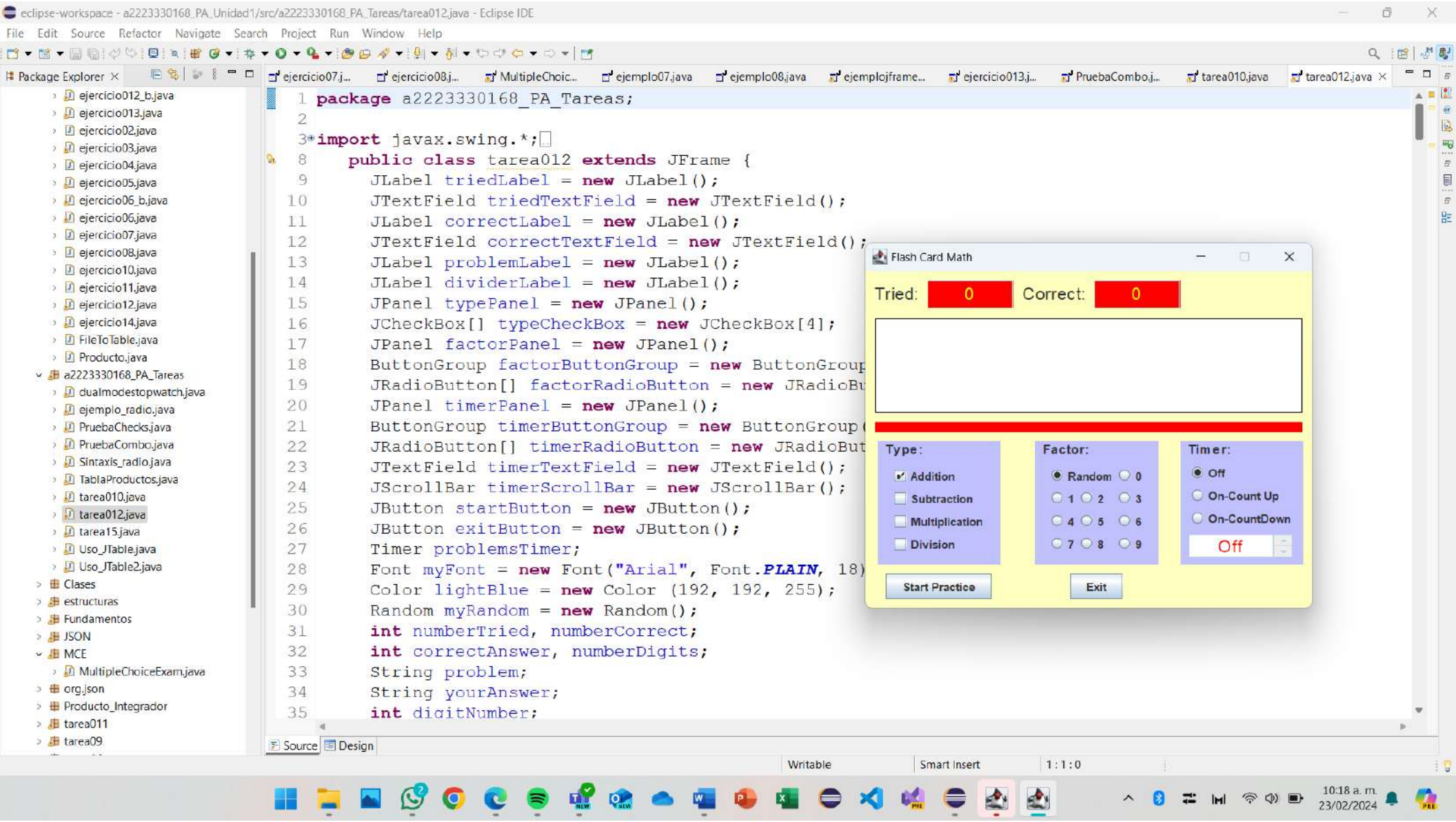
eclipse-workspace - a2223330168_PA_Unidad1/src/a2223330168_PA_Tareas/tarea012.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
ejercicio012_b.java
ejercicio013.java
ejercicio02.java
ejercicio03.java
ejercicio04.java
ejercicio05.java
ejercicio06_b.java
ejercicio06.java
ejercicio07.java
ejercicio08.java
ejercicio10.java
ejercicio11.java
ejercicio12.java
ejercicio14.java
FileToTable.java
Producto.java
a2223330168_PA_Tareas
dualmodestopwatch.java
ejemplo_radio.java
PruebaChecks.java
PruebaCombo.java
Sintaxis_radio.java
TablaProductos.java
tarea010.java
tarea012.java
tarea15.java
Uso_JTable.java
Uso_JTable2.java
Clases
estructuras
Fundamentos
JSON
MCE
MultipleChoiceExam.java
org.json
Producto_Integrador
tarea011
tarea09
ejercicio07.j...
ejercicio08.j...
MultipleChoic...
ejemplo07.java
ejemplo08.java
ejemplojframe...
ejercicio013.j...
PruebaCombo.j...
tarea010.java
tarea012.java
1 package a2223330168_PA_Tareas;
2
3 import javax.swing.*;
8 public class tarea012 extends JFrame {
9 JLabel triedLabel = new JLabel();
10 JTextField triedTextField = new JTextField();
11 JLabel correctLabel = new JLabel();
12 JTextField correctTextField = new JTextField();
13 JLabel problemLabel = new JLabel();
14 JLabel dividerLabel = new JLabel();
15 JPanel typePanel = new JPanel();
16 JCheckBox[] typeCheckBox = new JCheckBox[4];
17 JPanel factorPanel = new JPanel();
18 ButtonGroup factorButtonGroup = new ButtonGroup
19 JRadioButton[] factorRadioButton = new JRadioBu
20 JPanel timerPanel = new JPanel();
21 ButtonGroup timerButtonGroup = new ButtonGroup(
22 JRadioButton[] timerRadioButton = new JRadioBut
23 JTextField timerTextField = new JTextField();
24 JScrollBar timerScrollBar = new JScrollBar();
25 JButton startButton = new JButton();
26 JButton exitButton = new JButton();
27 Timer problemsTimer;
28 Font myFont = new Font("Arial", Font.PLAIN, 18)
29 Color lightBlue = new Color (192, 192, 255);
30 Random myRandom = new Random();
31 int numberTried, numberCorrect;
32 int correctAnswer, numberDigits;
33 String problem;
34 String yourAnswer;
35 int digitNumber;
Source Design
Writable Smart Insert 1 : 1 : 0
Flash Card Math
Tried: 0 Correct: 0
Type:
Addition
Subtraction
Multiplication
Division
Factor:
Random 0
1 2 3
4 5 6
7 8 9
Timer:
Off
On-Count Up
On-CountDown
Off
Start Practice
Exit
10:18 a. m.
23/02/2024

```java
package a2223330168_PA_Tareas;

import javax.swing.*;
   public class tarea012 extends JFrame {
     JLabel triedLabel = new JLabel();
     JTextField triedTextField = new JTextField();
     JLabel correctLabel = new JLabel();
     JTextField correctTextField = new JTextField();
     JLabel problemLabel = new JLabel();
     JLabel dividerLabel = new JLabel();
     JPanel typePanel = new JPanel();
     JCheckBox[] typeCheckBox = new JCheckBox[4];
     JPanel factorPanel = new JPanel();
     ButtonGroup factorButtonGroup = new ButtonGroup();
     JRadioButton[] factorRadioButton = new JRadioButton[11];
     JPanel timerPanel = new JPanel();
     ButtonGroup timerButtonGroup = new ButtonGroup();
     JRadioButton[] timerRadioButton = new JRadioButton[3];
     JTextField timerTextField = new JTextField();
     JScrollBar timerScrollBar = new JScrollBar();
     JButton startButton = new JButton();
     JButton exitButton = new JButton();
     Timer problemsTimer;
     Font myFont = new Font("Arial", Font.PLAIN, 18);
     Color lightBlue = new Color (192, 192, 255);
     Random myRandom = new Random();
     int numberTried, numberCorrect;
     int correctAnswer, numberDigits;
     String problem;
```

```
34      String yourAnswer;
35      int digitNumber;
36      int problemTime;
37
38      public static void main(String args[]) {
39      // create frame
40      new tarea012().show();}
41
42      public tarea012(){
43      // frame constructor
44      setTitle("Flash Card Math");
45      getContentPane().setBackground(new Color (255, 255, 192));
46      setResizable(false);
47      addWindowListener(new WindowAdapter() {
48          public void windowClosing(WindowEvent evt) {
49              exitForm(evt);
50              }
51});
52
53      getContentPane().setLayout(new GridBagLayout());
54      GridBagConstraints gridConstraints;
55      triedLabel.setText("Tried:");
56      triedLabel.setFont(myFont);
57      gridConstraints = new GridBagConstraints();
58      gridConstraints.gridx= 0;
59      gridConstraints.gridy = 0;
60      gridConstraints.anchor = GridBagConstraints.WEST;
61      gridConstraints.insets = new Insets(10, 10, 0, 10);
62      getContentPane().add(triedLabel,gridConstraints);
```

```
63      triedTextField.setText("0");
64      triedTextField.setPreferredSize(new Dimension(90,30));
65      triedTextField.setEditable(false);
66      triedTextField.setBackground(Color.RED);
67      triedTextField.setForeground(Color.YELLOW);
68      triedTextField.setHorizontalAlignment(SwingConstants.CENTER);
69      triedTextField.setFont(myFont);
70      gridConstraints = new GridBagConstraints();
71      gridConstraints.gridx = 1;
72      gridConstraints.gridy = 0;
73      gridConstraints.insets = new Insets(10, 0, 0, 0);getContentPane().add
  (triedTextField,gridConstraints);
74      correctLabel.setText("Correct:");
75      correctLabel.setFont(myFont);
76      gridConstraints = new GridBagConstraints();
77      gridConstraints.gridx = 2;
78      gridConstraints.gridy = 0;
79      gridConstraints.anchor = GridBagConstraints.EAST;
80      gridConstraints.insets = new Insets(10, 10, 0, 10);
81      getContentPane().add(correctLabel,gridConstraints);
82      correctTextField.setText("0");
83      correctTextField.setPreferredSize(new Dimension(90,30));
84      correctTextField.setEditable(false);
85      correctTextField.setBackground(Color.RED);
86      correctTextField.setForeground(Color.YELLOW);
87      correctTextField.setHorizontalAlignment(SwingConstants.CENTER);
88      correctTextField.setFont(myFont);
89      gridConstraints = new GridBagConstraints();
90      gridConstraints.gridx = 3;
```

```
 91      gridConstraints.gridy = 0;
 92      gridConstraints.insets = new Insets(10, 0, 0, 0);
 93      getContentPane().add(correctTextField, gridConstraints);
 94      problemLabel.setText("");
 95      problemLabel.setBorder(BorderFactory.createLineBorder(Color.BLACK));
 96      problemLabel.setPreferredSize(new Dimension(450,    100));
 97      problemLabel.setBackground(Color.WHITE);
 98      problemLabel.setOpaque(true);
 99      problemLabel.setFont(new Font("Comic Sans MS", Font.PLAIN,48));
100      problemLabel.setHorizontalAlignment(SwingConstants.CENTER);
101      gridConstraints = new GridBagConstraints();
102      gridConstraints.gridx = 0;
103      gridConstraints.gridy = 1;
104      gridConstraints.gridwidth = 5;
105      gridConstraints.insets = new Insets(10, 10, 0, 10);
106      getContentPane().add(problemLabel, gridConstraints);
107      problemLabel.addKeyListener(new KeyAdapter() {
108
109          public void keyPressed(KeyEvent e) {
110              problemLabelKeyPressed(e);}
111          });
112      dividerLabel.setPreferredSize(new Dimension(450, 10));
113      dividerLabel.setBackground(Color.RED);
114      dividerLabel.setOpaque(true);
115      gridConstraints = new GridBagConstraints();
116      gridConstraints.gridx = 0;
117      gridConstraints.gridy = 2;
118      gridConstraints.gridwidth = 5;
119      gridConstraints.insets = new Insets(10, 10, 10, 10);
```

```
120     getContentPane().add(dividerLabel, gridConstraints);
121     UIManager.put("TitledBorder.font", new Font("Arial", Font.BOLD, 14));
122     typePanel.setPreferredSize(new Dimension(130, 130));
123     typePanel.setBorder(BorderFactory.createTitledBorder("Type:"));
124     typePanel.setBackground(lightBlue);
125     typePanel.setLayout(new GridBagLayout()); gridConstraints = new
   GridBagConstraints();
126
127     gridConstraints.gridx = 0;
128     gridConstraints.gridy = 3;
129     gridConstraints.gridwidth = 2;
130     gridConstraints.anchor = GridBagConstraints.NORTH;getContentPane().add
   (typePanel, gridConstraints); for (int i = 0; i < 4;i++) {
131     typeCheckBox[i] = new JCheckBox();
132     typeCheckBox[i].setBackground(lightBlue);
133     gridConstraints = new GridBagConstraints();
134     gridConstraints.gridx = 0;
135     gridConstraints.gridy = i;
136     gridConstraints.anchor = GridBagConstraints. WEST;
137     typePanel.add(typeCheckBox[i],gridConstraints);
138     typeCheckBox[i].addActionListener(new ActionListener() {
139
140         public void actionPerformed(ActionEvent e) {
141      typeCheckBoxActionPerformed(e);}
142    });
143}
144     typeCheckBox[0].setText("Addition");
145     typeCheckBox[1].setText("Subtraction");
146     typeCheckBox[2].setText("Multiplication");
```

```
147      typeCheckBox[3].setText("Division");typeCheckBox[0].setSelected(true);
148      factorPanel.setPreferredSize(new Dimension(130, 130));
149      factorPanel.setBorder(BorderFactory.createTitledBorder("Factor:"));
150      factorPanel.setBackground(lightBlue);
151      factorPanel.setLayout(new GridBagLayout());
152      gridConstraints = new GridBagConstraints();
153      gridConstraints.gridx = 2;
154      gridConstraints.gridy = 3;
155      gridConstraints.gridwidth = 2;
156      gridConstraints.anchor = GridBagConstraints.NORTH;
157      getContentPane().add(factorPanel, gridConstraints); int x = 2;
158      int y = 0;
159      for (int i = 0; i < 11; i++)
160{
161
162     factorRadioButton[i] = new JRadioButton();
163     factorRadioButton[i].setText(String.valueOf(i));
164     factorRadioButton[i].setBackground(lightBlue);
165     factorButtonGroup.add(factorRadioButton[i]);
166     gridConstraints = new GridBagConstraints();
167     if (i < 10) {
168     gridConstraints.gridx = x;
169     gridConstraints.gridy = y;
170     }
171     else {
172     gridConstraints.gridx = 0;
173     gridConstraints.gridy = 0;
174     gridConstraints.gridwidth = 2;
175     }
```

```
176     gridConstraints.anchor = GridBagConstraints.WEST;
177     factorPanel.add(factorRadioButton[i], gridConstraints);
178     factorRadioButton[i].addActionListener(new ActionListener() {
179     public void actionPerformed(ActionEvent e) {
180     factorRadioButtonActionPerformed(e); }
181     });
182     x++;
183     if(x > 2)
184     {
185     x = 0;
186     y++;
187
188     }
189     }
190
191      factorRadioButton[10].setText("Random");
192      factorRadioButton[10].setSelected(true);
193      timerPanel.setPreferredSize(new Dimension(130, 130));
194      timerPanel.setBorder(BorderFactory.createTitledBorder("Timer:"));
195      timerPanel.setBackground(lightBlue);
196      timerPanel.setLayout(new GridBagLayout());
197      gridConstraints = new GridBagConstraints();
198      gridConstraints.gridx = 4;
199      gridConstraints.gridy = 3;
200      gridConstraints.insets = new Insets(0, 0, 0, 10);
201      gridConstraints.anchor = GridBagConstraints.NORTH;
202      getContentPane().add(timerPanel, gridConstraints);
203      for (int i = 0; i < 3; i++)
204 {
```

```
205         timerRadioButton[i] = new JRadioButton();
206         timerRadioButton[i].setBackground(lightBlue);
207         timerButtonGroup.add(timerRadioButton[i]);
208         gridConstraints = new GridBagConstraints();
209         gridConstraints.gridx = 0;
210         gridConstraints.gridy = i;
211         gridConstraints.gridwidth = 2;
212         gridConstraints.anchor = GridBagConstraints. WEST;
213         timerPanel.add(timerRadioButton[i],gridConstraints);
214         timerRadioButton[i].addActionListener(new ActionListener() {
215
216             public void actionPerformed(ActionEvent e) {
217                 timerRadioButtonActionPerformed(e); }
218             });
219}
220
221     timerRadioButton[0].setText("Off");
222     timerRadioButton[1].setText("On-Count Up");
223     timerRadioButton[2].setText("On-CountDown");
224     timerRadioButton[0].setSelected(true);
225     timerTextField.setText("Off");
226     timerTextField.setPreferredSize(new Dimension (90,25));
227     timerTextField.setEditable(false);
228     timerTextField.setBackground(Color.WHITE);
229     timerTextField.setForeground(Color.RED);
230     timerTextField.setHorizontalAlignment(SwingConstants.CENTER);
231     timerTextField.setFont(myFont);
232     gridConstraints = new GridBagConstraints();
233     gridConstraints.gridx = 0;
```

```
234     gridConstraints.gridy = 3;
235     gridConstraints.anchor = GridBagConstraints. WEST;
236     gridConstraints.insets = new Insets(5, 0, 0, 0);
237     timerPanel.add(timerTextField,gridConstraints);
238     timerScrollBar.setPreferredSize(new Dimension(20, 25));
239     timerScrollBar.setMinimum(1);
240     timerScrollBar.setMaximum(60);
241     timerScrollBar.setValue(1);
242     timerScrollBar.setBlockIncrement(1);
243     timerScrollBar.setUnitIncrement(1);
244     timerScrollBar.setOrientation(JScrollBar.VERTICAL);
245     timerScrollBar.setEnabled(false);
246     gridConstraints = new GridBagConstraints();
247     gridConstraints.gridx = 1;
248     gridConstraints.gridy = 3;
249     gridConstraints.anchor = GridBagConstraints.WEST;
250     gridConstraints.insets = new Insets(5, 0, 0, 0);
251     timerPanel.add(timerScrollBar, gridConstraints);
252     timerScrollBar.addAdjustmentListener(new AdjustmentListener() {
253
254         public void adjustmentValueChanged(AdjustmentEvent e) {
255             timerScrollBarAdjustmentValueChanged(e); }
256           });
257     startButton.setText("Start Practice");
258     gridConstraints = new GridBagConstraints();
259     gridConstraints.gridx = 0;
260     gridConstraints.gridy = 4;
261     gridConstraints.gridwidth = 2;
262     gridConstraints.insets = new Insets(10, 0, 10, 0);
```

```
263     getContentPane().add(startButton, gridConstraints);
264     startButton.addActionListener(new ActionListener() {
265
266         public void actionPerformed(ActionEvent e) {
267             startButtonActionPerformed(e);
268             }
269         });
270     exitButton.setText("Exit");
271     gridConstraints = new GridBagConstraints();
272     gridConstraints.gridx = 2;
273     gridConstraints.gridy = 4;
274     gridConstraints.gridwidth = 2;
275     gridConstraints.insets = new Insets(10, 0, 10, 0);
276     getContentPane().add(exitButton, gridConstraints);
277     exitButton.addActionListener(new ActionListener() {
278         public void actionPerformed(ActionEvent e) {
279exitButtonActionPerformed(e);
280}
281});
282problemsTimer = new Timer (1000, new ActionListener() {
283public void actionPerformed(ActionEvent e) {
284problemsTimerActionPerformed(e);
285}
286});
287pack();
288Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
289setBounds((int)((screenSize.width - getWidth())), (int) ((screenSize.height -
   getHeight()))), getWidth(), getHeight());}
290private void exitForm(WindowEvent evt) {
```

```
291System.exit(0);
292}
293private void typeCheckBoxActionPerformed(ActionEvent e) {
294int numberChecks;
295int clickedBox = 0;
296// determine which box was clicked
297String s = e.getActionCommand();
298if (s.equals("Addition"))
299clickedBox = 0;
300else if (s.equals("Subtraction")) clickedBox = 1;
301else if (s.equals("Multiplication")) clickedBox = 2;
302else if (s.equals("Division"))
303clickedBox = 3;
304// determine how many boxes are checked
305numberChecks = 0;
306if (typeCheckBox[0].isSelected()) numberChecks++;
307if (typeCheckBox[1].isSelected()) numberChecks++;
308if (typeCheckBox[2].isSelected()) numberChecks++;
309if (typeCheckBox[3].isSelected()) {
310    numberChecks++;
311// make sure zero not selected factor
312    if(factorRadioButton[0].isSelected())
313        factorRadioButton[1].doClick();
314        factorRadioButton[0].setEnabled(false); }
315else
316{
317factorRadioButton[0].setEnabled(true); }
318// if all boxes unchecked, recheck last clicked box
319if (numberChecks == 0)
```

```
320     typeCheckBox[clickedBox].setSelected(true);
321     problemLabel.requestFocus();
322     }
323     private void factorRadioButtonActionPerformed(ActionEvent e){
324     problemLabel.requestFocus();
325     }
326     private void timerRadioButtonActionPerformed(ActionEvent e) {
327     if (timerRadioButton[0].isSelected()) {
328     timerTextField.setText("Off");
329     timerScrollBar.setEnabled(false); }
330     else if (timerRadioButton[1].isSelected()) {
331     problemTime = 0;
332     timerTextField.setText(getTime(problemTime));
333     timerScrollBar.setEnabled(false); }
334     else if (timerRadioButton[2].isSelected()) {
335     problemTime = 30 * timerScrollBar.getValue();
336     timerTextField.setText(getTime(problemTime));
337     timerScrollBar.setEnabled(true);
338     }
339     }
340     private void timerScrollBarAdjustmentValueChanged(AdjustmentEvent e) {
341     timerTextField.setText(getTime(30 * timerScrollBar.getValue()));}
342
343     private void startButtonActionPerformed(ActionEvent e){
344     int score;
345     String message = "";
346     if (startButton.getText().equals("Start Practice")) {
347         startButton.setText("Stop Practice");
348         exitButton.setEnabled(false);
```

```
349         numberTried = 0;
350         numberCorrect = 0;
351         triedTextField.setText("0");
352         correctTextField.setText("0");
353         timerRadioButton[0].setEnabled(false);
354         timerRadioButton[1].setEnabled(false);
355         timerRadioButton[2].setEnabled(false);
356         timerScrollBar.setEnabled(false);
357         if(!timerRadioButton[01].isSelected()) {
358         if (timerRadioButton[1].isSelected()) problemTime = 0;
359         else
360         problemTime = 30 * timerScrollBar.getValue();
361         timerTextField.setText(getTime(problemTime));
362         problemsTimer.start();
363         }
364         problemLabel.setText(getProblem());}
365     else
366         {
367         timerRadioButton[0].setEnabled(true);
368         timerRadioButton[1].setEnabled(true);
369         timerRadioButton[2].setEnabled(true);
370         if (timerRadioButton[2].isSelected())
371             timerScrollBar.setEnabled(true);
372         problemsTimer.stop();
373         startButton.setText("Start Practice");
374         exitButton.setEnabled(true);
375         problemLabel.setText("");
376         if (numberTried > 0){
377             score = (int) (100 * (double) (numberCorrect) / numberTried);
```

```
378             message = "Problems Tried: " + String.valueOf(numberTried) + "\n";
379             message += "Problems Correct: " + String.valueOf(numberCorrect) + " ("
   + String.valueOf(score) + " %)" + "\n";
380             if(timerRadioButton[0].isSelected()) {
381             message += "Timer Off";
382             }
383             else
384             {
385             if(timerRadioButton[2].isSelected()) {
386             problemTime = 30 * timerScrollBar.getValue() - problemTime; }
387             message += "Elapsed Time: " + getTime(problemTime) + "\n";
388             message += "Time Per Problem: " + new DecimalFormat("0.00").format
   ((double)(problemTime)/numberTried) + " sec";
389             }
390             JOptionPane.showConfirmDialog(null, message, "Results",
391             JOptionPane.DEFAULT_OPTION, JOptionPane.INFORMATION_MESSAGE); }
392         }
393     }
394
395             private void exitButtonActionPerformed(ActionEvent e){
396             System.exit(0);
397             }
398             private void problemLabelKeyPressed(KeyEvent e) {
399             if (startButton.getText().equals("Start Practice")) return;
400             // only allow number keys
401             if (e.getKeyChar() >= '0' && e.getKeyChar() <= '9') {
402                 yourAnswer += e.getKeyChar();
403                 problemLabel.setText(problem + yourAnswer);
404                 if (digitNumber != numberDigits) {
```

```
405                     digitNumber++;
406             problemLabel.setText(problemLabel.getText() + "?"); return;
407                 }
408     else
409     {
410     numberTried++;
411     // check answer
412     if (Integer.valueOf(yourAnswer).intValue()== correctAnswer)
413     {
414     numberCorrect++;
415     }
416     triedTextField.setText(String.valueOf(numberTried));
417     correctTextField.setText(String.valueOf(numberCorrect));
418     problemLabel.setText(getProblem());}
419         }
420         }
421     private void problemsTimerActionPerformed(ActionEvent e){
422     if (timerRadioButton[1].isSelected()) {
423         problemTime++;
424     timerTextField.setText(getTime(problemTime));
425     if (problemTime >= 1800) {
426         startButton.doClick();
427         return;
428     }
429     }
430     else
431     {
432     problemTime--;
433     timerTextField.setText(getTime(problemTime));
```

```
434     if (problemTime == 0)
435     {
436     startButton.doClick();
437     return;
438     }
439     }
440     }
441
442     private String getProblem() {
443
444
445     int pType, p, number, factor;
446     p = 0;
447     do
448         {
449         pType = myRandom.nextInt(4) + 1;
450     if (pType == 1 && typeCheckBox[0].isSelected()) {
451     // Addition
452         p = pType;
453         number = myRandom.nextInt(10);
454         factor = getFactor(1);
455         correctAnswer = number + factor;
456         problem = String.valueOf(number) + " + " + String.valueOf(factor) + " = ";
    }
457         else if (pType == 2 && typeCheckBox[1].isSelected()) {
458         // Subtraction
459         p = pType;
460         factor = getFactor(2);
461         correctAnswer = myRandom.nextInt(10);
```

```
462         number = correctAnswer + factor;
463         problem = String.valueOf(number) + " - " + String.valueOf(factor) + " =";
    }
464         else if (pType == 3 && typeCheckBox[2].isSelected()) {
465         // Multiplication
466         p = pType;
467         number = myRandom.nextInt(10);
468         factor = getFactor(3);
469         correctAnswer = number * factor;
470         problem = String.valueOf(number) + " × " + String.valueOf(factor) + " = ";
    }
471         else if (pType == 4 && typeCheckBox[3].isSelected()) {
472         // Division
473         p = pType;
474         factor = getFactor(4);
475         correctAnswer = myRandom.nextInt(10);
476         number = correctAnswer * factor;
477         problem = String.valueOf(number) + " / " + String.valueOf(factor) + " =";
    }
478         }
479         while (p == 0);
480             yourAnswer ="";
481             digitNumber = 1;
482             problemLabel.requestFocus();
483             if (correctAnswer < 10)
484             {
485             numberDigits = 1;
486             return (problem + "?");
487             }
```

```
488             else
489             {
490             numberDigits = 2;
491             return (problem +"??");
492             }
493     }
494             private int getFactor(int p)
495             {
496             if (factorRadioButton[10].isSelected()) {
497             //random
498             if (p == 4)
499             return (myRandom.nextInt(9) + 1); else
500                 return (myRandom.nextInt(10));
501             }
502             else
503             {
504                 for (int i = 0; i < 10; i++)
505                 {
506                 if (factorRadioButton[i].isSelected())
507                     return(i);
508                 }
509                 return (0);
510             }
511             }
512             private String getTime(int s)
513             {
514                 int min, sec;
515                 String ms, ss;
516                 min = (int) (s / 60);
```

```
517                 sec = s - 60 * min;
518                 ms = String.valueOf(min);
519                 ss = String.valueOf(sec);
520                 if (sec < 10)
521                 ss = "0" + ss;
522                 return (ms + ":" + ss);
523
524 }
525 }
```