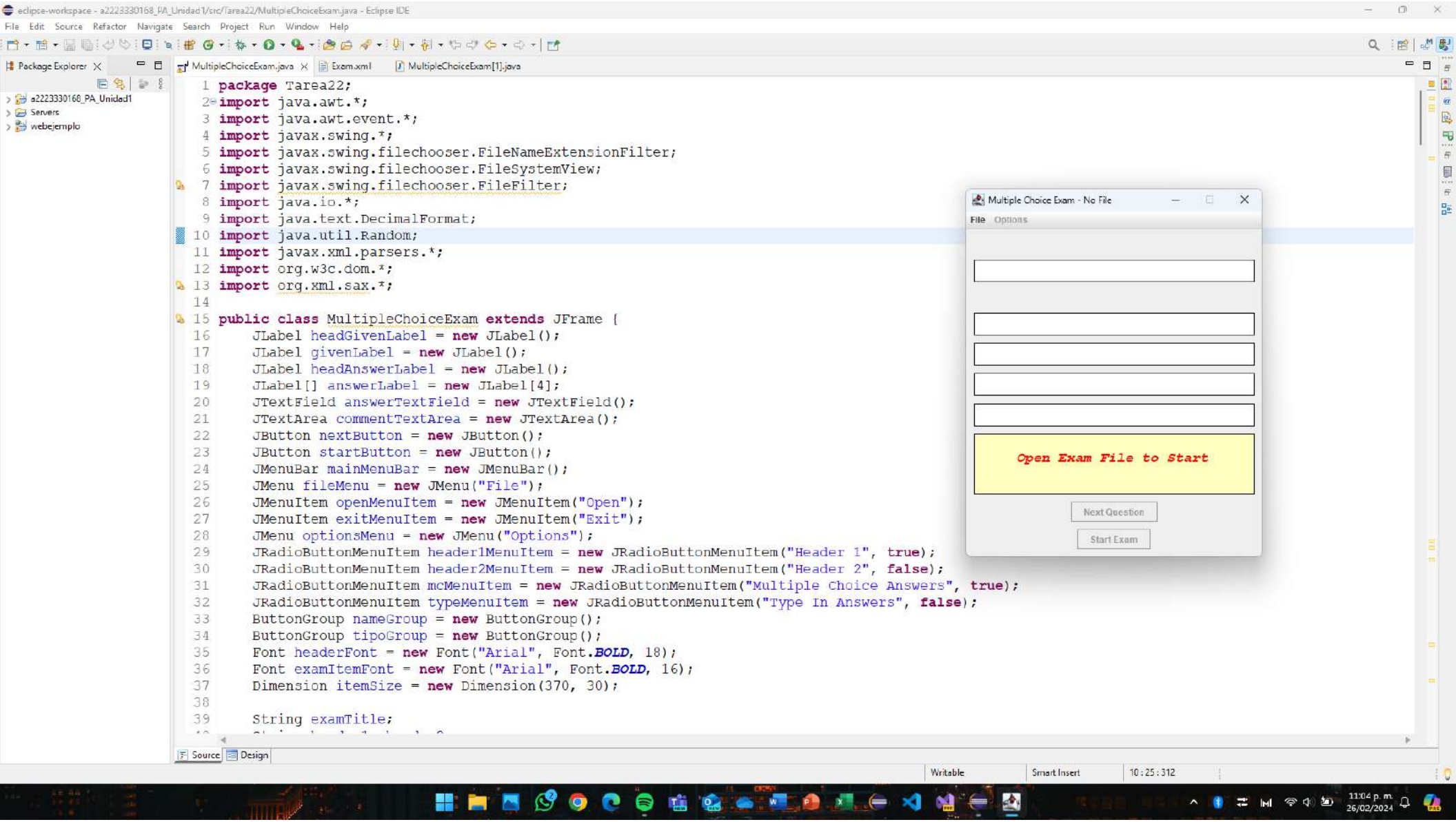
eclipse-workspace - a2223330168_PA_Unidad1/src/Tarea22/MultipleChoiceExam.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
a2223330168_PA_Unidad1
Servers
webejemplo
MultipleChoiceExam.java
Exam.xml
MultipleChoiceExam[1].java
1 package Tarea22;
2 import java.awt.*;
3 import java.awt.event.*;
4 import javax.swing.*;
5 import javax.swing.filechooser.FileNameExtensionFilter;
6 import javax.swing.filechooser.FileSystemView;
7 import javax.swing.filechooser.FileFilter;
8 import java.io.*;
9 import java.text.DecimalFormat;
10 import java.util.Random;
11 import javax.xml.parsers.*;
12 import org.w3c.dom.*;
13 import org.xml.sax.*;
14
15 public class MultipleChoiceExam extends JFrame {
16 JLabel headGivenLabel = new JLabel();
17 JLabel givenLabel = new JLabel();
18 JLabel headAnswerLabel = new JLabel();
19 JLabel[] answerLabel = new JLabel[4];
20 JTextField answerTextField = new JTextField();
21 JTextArea commentTextArea = new JTextArea();
22 JButton nextButton = new JButton();
23 JButton startButton = new JButton();
24 JMenuBar mainMenuBar = new JMenuBar();
25 JMenu fileMenu = new JMenu("File");
26 JMenuItem openMenuItem = new JMenuItem("Open");
27 JMenuItem exitMenuItem = new JMenuItem("Exit");
28 JMenu optionsMenu = new JMenu("Options");
29 JRadioButtonMenuItem header1MenuItem = new JRadioButtonMenuItem("Header 1", true);
30 JRadioButtonMenuItem header2MenuItem = new JRadioButtonMenuItem("Header 2", false);
31 JRadioButtonMenuItem mcMenuItem = new JRadioButtonMenuItem("Multiple Choice Answers", true);
32 JRadioButtonMenuItem typeMenuItem = new JRadioButtonMenuItem("Type In Answers", false);
33 ButtonGroup nameGroup = new ButtonGroup();
34 ButtonGroup tipoGroup = new ButtonGroup();
35 Font headerFont = new Font("Arial", Font.BOLD, 18);
36 Font examItemFont = new Font("Arial", Font.BOLD, 16);
37 Dimension itemSize = new Dimension(370, 30);
38
39 String examTitle;
Source
Design
Writable
Smart Insert
10 : 25 : 312
Multiple Choice Exam - No File
File Options
Open Exam File to Start
Next Question
Start Exam
11:04 p. m.
26/02/2024

```java
package Tarea22;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.filechooser.FileSystemView;
import javax.swing.filechooser.FileFilter;
import java.io.*;
import java.text.DecimalFormat;
import java.util.Random;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;

public class MultipleChoiceExam extends JFrame {
    JLabel headGivenLabel = new JLabel();
    JLabel givenLabel = new JLabel();
    JLabel headAnswerLabel = new JLabel();
    JLabel[] answerLabel = new JLabel[4];
    JTextField answerTextField = new JTextField();
    JTextArea commentTextArea = new JTextArea();
    JButton nextButton = new JButton();
    JButton startButton = new JButton();
    JMenuBar mainMenuBar = new JMenuBar();
    JMenu fileMenu = new JMenu("File");
    JMenuItem openMenuItem = new JMenuItem("Open");
    JMenuItem exitMenuItem = new JMenuItem("Exit");
    JMenu optionsMenu = new JMenu("Options");
    JRadioButtonMenuItem header1MenuItem = new JRadioButtonMenuItem("Header 1",
```

```
   true);
30     JRadioButtonMenuItem header2MenuItem = new JRadioButtonMenuItem("Header 2",
   false);
31     JRadioButtonMenuItem mcMenuItem = new JRadioButtonMenuItem("Multiple Choice
   Answers", true);
32     JRadioButtonMenuItem typeMenuItem = new JRadioButtonMenuItem("Type In
   Answers", false);
33     ButtonGroup nameGroup = new ButtonGroup();
34     ButtonGroup tipoGroup = new ButtonGroup();
35     Font headerFont = new Font("Arial", Font.BOLD, 18);
36     Font examItemFont = new Font("Arial", Font.BOLD, 16);
37     Dimension itemSize = new Dimension(370, 30);
38
39     String examTitle;
40     String header1, header2;
41     int numberTerms;
42     String[] term1 = new String[100];
43     String[] term2 = new String[100];
44     int numberTried, numberCorrect;
45     int correctAnswer;
46     Random myRandom = new Random();
47
48     public static void main(String[] args) {
49         new MultipleChoiceExam().show();
50     }
51
52     public MultipleChoiceExam() {
53         setTitle("Multiple Choice Exam - No File");
54         setResizable(false);
```

```
55          addWindowListener(new WindowAdapter() {
56              public void windowClosing(WindowEvent evt) {
57                  exitForm(evt);
58              }
59          });
60          getContentPane().setLayout(new GridBagLayout());
61          GridBagConstraints gridConstraints;
62          headGivenLabel.setPreferredSize(itemSize);
63          headGivenLabel.setFont(headerFont);
64          gridConstraints = new GridBagConstraints();
65          gridConstraints.gridx = 0;
66          gridConstraints.gridy = 0;
67          gridConstraints.insets = new Insets(10, 10, 0, 10);
68          getContentPane().add(headGivenLabel, gridConstraints);
69          givenLabel.setPreferredSize(itemSize);
70          givenLabel.setFont(examItemFont);
71          givenLabel.setBorder(BorderFactory.createLineBorder(Color.BLACK));
72          givenLabel.setBackground(Color.WHITE);
73          givenLabel.setForeground(Color.BLUE);
74          givenLabel.setOpaque(true);
75          givenLabel.setHorizontalAlignment(SwingConstants.CENTER);
76          gridConstraints = new GridBagConstraints();
77          gridConstraints.gridx = 0;
78          gridConstraints.gridy = 1;
79          gridConstraints.insets = new Insets(0, 10, 0, 10);
80          getContentPane().add(givenLabel, gridConstraints);
81          headAnswerLabel.setPreferredSize(itemSize);
82          headAnswerLabel.setFont(headerFont);
83          gridConstraints = new GridBagConstraints();
```

```
 84         gridConstraints.gridx = 0;
 85         gridConstraints.gridy = 2;
 86         gridConstraints.insets = new Insets(10, 10, 0, 10);
 87         getContentPane().add(headAnswerLabel, gridConstraints);
 88         for (int i = 0; i < 4; i++) {
 89             answerLabel[i] = new JLabel();
 90             answerLabel[i].setPreferredSize(itemSize);
 91             answerLabel[i].setFont(examItemFont);
 92             answerLabel[i].setBorder(BorderFactory.createLineBorder(Color.BLACK));
 93             answerLabel[i].setBackground(Color.WHITE);
 94             answerLabel[i].setForeground(Color.BLUE);
 95             answerLabel[i].setOpaque(true);
 96             answerLabel[i].setHorizontalAlignment(SwingConstants.CENTER);
 97             gridConstraints = new GridBagConstraints();
 98             gridConstraints.gridx = 0;
 99             gridConstraints.gridy = i + 3;
100             gridConstraints.insets = new Insets(0, 10, 10, 10);
101             getContentPane().add(answerLabel[i], gridConstraints);
102         }
103         answerTextField.setPreferredSize(itemSize);
104         answerTextField.setFont(examItemFont);
105         answerTextField.setBackground(Color.WHITE);
106         answerTextField.setForeground(Color.BLUE);
107         answerTextField.setVisible(false);
108         gridConstraints = new GridBagConstraints();
109         gridConstraints.gridx = 0;
110         gridConstraints.gridy = 3;
111         gridConstraints.insets = new Insets(0, 10, 10, 10);
112         getContentPane().add(answerTextField, gridConstraints);
```

```
113         commentTextArea.setPreferredSize(new Dimension(370, 80));
114         commentTextArea.setFont(new Font("Courier New", Font.BOLD + Font.ITALIC,
    18));
115         commentTextArea.setBorder(BorderFactory.createLineBorder(Color.BLACK));
116         commentTextArea.setEditable(false);
117         commentTextArea.setBackground(new Color(255, 255, 192));
118         commentTextArea.setForeground(Color.RED);
119         gridConstraints = new GridBagConstraints();
120         gridConstraints.gridx = 0;
121         gridConstraints.gridy = 7;
122         gridConstraints.insets = new Insets(0, 10, 10, 10);
123         getContentPane().add(commentTextArea, gridConstraints);
124         nextButton.setText("Next Question");
125         gridConstraints = new GridBagConstraints();
126         gridConstraints.gridx = 0;
127         gridConstraints.gridy = 8;
128         gridConstraints.insets = new Insets(0, 0, 10, 0);
129         getContentPane().add(nextButton, gridConstraints);
130         nextButton.addActionListener(new ActionListener() {
131             public void actionPerformed(ActionEvent e) {
132                 nextButtonActionPerformed(e);
133             }
134         });
135         startButton.setText("Start Exam");
136         gridConstraints = new GridBagConstraints();
137         gridConstraints.gridx = 0;
138         gridConstraints.gridy = 9;
139         gridConstraints.insets = new Insets(0, 0, 10, 0);
140         getContentPane().add(startButton, gridConstraints);
```

```
141         startButton.addActionListener(new ActionListener() {
142             public void actionPerformed(ActionEvent e) {
143                 startButtonActionPerformed(e);
144             }
145         });
146         setJMenuBar(mainMenuBar);
147         mainMenuBar.add(fileMenu);
148         fileMenu.add(openMenuItem);
149         fileMenu.addSeparator();
150         fileMenu.add(exitMenuItem);
151         mainMenuBar.add(optionsMenu);
152         optionsMenu.add(header1MenuItem);
153         optionsMenu.add(header2MenuItem);
154         optionsMenu.addSeparator();
155         optionsMenu.add(mcMenuItem);
156         optionsMenu.add(typeMenuItem);
157         nameGroup.add(header1MenuItem);
158         nameGroup.add(header2MenuItem);
159         tipoGroup.add(mcMenuItem);
160         tipoGroup.add(typeMenuItem);
161         openMenuItem.addActionListener(new ActionListener() {
162             public void actionPerformed(ActionEvent e) {
163                 openMenuItemActionPerformed(e);
164             }
165         });
166         exitMenuItem.addActionListener(new ActionListener() {
167             public void actionPerformed(ActionEvent e) {
168                 exitMenuItemActionPerformed(e);
169             }
```

```
170         });
171         header1MenuItem.addActionListener(new ActionListener() {
172             public void actionPerformed(ActionEvent e) {
173                 header1MenuItemActionPerformed(e);
174             }
175         });
176         header2MenuItem.addActionListener(new ActionListener() {
177             public void actionPerformed(ActionEvent e) {
178                 header2MenuItemActionPerformed(e);
179             }
180         });
181         mcMenuItem.addActionListener(new ActionListener() {
182             public void actionPerformed(ActionEvent e) {
183                 mcMenuItemActionPerformed(e);
184             }
185         });
186         typeMenuItem.addActionListener(new ActionListener() {
187             public void actionPerformed(ActionEvent e) {
188                 typeMenuItemActionPerformed(e);
189             }
190         });
191         pack();
192         Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
193         setBounds((int) (0.5 * (screenSize.width - getWidth())), (int) (0.5 *
    (screenSize.height - getHeight())), getWidth(), getHeight());
194         startButton.setEnabled(false);
195         nextButton.setEnabled(false);
196         optionsMenu.setEnabled(false);
197         commentTextArea.setText(centerTextArea("Open Exam File to Start"));
```

```
198     }
199
200     private void exitForm(WindowEvent evt) {
201         System.exit(0);
202     }
203
204     private void nextButtonActionPerformed(ActionEvent e) {
205         nextButton.setEnabled(false);
206         nextQuestion();
207     }
208
209     private void startButtonActionPerformed(ActionEvent e) {
210         String message;
211         if (startButton.getText().equals("Start Exam")) {
212             startButton.setText("Stop Exam");
213             nextButton.setEnabled(false);
214             numberTried = 0;
215             numberCorrect = 0;
216             commentTextArea.setText("");
217             fileMenu.setEnabled(false);
218             optionsMenu.setEnabled(false);
219             nextQuestion();
220         } else {
221             startButton.setText("Start Exam");
222             nextButton.setEnabled(false);
223             if (numberTried > 0) {
224                 message = "Questions Tried: " + String.valueOf(numberTried) +
    "\n";
225                 message += "Questions Correct: " + String.valueOf(numberCorrect) +
```

```
    "\n\n";
226                 message += "Your Score: " + new DecimalFormat("0.0").format(100.0
    * ((double) numberCorrect / numberTried)) + "%";
227                 JOptionPane.showConfirmDialog(null, message, examTitle + "
    Results", JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE);
228             }
229             givenLabel.setText("");
230             answerLabel[0].setText("");
231             answerLabel[1].setText("");
232             answerLabel[2].setText("");
233             answerLabel[3].setText("");
234             answerTextField.setText("");
235             commentTextArea.setText(centerTextArea("Choose Options\nClick Start
    Exam"));
236             fileMenu.setEnabled(true);
237             optionsMenu.setEnabled(true);
238         }
239     }
240
241     private void openMenuItemActionPerformed(ActionEvent e) {
242         JFileChooser openFileChooser = new JFileChooser
    (FileSystemView.getFileSystemView().getHomeDirectory());
243         openFileChooser.setDialogTitle("Select an XML File");
244         openFileChooser.setAcceptAllFileFilterUsed(false);
245         FileNameExtensionFilter filter = new FileNameExtensionFilter("XML Files",
    "xml");
246         openFileChooser.addChoosableFileFilter(filter);
247
248         int returnValue = openFileChooser.showOpenDialog(null);
```

```
249         if (returnValue == JFileChooser.APPROVE_OPTION) {
250             File selectedFile = openFileChooser.getSelectedFile();
251             try {
252                 DocumentBuilderFactory dbFactory =
    DocumentBuilderFactory.newInstance();
253                 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
254                 Document doc = dBuilder.parse(selectedFile);
255                 doc.getDocumentElement().normalize();
256 
257                 examTitle = doc.getElementsByTagName("title").item
    (0).getTextContent();
258 
259                 NodeList headers = doc.getElementsByTagName("headers");
260                 Element headersElement = (Element) headers.item(0);
261                 header1 = headersElement.getElementsByTagName("header1").item
    (0).getTextContent();
262                 header2 = headersElement.getElementsByTagName("header2").item
    (0).getTextContent();
263 
264                 NodeList questions = doc.getElementsByTagName("question");
265                 numberTerms = questions.getLength();
266                 for (int i = 0; i < numberTerms; i++) {
267                     Element question = (Element) questions.item(i);
268                     term1[i] = question.getElementsByTagName("term1").item
    (0).getTextContent();
269                     term2[i] = question.getElementsByTagName("term2").item
    (0).getTextContent();
270                 }
271 
```

```
272                 setTitle("Multiple Choice Exam - " + examTitle);
273                 header1MenuItem.setText(header1 + ", Given " + header2);
274                 header2MenuItem.setText(header2 + ", Given " + header1);
275 
276                 if (header1MenuItem.isSelected()) {
277                     headGivenLabel.setText(header2);
278                     headAnswerLabel.setText(header1);
279                 } else {
280                     headGivenLabel.setText(header1);
281                     headAnswerLabel.setText(header2);
282                 }
283 
284                 startButton.setEnabled(true);
285                 optionsMenu.setEnabled(true);
286                 commentTextArea.setText(centerTextArea("File Loaded, Choose
    Options\nClick Start Exam"));
287 
288             } catch (Exception ex) {
289                 JOptionPane.showConfirmDialog(null, "Error reading in input file -
    make sure file is correct format.", "Multiple Choice Exam File Error",
    JOptionPane.DEFAULT_OPTION, JOptionPane.ERROR_MESSAGE);
290                 return;
291             }
292         }
293     }
294 
295     private void exitMenuItemActionPerformed(ActionEvent e) {
296         System.exit(0);
297     }
```

```
298
299     private void header1MenuItemActionPerformed(ActionEvent e) {
300         headGivenLabel.setText(header2);
301         headAnswerLabel.setText(header1);
302     }
303
304     private void header2MenuItemActionPerformed(ActionEvent e) {
305         headGivenLabel.setText(header1);
306         headAnswerLabel.setText(header2);
307     }
308
309     private void mcMenuItemActionPerformed(ActionEvent e) {
310         answerLabel[0].setVisible(true);
311         answerLabel[1].setVisible(true);
312         answerLabel[2].setVisible(true);
313         answerLabel[3].setVisible(true);
314         answerTextField.setVisible(false);
315     }
316
317     private void typeMenuItemActionPerformed(ActionEvent e) {
318         answerLabel[0].setVisible(false);
319         answerLabel[1].setVisible(false);
320         answerLabel[2].setVisible(false);
321         answerLabel[3].setVisible(false);
322         answerTextField.setVisible(true);
323     }
324
325     private String parseLeft(String s) {
326         int cl = s.indexOf(",");
```

```
327         return (s.substring(0, cl));
328     }
329 
330     private String parseRight(String s) {
331         int cl = s.indexOf(",");
332         return (s.substring(cl + 1));
333     }
334 
335     private String centerTextArea(String s) {
336         int charsPerLine = 33;
337         String sOut = "";
338         int j = s.indexOf("\n");
339         int nSpaces;
340         if (j == -1) {
341             sOut = "\n" + spacePadding((int) ((charsPerLine - s.length()) / 2)) +
    s;
342         } else {
343             String l = s.substring(0, j);
344             sOut = "\n" + spacePadding((int) ((charsPerLine - l.length()) / 2)) +
    l;
345             l = s.substring(j + 1);
346             sOut += "\n" + spacePadding((int) ((charsPerLine - l.length()) / 2)) +
    l;
347         }
348         return (sOut);
349     }
350 
351     private String spacePadding(int n) {
352         String s = "";
```

```
353         if (n != 0)
354             for (int i = 0; i < n; i++)
355                 s += " ";
356         return (s);
357     }
358 
359     private void nextQuestion() {
360         boolean[] termUsed = new boolean[numberTerms];
361         int[] index = new int[4];
362         int j;
363         commentTextArea.setText("");
364         correctAnswer = myRandom.nextInt(numberTerms);
365         if (header1MenuItem.isSelected()) {
366             givenLabel.setText(term2[correctAnswer]);
367         } else {
368             givenLabel.setText(term1[correctAnswer]);
369         }
370         if (mcMenuItem.isSelected()) {
371             for (int i = 0; i < numberTerms; i++) {
372                 termUsed[i] = false;
373             }
374             for (int i = 0; i < 4; i++) {
375                 do {
376                     j = myRandom.nextInt(numberTerms);
377                 } while (termUsed[j] || j == correctAnswer);
378                 termUsed[j] = true;
379                 index[i] = j;
380             }
381             index[myRandom.nextInt(4)] = correctAnswer;
```

```
382             if (header1MenuItem.isSelected()) {
383                 answerLabel[0].setText(term1[index[0]]);
384                 answerLabel[1].setText(term1[index[1]]);
385                 answerLabel[2].setText(term1[index[2]]);
386                 answerLabel[3].setText(term1[index[3]]);
387             } else {
388                 answerLabel[0].setText(term2[index[0]]);
389                 answerLabel[1].setText(term2[index[1]]);
390                 answerLabel[2].setText(term2[index[2]]);
391                 answerLabel[3].setText(term2[index[3]]);
392             }
393         } else {
394             answerTextField.setEditable(true);
395             answerTextField.setText("");
396             answerTextField.requestFocus();
397         }
398     }
399 
400     private void updateScore(boolean correct) {
401         if (correct) {
402             numberCorrect++;
403             commentTextArea.setText(centerTextArea("Correct!"));
404         } else
405             commentTextArea.setText(centerTextArea("Sorry ... Correct Answer
    Shown"));
406 
407         if (mcMenuItem.isSelected()) {
408             if (header1MenuItem.isSelected())
409                 answerLabel[0].setText(term1[correctAnswer]);
```

```
410             else
411                 answerLabel[0].setText(term2[correctAnswer]);
412             answerLabel[1].setText("");
413             answerLabel[2].setText("");
414             answerLabel[3].setText("");
415         } else {
416             if (header1MenuItem.isSelected())
417                 answerTextField.setText(term1[correctAnswer]);
418             else
419                 answerTextField.setText(term2[correctAnswer]);
420         }
421         startButton.setEnabled(true);
422         nextButton.setEnabled(true);
423         nextButton.requestFocus();
424     }
425 
426     private String soundex(String w) {
427         String wTemp, s = "";
428         int l;
429         int wPrev, wSnd, cIndex;
430         int[] wSound = {0, 1, 2, 3, 0, 1, 2, 0, 0, 2, 2, 4, 5, 5, 0, 1, 2, 6, 2,
    3, 0, 1, 0, 2, 0, 2};
431         wTemp = w.toUpperCase();
432         l = w.length();
433         if (l != 0) {
434             s = String.valueOf(w.charAt(0));
435             wPrev = 0;
436             if (l > 1) {
437                 for (int i = 1; i < l; i++) {
```

```
438                     cIndex = (int) wTemp.charAt(i) - 65;
439                     if (cIndex >= 0 && cIndex <= 25) {
440                         wSnd = wSound[cIndex] + 48;
441                         if (wSnd != 48 && wSnd != wPrev) {
442                             s += String.valueOf((char) wSnd);
443                         }
444                         wPrev = wSnd;
445                     }
446                 }
447             }
448         } else
449             s = " ";
450         return s;
451     }
452
453}
454
```