new JButton();
34         regButton = new JButton();
35         userField = new JTextField();
36         jLabel1 = new JLabel();
37         passField = new JPasswordField();
38         jLabel2 = new JLabel();
39         header =
   Toolkit.getDefaultToolkit().getImage("backgroundphotos/header.jpg");
40         this.setTitle("Math Helper");
41         setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
42
43         loginButton.setText("Login");
44         loginButton.addActionListener(new ActionListener(){
45             public void actionPerformed(ActionEvent evt){
46                 loginButtonActionPerformed(evt);
47             }
48         });
49
50         regButton.setText("Register");
51         regButton.addActionListener(new ActionListener() {
52             public void actionPerformed(ActionEvent evt) {
```

```
53                  regButtonActionPerformed(evt);
54              }
55          });
56
57          jLabel1.setText("Enter Username Here:");
58
59          jLabel2.setText("Enter Password Here:");
60
61          GroupLayout layout = new GroupLayout(getContentPane());
62          getContentPane().setLayout(layout);
63          layout.setHorizontalGroup(
64              layout.createParallelGroup(GroupLayout.Alignment.LEADING)
65              .addGroup(layout.createSequentialGroup()
66
  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
67                      .addGroup(layout.createSequentialGroup()
68                          .addGap(61, 61, 61)
69
  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
70                              .addComponent(jLabel1)
71                              .addComponent(jLabel2))
72
  .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
73
  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING, false)
74                              .addComponent(loginButton)
75                              .addComponent(passField, GroupLayout.DEFAULT_SIZE,
  90, Short.MAX_VALUE)
76                              .addComponent(userField)))
77                      .addGroup(layout.createSequentialGroup()
78                          .addGap(188, 188, 188)
79                          .addComponent(regButton)))
80                  .addContainerGap(142, Short.MAX_VALUE))
81          );
82          layout.setVerticalGroup(
83              layout.createParallelGroup(GroupLayout.Alignment.LEADING)
84              .addGroup(layout.createSequentialGroup()
85                  .addContainerGap(90, Short.MAX_VALUE)
86
  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
87                      .addComponent(userField, GroupLayout.PREFERRED_SIZE, 27,
  GroupLayout.PREFERRED_SIZE)
88                      .addComponent(jLabel1))
89                  .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
90
  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
91                      .addComponent(passField, GroupLayout.PREFERRED_SIZE, 29,
```

```
   GroupLayout.PREFERRED_SIZE)
 92                     .addComponent(jLabel2))
 93                 .addGap(50, 50, 50)
 94                 .addComponent(loginButton)
 95                 .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
 96                 .addComponent(regButton)
 97                 .addGap(41, 41, 41))
 98         );
 99 
100         pack();
101     }
102     /**
103      * Method to check if the user info is valid and if it is valid, it will
   open the selection menu for the user
104      * @param evt Detects when the button is pressed
105      */
106     private void loginButtonActionPerformed(ActionEvent evt){
107         String username = userField.getText().trim();
108         char passwordArr[] = passField.getPassword();
109         String password = passwordConverter(passwordArr);
110         long userHash = Hasher.getHash(username);
111         long passHash = Hasher.getHash(password);
112         try{
113             BufferedReader reader = new BufferedReader(new FileReader("Text
   Files/userinfo.txt"));
114                 String holder = reader.readLine();
115                 while (holder != null && !holder.equals("")){
116                     String hold[] = holder.split(" ", 2);
117                     if (userHash == Long.parseLong(hold[0]) && passHash ==
   Long.parseLong(hold[1])){
118                         this.setVisible(false);
119                         SelectionMenu sm = new SelectionMenu();
120                         sm.setVisible(true);
121                         sm.setLocationRelativeTo(null);
122                         return;
123                     }
124                     holder = reader.readLine();
125                 }
126             displayError = new JOptionPane();
127             JFrame f = new JFrame();
128             JOptionPane.showMessageDialog(f,"Your username and password
   combination is incorrect.");
129             reader.close();
130         }
131         catch (IOException e){
132             displayError = new JOptionPane();
133             JFrame f = new JFrame();
```

```
134             JOptionPane.showMessageDialog(f,"An Unexpected Error has
    occured");
135         }
136     }
137 
138     /**
139      * Method to convert the password from a char array into a string
140      * @param passwordArr char array of the password
141      * @return returns a string as the password
142      */
143     public String passwordConverter(char[] passwordArr) {
144         String password = "";
145         for (int i = 0; i < passwordArr.length; i ++) {
146             if (passwordArr[i] != ' ')
147                 password += passwordArr[i];
148         }
149         return password;
150     }
151     /**
152      * Method to open up the register screen for the user
153      * @param evt Detects when the button is pressed
154      */
155     private void regButtonActionPerformed(ActionEvent evt) {
156         RegisterScreen rs = new RegisterScreen();
157         rs.setVisible(true);
158         rs.setLocationRelativeTo(null);
159         rs.setResizable(false);
160     }
161 
162 }
163 
```

```java
package GUIMenus;


import javax.swing.*;

public class RegisterScreen extends JFrame {

     // Variables declaration
    private JButton regButton;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JLabel jLabel3;
    private JPasswordField jPasswordField1;
    private JPasswordField jPasswordField2;
    private JTextField jTextField1;
    private JOptionPane jOptionPane1;
    // End of variables declaration

    //Constructor for login screen.
    public RegisterScreen() {
        initComponents();
    }

    /**Method to setup all the GUI of the register screen.
     *
     */
    private void initComponents() {

        //Finishing the declaration of all the GUI components.
        jPasswordField1 = new JPasswordField();
        jLabel1 = new JLabel();
        jPasswordField2 = new JPasswordField();
        jLabel2 = new JLabel();
        jTextField1 = new JTextField();
        jLabel3 = new JLabel();
        regButton = new JButton();

        this.setTitle("Math Helper");
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setText("Enter Your Password");

        jLabel2.setText("Confirm Your Password");

        jLabel3.setText("Enter Your Username");

        regButton.setText("Register");
```

```
51          regButton.addActionListener(new ActionListener() {
52              public void actionPerformed(ActionEvent evt) {
53                  regButtonActionPerformed(evt);
54              }
55          });
56
57          //Setting up the layout of the screen
58          GroupLayout layout = new GroupLayout(getContentPane());
59          getContentPane().setLayout(layout);
60          layout.setHorizontalGroup(
61              layout.createParallelGroup(GroupLayout.Alignment.LEADING)
62              .addGroup(layout.createSequentialGroup()
63                  .addContainerGap(85, Short.MAX_VALUE)
64
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
65                      .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
66
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING, false)
67                              .addGroup(layout.createSequentialGroup()
68
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
69                                      .addComponent(jLabel3)
70                                      .addComponent(jLabel1))
71                                  .addGap(28, 28, 28))
72                              .addGroup(layout.createSequentialGroup()
73                                  .addComponent(jLabel2,
    GroupLayout.DEFAULT_SIZE, GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
74                                  .addGap(18, 18, 18)))
75
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING, false)
76                              .addComponent(jTextField1,
    GroupLayout.DEFAULT_SIZE, 73, Short.MAX_VALUE)
77                              .addComponent(jPasswordField1)
78                              .addComponent(jPasswordField2))
79                          .addGap(124, 124, 124))
80                      .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
81                          .addComponent(regButton)
82                          .addGap(167, 167, 167))))
83          );
84          layout.setVerticalGroup(
85              layout.createParallelGroup(GroupLayout.Alignment.LEADING)
86              .addGroup(layout.createSequentialGroup()
87                  .addGap(108, 108, 108)
88
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
```

```
89                      .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE,
   GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
90                      .addComponent(jLabel3))
91                  .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
92
   .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
93                      .addComponent(jPasswordField1, GroupLayout.PREFERRED_SIZE,
   GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
94                      .addComponent(jLabel1))
95                  .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
96
   .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
97                      .addComponent(jLabel2)
98                      .addComponent(jPasswordField2, GroupLayout.PREFERRED_SIZE,
   GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
99                  .addGap(18, 18, 18)
100                 .addComponent(regButton)
101                 .addContainerGap(79, Short.MAX_VALUE))
102         );
103
104         pack();
105     }
106
107     /**
108      * Method to process the login information and to check if it matches with
   an account already
109      * @param evt Detects when the button is pressed
110      */
111     private void regButtonActionPerformed(ActionEvent evt) {
112         String user = jTextField1.getText().trim();
113         String password1 = jPasswordField1.getText().trim();
114         String password2 = jPasswordField2.getText().trim();
115         boolean valid = isValid(user, password1, password2);
116
117         if (valid){
118             long userHash = Hasher.getHash(user);
119             long passwordHash = Hasher.getHash(password1);
120             try {
121                 BufferedReader reader = new BufferedReader(new
   FileReader("Text Files/userinfo.txt"));
122                 String holder = reader.readLine();
123                 while (holder != null && !holder.equals("")){
124                     String hold[] = holder.split(" ", 2);
125                     if (userHash == Long.parseLong(hold[0])){
126                         jOptionPane1 = new JOptionPane();
127                         JFrame f = new JFrame();
128                         JOptionPane.showMessageDialog(f,"This username has
```

```
already been taken");
                        valid = false;
                        break;
                    }
                    holder = reader.readLine();
                }
                reader.close();
                writeInfo(valid, userHash, passwordHash);
            }
            catch (IOException e){
                jOptionPane1 = new JOptionPane();
                JFrame f = new JFrame();
                JOptionPane.showMessageDialog(f,"An Unexpected Error has
occured");
            }
        }

    }

    /**
     * Method to check if the user has inputted passwords and usernames which
meet all of the requirements
     * @param user String representing the inputted username
     * @param password1 String representing the first password
     * @param password2 String representing the confirmed password
     * @return returns a boolean variable to determine if the input is valid
or not
     */
    public boolean isValid(String user, String password1, String password2) {
        boolean valid = true;
        if (password1.length() < 7 || password1.length() > 14){
            jOptionPane1 = new JOptionPane();
            JFrame f = new JFrame();
            JOptionPane.showMessageDialog(f,"Please enter a password between 7
-14 characters long");
            valid = false;
        }

        else if (user.length() < 3 || user.length() > 10){
            jOptionPane1 = new JOptionPane();
            JFrame f = new JFrame();
            JOptionPane.showMessageDialog(f,"Please enter a username between 3
-10 characters long");
            valid = false;
        }

        else if (!password1.equals(password2)){
```

```
170             jOptionPane1 = new JOptionPane();
171             JFrame f = new JFrame();
172             JOptionPane.showMessageDialog(f,"The passwords do not match");
173             valid = false;
174         }
175 
176         else if (password1.contains(" ")){
177             jOptionPane1 = new JOptionPane();
178             JFrame f = new JFrame();
179             JOptionPane.showMessageDialog(f,"The password contains a space.
    Please delete the space.");
180             valid = false;
181         }
182         return valid;
183     }
184 
185     /**
186      * Method to write the encrypted hash of the username and password of the
    user if the given input is valid and is not duplicate
187      * @param valid Boolean representing if the given user info is valid
188      * @param userHash The encrypted hash of the username
189      * @param passwordHash The encrypted hash of the password
190      */
191     public void writeInfo(boolean valid, long userHash, long passwordHash) {
192         if (valid) {
193             try{
194                 BufferedWriter writer = new BufferedWriter(new
    FileWriter("Text Files/userinfo.txt", true));
195                 writer.newLine();
196                 writer.append(userHash + " " + passwordHash);
197                 writer.close();
198                 this.setVisible(false);
199             }
200             catch (IOException e){
201                 jOptionPane1 = new JOptionPane();
202                 JFrame f = new JFrame();
203                 JOptionPane.showMessageDialog(f,"An Unexpected Error has
    occured");
204             }
205         }
206     }
207 
208 }
209 
```

```java
package AbstractDataType;

import Objects.Problem;

public class LinkedList{

    private Node head;
    private Node tail;
    private int size;

    public LinkedList(){
        head = null;
        tail = null;
        size = 0;
    }

    /**
     * Method to add a new node to the head of the LinkedList
     * @param n a new node
     */
    public void addFirst(Node n){
        if (head == null){
            head = tail = n;
            n.setPrev(null);
            n.setNext(null);
        }
        else{
            n.setNext(head);
            n.setPrev(null);
            head.setPrev(n);
            head = n;
        }
        size ++;
    }

    /**
     * Method to return the head of the LinkedList
     * @return head of LinkedList
     */
    public Node getHead(){
        return head;
    }

    /**
     * Method to add a node to the tail of the LinkedList
     * @param n a new node
     */
```

```
48      public void addLast(Node n){
49          if (tail == null){
50              head = tail = n;
51              n.setPrev(null);
52              n.setNext(null);
53          }
54          else{
55              tail.setNext(n);
56              n.setPrev(tail);
57              n.setNext(null);
58              tail = n;
59          }
60          size++;
61      }
62
63      /**
64       * Method to delete a specific node
65       * @param n a node to delete
66       */
67      public void delete(Node n){
68          if (n == null || head == null){
69              return;
70          }
71          else if (n == head){
72              head = head.getNext();
73              if (head != null)
74                  head.setPrev(null);
75          }
76          else if (n == tail){
77              tail = tail.getPrev();
78              if (tail != null)
79                  tail.setNext(null);
80          }
81          else{
82              n.getPrev().setNext(n.getNext());
83              n.getNext().setNext(n.getPrev());
84              n.setNext(null);
85              n.setPrev(null);
86          }
87          size--;
88          return;
89      }
90
91      /**
92       * Recursive method to obtain a value of a certain index of a LinkedList
93       * @param indexGoal the index to obtain
94       * @param curIndex the starting index
```

```
95      * @param curNode the starting node
96      * @return the node of indexGoal
97      */
98     public Node getIndex(int indexGoal, int curIndex, Node curNode){
99         if (indexGoal == curIndex){
100            return curNode;
101        }
102        return getIndex(indexGoal, curIndex+1, curNode.getNext());
103    }
104
105    /**
106     * Helper method for the recursive method getIndex
107     * @param indexGoal the index to obtain
108     * @return the node of indexGoal
109     */
110    public Node getIndexHelper(int indexGoal){
111        return getIndex(indexGoal, 0, head);
112    }
113
114    /**
115     * Method that uses bubble sort to sort the LinkedList lexicographically
    based on the names
116     */
117    public void sortAlpha(){
118        if (head == null)
119            return;
120        for (int i = 0; i < size; i ++){
121            Node n = head;
122            for (int k = 0; k < size-i-1; k ++){
123                String strA = ((Problem)(n.getStore())).getName();
124                String strB = ((Problem)(n.getNext().getStore())).getName();
125                if (strB.compareTo(strA) < 0){
126                    swapValues(n, n.getNext());
127                }
128                n = n.getNext();
129            }
130        }
131    }
132
133    /**
134     * Method that uses bubble sort to sort the LinkedList based on the
    difficulties of the problems
135     */
136    public void sortNum(){
137        if (head == null)
138            return;
139
```

```
140         for (int i = 0; i < size; i ++){
141             Node n = head;
142             for (int k = 0; k < size-i-1; k ++){
143                 int a = Integer.parseInt(((Problem)
    (n.getStore())).getDifficulty());
144                 int b = Integer.parseInt(((Problem)
    (n.getNext().getStore())).getDifficulty());
145                 if (b < a){
146                     swapValues(n, n.getNext());
147                 }
148                 n = n.getNext();
149             }
150         }
151     }
152 
153     /**
154      * Method that swaps the stored value of two nodes of the LinkedList
155      * @param a one of the nodes that is to be swapped
156      * @param b the other node that is to be swapped
157      */
158     public void swapValues(Node a, Node b){
159         Problem temp = (Problem)(a.getStore());
160         a.setStore(b.getStore());
161         b.setStore(temp);
162     }
163 
164     /**
165      * Method that returns the tail of the LinkedList
166      * @return tail of the LinkedList
167      */
168     public Node getTail(){
169         return tail;
170     }
171 
172     /**
173      * Method that returns the size of the LinkedList
174      * @return the number of nodes in the LinkedList
175      */
176     public int getSize(){
177         return size;
178     }
179 
180 }
```

```java
package AbstractDataType;

public class Node{

    private Node next;
    private Node previous;
    private Object store;

    public Node(Object obj){
        store = obj;
        next = null;
        previous = null;
    }

    public Node(Object obj, Node next, Node previous){
        store = obj;
        this.next = next;
        this.previous = previous;
    }

    /**
     * Method to set the value of the stored object to a new value as encapsulation is used
     * @param obj a new object/value for the node
     */
    public void setStore(Object obj){
        store = obj;
    }

    /**
     * Method to return the stored object
     * @return returns the stored object
     */
    public Object getStore(){
        return store;
    }

    /**
     * Method to set the value of the next node of the current node
     * @param next the new "next" node that this node is connected to
     */
    public void setNext(Node next){
        this.next = next;
    }

    /**
     * Method to return the next node
```

```
47      * @return the next node
48      */
49     public Node getNext(){
50         return next;
51     }
52 
53     /**
54      * Method to set the previous node to another value
55      * @param previous the new previous node
56      */
57     public void setPrev(Node previous){
58         this.previous = previous;
59     }
60 
61     /**
62      * Method to get and obtain the previous node
63      * @return the previous node
64      */
65     public Node getPrev(){
66         return previous;
67     }
68 
69 }
```

```java
package Helper;

public class Hasher{

    public Hasher(){

    }

    /**
     * Method that hashes and encrypts a string using the prime modulus of 131
     * @param str Given string as input to hash
     * @return returns the encrypted hash
     */
    public static long getHash(String str){
        long code = 7;
        //Prime Modulus for Hashing
        final int prime = 131;
        for (int i = 0; i < str.length(); i ++){
            code = code * prime + str.charAt(i);
        }
        return code;
    }

}
```

```java
package Objects;

import AbstractDataType.LinkedList;

public class Problem{
    private String name;
    private String statement;
    private String difficulty;
    private LinkedList areas;

    public Problem(){
    }

    public Problem(String name, String statement, String difficulty, String types){
        this.name = name;
        this.statement = statement;
        this.difficulty = difficulty;
        areas = new LinkedList();
        String temp = "";
        for (int i = 0; i < types.length(); i ++){
            if (types.charAt(i) != ','){
                temp += types.charAt(i);
            }
            else{
                Node n = new Node(temp);
                areas.addLast(n);
                temp = "";
            }
        }
    }

    public String getTypes(){
        String typeFormatted = "";
        Node base = areas.getHead();
        for (int i = 0; i < areas.getSize(); i ++){
            if (i == 0)
                base = areas.getHead();
            else
                base = base.getNext();
            typeFormatted += ((String)(base.getStore()));
            if (i != areas.getSize()-1)
                typeFormatted += ", ";
        }
        return typeFormatted;
    }
    public LinkedList getAreas(){
```

```
48          return areas;
49      }
50      public String getProblem(){
51          return statement;
52      }
53
54      public String getDifficulty(){
55          return difficulty;
56      }
57
58      public String getName(){
59          return name;
60      }
61
62  }
```

SAProblem.java

```java
package Objects;

public class SAProblem extends Problem{
    private String answer;


    public SAProblem(){
        super();
    }

    public SAProblem(String name, String statement, String difficulty, String answer, String types){
        super(name, statement, difficulty, types);
        this.answer = answer;

    }
    /**
     * Method to return the answer as encapsulation is used
     * @return returns the answer as a String
     */
    public String getAnswer(){
        return answer;
    }

    /**
     * Method to check if the answer is correct or not
     * @param answer the answer inputted by the user
     * @return returns a boolean representing the validity of the answer
     */
    public boolean checkAnswer(String answer){
        return this.answer.equals(answer);
    }
}


```

MCProblem.java

```java
package Objects;

public class MCProblem extends Problem{
    private char answer;


    public MCProblem(){
        super();
    }

    public MCProblem(String name, String statement, String difficulty, char answer, String types){
        super(name, statement, difficulty, types);
        this.answer = answer;

    }

    /**
     * Method to return the answer as encapsulation is used
     * @return returns the answer as a char
     */
    public char getAnswer(){
        return answer;
    }

    /**
     * Method to check if the answer is correct or not
     * @param answer the answer inputted by the user
     * @return returns a boolean representing the validity of the answer
     */
    public boolean checkAnswer(char answer){
        return this.answer == answer || (char)((int)(this.answer+32)) == answer;
    }
}


```

```java
package GUIMenus;

import java.awt.*;

public class SelectionMenu extends JFrame {

    // Variables declaration - do not modify
    private JButton studentButton;
    private JButton teacherButton;
    // End of variables declaration

    public SelectionMenu() {
        initComponents();
    }

    /**
     * Method to setup all the GUI of the register screen.
     */
    private void initComponents() {

        studentButton = new JButton();
        teacherButton = new JButton();

        this.setTitle("Math Helper");
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

        studentButton.setText("Student Mode");
        studentButton.setBackground(new Color(51,153,255));
        studentButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                studentButtonActionPerformed(evt);
            }
        });

        teacherButton.setText("Teacher Mode");
        teacherButton.setBackground(new Color(51,153,255));
        teacherButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                teacherButtonActionPerformed(evt);
            }
        });

        //Setting up layout of the GUI
        GroupLayout layout = new GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(GroupLayout.Alignment.LEADING)
```

```
51              .addGroup(layout.createSequentialGroup()
52                  .addGap(37, 37, 37)
53                  .addComponent(studentButton, GroupLayout.PREFERRED_SIZE, 148,
  GroupLayout.PREFERRED_SIZE)
54                  .addGap(29, 29, 29)
55                  .addComponent(teacherButton, GroupLayout.PREFERRED_SIZE, 148,
  GroupLayout.PREFERRED_SIZE)
56                  .addContainerGap(38, Short.MAX_VALUE))
57          );
58          layout.setVerticalGroup(
59              layout.createParallelGroup(GroupLayout.Alignment.LEADING)
60              .addGroup(layout.createSequentialGroup()
61                  .addGap(102, 102, 102)
62
  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
63                      .addComponent(studentButton, GroupLayout.PREFERRED_SIZE,
  94, GroupLayout.PREFERRED_SIZE)
64                      .addComponent(teacherButton, GroupLayout.PREFERRED_SIZE,
  94, GroupLayout.PREFERRED_SIZE))
65                  .addContainerGap(104, Short.MAX_VALUE))
66          );
67
68          pack();
69      }
70
71      /**
72       * Method to open up student mode
73       * @param evt Detects when the button is pressed
74       */
75      private void studentButtonActionPerformed(ActionEvent evt) {
76         StudentMode sm = new StudentMode();
77         sm.setVisible(true);
78         sm.setLocationRelativeTo(null);
79         this.setVisible(false);
80         this.dispose();
81      }
82
83      /**
84       * Method to open up teahcer mode
85       * @param evt Detects when the button is pressed
86       */
87      private void teacherButtonActionPerformed(ActionEvent evt) {
88          TeacherMode tm = new TeacherMode();
89          tm.setVisible(true);
90          tm.setLocationRelativeTo(null);
91          tm.setResizable(false);
92          this.setVisible(false);
```

```
93          this.dispose();
94      }
95
96  }
97
```

```java
package GUIMenus;

import java.awt.*;

public class TeacherMode extends JFrame {

    // Variables declaration
    private JButton submitBut;
    private JButton backBut;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JLabel jLabel3;
    private JLabel jLabel4;
    private JLabel jLabel5;
    private JLabel jLabel6;
    private JLabel jLabel7;
    private JLabel jLabel9;
    private JList<String> jList1;
    private JRadioButton mcRadioBut;
    private JRadioButton saRadioBut;
    private JRadioButton geoRadioBut;
    private JRadioButton numRadioBut;
    private JRadioButton probRadioBut;
    private JRadioButton otherRadioBut;
    private JRadioButton algebraRadioBut;
    private JScrollPane jScrollPane1;
    private JScrollPane jScrollPane2;
    private JScrollPane jScrollPane3;
    private JTextArea jTextArea1;
    private JTextField jTextField1;
    private JTextField jTextField2;
    private JTextPane jTextPane1;
    private JOptionPane messagePane;
    private ButtonGroup g;
    // End of variables declaration


    public TeacherMode() {
        initComponents();
    }

    /**
     * Method to setup all the GUI of the register screen.
     */
    private void initComponents() {

        jScrollPane2 = new JScrollPane();
```

```
52          jTextPane1 = new JTextPane();
53          jLabel1 = new JLabel();
54          jLabel2 = new JLabel();
55          jLabel3 = new JLabel();
56          jScrollPane1 = new JScrollPane();
57          jTextArea1 = new JTextArea();
58          jTextField1 = new JTextField();
59          jLabel4 = new JLabel();
60          jScrollPane3 = new JScrollPane();
61          jList1 = new JList<>();
62          jLabel5 = new JLabel();
63          mcRadioBut = new JRadioButton();
64          saRadioBut = new JRadioButton();
65          jLabel6 = new JLabel();
66          jTextField2 = new JTextField();
67          jLabel7 = new JLabel();
68          jLabel9 = new JLabel();
69          geoRadioBut = new JRadioButton();
70          numRadioBut = new JRadioButton();
71          probRadioBut = new JRadioButton();
72          otherRadioBut = new JRadioButton();
73          algebraRadioBut = new JRadioButton();
74          submitBut = new JButton();
75          backBut = new JButton();
76          g = new ButtonGroup();
77
78          this.setTitle("Math Helper - Teacher Mode");
79          jScrollPane2.setViewportView(jTextPane1);
80
81          setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
82
83          jLabel1.setFont(new Font("Tahoma", 0, 14)); // NOI18N
84          jLabel1.setText("To insert a new question, please enter in the
    following fields as prompted");
85
86          jLabel2.setText("Problem Name:");
87
88          jLabel3.setText("Problem Statement:");
89
90          jTextArea1.setColumns(20);
91          jTextArea1.setRows(5);
92          jScrollPane1.setViewportView(jTextArea1);
93          jTextArea1.setLineWrap(true);
94          jTextArea1.setWrapStyleWord(true);
95
96
97          jLabel4.setText("Relative Problem Difficulty (Please select one the
```

```
   relative difficulty):");
 98
 99          jList1.setModel(new AbstractListModel<String>() {
100              String[] strings = { "1", "2", "3", "4", "5" };
101              public int getSize() { return strings.length; }
102              public String getElementAt(int i) { return strings[i]; }
103          });
104          jScrollPane3.setViewportView(jList1);
105
106          jLabel5.setText("Problem Type (Please select one):");
107
108          mcRadioBut.setText("Multiple Choice");
109
110          saRadioBut.setText("Short Answer");
111
112          g.add(mcRadioBut);
113          g.add(saRadioBut);
114
115          jLabel6.setText("Answer (Type in the correct choice for Multiple
   Choice questions and the full number for short answer questions)");
116
117          jLabel7.setText("Areas of Mathematics:");
118
119          jLabel9.setFont(new java.awt.Font("Tahoma", 0, 10)); // NOI18N
120          jLabel9.setText("Choose at least one of these areas in which you think
   the question falls under");
121
122          geoRadioBut.setText("Geometry");
123
124
125          numRadioBut.setText("Number Theory");
126
127
128          probRadioBut.setText("Probability");
129
130
131          otherRadioBut.setText("Other");
132
133
134          algebraRadioBut.setText("Algebra");
135
136
137          submitBut.setText("Submit");
138          submitBut.addActionListener(new ActionListener() {
139              public void actionPerformed(ActionEvent evt) {
140                  submitButActionPerformed(evt);
141              }
```

```
142         });
143         backBut.setText("Back");
144         backBut.addActionListener(new ActionListener() {
145             public void actionPerformed(ActionEvent evt) {
146                 backButActionPerformed(evt);
147             }
148         });
149 
150         //Setting up layout of the GUI
151         GroupLayout layout = new GroupLayout(getContentPane());
152         getContentPane().setLayout(layout);
153         layout.setHorizontalGroup(
154             layout.createParallelGroup(GroupLayout.Alignment.LEADING)
155             .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
156                 .addGap(31, 31, 31)
157 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING)
158                     .addGroup(layout.createSequentialGroup()
159                         .addComponent(jLabel9)
160 
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
161                         .addComponent(backBut)
162 
    .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
163                         .addComponent(submitBut))
164                     .addGroup(GroupLayout.Alignment.LEADING,
    layout.createSequentialGroup()
165                         .addComponent(jLabel7)
166 
    .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
167                         .addComponent(geoRadioBut)
168 
    .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
169                         .addComponent(numRadioBut)
170                         .addGap(10, 10, 10)
171                         .addComponent(probRadioBut)
172                         .addGap(10, 10, 10)
173                         .addComponent(algebraRadioBut)
174 
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
175                         .addComponent(otherRadioBut,
    GroupLayout.PREFERRED_SIZE, 61, GroupLayout.PREFERRED_SIZE))
176                     .addGroup(layout.createSequentialGroup()
177                         .addComponent(jLabel4)
```

```
178
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 316, Short.MAX_VALUE)
179                         .addComponent(jScrollPane3,
    GroupLayout.PREFERRED_SIZE, 70, GroupLayout.PREFERRED_SIZE))
180                     .addGroup(layout.createSequentialGroup()
181                         .addComponent(jLabel6)
182
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
183                         .addComponent(jTextField2, GroupLayout.PREFERRED_SIZE,
    102, GroupLayout.PREFERRED_SIZE))
184                     .addGroup(layout.createSequentialGroup()
185                         .addComponent(jLabel5)
186
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
187                         .addComponent(mcRadioBut)
188
    .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
189                         .addComponent(saRadioBut))
190                     .addGroup(layout.createSequentialGroup()
191
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
192                             .addComponent(jLabel3)
193                             .addComponent(jLabel2))
194
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
195                             .addGroup(layout.createSequentialGroup()
196                                 .addGap(26, 26, 26)
197                                 .addComponent(jTextField1))
198                             .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
199
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
200                                 .addComponent(jScrollPane1,
    GroupLayout.PREFERRED_SIZE, 318, GroupLayout.PREFERRED_SIZE))))
201                     .addGroup(GroupLayout.Alignment.LEADING,
    layout.createSequentialGroup()
202                         .addComponent(jLabel1)
203                         .addGap(0, 0, Short.MAX_VALUE)))
204                 .addGap(37, 37, 37))
205         );
206         layout.setVerticalGroup(
207             layout.createParallelGroup(GroupLayout.Alignment.LEADING)
208             .addGroup(layout.createSequentialGroup()
209                 .addGap(37, 37, 37)
```

```
210                 .addComponent(jLabel1)
211                 .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
212 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
213                     .addComponent(jLabel2)
214                     .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE,
    GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
215                 .addGap(18, 18, 18)
216 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
217                     .addComponent(jLabel3)
218                     .addComponent(jScrollPane1, GroupLayout.PREFERRED_SIZE,
    GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
219                 .addGap(18, 18, 18)
220 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
221                     .addComponent(jLabel4)
222                     .addComponent(jScrollPane3, GroupLayout.PREFERRED_SIZE,
    84, GroupLayout.PREFERRED_SIZE))
223                 .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 26,
    Short.MAX_VALUE)
224 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
225                     .addComponent(jLabel5)
226                     .addComponent(mcRadioBut)
227                     .addComponent(saRadioBut))
228                 .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
229 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
230                     .addComponent(jTextField2, GroupLayout.PREFERRED_SIZE,
    GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
231                     .addComponent(jLabel6))
232                 .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
233 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
234                     .addComponent(jLabel7)
235                     .addComponent(geoRadioBut)
236                     .addComponent(numRadioBut)
237                     .addComponent(probRadioBut)
238                     .addComponent(otherRadioBut)
239                     .addComponent(algebraRadioBut))
240                 .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
241 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
242 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
243                         .addComponent(submitBut)
```

```
244                         .addComponent(backBut))
245                     .addComponent(jLabel9))
246                 .addGap(5, 5, 5))
247         );
248 
249         pack();
250     }
251 
252     /**
253      * Method that checks the validity of the inputs for the problem and writes the problem in the text file if it is valid
254      * @param evt Detects when the button is pressed
255      */
256     private void submitButActionPerformed(ActionEvent evt){
257         //Need function to check if name is duplicate
258         String name = jTextField1.getText();
259         String problem = jTextArea1.getText();
260         String answer = jTextField2.getText();
261         String areas = "";
262         String type = "";
263         String difficulty = "";
264         if(mcRadioBut.isSelected()){
265             type = "MC";
266         }
267         else if (saRadioBut.isSelected()){
268             type = "SA";
269         }
270         else{
271             messagePane = new JOptionPane();
272             JFrame f = new JFrame();
273             JOptionPane.showMessageDialog(f,"Please select a problem type");
274             return;
275         }
276         if (type.equals("MC")){
277             if (answer.length() != 1){
278                 messagePane = new JOptionPane();
279                 JFrame f = new JFrame();
280                 JOptionPane.showMessageDialog(f,"The answer should be a singular character representing the letter that corresponds with the correct solution.");
281                 return;
282             }
283         }
284         int temp = jList1.getSelectedIndex()+1;
285         difficulty += temp;
286 
287         if (difficulty.equals("")){
```

```
288             messagePane = new JOptionPane();
289             JFrame f = new JFrame();
290             JOptionPane.showMessageDialog(f,"Please select the problem's
    difficulty");
291             return;
292         }
293 
294         areas = areasSelected();
295         if (areas.equals("")){
296             messagePane = new JOptionPane();
297             JFrame f = new JFrame();
298             JOptionPane.showMessageDialog(f,"Please select at least one
    problem area");
299             return;
300         }
301 
302         try{
303             BufferedWriter writer = new BufferedWriter(new FileWriter("Text
    Files/problems.txt", true));
304             writer.newLine();
305             writer.append("/");
306             writer.newLine();
307             writer.append(name);
308             writer.newLine();
309             writer.append(problem);
310             writer.newLine();
311             writer.append("*");
312             writer.newLine();
313             writer.append(difficulty);
314             writer.newLine();
315             writer.append(areas);
316             writer.newLine();
317             writer.append(type);
318             writer.newLine();
319             writer.append(answer);
320             writer.close();
321             messagePane = new JOptionPane();
322             JFrame f = new JFrame();
323             JOptionPane.showMessageDialog(f,"Success");
324         }
325         catch (IOException e){
326             messagePane = new JOptionPane();
327             JFrame f = new JFrame();
328             JOptionPane.showMessageDialog(f,"An Unexpected Error has
    occured");
329         }
330 
```

```
331
332     }
333
334     /**
335      * Method that helps generate the string that contains all of the selected
    Mathematical areas
336      * @return
337      */
338     public String areasSelected() {
339         String areas = "";
340         if (geoRadioBut.isSelected()){
341             areas += "Geometry,";
342         }
343         if (numRadioBut.isSelected()){
344             areas += "Number Theory,";
345         }
346         if (probRadioBut.isSelected()){
347             areas += "Probability,";
348         }
349         if (algebraRadioBut.isSelected()){
350             areas += "Algebra,";
351         }
352         if (otherRadioBut.isSelected()){
353             areas += "Other,";
354         }
355         return areas;
356     }
357
358     /**
359      * Method that helps go back to the SelectionMenu
360      * @param evt Detects when the button is pressed
361      */
362     private void backButActionPerformed(ActionEvent evt){
363         SelectionMenu sm = new SelectionMenu();
364         this.setVisible(false);
365         sm.setLocationRelativeTo(null);
366         sm.setVisible(true);
367     }
368
369 }
```

```java
package GUIMenus;

import java.awt.*;

public class StudentMode extends JFrame {

    // Variables declaration - do not modify
    private JButton searchButton;
    private JButton sortButton;
    private JButton generateButton;
    private JButton backBut;
    private JRadioButton sortName;
    private JRadioButton sortDiff;
    private JTextField jTextField1;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JLabel jLabel3;
    private JOptionPane jOptionPane1;
    private JScrollPane jScrollPane1;
    private JTable jTable1;
    private ButtonGroup g;
    private LinkedList problems;
    private LinkedList allProblems = new LinkedList();
    // End of variables declaration

    /**
     * Creates new form StudentMode
     */
    public StudentMode() {
        storeElements();
        initComponents();
    }


    /**
     * Method to setup all the GUI of the register screen.
     */
    private void initComponents() {
        this.setTitle("Math Helper - Student Mode");

        jScrollPane1 = new JScrollPane();
        searchButton = new JButton();
        jTextField1 = new JTextField();
        sortButton = new JButton();
        sortName = new JRadioButton();
        sortDiff = new JRadioButton();
        generateButton = new JButton();
```

```
58          backBut = new JButton();
59          problems = allProblems;
60          g = new ButtonGroup();
61
62          jLabel1 = new JLabel();
63          jLabel2 = new JLabel();
64          jLabel3 = new JLabel();
65          setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
66          initializeTable();
67          fillInTable();
68          jLabel1.setFont(new Font("Tahoma", 0, 16));
69          jLabel1.setText("List of Problems");
70          jLabel2.setFont(new Font("Tahoma", 0, 10));
71          jLabel2.setText("Double click the problem name to open up a problem");
72          jLabel3.setFont(new Font("Tahoma", 0, 10));
73          jLabel3.setText("Select the sort option and click sort");
74
75          searchButton.setText("Search");
76          searchButton.addActionListener(new ActionListener() {
77              public void actionPerformed(ActionEvent evt) {
78                  searchButtonActionPerformed(evt);
79              }
80          });
81
82          sortButton.setText("Sort");
83          sortButton.addActionListener(new ActionListener() {
84              public void actionPerformed(ActionEvent evt) {
85                  sortButtonActionPerformed(evt);
86              }
87          });
88
89
90          sortName.setText("Sort by Name");
91
92          sortDiff.setText("Sort by Difficulty");
93
94          g.add(sortName);
95          g.add(sortDiff);
96
97          generateButton.setText("Generate Problem Sets");
98          generateButton.addActionListener(new ActionListener() {
99              public void actionPerformed(ActionEvent evt) {
100                 generateButtonActionPerformed(evt);
101             }
102         });
103
104         backBut.setText("Back");
```

```
105         backBut.addActionListener(new ActionListener() {
106             public void actionPerformed(ActionEvent evt) {
107                 backButActionPerformed(evt);
108             }
109         });
110         //Setting up layout of the GUI
111        GroupLayout layout = new GroupLayout(getContentPane());
112         getContentPane().setLayout(layout);
113         layout.setHorizontalGroup(
114             layout.createParallelGroup(GroupLayout.Alignment.LEADING)
115             .addGroup(layout.createSequentialGroup()
116                 .addContainerGap()
117 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
118                     .addGroup(layout.createSequentialGroup()
119 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
120                         .addComponent(jScrollPane1)
121                         .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
122 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
123                             .addGroup(layout.createSequentialGroup()
124                                 .addComponent(searchButton)
125 
    .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
126                                 .addComponent(jTextField1,
    GroupLayout.PREFERRED_SIZE, 205, GroupLayout.PREFERRED_SIZE)
127 
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
128                             .addGroup(layout.createSequentialGroup()
129                                 .addComponent(generateButton)
130 
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
131 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
132                             .addGroup(layout.createSequentialGroup()
133                                 .addGap(192, 192, 192)
134                                 .addComponent(backBut))
135                             .addGroup(layout.createSequentialGroup()
136                                 .addComponent(sortButton)
137 
    .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
138                                 .addComponent(sortName)
139 
```

```
    .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
140                                             .addComponent(sortDiff)))))
141                         .addContainerGap())
142                     .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
143                         .addComponent(jLabel2)
144
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
145                         .addComponent(jLabel3)
146                         .addContainerGap())))
147             .addGroup(layout.createSequentialGroup()
148                 .addGap(260, 260, 260)
149                 .addComponent(jLabel1)
150                 .addGap(0, 0, Short.MAX_VALUE))
151         );
152         layout.setVerticalGroup(
153             layout.createParallelGroup(GroupLayout.Alignment.LEADING)
154             .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
155                 .addContainerGap(26, Short.MAX_VALUE)
156                 .addComponent(jLabel1)
157                 .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
158
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
159                     .addComponent(generateButton)
160                     .addComponent(backBut))
161                 .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
162
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
163                     .addComponent(searchButton)
164                     .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE,
    GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
165                     .addComponent(sortButton)
166                     .addComponent(sortName)
167                     .addComponent(sortDiff))
168                 .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
169
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
170                     .addComponent(jLabel3)
171                     .addComponent(jLabel2))
172                 .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
173                 .addComponent(jScrollPane1, GroupLayout.PREFERRED_SIZE, 370,
    GroupLayout.PREFERRED_SIZE)
174                 .addGap(21, 21, 21))
175         );
176
```

```
177         pack();
178     }
179 
180     /**
181      * Method that initializes and creates the table
182      */
183     public void initializeTable(){
184         jTable1 = new JTable(){
185             public boolean isCellEditable(int row, int column){
186                 return false;
187             }
188         };
189         jTable1.setModel(new DefaultTableModel(
190             new Object [][] {
191 
192             },
193             new String [] {
194                 "Problem Name", "Problem Area", "Relative Difficulty",
    "Problem Type"
195             }
196         ));
197 
198         jScrollPane1.setViewportView(jTable1);
199         if (jTable1.getColumnModel().getColumnCount() > 0) {
200             jTable1.getColumnModel().getColumn(0).setMinWidth(200);
201             jTable1.getColumnModel().getColumn(0).setPreferredWidth(200);
202             jTable1.getColumnModel().getColumn(1).setMinWidth(200);
203             jTable1.getColumnModel().getColumn(1).setPreferredWidth(200);
204         }
205 
206         jTable1.addMouseListener(new MouseAdapter(){
207             public void mousePressed(MouseEvent e){
208                 if (e.getClickCount() == 2){
209                     JTable target = (JTable)e.getSource();
210                     int row = target.getSelectedRow();
211                     int column = target.getSelectedColumn();
212                     if (column == 0){
213                         Node n = problems.getIndex(row, 0,
    problems.getHead());
214                         QuestionDisplay q;
215                         if (n.getStore() instanceof SAProblem)
216                             q = new QuestionDisplay((SAProblem)
    (n.getStore()));
217                         else
218                             q = new QuestionDisplay((MCProblem)
    (n.getStore()));
219                         q.setVisible(true);
```

```
220                         q.setLocationRelativeTo(null);
221                     }
222                 }
223             }
224         });
225     }
226 
227     /**
228      * Method that fills in the table
229      */
230     public void fillInTable(){
231         DefaultTableModel tableModel = (DefaultTableModel) jTable1.getModel();
232         String [] data = new String[4];
233         Node temp = problems.getHead();
234         for (int i = 0; i < problems.getSize(); i ++){
235             if (temp.getStore() instanceof MCProblem){
236                 data[0] = ((MCProblem)(temp.getStore())).getName();
237                 data[1] = ((MCProblem)(temp.getStore())).getTypes();
238                 data[2] = ((MCProblem)(temp.getStore())).getDifficulty();
239                 data[3] = "MC";
240                 tableModel.addRow(data);
241             }
242             else{
243                 data[0] = ((SAProblem)(temp.getStore())).getName();
244                 data[1] = ((SAProblem)(temp.getStore())).getTypes();
245                 data[2] = ((SAProblem)(temp.getStore())).getDifficulty();
246                 data[3] = "SA";
247                 tableModel.addRow(data);
248             }
249 
250             temp = temp.getNext();
251         }
252         revalidate();
253         repaint();
254     }
255 
256     /**
257      * Method that transfers all the problems from the text file into a
    LinkedList
258      */
259     public void storeElements(){
260         try{
261             BufferedReader reader = new BufferedReader(new FileReader("Text
    Files/problems.txt"));
262             String line = reader.readLine();
263             while (line != null && !line.equals("")){
264                 String name = reader.readLine();
```

```
265                 String statement = "";
266                 String temp = reader.readLine();
267                 while (temp != null && !temp.equals("*")){
268                     if (!statement.equals(""))
269                         statement += ("@" + temp);
270                     else
271                         statement += temp;
272                     temp = reader.readLine();
273                 }
274                 String difficulty = reader.readLine();
275                 String area = reader.readLine();
276                 String type = reader.readLine();
277                 String answerSA;
278                 char answerMC;
279                 MCProblem probMC;
280                 SAProblem probSA;
281                 Node n;
282                 if (type.equals("SA")){
283                     answerSA = reader.readLine();
284                     probSA = new SAProblem(name, statement, difficulty,
    answerSA, area);
285                     n = new Node(probSA);
286                     allProblems.addFirst(n);
287                 }
288                 else{
289                     String hold = reader.readLine();
290                     answerMC = hold.charAt(0);
291                     probMC = new MCProblem(name, statement, difficulty,
    answerMC, area);
292                     n = new Node(probMC);
293                     allProblems.addFirst(n);
294                 }
295                 line = reader.readLine();
296             }
297             reader.close();
298         }
299         catch (IOException e){
300             jOptionPane1 = new JOptionPane();
301             JFrame f = new JFrame();
302             JOptionPane.showMessageDialog(f,"An Unexpected Error has
    occured");
303         }
304 
305     }
306 
307     /**
308      * Method which helps display the new table and initializes a new
```

```
   LinkedList containing only the search results wanted
309      * @param evt Detects when the button is pressed
310      */
311     private void searchButtonActionPerformed(ActionEvent evt){
312         String search = jTextField1.getText();
313         if (search == null || search.equals(""))
314             problems = allProblems;
315         else{
316             problems = new LinkedList();
317             Node n = allProblems.getHead();
318             String curr = "";
319             for (int i = 0; i < allProblems.getSize(); i ++){
320                 if (n.getStore() instanceof SAProblem)
321                     curr = ((SAProblem)(n.getStore())).getName();
322                 else
323                     curr = ((MCProblem)(n.getStore())).getName();
324                 if (curr.toLowerCase().contains(search.toLowerCase()))
325                     problems.addFirst(new Node(n.getStore()));
326                 n = n.getNext();
327             }
328         }
329         initializeTable();
330         fillInTable();
331     }
332 
333     /**
334      * Method which checks to see what kind of sort should be performed and
   also executes the sort
335      * @param evt Detects when the button is pressed
336      */
337     private void sortButtonActionPerformed(ActionEvent evt){
338         boolean selected = false;
339         if (sortName.isSelected()){
340             selected = true;
341             problems.sortAlpha();
342         }
343         else if (sortDiff.isSelected()){
344             selected = true;
345             problems.sortNum();
346         }
347         if (selected){
348             initializeTable();
349             fillInTable();
350         }
351     }
352 
353     /**
```

```
354      * Method which calls on the generate QuestionsetMenu class
355      * @param evt Detects when the button is pressed
356      */
357     private void generateButtonActionPerformed(ActionEvent evt){
358         QuestionsetMenu questions = new QuestionsetMenu(allProblems);
359         questions.setVisible(true);
360         questions.setLocationRelativeTo(null);
361     }
362 
363     /**
364      * Method which disposes of the current jFrame and reverts back to the
    SelectionMenu jFrame
365      * @param evt Detects when the button is pressed
366      */
367     private void backButActionPerformed(ActionEvent evt) {
368         this.setVisible(false);
369         this.dispose();
370         SelectionMenu sm = new SelectionMenu();
371         sm.setVisible(true);
372         sm.setLocationRelativeTo(null);
373         sm.setResizable(false);
374     }
375 
376 }
377 
```

```java
package GUIMenus;

import java.awt.*;

public class QuestionDisplay extends JFrame {

     // Variables declaration - do not modify
    private JButton doneButton;
    private JButton showAnsBut;
    private JButton checkAnsBut;
    private JLabel jLabel1;
    private JScrollPane jScrollPane1;
    private JTextArea jTextArea1;
    private JTextField jTextField1;
    private JTextField jTextField2;
    private MCProblem probMC;
    private SAProblem probSA;
    private JOptionPane jOptionPane1;
    private boolean indic = false;
    // End of variables declaration


    public QuestionDisplay(MCProblem prob) {
        probMC = prob;
        indic = false;
        initComponents();
    }

    public QuestionDisplay(SAProblem prob){
        probSA = prob;
        indic = true;
        initComponents();
    }

    /**
     * Method to setup all the GUI of the register screen.
     */
    private void initComponents() {

        doneButton = new JButton();
        showAnsBut = new JButton();
        jTextField1 = new JTextField();
        checkAnsBut = new JButton();
        jTextField2 = new JTextField();
        jScrollPane1 = new JScrollPane();
        jTextArea1 = new JTextArea();
        jLabel1 = new JLabel();
```

```
53
54          this.setTitle("Math Helper");
55          setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
56
57          doneButton.setText("Done");
58          doneButton.addActionListener(new ActionListener() {
59              public void actionPerformed(ActionEvent evt) {
60                  doneButtonActionPerformed(evt);
61              }
62          });
63
64          showAnsBut.setText("Show Answer");
65          showAnsBut.addActionListener(new ActionListener() {
66              public void actionPerformed(ActionEvent evt) {
67                  showAnsButActionPerformed(evt);
68              }
69          });
70
71          jTextField1.setEditable(false);
72
73          checkAnsBut.setText("Check Answer");
74          checkAnsBut.addActionListener(new ActionListener() {
75              public void actionPerformed(ActionEvent evt) {
76                  checkAnsButActionPerformed(evt);
77              }
78          });
79
80          jTextArea1.setEditable(false);
81          jTextArea1.setColumns(20);
82          jTextArea1.setRows(5);
83          jScrollPane1.setViewportView(jTextArea1);
84
85          jLabel1.setText("Enter Answer Below");
86
87          displayProblem();
88          jTextArea1.setLineWrap(true);
89          jTextArea1.setWrapStyleWord(true);
90
91          //Setting up layout of the GUI
92          GroupLayout layout = new GroupLayout(getContentPane());
93          getContentPane().setLayout(layout);
94          layout.setHorizontalGroup(
95              layout.createParallelGroup(GroupLayout.Alignment.LEADING)
96              .addGroup(layout.createSequentialGroup()
97                  .addContainerGap()
98
  .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
```

```
 99                      .addGroup(layout.createSequentialGroup()
100                          .addComponent(checkAnsBut)
101 
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
102 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
103                              .addGroup(layout.createSequentialGroup()
104                                  .addComponent(jLabel1)
105                                  .addGap(0, 0, Short.MAX_VALUE))
106                              .addGroup(layout.createSequentialGroup()
107                                  .addComponent(jTextField2,
    GroupLayout.PREFERRED_SIZE, 78, GroupLayout.PREFERRED_SIZE)
108 
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 86, Short.MAX_VALUE)
109                                  .addComponent(showAnsBut)
110 
    .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
111                                  .addComponent(doneButton)))
112                          .addContainerGap())
113                      .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
114                          .addGap(0, 0, Short.MAX_VALUE)
115 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING, false)
116                              .addComponent(jTextField1)
117                              .addComponent(jScrollPane1,
    GroupLayout.DEFAULT_SIZE, 341, Short.MAX_VALUE))
118                          .addGap(57, 57, 57))))
119         );
120         layout.setVerticalGroup(
121             layout.createParallelGroup(GroupLayout.Alignment.LEADING)
122             .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
123                 .addGap(31, 31, 31)
124                 .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE,
    GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
125                 .addGap(18, 18, 18)
126                 .addComponent(jScrollPane1, GroupLayout.DEFAULT_SIZE, 166,
    Short.MAX_VALUE)
127                 .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
128                 .addComponent(jLabel1)
129                 .addGap(3, 3, 3)
130 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
131                     .addComponent(doneButton)
132                     .addComponent(showAnsBut)
133                     .addComponent(checkAnsBut)
```

```
134                     .addComponent(jTextField2, GroupLayout.PREFERRED_SIZE,
   GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
135                 .addContainerGap())
136         );
137 
138         pack();
139     }
140 
141     /**
142      * Method that displays the problem
143      */
144     public void displayProblem(){
145         if (indic){
146             String name = probSA.getName();
147             String statement = formatStatement(probSA.getProblem(), 0);
148             jTextField1.setText(name);
149             jTextArea1.setText(statement);
150         }
151         else{
152             String name = probMC.getName();
153             String statement = formatStatement(probMC.getProblem(), 0);
154             jTextField1.setText(name);
155             jTextArea1.setText(statement);
156         }
157     }
158 
159     /**
160      * Method that recursively formats the problem statement so that the line
   spacing is correct
161      * @param statement the problem statement itself
162      * @param index the index that the method is currently walking through
163      * @return
164      */
165     public String formatStatement(String statement, int index){
166         if (index == statement.length())
167             return statement;
168         if (statement.charAt(index) == '@')
169             return
   formatStatement(statement.substring(0,index)+"\n"+statement.substring(index+1)
   , index+1);
170         return formatStatement(statement, index+1);
171     }
172 
173     /**
174      * Method that disposes of the current jFrame
175      * @param evt Detects when the button is pressed
176      */
```

```
177     private void doneButtonActionPerformed(ActionEvent evt) {
178         this.setVisible(false);
179         this.dispose();
180     }
181 
182     /**
183      * Method that displays the correct answer
184      * @param evt Detects when the button is pressed
185      */
186     private void showAnsButActionPerformed(ActionEvent evt) {
187         jOptionPane1 = new JOptionPane();
188         JFrame f = new JFrame();
189         if (indic)
190             JOptionPane.showMessageDialog(f,"The answer is " +
    probSA.getAnswer());
191         else
192             JOptionPane.showMessageDialog(f,"The answer is " +
    probMC.getAnswer());
193     }
194 
195     /**
196      * Method that checks if the user given answer is correct and displays to
    the user if it is correct or not
197      * @param evt Detects when the button is pressed
198      */
199     private void checkAnsButActionPerformed(ActionEvent evt) {
200         //Check Answer method
201         String answer;
202         if (!jTextField2.getText().trim().equals(""))
203             answer = jTextField2.getText().trim();
204         else
205             return;
206         //Check for short answer
207         if (indic){
208             if (probSA.checkAnswer(answer)){
209                 jOptionPane1 = new JOptionPane();
210                 JFrame f = new JFrame();
211                 JOptionPane.showMessageDialog(f,"Correct");
212             }
213             else{
214                 jOptionPane1 = new JOptionPane();
215                 JFrame f = new JFrame();
216                 JOptionPane.showMessageDialog(f,"Incorrect");
217             }
218         }
219         //Check for multiple choice
220         else{
```

```
221             if (answer.length() > 1){
222                 jOptionPane1 = new JOptionPane();
223                 JFrame f = new JFrame();
224                 JOptionPane.showMessageDialog(f,"Invalid answer please type in
    a letter that corresponds with one of the choices");
225                 return;
226             }
227             if (probMC.checkAnswer(answer.charAt(0))){
228                 jOptionPane1 = new JOptionPane();
229                 JFrame f = new JFrame();
230                 JOptionPane.showMessageDialog(f,"Correct");
231             }
232             else{
233                 jOptionPane1 = new JOptionPane();
234                 JFrame f = new JFrame();
235                 JOptionPane.showMessageDialog(f,"Incorrect");
236             }
237         }
238     }
239
240
241
242 }
```

```java
package GUIMenus;

import javax.swing.*;

public class QuestionsetMenu extends JFrame {

    // Variables declaration - do not modify
    private JButton cancelBut;
    private JButton doneBut;
    private JLabel jLabel1;
    private JLabel jLabel2;
    private JLabel jLabel3;
    private JLabel jLabel4;
    private JRadioButton mcRadBut;
    private JRadioButton saRadBut;
    private JRadioButton geoRadBut;
    private JRadioButton numRadBut;
    private JRadioButton probRadBut;
    private JRadioButton otherRadBut;
    private JRadioButton algebraRadBut;
    private JTextField jTextField1;
    private JTextField jTextField2;
    private JOptionPane jOptionPane1;
    private ButtonGroup g;
    private LinkedList allProblems;
    // End of variables declaration

    public QuestionsetMenu(){
    }

    public QuestionsetMenu(LinkedList allProblems) {
        this.allProblems = allProblems;
        initComponents();
    }

    /**
     * Method to setup all the GUI of the register screen.
     */
    private void initComponents() {

        jLabel1 = new JLabel();
        jTextField1 = new JTextField();
        mcRadBut = new JRadioButton();
        saRadBut = new JRadioButton();
        cancelBut = new JButton();
        doneBut = new JButton();
        jTextField2 = new JTextField();
```

```
56          jLabel2 = new JLabel();
57          jLabel3 = new JLabel();
58          jLabel4 = new JLabel();
59          geoRadBut = new JRadioButton();
60          numRadBut = new JRadioButton();
61          probRadBut = new JRadioButton();
62          otherRadBut = new JRadioButton();
63          algebraRadBut = new JRadioButton();
64          g = new ButtonGroup();
65
66          setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
67
68          jLabel1.setFont(new Font("Tahoma", 0, 12)); // NOI18N
69          jLabel1.setText("Problem Set Generator Filter");
70
71          mcRadBut.setText("Multiple Choice Only");
72
73          saRadBut.setText("Short Answer Only");
74
75          g.add(mcRadBut);
76          g.add(saRadBut);
77
78          cancelBut.setText("Cancel");
79          cancelBut.addActionListener(new ActionListener() {
80              public void actionPerformed(ActionEvent evt) {
81                  cancelButActionPerformed(evt);
82              }
83          });
84
85          doneBut.setText("Done");
86          doneBut.addActionListener(new ActionListener() {
87              public void actionPerformed(ActionEvent evt) {
88                  doneButActionPerformed(evt);
89              }
90          });
91
92          jLabel2.setFont(new Font("Tahoma", 0, 10)); // NOI18N
93          jLabel2.setText("Restricts difficulty to the inputted difficulty");
94
95          jLabel3.setText("Number of Problems");
96
97          jLabel4.setText("Difficulty Restriction");
98
99          geoRadBut.setText("Geometry");
100
101         numRadBut.setText("Number Theory");
102
```

```
103         probRadBut.setText("Probability");
104 
105         otherRadBut.setText("Other");
106 
107         algebraRadBut.setText("Algebra");
108 
109         //Setting up layout of the GUI
110         GroupLayout layout = new GroupLayout(getContentPane());
111         getContentPane().setLayout(layout);
112         layout.setHorizontalGroup(
113             layout.createParallelGroup(GroupLayout.Alignment.LEADING)
114             .addGroup(layout.createSequentialGroup()
115 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
116                     .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
117                         .addGap(19, 19, 19)
118 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
119                             .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
120                                 .addComponent(jLabel3)
121 
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
122                                 .addComponent(jTextField1,
    GroupLayout.PREFERRED_SIZE, 120, GroupLayout.PREFERRED_SIZE))
123                             .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
124                                 .addComponent(jLabel4)
125 
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED,
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
126                                 .addComponent(jTextField2,
    GroupLayout.PREFERRED_SIZE, 120, GroupLayout.PREFERRED_SIZE))))
127                     .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
128                         .addContainerGap(10, Short.MAX_VALUE)
129 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
130                             .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
131                                 .addComponent(mcRadBut)
132 
    .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
133                                 .addComponent(saRadBut))
134                             .addComponent(jLabel2,
```

```
    GroupLayout.Alignment.TRAILING)
135                             .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
136                                 .addComponent(cancelBut)
137
    .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
138                                 .addComponent(doneBut))
139                             .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
140
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
141                                     .addGroup(GroupLayout.Alignment.TRAILING,
    layout.createSequentialGroup()
142                                         .addComponent(geoRadBut)
143                                         .addGap(18, 18, 18))
144                                     .addGroup(layout.createSequentialGroup()
145                                         .addComponent(probRadBut)
146                                         .addGap(16, 16, 16)))
147
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
148                                     .addComponent(algebraRadBut)
149                                     .addGroup(layout.createSequentialGroup()
150                                         .addComponent(numRadBut)
151                                         .addGap(18, 18, 18)
152                                         .addComponent(otherRadBut)))
153                                 .addGap(8, 8, 8)))))
154                 .addContainerGap())
155             .addGroup(layout.createSequentialGroup()
156                 .addGap(63, 63, 63)
157                 .addComponent(jLabel1)
158                 .addContainerGap(GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
159         );
160         layout.setVerticalGroup(
161             layout.createParallelGroup(GroupLayout.Alignment.LEADING)
162             .addGroup(layout.createSequentialGroup()
163                 .addContainerGap()
164                 .addComponent(jLabel1)
165                 .addGap(11, 11, 11)
166
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
167                     .addComponent(jTextField1, GroupLayout.PREFERRED_SIZE,
    GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE)
168                     .addComponent(jLabel3))
169                 .addGap(18, 18, 18)
170
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.TRAILING)
171                     .addComponent(jLabel4)
```

```
172                     .addComponent(jTextField2, GroupLayout.PREFERRED_SIZE,
    GroupLayout.DEFAULT_SIZE, GroupLayout.PREFERRED_SIZE))
173                 .addGap(5, 5, 5)
174                 .addComponent(jLabel2)
175                 .addGap(13, 13, 13)
176 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
177                     .addComponent(mcRadBut)
178                     .addComponent(saRadBut))
179                 .addGap(26, 26, 26)
180 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
181                     .addComponent(geoRadBut)
182                     .addComponent(numRadBut)
183                     .addComponent(otherRadBut))
184                 .addPreferredGap(LayoutStyle.ComponentPlacement.UNRELATED)
185 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
186                     .addComponent(probRadBut)
187                     .addComponent(algebraRadBut))
188                 .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED, 3,
    Short.MAX_VALUE)
189 
    .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
190                     .addComponent(doneBut)
191                     .addComponent(cancelBut)))
192         );
193 
194         pack();
195     }
196 
197     /**
198      * Method that disposes of this jFrame
199      * @param evt Detects when the button is pressed
200      */
201     private void cancelButActionPerformed(ActionEvent evt){
202         this.setVisible(false);
203         dispose();
204     }
205 
206     /**
207      * Method that checks if the user's inputs are valid and the overall
    method that runs the problem set generation process
208      * @param evt Detects when the button is pressed
209      */
210     private void doneButActionPerformed(ActionEvent evt){
211         String problems = jTextField1.getText();
```

```
212         String difficulty = jTextField2.getText();
213         LinkedList areas = new LinkedList();
214         boolean mc = true;
215         boolean sa = true;
216         boolean selectedType = false;
217         if (difficulty.equals("") || difficulty == null)
218             difficulty = "5";
219         if (mcRadBut.isSelected())
220             sa = false;
221         else if (saRadBut.isSelected())
222             mc = false;
223         if (geoRadBut.isSelected()){
224             Node n = new Node("Geometry");
225             selectedType = true;
226             areas.addFirst(n);
227         }
228         if (numRadBut.isSelected()){
229             Node n = new Node("Number Theory");
230             selectedType = true;
231             areas.addFirst(n);
232         }
233         if (probRadBut.isSelected()){
234             Node n = new Node("Probability");
235             selectedType = true;
236             areas.addFirst(n);
237         }
238         if (otherRadBut.isSelected()){
239             Node n = new Node("Other");
240             selectedType = true;
241             areas.addFirst(n);
242         }
243         if (algebraRadBut.isSelected()){
244             Node n = new Node("Algebra");
245             selectedType = true;
246             areas.addFirst(n);
247         }
248         if (!selectedType)
249             areas = includeAll(areas);
250         int numProb = Integer.parseInt(problems);
251         Node curr = allProblems.getHead();
252         MCProblem probMC = new MCProblem();
253         SAProblem probSA = new SAProblem();
254         LinkedList usuableProbs = new LinkedList();
255         for (int i = 0; i < allProblems.getSize(); i ++){
256             if (mc && sa){
257                 if (curr.getStore() instanceof MCProblem){
258                     probMC = (MCProblem)(curr.getStore());
```

```
259                     boolean legible = checkLegibility(areas, difficulty,
    probMC);
260                     if (legible)
261                         usuableProbs.addFirst(new Node(probMC));
262                 }
263                 else{
264                     probSA = (SAProblem)(curr.getStore());
265                     boolean legible = checkLegibility(areas, difficulty,
    probSA);
266                     if (legible)
267                         usuableProbs.addFirst(new Node(probSA));
268                 }
269             }
270             else if (sa){
271                 if (curr.getStore() instanceof SAProblem){
272                     probSA = (SAProblem)(curr.getStore());
273                     boolean legible = checkLegibility(areas, difficulty,
    probSA);
274                     if (legible)
275                         usuableProbs.addFirst(new Node(probSA));
276                 }
277             }
278             else{
279                 if (curr.getStore() instanceof MCProblem){
280                     probMC = (MCProblem)(curr.getStore());
281                     boolean legible = checkLegibility(areas, difficulty,
    probMC);
282                     if (legible)
283                         usuableProbs.addFirst(new Node(probMC));
284                 }
285             }
286             curr = curr.getNext();
287         }
288         if (numProb > usuableProbs.getSize()){
289             jOptionPane1 = new JOptionPane();
290             JFrame f = new JFrame();
291             JOptionPane.showMessageDialog(f,"There are not enough problems to
    support your request");
292             return;
293         }
294         this.setVisible(false);
295         generateProblems(numProb, usuableProbs);
296     }
297 
298     /**
299      * Method that stores all of the problem types into the LinkedList
300      * @param areas LinkedList that stores the problem areas
```

```
301      * @return returns the given LinkedList with the appropriate additions
302      */
303     private LinkedList includeAll(LinkedList areas){
304         areas.addFirst(new Node("Geometry"));
305         areas.addFirst(new Node("Number Theory"));
306         areas.addFirst(new Node("Algebra"));
307         areas.addFirst(new Node("Other"));
308         areas.addFirst(new Node("Probability"));
309         return areas;
310     }
311 
312     /**
313      * Method that generates random problems based on the given LinkedList
314      * @param numProb number of problems user wants to generate
315      * @param usuableProbs number of problems that fufill the users problem
    criterias
316      */
317     private void generateProblems(int numProb, LinkedList usuableProbs){
318         QuestionDisplay question;
319         for (int i = 0; i < numProb; i ++){
320             //Generate a random number from 0 to the size of usuableProbs-1
321             int random = (int)(Math.random()*usuableProbs.getSize());
322 
323             //Get a node from that index
324             Node randNode = usuableProbs.getIndexHelper(random);
325 
326             //Use the QuestionDisplay class to display the question which
    could be either multiple choice style or short answer
327             if (randNode.getStore() instanceof MCProblem)
328                 question = new QuestionDisplay((MCProblem)
    (randNode.getStore()));
329             else
330                 question = new QuestionDisplay((SAProblem)
    (randNode.getStore()));
331 
332             question.setVisible(true);
333             question.setLocationRelativeTo(null);
334             //Delete the node storing the question as said question has
    already been used
335             usuableProbs.delete(randNode);
336         }
337     }
338 
339     /**
340      * Method that checks the legibility of a particular problem (Method
    overloading with the other checkLegibility method)
341      * @param areas the problem areas the user wants
```

```
342     * @param difficulty the difficulty limitation of the problem the user
    wants
343     * @param probMC the singular problem itself
344     * @return a boolean variable that is true if the problem meets the user
    requirements false if otherwise
345     */
346    private boolean checkLegibility(LinkedList areas, String difficulty,
    MCProblem probMC){
347        if (Integer.parseInt(probMC.getDifficulty()) >
    Integer.parseInt(difficulty)){
348            return false;
349        }
350        LinkedList probAreas = probMC.getAreas();
351        Node n = probAreas.getHead();
352        for (int i = 0; i < probAreas.getSize(); i ++){
353            String currType = (String)(n.getStore());
354            boolean isValid = containsType(areas, currType);
355            if (!isValid)
356                return false;
357        }
358        return true;
359    }
360    /**
361     * Method that checks the legibility of a particular problem (Method
    overloading with the other checkLegibility method)
362     * @param areas the problem areas the user wants
363     * @param difficulty the difficulty limitation of the problem the user
    wants
364     * @param probSA the singular problem itself
365     * @return a boolean variable that is true if the problem meets the user
    requirements false if otherwise
366     */
367    private boolean checkLegibility(LinkedList areas, String difficulty,
    SAProblem probSA){
368        if (!probSA.getDifficulty().equalsIgnoreCase(difficulty))
369            return false;
370        LinkedList probAreas = probSA.getAreas();
371        Node n = probAreas.getHead();
372        for (int i = 0; i < probAreas.getSize(); i ++){
373            String currType = (String)(n.getStore());
374            boolean isValid = containsType(areas, currType);
375            if (!isValid)
376                return false;
377        }
378        return true;
379    }
380
```

```
381     /**
382      * Method that checks if a particular problem type is one that the user
    wishes to see in their custom problem set
383      * @param areas the particular problem types the user desires
384      * @param str the singular problem type of a problem
385      * @return a boolean variable that is true if it is contained within the
    ones the user desires false if otherwise
386      */
387     private boolean containsType(LinkedList areas, String str){
388         boolean doesContain = false;
389         Node n = areas.getHead();
390         for (int i = 0; i < areas.getSize(); i++){
391             String curr = (String)(n.getStore());
392             if (curr.equalsIgnoreCase(str)){
393                 doesContain = true;
394                 break;
395             }
396             n = n.getNext();
397         }
398         return doesContain;
399     }
400 }
401
```