```
1  using System;
2  using System.Drawing;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using System.Windows.Forms;
6  using System.IO;
7
8
9
10 namespace Karela
11 {
12     public partial class main : Form
13     {
14         //************************************
15         //Running Loops and If statement variables
16         //************************************
17         static bool forLoop = false;                    //If the program is running a
             for loop
18         static string condition = "";                   //Condition that needs to be
             met for loop
19         static string[] x;                              //holds split conditions from
             statements
20         static bool greaterthan = false;                //Show whether for loop is
             greater than or less than
21         static int startValue = 0;                      //Starting value for a for loop
22         static bool up = true;                          //Whether to increment up or
             down for a for loop
23         static int loopVariable = 0;                    //The varible used to increment
              a for loop
24         static bool runElse = false;                    //Indicates whether or not the
             else needs to run
25
26
27         //************************************
28         //General running of program
29         //************************************
30         int line = 0;                                   //Current line number being run
31         int startIndex = 0;                             //Index used for highlighting
             text
32         static String s;                                //Stores text from RTB
33         public String[] prog;                           //Stores split program
34         static int[,] compoundStatements = new int[10, 4]; //Keeps track of line nums
             for beginning and end of statements
35         public bool progDone = false;                   //Indicates whether the program
              has run completely or not
36         static bool error = false;                      //Indicates if prog has stopped
              due to an error
37         public bool progSet = false;                    //Indicates if prog was already
              set in the lessson form
```

```
38          int progLength = 0;                                 //Length of the program
39          int tableCol = 0;                                   //Column in compoundStatement
              table
40          int tableRow = 0;                                   //Row in compoundStatement
              table
41          public bool fromLesson = false;                     // Shows if coming from lesson
              form
42
43          //*************************************
44          // Variables to keep track of place in grid and program
45          //*************************************
46
47          box[,] grid = new box[7, 7];                        //Array to keep track of
              picture boxes in grid
48          static int activeRow = 6;                           //Row that Karela is on
49          static int activeColumn = 0;                        //Column that Karela is on
50          int direction = 1;                          //Direction Karela is facing
51          int gridSize = 7;                                   //Size of grid
52
53          //*************************************
54          // Variables for saving/opening files and opening lessons
55          //*************************************
56          int lesson = 1;                     //Indicates what number lesson is selected
57          bool fileModified = false;          //Indicates if a program has been modified
58          string fileName = "Untitled";       //Default file name
59
60
61          //*************************************
62          // Image variables
63          //*************************************
64          Image karela = Prototype.Properties.Resources.karela;
65          Image blank = Prototype.Properties.Resources.white;
66          Image block = Prototype.Properties.Resources.blocked;
67          Image KarelaPaint = Prototype.Properties.Resources.KarelaPainted;
68          Image painted = Prototype.Properties.Resources.painted;
69
70          public main()
71          {
72              InitializeComponent();
73              labelAction.Text = "";
74
75              //Programmatically create pictureboxes in grid
76              for (int i = 0; i < gridSize; i++)
77              {
78                  for (int j = 0; j < gridSize; j++)
79                  {
80                      PictureBox p = new PictureBox();
81
82                      p.Anchor = ((System.Windows.Forms.AnchorStyles)
```

```
                        ((((System.Windows.Forms.AnchorStyles.Top |
                        System.Windows.Forms.AnchorStyles.Bottom)
 83                     | System.Windows.Forms.AnchorStyles.Left)
 84                     | System.Windows.Forms.AnchorStyles.Right)));
 85                     p.Image = blank;
 86                     p.Size = new System.Drawing.Size(119, 118);
 87                     p.SizeMode = System.Windows.Forms.PictureBoxSizeMode.CenterImage;
 88                     p.Click += new System.EventHandler(this.pictureBox_block);
 89                     grid[i, j] = new box(p, i, j);
 90                 }
 91             }
 92         }
 93
 94         private void main_Load(object sender, EventArgs e)
 95         {
 96
 97             //Add pictureboxes to table layout
 98             for (int i = 0; i < gridSize; i++)
 99             {
100                 for (int j = 0; j < gridSize; j++)
101                 {
102                     tableLayoutPanelGrid.Controls.Add(grid[i, j].getPictureBox(), j,
                        i);
103                 }
104             }
105
106             //Set bottom corner as active image
107             grid[6, 0].setImage(karela);
108
109         }
110
111
112         //Blocks a picturebox when clicked
113         private void pictureBox_block(object sender, EventArgs e)
114         {
115
116                 PictureBox pb = (PictureBox)sender;
117
118
119                 if (pb.Parent.Equals(tableLayoutPanelGrid))
120                 {
121
122                     int r = tableLayoutPanelGrid.GetRow(pb);
123                     int c = tableLayoutPanelGrid.GetColumn(pb);
124                 if (r != activeRow || c != activeColumn)
125                 {
126                     if (grid[r, c].getBlocked())
127                     {
128                         pb.Image = blank;
```

```
129                         grid[r, c].setBlocked();
130                     }
131                     else
132                     {
133                         pb.Image = block;
134                         grid[r, c].setBlocked();
135                     }
136                 }
137 
138                 }
139 
140 
141         }
142 
143 
144         //************************************
145         // Dragging events
146         //************************************
147         private void RTBProg_DragEnter(object sender, DragEventArgs e)
148         {
149             RTBprog.Focus();
150             if (e.Data.GetDataPresent(DataFormats.Text))
151             {
152                 e.Effect = DragDropEffects.Copy;
153             }
154         }
155         private void RTBProg_DragDrop(object sender,
156         System.Windows.Forms.DragEventArgs e)
157         {
158 
159 
160             RTBprog.Focus();
161             int index = RTBprog.GetCharIndexFromPosition(RTBprog.PointToClient
                  (Cursor.Position));
162             RTBprog.SelectionStart = index;
163             RTBprog.SelectionLength = 0;
164             Point cp = RTBprog.GetPositionFromCharIndex(index);
165             char c = RTBprog.GetCharFromPosition(cp);
166 
167             //If statements to determine if new line should be entered
168             if (RTBprog.GetCharFromPosition(cp) == '\0')
169             {
170 
171                 RTBprog.Text = RTBprog.Text.Insert(RTBprog.SelectionStart,
                      e.Data.GetData(DataFormats.Text).ToString());
172 
173             }
174             else if (RTBprog.GetCharFromPosition(cp) == '\n')
175             {
```

```
176
177                 RTBprog.Text = RTBprog.Text.Insert(RTBprog.SelectionStart + 1,
                      e.Data.GetData(DataFormats.Text).ToString());
178
179             }
180             else
181             {
182
183                 RTBprog.Text = RTBprog.Text.Insert(RTBprog.SelectionStart,
                      e.Data.GetData(DataFormats.Text).ToString());
184             }
185             RTBprog.Select(RTBprog.Text.Length, 0);
186
187         }
188         private void pictureBox_MouseDown(object sender,
              System.Windows.Forms.MouseEventArgs e)
189         {
190
191             //Assign action name to be dropped into textbox
192             PictureBox pb = (PictureBox)sender;
193             if (pb.Name == "pictureBoxMove")
194             {
195                 pb.DoDragDrop("Move\n", DragDropEffects.All);
196             }
197             else if (pb.Name == "pictureBoxTurnLeft")
198             {
199                 pb.DoDragDrop("TurnLeft\n", DragDropEffects.All);
200             }
201             else if (pb.Name == "pictureBoxTurnRight")
202             {
203                 pb.DoDragDrop("TurnRight\n", DragDropEffects.All);
204             }
205             else if (pb.Name == "pictureBoxTurnAround")
206             {
207                 pb.DoDragDrop("TurnAround\n", DragDropEffects.All);
208             }
209             else if (pb.Name == "pictureBoxPaint")
210             {
211                 pb.DoDragDrop("Paint\n", DragDropEffects.All);
212             }
213             else if (pb.Name == "pictureBoxErase")
214             {
215                 pb.DoDragDrop("Erase\n", DragDropEffects.All);
216             }
217             else if (pb.Name == "pictureBoxWhile")
218             {
219                 pb.DoDragDrop("while(){ \n}\n", DragDropEffects.All);
220             }
221             else if (pb.Name == "pictureBoxFor")
```

```
222             {
223                 pb.DoDragDrop("for( : : ){\n}\n", DragDropEffects.All);
224             }
225             else if (pb.Name == "pictureBoxIf")
226             {
227                 pb.DoDragDrop("if( ){\n}\n", DragDropEffects.All);
228             }
229             else if (pb.Name == "pictureBoxIfElse")
230             {
231                 pb.DoDragDrop("if( ){\n}\nElse{\n}\n", DragDropEffects.All);
232             }
233             else if (pb.Name == "pictureBoxPainted")
234             {
235                 pb.DoDragDrop("isPainted", DragDropEffects.All);
236             }
237             else if (pb.Name == "pictureBoxNotPainted")
238             {
239                 pb.DoDragDrop("notPainted", DragDropEffects.All);
240             }
241             else if (pb.Name == "pictureBoxNotBlocked")
242             {
243                 pb.DoDragDrop("notBlocked", DragDropEffects.All);
244             }
245             else if (pb.Name == "pictureBoxLeftBlocked")
246             {
247                 pb.DoDragDrop("leftBlocked", DragDropEffects.All);
248 
249             }
250             else if (pb.Name == "pictureBoxRightBlocked")
251             {
252                 pb.DoDragDrop("rightBlocked", DragDropEffects.All);
253             }
254             else if (pb.Name == "pictureBoxBehindBlocked")
255             {
256                 pb.DoDragDrop("behindBlocked", DragDropEffects.All);
257             }
258         }
259 
260 
261         //************************************
262         // Running the user's program
263         //************************************
264 
265         //Split program if not done in lesson
266         public void setProg()
267         {
268             s = RTBprog.Text.ToString().ToLower();
269             prog = s.Split('\n');
270 
```

```
271         }
272
273         //When run is clicked
274         public void buttonRun_Click(object sender, EventArgs e)
275         {
276
277             //Disable buttons and get program
278             buttonRun.Enabled = false;
279             buttonReset.Enabled = false;
280             RTBprog.Enabled = false;
281             buttonStep.Enabled = false;
282
283             //Check is program was set in lesson
284             if (!progSet)
285             {
286                 setProg();
287             }
288
289             //Set bools
290             forLoop = false;
291             greaterthan = false;
292             startValue = 0;
293             up = true;
294
295                 //Reset the grid before starting new program
296                 resetGrid();
297                 tableLayoutPanelGrid.BackColor = Color.Green; //Indicate that prog is
                      running
298                 int startIndex = 0; //Start value for highlighting
299
300                 //Create table of compound statements
301                 preCompile();
302
303                 //Check if returing from lesson
304                 if (fromLesson)
305                 {
306                     runLesson(); //Runs without highlighting in textbox
307                 }
308                 else
309                 {
310                     runHighlight(); //Runs with highlighting in textbox
311                 }
312
313
314                 line = 0;
315
316         }
317
318         //When step is clicked
```

```
319         public async void buttonStep_Click(object sender, EventArgs e)
320         {
321
322             //Disable buttons and get program
323             tableLayoutPanelGrid.BackColor = Color.Green;
324             buttonRun.Enabled = false;
325             buttonReset.Enabled = false;
326             RTBprog.Enabled = false;
327             buttonStep.Enabled = false;
328
329             //Check is prog was set in lesson
330             if (!progSet)
331             {
332                 setProg();
333             }
334             string progItem;
335
336             if(line == 0) //Precompiles if start of the prog
337             {
338                 preCompile(); //Gets compound statements
339             }
340
341             if (progDone)  //If program is complete then reset values and grid
342             {
343                 resetGrid();
344                 progDone = false;
345                 line = 0;
346             }
347             else
348             {
349
350                 if (line < prog.Length && !error) //Check that program isn't complete
351                 {
352                     progItem = prog[line];
353                     if (progItem != " " && progItem != "" && progItem != "\r" &&
                          progItem != "\n")
354                     {
355                         if (inTable(line)) //Check if line is in compound statement
                          table
356                         {
357                             if (!fromLesson) //Check if from lesson
358                             {
359                                 startIndex = RTBprog.GetFirstCharIndexFromLine(line);
360                                 RTBprog.Select(startIndex, RTBprog.Lines
                          [line].Length);
361                                 RTBprog.SelectionBackColor =
                          System.Drawing.Color.Yellow;
362                             }
363                             labelAction.Text = "Running: " + progItem;
```

```
364
365                             if (tableCol == 1) //Check if it is a while
366                             {
367                                 if (compoundStatements[returnRow(line), 0] == 1) //
                    Start of a while loop
368                                 {
369                                     x = progItem.Split('('); //Get condition
370                                     condition = x[1];
371
372                                     //Check if condition is true and assign line
373                                     if (condition.Contains("notblocked"))
374                                     {
375                                         if (moveIsValid())
376                                         {
377                                             line++;
378                                         }
379                                         else
380                                         {
381                                             line = compoundStatements[returnRow(line),
                     2] + 1;
382                                         }
383
384                                     }
385                                     else if (condition.Contains("ispainted"))
386                                     {
387                                         if (grid[activeRow, activeColumn].getPaint())
388                                         {
389                                             line++;
390                                         }
391                                         else
392                                         {
393                                             line = compoundStatements[returnRow(line),
                     2] + 1;
394                                         }
395
396                                     }
397                                     else if (condition.Contains("notpainted"))
398                                     {
399                                         if (!grid[activeRow, activeColumn].getPaint())
400                                         {
401                                             line++;
402                                         }
403                                         else
404                                         {
405                                             line = compoundStatements[returnRow(line),
                     2] + 1;
406                                         }
407
408                                     }
```

```
409                                 else if (condition.Contains("leftblocked"))
410                                 {
411                                     if (direction == 1)
412                                     {
413                                         if (grid[activeRow - 1,
                    activeColumn].getBlocked())
414                                         {
415                                             line++;
416                                         }
417                                         else
418                                         {
419                                             line = compoundStatements[returnRow
                    (line), 2] + 1;
420                                         }
421                                     }
422                                     else if (direction == 2)
423                                     {
424                                         if (grid[activeRow, activeColumn -
                    1].getBlocked())
425                                         {
426                                             line++;
427                                         }
428                                         else
429                                         {
430                                             line = compoundStatements[returnRow
                    (line), 2] + 1;
431                                         }
432                                     }
433                                     else if (direction == 3)
434                                     {
435                                         if (grid[activeRow + 1,
                    activeColumn].getBlocked())
436                                         {
437                                             line++;
438                                         }
439                                         else
440                                         {
441                                             line = compoundStatements[returnRow
                    (line), 2] + 1;
442                                         }
443                                     }
444                                     else
445                                     {
446                                         if (grid[activeRow, activeColumn +
                    1].getBlocked())
447                                         {
448                                             line++;
449                                         }
450                                         else
```

```
451                                         {
452                                             line = compoundStatements[returnRow
                        (line), 2] + 1;
453                                         }
454                                     }
455
456                                 }
457                             }
458                             else if (compoundStatements[returnRow(line), 0] ==
                        2) //Start of an if statement
459                             {
460
461                                 x = progItem.Split('('); //Get condition
462                                 condition = x[1];
463
464                                 //Check if condition is true
465                                 if (condition.Contains("notblocked"))
466                                 {
467                                     if (moveIsValid())
468                                     {
469                                         line++;
470                                     }
471                                     else
472                                     {
473                                         runElse = true;
474                                         line = compoundStatements[returnRow(line),
                         2];
475                                     }
476
477                                 }
478                                 else if (condition.Contains("ispainted"))
479                                 {
480                                     if (grid[activeRow, activeColumn].getPaint())
481                                     {
482                                         line++;
483                                     }
484                                     else
485                                     {
486                                         runElse = true;
487                                         line = compoundStatements[returnRow(line),
                         2];
488                                     }
489
490                                 }
491                                 else if (condition.Contains("notpainted"))
492                                 {
493                                     if (!grid[activeRow, activeColumn].getPaint())
494                                     {
495                                         line++;
```

```
                                    }
                                    else
                                    {
                                        runElse = true;
                                        line = compoundStatements[returnRow(line), 2];
                                    }

                                }
                                else if (condition.Contains("leftblocked"))
                                {
                                    if (direction == 1)
                                    {
                                        if (grid[activeRow - 1, activeColumn].getBlocked())
                                        {
                                            line++;
                                        }
                                        else
                                        {
                                            runElse = true;
                                            line = compoundStatements[returnRow(line), 2];
                                        }
                                    }
                                    else if (direction == 2)
                                    {
                                        if (grid[activeRow, activeColumn - 1].getBlocked())
                                        {
                                            line++;
                                        }
                                        else
                                        {
                                            runElse = true;
                                            line = compoundStatements[returnRow(line), 2];
                                        }
                                    }
                                    else if (direction == 3)
                                    {
                                        if (grid[activeRow + 1, activeColumn].getBlocked())
                                        {
                                            line++;
                                        }
                                        else
                                        {
                                            runElse = true;
```

```
539                                             line = compoundStatements[returnRow(line), 2];
540                                         }
541                                     }
542                                     else
543                                     {
544                                         if (grid[activeRow, activeColumn + 1].getBlocked())
545                                         {
546                                             line++;
547                                         }
548                                         else
549                                         {
550                                             runElse = true;
551                                             line = compoundStatements[returnRow(line), 2];
552                                         }
553                                     }
554 
555                                 }
556                             }
557                             else if (compoundStatements[returnRow(line), 0] == 3) //Start of an else statement
558                             {
559                                 if (!runElse) //Go to end of else if it shouldn't be run
560                                 {
561                                     line = compoundStatements[returnRow(line), 2] + 1;
562                                 }
563                                 else //Goes to next line to run else
564                                 {
565                                     line++;
566                                 }
567 
568                             }
569                             else //If a for loop
570                             {
571                                 if (!forLoop) //If start of a for loop
572                                 {
573                                     //Set bool and get values for for loop
574                                     forLoop = true;
575                                     x = progItem.Split(':');
576                                     condition = x[1];
577                                     string y = x[0];
578                                     startValue = int.Parse(x[0].Split('=')[1]);
579                                     if (progItem.Contains('>'))
580                                     {
581                                         greaterthan = true;
```

```
582                                     }
583                                     else
584                                     {
585                                         greaterthan = false;
586                                     }
587 
588                                     if (!progItem.Contains('+'))
589                                     {
590                                         up = false;
591                                     }
592                                     else
593                                     {
594                                         up = true;
595                                     }
596                                     loopVariable = startValue;
597                                     line++;
598                                 }
599                                 else
600                                 {
601                                     //Check for loop values and condition
602                                     int endCondition = int.Parse(condition.Split
                        ('<')[1]);
603                                     if (!greaterthan)
604                                     {
605                                         if (up)
606                                         {
607                                             loopVariable++;
608                                             if (loopVariable > endCondition)
609                                             {
610                                                 forLoop = false;
611                                                 line = compoundStatements
                        [returnRow(line), 2] + 1;
612                                             }
613                                             else
614                                             {
615                                                 line++;
616                                             }
617                                         }
618                                         else
619                                         {
620                                             loopVariable--;
621                                             if (loopVariable > endCondition)
622                                             {
623                                                 forLoop = false;
624                                                 line = compoundStatements
                        [returnRow(line), 2] + 1;
625                                             }
626                                             else
627                                             {
```

```
628                                                 line++;
629                                             }
630                                         }
631                                     }
632                                     else
633                                     {
634                                         if (up)
635                                         {
636                                             loopVariable++;
637                                             if (loopVariable < endCondition)
638                                             {
639                                                 forLoop = false;
640                                                 line = compoundStatements
                        [returnRow(line), 2] + 1;
641                                             }
642                                             else
643                                             {
644                                                 line++;
645                                             }
646                                         }
647                                         else
648                                         {
649                                             loopVariable--;
650                                             if (loopVariable < endCondition)
651                                             {
652                                                 forLoop = false;
653                                                 line = compoundStatements
                        [returnRow(line), 2] + 1;
654                                             }
655                                             else
656                                             {
657                                                 line++;
658                                             }
659
660                                         }
661                                     }
662
663                                 }
664
665
666                             }
667                         }
668                         else
669                         {
670                             if (compoundStatements[returnRow(line), 0] == 1) //End
                     of a while
671                             {
672                                 line = compoundStatements[returnRow(line), 1];
673                             }
```

```
674                                 else if (compoundStatements[returnRow(line), 0] ==
                        4) //End of for loop
675                                 {
676                                     line = compoundStatements[returnRow(line), 1];
677                                 }
678                                 else
679                                 {
680                                     line = compoundStatements[returnRow(line), 2] + 1;
681                                 }
682                             }
683                             await Task.Delay(1000);
684                             RTBprog.SelectionBackColor =
                        System.Drawing.Color.Transparent;
685                           }
686                           else
687                           {
688                               if (!fromLesson) //Check if from lesson
689                               {
690                                   startIndex = RTBprog.GetFirstCharIndexFromLine(line);
691                                   RTBprog.Select(startIndex, RTBprog.Lines
                        [line].Length);
692                                   RTBprog.SelectionBackColor =
                        System.Drawing.Color.Yellow;
693                             }
694
695                             //Run the program
696                             labelAction.Text = "Running: " + progItem;
697                             checkAction(progItem);
698                             await Task.Delay(1000);
699                             line++;
700                             RTBprog.SelectionBackColor =
                        System.Drawing.Color.Transparent;
701                           }
702                       }
703                       else
704                       {
705                           line++;
706                       }
707                   }
708                   else
709                   {
710                       if (error)
711                       {
712                           labelAction.Text = "Error on line: " + line + " " + prog
                          [line];
713                       }
714                       else
715                       {
716                           labelAction.Text = "Done Running";
```

```
717                     }
718                     //Enable buttons and reset variables
719
720                     tableLayoutPanelGrid.BackColor = Color.YellowGreen;
721                     buttonRun.Enabled = true;
722                     buttonReset.Enabled = true;
723                     RTBprog.Enabled = true;
724                     loopVariable = 0;
725                     labelAction.Text = "Done Running";
726
727                     progDone = true;
728                     progSet = false;
729                     //end of stepping
730                 }
731             }
732             buttonStep.Enabled = true;
733         }
734
735         //When reset is clicked
736         public void buttonReset_Click(object sender, EventArgs e)
737         {
738             resetGrid();
739         }
740
741         //Runs and highlights text
742         public async void runLesson()
743         {
744             buttonRun.Enabled = false;
745             buttonReset.Enabled = false;
746             RTBprog.Enabled = false;
747             buttonStep.Enabled = false;
748             await Task.Delay(1000);
749             string progItem;
750             while (line < progLength)
751             {
752                 progItem = prog[line];
753                 if (progItem != " " && progItem != "" && progItem != "\r" &&
                      progItem != "\n")
754                 {
755                     if (inTable(line))
756                     {
757
758                         labelAction.Text = "Running: " + progItem;
759                         if (tableCol == 1)
760                         {
761                             if (compoundStatements[returnRow(line), 0] == 1) //Start
                          of a while loop
762                             {
763                                 x = progItem.Split('('); //Get condition
```

```
764                                 condition = x[1];
765
766                                 //Check if condition is true
767                                 if (condition.Contains("notblocked"))
768                                 {
769                                     if (moveIsValid())
770                                     {
771                                         line++;
772                                     }
773                                     else
774                                     {
775                                         line = compoundStatements[returnRow(line), 2]
                        + 1;
776                                     }
777
778                                 }
779                                 else if (condition.Contains("ispainted"))
780                                 {
781                                     if (grid[activeRow, activeColumn].getPaint())
782                                     {
783                                         line++;
784                                     }
785                                     else
786                                     {
787                                         line = compoundStatements[returnRow(line), 2]
                        + 1;
788                                     }
789
790                                 }
791                                 else if (condition.Contains("notpainted"))
792                                 {
793                                     if (!grid[activeRow, activeColumn].getPaint())
794                                     {
795                                         line++;
796                                     }
797                                     else
798                                     {
799                                         line = compoundStatements[returnRow(line), 2]
                        + 1;
800                                     }
801
802                                 }
803                                 else if (condition.Contains("leftblocked"))
804                                 {
805                                     if (direction == 1)
806                                     {
807                                         if (grid[activeRow - 1,
                        activeColumn].getBlocked())
808                                         {
```

```
809                                         line++;
810                                     }
811                                     else
812                                     {
813                                         line = compoundStatements[returnRow(line),
     2] + 1;
814                                     }
815                                 }
816                                 else if (direction == 2)
817                                 {
818                                     if (grid[activeRow, activeColumn -
    1].getBlocked())
819                                     {
820                                         line++;
821                                     }
822                                     else
823                                     {
824                                         line = compoundStatements[returnRow(line),
     2] + 1;
825                                     }
826                                 }
827                                 else if (direction == 3)
828                                 {
829                                     if (grid[activeRow + 1,
    activeColumn].getBlocked())
830                                     {
831                                         line++;
832                                     }
833                                     else
834                                     {
835                                         line = compoundStatements[returnRow(line),
     2] + 1;
836                                     }
837                                 }
838                                 else
839                                 {
840                                     if (grid[activeRow, activeColumn +
    1].getBlocked())
841                                     {
842                                         line++;
843                                     }
844                                     else
845                                     {
846                                         line = compoundStatements[returnRow(line),
     2] + 1;
847                                     }
848                                 }
849
850                             }
```

```
851                                 }
852                                 else if (compoundStatements[returnRow(line), 0] == 2) //
                                Start of an if statement
853                                 {
854
855                                     x = progItem.Split('('); //Get condition
856                                     condition = x[1];
857
858                                     //Check if condition is true
859                                     if (condition.Contains("notblocked"))
860                                     {
861                                         if (moveIsValid())
862                                         {
863                                             line++;
864                                         }
865                                         else
866                                         {
867                                             runElse = true;
868                                             line = compoundStatements[returnRow(line), 2];
869                                         }
870
871                                     }
872                                     else if (condition.Contains("ispainted"))
873                                     {
874                                         if (grid[activeRow, activeColumn].getPaint())
875                                         {
876                                             line++;
877                                         }
878                                         else
879                                         {
880                                             runElse = true;
881                                             line = compoundStatements[returnRow(line), 2];
882                                         }
883
884                                     }
885                                     else if (condition.Contains("notpainted"))
886                                     {
887                                         if (!grid[activeRow, activeColumn].getPaint())
888                                         {
889                                             line++;
890                                         }
891                                         else
892                                         {
893                                             runElse = true;
894                                             line = compoundStatements[returnRow(line), 2];
895                                         }
896
897                                     }
898                                     else if (condition.Contains("leftblocked"))
```

```
                                    {
                                        if (direction == 1)
                                        {
                                            if (grid[activeRow - 1,
                        activeColumn].getBlocked())
                                            {
                                                line++;
                                            }
                                            else
                                            {
                                                runElse = true;
                                                line = compoundStatements[returnRow(line),
                         2];
                                            }
                                        }
                                        else if (direction == 2)
                                        {
                                            if (grid[activeRow, activeColumn -
                        1].getBlocked())
                                            {
                                                line++;
                                            }
                                            else
                                            {
                                                runElse = true;
                                                line = compoundStatements[returnRow(line),
                         2];
                                            }
                                        }
                                        else if (direction == 3)
                                        {
                                            if (grid[activeRow + 1,
                        activeColumn].getBlocked())
                                            {
                                                line++;
                                            }
                                            else
                                            {
                                                runElse = true;
                                                line = compoundStatements[returnRow(line),
                         2];
                                            }
                                        }
                                        else
                                        {
                                            if (grid[activeRow, activeColumn +
                        1].getBlocked())
                                            {
                                                line++;
```

```
941                                         }
942                                         else
943                                         {
944                                             runElse = true;
945                                             line = compoundStatements[returnRow(line), 2];
946                                         }
947                                     }
948 
949                                 }
950                             }
951                             else if (compoundStatements[returnRow(line), 0] == 3) // Start of an else statement
952                             {
953                                 if (!runElse) //Go to end of else if it shouldn't be run
954                                 {
955                                     line = compoundStatements[returnRow(line), 2] + 1;
956                                 }
957                                 else //Goes to next line to run else
958                                 {
959                                     line++;
960                                 }
961 
962                             }
963                             else //If a for loop
964                             {
965                                 if (!forLoop)
966                                 {
967                                     forLoop = true;
968                                     x = progItem.Split(':');
969                                     condition = x[1];
970                                     string y = x[0];
971                                     startValue = int.Parse(x[0].Split('=')[1]);
972                                     if (progItem.Contains('>'))
973                                     {
974                                         greaterthan = true;
975                                     }
976                                     else
977                                     {
978                                         greaterthan = false;
979                                     }
980 
981                                     if (!progItem.Contains('+'))
982                                     {
983                                         up = false;
984                                     }
985                                     else
986                                     {
```

```
                                    up = true;
                                }
                                loopVariable = startValue;
                                line++;
                            }
                            else
                            {
                                int endCondition;
                                if (!greaterthan)
                                {
                                    endCondition = int.Parse(condition.Split('<')[1]);
                                    if (up)
                                    {
                                        loopVariable++;
                                        if (loopVariable >= endCondition)
                                        {
                                            forLoop = false;
                                            line = compoundStatements[returnRow(line), 2] + 1;
                                        }
                                        else
                                        {
                                            line++;
                                        }
                                    }
                                    else
                                    {

                                        loopVariable--;
                                        if (loopVariable >= endCondition)
                                        {
                                            forLoop = false;
                                            line = compoundStatements[returnRow(line), 2] + 1;
                                        }
                                        else
                                        {
                                            line++;
                                        }
                                    }
                                }
                                else
                                {
                                    endCondition = int.Parse(condition.Split('>')[1]);
                                    if (up)
                                    {
                                        loopVariable++;
```

```
1032                                        if (loopVariable <= endCondition)
1033                                        {
1034                                            forLoop = false;
1035                                            line = compoundStatements[returnRow
                        (line), 2] + 1;
1036                                        }
1037                                        else
1038                                        {
1039                                            line++;
1040                                        }
1041                                    }
1042                                    else
1043                                    {
1044                                        loopVariable--;
1045                                        if (loopVariable <= endCondition)
1046                                        {
1047                                            forLoop = false;
1048                                            line = compoundStatements[returnRow
                        (line), 2] + 1;
1049                                        }
1050                                        else
1051                                        {
1052                                            line++;
1053                                        }
1054
1055                                    }
1056                                }
1057
1058                                }
1059
1060
1061                            }
1062                        }
1063                        else
1064                        {
1065                            if (compoundStatements[returnRow(line), 0] == 1) //End of
                        a while
1066                            {
1067                                line = compoundStatements[returnRow(line), 1];
1068                            }
1069                            else if (compoundStatements[returnRow(line), 0] == 4) //
                        End of for loop
1070                            {
1071                                line = compoundStatements[returnRow(line), 1];
1072                            }
1073                            else
1074                            {
1075                                line = compoundStatements[returnRow(line), 2] + 1;
1076                            }
```

```
1077                            }
1078                            await Task.Delay(1000);
1079
1080                        }
1081                        else
1082                        {
1083
1084                            labelAction.Text = "Running: " + progItem;
1085                            checkAction(progItem);
1086                            await Task.Delay(1000);
1087                            line++;
1088
1089                        }
1090
1091                    }
1092                    else { line++; }
1093                }
1094
1095                tableLayoutPanelGrid.BackColor = Color.YellowGreen;
1096                buttonRun.Enabled = true;
1097                buttonReset.Enabled = true;
1098                buttonStep.Enabled = true;
1099                RTBprog.Enabled = true;
1100                labelAction.Text = "Done Running";
1101                progDone = true;
1102                progSet = false;
1103
1104
1105            }
1106
1107            //Runs without highlighting text
1108            public async void runHighlight()
1109            {
1110
1111                await Task.Delay(1000);
1112                string progItem;
1113
1114
1115                while (line < progLength && !error)
1116                {
1117                    progItem = prog[line];
1118                    if (progItem != " " && progItem != "" && progItem != "\r" &&
                        progItem != "\n")
1119                    {
1120                        if (inTable(line))
1121                        {
1122                            if (!fromLesson)
1123                            {
1124                                startIndex = RTBprog.GetFirstCharIndexFromLine(line);
```

```
1125                            RTBprog.Select(startIndex, RTBprog.Lines[line].Length);
1126                            RTBprog.SelectionBackColor = System.Drawing.Color.Yellow;
1127
1128                        }
1129                        labelAction.Text = "Running: " + progItem;
1130                        if (tableCol == 1)
1131                        {
1132                            if (compoundStatements[returnRow(line), 0] == 1) //Start
                        of a while loop
1133                            {
1134                                x = progItem.Split('('); //Get condition
1135                                condition = x[1];
1136
1137                                //Check if condition is true
1138                                if (condition.Contains("notblocked"))
1139                                {
1140                                    if (moveIsValid())
1141                                    {
1142                                        line++;
1143                                    }
1144                                    else
1145                                    {
1146                                        line = compoundStatements[returnRow(line), 2]
                        + 1;
1147                                    }
1148
1149                                }
1150                                else if (condition.Contains("ispainted"))
1151                                {
1152                                    if (grid[activeRow, activeColumn].getPaint())
1153                                    {
1154                                        line++;
1155                                    }
1156                                    else
1157                                    {
1158                                        line = compoundStatements[returnRow(line), 2]
                        + 1;
1159                                    }
1160
1161                                }
1162                                else if (condition.Contains("notpainted"))
1163                                {
1164                                    if (!grid[activeRow, activeColumn].getPaint())
1165                                    {
1166                                        line++;
1167                                    }
1168                                    else
1169                                    {
1170                                        line = compoundStatements[returnRow(line), 2]
```

```
                        + 1;
1171                            }
1172
1173                        }
1174                        else if (condition.Contains("leftblocked"))
1175                        {
1176                            if (direction == 1)
1177                            {
1178                                if (grid[activeRow - 1,
                        activeColumn].getBlocked())
1179                                {
1180                                    line++;
1181                                }
1182                                else
1183                                {
1184                                    line = compoundStatements[returnRow(line),
                         2] + 1;
1185                                }
1186                            }
1187                            else if (direction == 2)
1188                            {
1189                                if (grid[activeRow, activeColumn -
                        1].getBlocked())
1190                                {
1191                                    line++;
1192                                }
1193                                else
1194                                {
1195                                    line = compoundStatements[returnRow(line),
                         2] + 1;
1196                                }
1197                            }
1198                            else if (direction == 3)
1199                            {
1200                                if (grid[activeRow + 1,
                        activeColumn].getBlocked())
1201                                {
1202                                    line++;
1203                                }
1204                                else
1205                                {
1206                                    line = compoundStatements[returnRow(line),
                         2] + 1;
1207                                }
1208                            }
1209                            else
1210                            {
1211                                if (grid[activeRow, activeColumn +
                        1].getBlocked())
```

```
                                        {
                                            line++;
                                        }
                                        else
                                        {
                                            line = compoundStatements[returnRow(line), 2] + 1;
                                        }
                                    }

                                }
                            }
                            else if (compoundStatements[returnRow(line), 0] == 2) // Start of an if statement
                            {

                                x = progItem.Split('('); //Get condition
                                condition = x[1];

                                //Check if condition is true
                                if (condition.Contains("notblocked"))
                                {
                                    if (moveIsValid())
                                    {
                                        line++;
                                    }
                                    else
                                    {
                                        runElse = true;
                                        line = compoundStatements[returnRow(line), 2];
                                    }

                                }
                                else if (condition.Contains("ispainted"))
                                {
                                    if (grid[activeRow, activeColumn].getPaint())
                                    {
                                        line++;
                                    }
                                    else
                                    {
                                        runElse = true;
                                        line = compoundStatements[returnRow(line), 2];
                                    }

                                }
                                else if (condition.Contains("notpainted"))
                                {
                                    if (!grid[activeRow, activeColumn].getPaint())
```

```
                                    {
                                        line++;
                                    }
                                    else
                                    {
                                        runElse = true;
                                        line = compoundStatements[returnRow(line), 2];
                                    }

                                }
                                else if (condition.Contains("leftblocked"))
                                {
                                    if (direction == 1)
                                    {
                                        if (grid[activeRow - 1, activeColumn].getBlocked())
                                        {
                                            line++;
                                        }
                                        else
                                        {
                                            runElse = true;
                                            line = compoundStatements[returnRow(line), 2];
                                        }
                                    }
                                    else if (direction == 2)
                                    {
                                        if (grid[activeRow, activeColumn - 1].getBlocked())
                                        {
                                            line++;
                                        }
                                        else
                                        {
                                            runElse = true;
                                            line = compoundStatements[returnRow(line), 2];
                                        }
                                    }
                                    else if (direction == 3)
                                    {
                                        if (grid[activeRow + 1, activeColumn].getBlocked())
                                        {
                                            line++;
                                        }
                                        else
                                        {
```

```
                                        runElse = true;
                                        line = compoundStatements[returnRow(line), 2];
                                    }
                                }
                                else
                                {
                                    if (grid[activeRow, activeColumn + 1].getBlocked())
                                    {
                                        line++;
                                    }
                                    else
                                    {
                                        runElse = true;
                                        line = compoundStatements[returnRow(line), 2];
                                    }
                                }

                            }
                        }
                        else if (compoundStatements[returnRow(line), 0] == 3) // Start of an else statement
                        {
                            if (!runElse) //Go to end of else if it shouldn't be run
                            {
                                line = compoundStatements[returnRow(line), 2] + 1;
                            }
                            else //Goes to next line to run else
                            {
                                line++;
                                runElse = false;
                            }

                        }
                        else //If a for loop
                        {
                            x = progItem.Split(':');
                            condition = x[1];
                            string y = x[0];
                            if (!forLoop)
                            {
                                forLoop = true;

                                startValue = int.Parse(x[0].Split('=')[1]);
                                if (progItem.Contains('>'))
                                {
```

```
1347                                    greaterthan = true;
1348                                }
1349                                else
1350                                {
1351                                    greaterthan = false;
1352                                }
1353
1354                                if (!progItem.Contains('+'))
1355                                {
1356                                    up = false;
1357                                }
1358                                else
1359                                {
1360                                    up = true;
1361                                }
1362                                loopVariable = startValue;
1363                                line++;
1364                            }
1365                            else
1366                            {
1367                                int endCondition;
1368                                if (!greaterthan)
1369                                {
1370                                    endCondition = int.Parse(condition.Split('<')
                        [1]);
1371                                    if (up)
1372                                    {
1373                                        loopVariable++;
1374                                        if (loopVariable > endCondition)
1375                                        {
1376                                            forLoop = false;
1377                                            line = compoundStatements[returnRow
                        (line), 2] + 1;
1378                                        }
1379                                        else
1380                                        {
1381                                            line++;
1382                                        }
1383                                    }
1384                                    else
1385                                    {
1386
1387                                        loopVariable--;
1388                                        if (loopVariable > endCondition)
1389                                        {
1390                                            forLoop = false;
1391                                            line = compoundStatements[returnRow
                        (line), 2] + 1;
1392                                        }
```

```
1393                                            else
1394                                            {
1395                                                line++;
1396                                            }
1397                                        }
1398                                    }
1399                                    else
1400                                    {
1401                                        endCondition = int.Parse(condition.Split('>')
                        [1]);
1402                                        if (up)
1403                                        {
1404                                            loopVariable++;
1405                                            if (loopVariable < endCondition)
1406                                            {
1407                                                forLoop = false;
1408                                                line = compoundStatements[returnRow
                        (line), 2] + 1;
1409                                            }
1410                                            else
1411                                            {
1412                                                line++;
1413                                            }
1414                                        }
1415                                        else
1416                                        {
1417                                            loopVariable--;
1418                                            if (loopVariable < endCondition)
1419                                            {
1420                                                forLoop = false;
1421                                                line = compoundStatements[returnRow
                        (line), 2] + 1;
1422                                            }
1423                                            else
1424                                            {
1425                                                line++;
1426                                            }
1427
1428                                        }
1429                                    }
1430
1431                                }
1432
1433
1434                            }
1435                        }
1436                        else
1437                        {
1438                            if (compoundStatements[returnRow(line), 0] == 1) //End of
```

```
                        a while
                              {
                                  line = compoundStatements[returnRow(line), 1];
                              }
                              else if (compoundStatements[returnRow(line), 0] == 4) //
                        End of for loop
                              {
                                  line = compoundStatements[returnRow(line), 1];
                              }
                              else
                              {
                                  line = compoundStatements[returnRow(line), 2] + 1;
                              }
                          }
                          await Task.Delay(1000);
                          RTBprog.SelectionBackColor = System.Drawing.Color.Transparent;
                      }
                      else
                      {

                          startIndex = RTBprog.GetFirstCharIndexFromLine(line);
                          RTBprog.Select(startIndex, RTBprog.Lines[line].Length);
                          RTBprog.SelectionBackColor = System.Drawing.Color.Yellow;

                          labelAction.Text = "Running: " + progItem;
                          checkAction(progItem);
                          await Task.Delay(1000);
                          line++;
                          RTBprog.SelectionBackColor = System.Drawing.Color.Transparent;
                      }

                  }
                  else { line++; }
              }

              if (error)
              {
                  int e = line - 1;
                  labelAction.Text = "Error on line: "+e+" "+prog[line-1];
              }
              else
              {
                  labelAction.Text = "Done Running";
              }

              tableLayoutPanelGrid.BackColor = Color.YellowGreen;
              buttonRun.Enabled = true;
              buttonReset.Enabled = true;
              buttonStep.Enabled = true;
```

```
1486                RTBprog.Enabled = true;
1487                progDone = true;
1488                progSet = false;
1489
1490            }
1491
1492
1493            //************************************
1494            // Helper functions
1495            //************************************
1496
1497
1498            //Returns row in compound statements table
1499            private int returnRow(int lineNum)
1500            {
1501                for (int i = 0; i < 10; i++)
1502                {
1503                    if (compoundStatements[i, 1] == lineNum || compoundStatements[i, 2] == ↵
                        lineNum)
1504                    {
1505                        return i;
1506                    }
1507
1508                }
1509                return -1;
1510            }
1511
1512            //Returns true if line number is in the compound statements table
1513            private bool inTable(int lineNum)
1514            {
1515                for (int i = 0; i < 10; i++)
1516                {
1517                    if (compoundStatements[i, 1] == lineNum)
1518                    {
1519                        tableRow = i;
1520                        tableCol = 1;
1521                        return true;
1522                    }
1523                    else if (compoundStatements[i, 2] == lineNum)
1524                    {
1525                        tableRow = i;
1526                        tableCol = 2;
1527                        return true;
1528                    }
1529                }
1530
1531                return false;
1532
1533            }
```

```
1534
1535        //Creates compound statements table
1536        private void preCompile()
1537        {
1538            int row = 0;
1539            int col = 0;
1540            int line = 0;
1541            int[] close = new int[10]; //Holds lines of closing brackets
1542
1543
1544            for (int i = 0; i < 10; i++)
1545            {
1546                compoundStatements[i, 2] = -1;
1547                compoundStatements[i, 1] = -1;
1548                compoundStatements[i, 0] = -1;
1549            }
1550
1551            foreach (string progItem in prog)
1552            {
1553
1554                if (progItem.Contains("while"))
1555                {
1556                    compoundStatements[row, col] = 1; //Shows me it is a while
1557                    col++;
1558                    compoundStatements[row, col] = line; //Line number of open
1559                    row++;
1560
1561
1562                }
1563                else if (progItem.Contains("if"))
1564                {
1565                    compoundStatements[row, col] = 2; //Shows me it is a if
1566                    col++;
1567
1568                    compoundStatements[row, col] = line; //Line number of open
1569                    row++;
1570
1571
1572                }
1573                else if (progItem.Contains("else")) //shows me it is a else
1574                {
1575                    compoundStatements[row, col] = 3;
1576                    col++;
1577
1578                    compoundStatements[row, col] = line; //Line number of open
1579                    row++;
1580
1581
1582                }
```

```
1583                else if (progItem.Contains("for"))
1584                {
1585                    compoundStatements[row, col] = 4;//Shows me it is a for
1586                    col++;
1587
1588                    compoundStatements[row, col] = line; //Line number of open
1589                    row++;
1590
1591
1592                }
1593                else if (progItem.Contains("}"))
1594                {
1595                    //Start at bottom and find the first zero starting at current row
1596
1597                    for (int i = row - 1; i >= 0; i--)
1598                    {
1599                        if (compoundStatements[i, 2] == -1)
1600                        {
1601                            compoundStatements[i, 2] = line;
1602                            break;
1603                        }
1604                    }
1605
1606
1607                }
1608
1609                line++;
1610
1611                if (col >= 1) //Reset columns if row is done
1612                {
1613                    col = 0;
1614                }
1615            }
1616
1617            progLength = line;
1618
1619        }
1620
1621        //Returns next row in current direction
1622        private int nextRow()
1623        {
1624            if (direction == 2)
1625            {
1626                return activeRow - 1;
1627            }
1628            else if (direction == 4)
1629            {
1630                return activeRow + 1;
1631            }
```

```
1632                 else
1633                 {
1634                     return activeRow;
1635                 }
1636             }
1637 
1638             //Returns next col in current direction
1639             private int nextCol()
1640             {
1641                 if (direction == 1)
1642                 {
1643                     return activeColumn + 1;
1644                 }
1645                 else if (direction == 3)
1646                 {
1647                     return activeColumn - 1;
1648                 }
1649                 else
1650                 {
1651                     return activeColumn;
1652                 }
1653             }
1654 
1655             //Checks if next move in direction is blocked
1656             private bool moveIsValid()
1657             {
1658                 int r = nextRow();
1659                 int c = nextCol();
1660                 if ((r <= gridSize - 1 && r >= 0) && (c <= gridSize - 1 && c >= 0)) //Make ↵
                        sure move is not out of range
1661                 {
1662 
1663                     if (grid[r, c].getBlocked())//Make sure move is not blocked
1664                     {
1665                         return false;
1666                     }
1667                     else
1668                     {
1669                         return true;
1670                     }
1671                 }
1672                 else
1673                 {
1674                     return false;
1675                 }
1676             }
1677 
1678             //Resets the grid
1679             public void resetGrid()
```

```
1680         {
1681             line = 0;
1682             tableLayoutPanelGrid.BackColor = Color.DarkGray;
1683             error = false;
1684             foreach (box b in grid) //Reset each box
1685             {
1686                 b.reset();
1687             }
1688 
1689             grid[6, 0].setImage(karela); //Reset Karela to start
1690             activeColumn = 0;
1691             activeRow = 6;
1692             while (direction != 1)
1693             {
1694                 turn();
1695             }
1696 
1697 
1698 
1699         }
1700 
1701         //Checks action and calls specific action function
1702         public void checkAction(string action)
1703         {
1704             if (action.Contains("move")) { move(); }
1705             else if (action.Contains("turnleft")) { turn(); }
1706             else if (action.Contains("turnright")) { turnRight(); }
1707             else if (action.Contains("turnaround")) { turnAround(); }
1708             else if (action.Contains("paint"))
1709             {
1710                 if (!grid[activeRow, activeColumn].getPaint())
1711                 {
1712                     grid[activeRow, activeColumn].setImage(KarelaPaint);
1713                     grid[activeRow, activeColumn].setPaint();
1714                 }
1715             }
1716             else if (action.Contains("erase"))
1717             {
1718                 grid[activeRow, activeColumn].setImage(karela);
1719                 grid[activeRow, activeColumn].setPaint();
1720             }
1721         }
1722 
1723 
1724         //************************************
1725         // Movement functions
1726         //************************************
1727         public void turn()
1728         {
```

```
            if (grid[activeRow, activeColumn].getPaint())
            {
                karela.RotateFlip(RotateFlipType.Rotate270FlipNone);
                KarelaPaint.RotateFlip(RotateFlipType.Rotate270FlipNone);
                grid[activeRow, activeColumn].setImage(KarelaPaint);

            }
            else
            {
                karela.RotateFlip(RotateFlipType.Rotate270FlipNone);
                KarelaPaint.RotateFlip(RotateFlipType.Rotate270FlipNone);
                grid[activeRow, activeColumn].setImage(karela);
            }
            if (direction + 1 <= 4)
            {
                direction++;
            }
            else
            {
                direction = 1;
            }

        }
        public void turnRight()
        {
            turn();
            turn();
            turn();
        }
        public void turnAround()
        {
            turn();
            turn();
        }
        public void move()
        {


            if (moveIsValid())
            {
                int r = nextRow();
                int c = nextCol();
                if (grid[activeRow, activeColumn].getPaint())
                {
                    grid[activeRow, activeColumn].setImage(painted);
                }
                else
                {
                    grid[activeRow, activeColumn].setImage(blank);
```

```
1778                }
1779
1780
1781                if (grid[r, c].getPaint())
1782                {
1783                    grid[r, c].setImage(KarelaPaint);
1784                }
1785                else
1786                {
1787                    grid[r, c].setImage(karela);
1788                }
1789
1790                activeRow = r;
1791                activeColumn = c;
1792
1793            }
1794            else
1795            {
1796                error = true;
1797            }
1798        }
1799
1800        //************************************
1801        // Lesson
1802        //************************************
1803
1804        //Opens lesson form on button click
1805        private void buttonLesson_Click(object sender, EventArgs e)
1806        {
1807            Lesson l = new Lesson(this, lesson);
1808            l.Show();
1809
1810        }
1811
1812        //Changes what lesson is selected
1813        private void listBoxLessons_SelectedIndexChanged(object sender, EventArgs e)
1814        {
1815            string selectedLesson = listBoxLessons.SelectedItems[0].ToString();
1816            if (selectedLesson == "Introduction")
1817            {
1818                lesson = 0;
1819            }
1820            if (selectedLesson == "Lesson 1")
1821            {
1822                lesson = 1;
1823            }
1824            else if (selectedLesson == "Lesson 2")
1825            {
1826                lesson = 2;
```

```
1827            }
1828            else if (selectedLesson == "Lesson 3")
1829            {
1830                lesson = 3;
1831            }
1832            else if (selectedLesson == "Lesson 4")
1833            {
1834                lesson = 4;
1835            }
1836            else if (selectedLesson == "Lesson 5")
1837            {
1838                lesson = 5;
1839            }
1840            else if (selectedLesson == "Lesson 6")
1841            {
1842                lesson = 6;
1843            }
1844            else if (selectedLesson == "Lesson 7")
1845            {
1846                lesson = 7;
1847            }
1848
1849        }
1850
1851        //************************************
1852        // Saving and loading files
1853        //************************************
1854
1855        //Opens save dialog to save program as .txt
1856        private void buttonSave_Click(object sender, EventArgs e)
1857        {
1858            saveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
1859            saveFileDialog1.FilterIndex = 2;
1860            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
1861            {
1862                File.WriteAllText(saveFileDialog1.FileName, RTBprog.Text);
1863                fileName = saveFileDialog1.FileName;
1864            }
1865
1866        }
1867        private void RTBprog_TextChanged(object sender, EventArgs e)
1868        {
1869            fileModified = true;
1870        }
1871
1872        //Opens load dialog to load a .txt file
1873        private void buttonLoad_Click(object sender, EventArgs e)
1874        {
1875            string filePath;
```

```
1876            if (!fileModified)
1877            {
1878                openFileDialog1.InitialDirectory = "c:\\";
1879                openFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|
                      *.*";
1880                openFileDialog1.FilterIndex = 2;
1881                openFileDialog1.RestoreDirectory = true;
1882
1883                if (openFileDialog1.ShowDialog() == DialogResult.OK)
1884                {
1885                    //Get the path of specified file
1886                    filePath = openFileDialog1.FileName;
1887                    RTBprog.Text = File.ReadAllText(filePath);
1888
1889                }
1890            }
1891            else
1892            {
1893                DialogResult result = MessageBox.Show(fileName, "Do you wish to
                      save?", MessageBoxButtons.YesNoCancel);
1894                if (result == DialogResult.Yes)
1895                {
1896                    saveFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|
                          *.*";
1897                    saveFileDialog1.FilterIndex = 2;
1898                    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
1899                    {
1900                        File.WriteAllText(saveFileDialog1.FileName, RTBprog.Text);
1901                        fileName = saveFileDialog1.FileName;
1902                    }
1903
1904                    fileModified = false;
1905                }
1906                else
1907                {
1908                    openFileDialog1.InitialDirectory = "c:\\";
1909                    openFileDialog1.Filter = "txt files (*.txt)|*.txt|All files (*.*)|
                          *.*";
1910                    openFileDialog1.FilterIndex = 2;
1911                    openFileDialog1.RestoreDirectory = true;
1912
1913                    if (openFileDialog1.ShowDialog() == DialogResult.OK)
1914                    {
1915                        //Get the path of specified file
1916                        filePath = openFileDialog1.FileName;
1917                        RTBprog.Text = File.ReadAllText(filePath);
1918                    }
1919                }
1920            }
```

```
1921
1922            }
1923
1924            //************************************
1925            // Enabling and disabling buttons on focus change
1926            //************************************
1927            private void main_Activate(object sender, EventArgs e)
1928            {
1929                buttonRun.Enabled = true;
1930                buttonReset.Enabled = true;
1931                buttonStep.Enabled = true;
1932                RTBprog.Enabled = true;
1933            }
1934            private void main_Deactivate(object sender, EventArgs e)
1935            {
1936                buttonRun.Enabled = false;
1937                buttonReset.Enabled = false;
1938                buttonStep.Enabled = false;
1939                RTBprog.Enabled = false;
1940            }
1941
1942            //Opens reference sheet
1943            private void buttonRef_Click(object sender, EventArgs e)
1944            {
1945                System.Diagnostics.Process.Start(Environment.CurrentDirectory+ "//Karela-
                      References-and-Syntax-Sheet.txt");
1946            }
1947
1948
1949        }
1950    }
1951
```