

```
1 import javafx.application.Application;
2 import java.util.*;
3
4 import javafx.application.Platform;
5 import javafx.collections.FXCollections;
6 import javafx.collections.ObservableList;
7 import javafx.geometry.Point2D;
8 import javafx.scene.control.Label;
9 import javafx.event.ActionEvent;
10 import javafx.event.EventHandler;
11 import javafx.event.EventType;
12 import javafx.scene.Scene;
13 import javafx.scene.layout.Pane;
14 import javafx.scene.layout.BorderPane;
15 import javafx.scene.layout.HBox;
16 import javafx.scene.Node;
17 import javafx.scene.control.Button;
18 import javafx.scene.control.TextField;
19 import javafx.scene.input.MouseEvent;
20 import javafx.event.EventHandler;
21 import javafx.scene.layout.StackPane;
22 import javafx.stage.Stage;
23
24 public class ExpressionEditor extends Application {
25     public static void main (String[] args) {
26         launch(args);
27     }
28
29     /**
30      * Mouse event handler for the entire pane that constitutes the
31      * ExpressionEditor
32     */
33     private static class MouseEventHandler implements EventHandler<MouseEvent>
34     {
35         private int focusLevel;
36         private Pane _pane;
37         private CompoundExpression _rootExpression;
38         private Expression selectedExp;
39         private Node deepCopy;
40         private CompoundExpression selectedExpParent;
41         private HBox helpMeP;
42         private Node selectedExpNode;
43
44         private List<Expression> expressionList;
45
46         private double curIndex;
47
48         private boolean wasSwapped;
49
50         MouseEventHandler(Pane pane_, CompoundExpression rootExpression_) {
51             _pane = pane_;
52             _rootExpression = rootExpression_;
53         }
54
55         public void handle(MouseEvent event) {
56             if (event.getEventType() == MouseEvent.MOUSE_PRESSED) {
57                 highlight(event);
58             }
59         }
60
61         private void highlight(MouseEvent event) {
62             if (selectedExp != null) {
63                 selectedExp.setMouseTransparent(true);
64             }
65
66             if (selectedExpParent != null) {
67                 selectedExpParent.setMouseTransparent(true);
68             }
69
70             if (deepCopy != null) {
71                 deepCopy.setMouseTransparent(true);
72             }
73
74             if (helpMeP != null) {
75                 helpMeP.setMouseTransparent(true);
76             }
77
78             if (_rootExpression != null) {
79                 _rootExpression.setMouseTransparent(true);
80             }
81
82             if (_pane != null) {
83                 _pane.setMouseTransparent(true);
84             }
85
86             if (expressionList != null) {
87                 expressionList.setMouseTransparent(true);
88             }
89
90             if (selectedExpNode != null) {
91                 selectedExpNode.setMouseTransparent(true);
92             }
93
94             if (wasSwapped) {
95                 wasSwapped = false;
96             }
97
98             if (curIndex > 0) {
99                 curIndex--;
100            }
101
102            if (curIndex < expressionList.size()) {
103                selectedExp = expressionList.get((int) curIndex);
104            }
105
106            if (selectedExp != null) {
107                selectedExp.setMouseTransparent(false);
108            }
109
110            if (selectedExpParent != null) {
111                selectedExpParent.setMouseTransparent(false);
112            }
113
114            if (deepCopy != null) {
115                deepCopy.setMouseTransparent(false);
116            }
117
118            if (helpMeP != null) {
119                helpMeP.setMouseTransparent(false);
120            }
121
122            if (_rootExpression != null) {
123                _rootExpression.setMouseTransparent(false);
124            }
125
126            if (_pane != null) {
127                _pane.setMouseTransparent(false);
128            }
129
130            if (expressionList != null) {
131                expressionList.setMouseTransparent(false);
132            }
133
134            if (selectedExpNode != null) {
135                selectedExpNode.setMouseTransparent(false);
136            }
137
138            wasSwapped = true;
139
140            if (selectedExp != null) {
141                selectedExp.setMouseTransparent(true);
142            }
143
144            if (selectedExpParent != null) {
145                selectedExpParent.setMouseTransparent(true);
146            }
147
148            if (deepCopy != null) {
149                deepCopy.setMouseTransparent(true);
150            }
151
152            if (helpMeP != null) {
153                helpMeP.setMouseTransparent(true);
154            }
155
156            if (_rootExpression != null) {
157                _rootExpression.setMouseTransparent(true);
158            }
159
160            if (_pane != null) {
161                _pane.setMouseTransparent(true);
162            }
163
164            if (expressionList != null) {
165                expressionList.setMouseTransparent(true);
166            }
167
168            if (selectedExpNode != null) {
169                selectedExpNode.setMouseTransparent(true);
170            }
171
172            if (selectedExp != null) {
173                selectedExp.setMouseTransparent(false);
174            }
175
176            if (selectedExpParent != null) {
177                selectedExpParent.setMouseTransparent(false);
178            }
179
180            if (deepCopy != null) {
181                deepCopy.setMouseTransparent(false);
182            }
183
184            if (helpMeP != null) {
185                helpMeP.setMouseTransparent(false);
186            }
187
188            if (_rootExpression != null) {
189                _rootExpression.setMouseTransparent(false);
190            }
191
192            if (_pane != null) {
193                _pane.setMouseTransparent(false);
194            }
195
196            if (expressionList != null) {
197                expressionList.setMouseTransparent(false);
198            }
199
200            if (selectedExpNode != null) {
201                selectedExpNode.setMouseTransparent(false);
202            }
203
204            if (selectedExp != null) {
205                selectedExp.setMouseTransparent(true);
206            }
207
208            if (selectedExpParent != null) {
209                selectedExpParent.setMouseTransparent(true);
210            }
211
212            if (deepCopy != null) {
213                deepCopy.setMouseTransparent(true);
214            }
215
216            if (helpMeP != null) {
217                helpMeP.setMouseTransparent(true);
218            }
219
220            if (_rootExpression != null) {
221                _rootExpression.setMouseTransparent(true);
222            }
223
224            if (_pane != null) {
225                _pane.setMouseTransparent(true);
226            }
227
228            if (expressionList != null) {
229                expressionList.setMouseTransparent(true);
230            }
231
232            if (selectedExpNode != null) {
233                selectedExpNode.setMouseTransparent(true);
234            }
235
236            if (selectedExp != null) {
237                selectedExp.setMouseTransparent(false);
238            }
239
240            if (selectedExpParent != null) {
241                selectedExpParent.setMouseTransparent(false);
242            }
243
244            if (deepCopy != null) {
245                deepCopy.setMouseTransparent(false);
246            }
247
248            if (helpMeP != null) {
249                helpMeP.setMouseTransparent(false);
250            }
251
252            if (_rootExpression != null) {
253                _rootExpression.setMouseTransparent(false);
254            }
255
256            if (_pane != null) {
257                _pane.setMouseTransparent(false);
258            }
259
260            if (expressionList != null) {
261                expressionList.setMouseTransparent(false);
262            }
263
264            if (selectedExpNode != null) {
265                selectedExpNode.setMouseTransparent(false);
266            }
267
268            if (selectedExp != null) {
269                selectedExp.setMouseTransparent(true);
270            }
271
272            if (selectedExpParent != null) {
273                selectedExpParent.setMouseTransparent(true);
274            }
275
276            if (deepCopy != null) {
277                deepCopy.setMouseTransparent(true);
278            }
279
280            if (helpMeP != null) {
281                helpMeP.setMouseTransparent(true);
282            }
283
284            if (_rootExpression != null) {
285                _rootExpression.setMouseTransparent(true);
286            }
287
288            if (_pane != null) {
289                _pane.setMouseTransparent(true);
290            }
291
292            if (expressionList != null) {
293                expressionList.setMouseTransparent(true);
294            }
295
296            if (selectedExpNode != null) {
297                selectedExpNode.setMouseTransparent(true);
298            }
299
300            if (selectedExp != null) {
301                selectedExp.setMouseTransparent(false);
302            }
303
304            if (selectedExpParent != null) {
305                selectedExpParent.setMouseTransparent(false);
306            }
307
308            if (deepCopy != null) {
309                deepCopy.setMouseTransparent(false);
310            }
311
312            if (helpMeP != null) {
313                helpMeP.setMouseTransparent(false);
314            }
315
316            if (_rootExpression != null) {
317                _rootExpression.setMouseTransparent(false);
318            }
319
320            if (_pane != null) {
321                _pane.setMouseTransparent(false);
322            }
323
324            if (expressionList != null) {
325                expressionList.setMouseTransparent(false);
326            }
327
328            if (selectedExpNode != null) {
329                selectedExpNode.setMouseTransparent(false);
330            }
331
332            if (selectedExp != null) {
333                selectedExp.setMouseTransparent(true);
334            }
335
336            if (selectedExpParent != null) {
337                selectedExpParent.setMouseTransparent(true);
338            }
339
340            if (deepCopy != null) {
341                deepCopy.setMouseTransparent(true);
342            }
343
344            if (helpMeP != null) {
345                helpMeP.setMouseTransparent(true);
346            }
347
348            if (_rootExpression != null) {
349                _rootExpression.setMouseTransparent(true);
350            }
351
352            if (_pane != null) {
353                _pane.setMouseTransparent(true);
354            }
355
356            if (expressionList != null) {
357                expressionList.setMouseTransparent(true);
358            }
359
360            if (selectedExpNode != null) {
361                selectedExpNode.setMouseTransparent(true);
362            }
363
364            if (selectedExp != null) {
365                selectedExp.setMouseTransparent(false);
366            }
367
368            if (selectedExpParent != null) {
369                selectedExpParent.setMouseTransparent(false);
370            }
371
372            if (deepCopy != null) {
373                deepCopy.setMouseTransparent(false);
374            }
375
376            if (helpMeP != null) {
377                helpMeP.setMouseTransparent(false);
378            }
379
380            if (_rootExpression != null) {
381                _rootExpression.setMouseTransparent(false);
382            }
383
384            if (_pane != null) {
385                _pane.setMouseTransparent(false);
386            }
387
388            if (expressionList != null) {
389                expressionList.setMouseTransparent(false);
390            }
391
392            if (selectedExpNode != null) {
393                selectedExpNode.setMouseTransparent(false);
394            }
395
396            if (selectedExp != null) {
397                selectedExp.setMouseTransparent(true);
398            }
399
400            if (selectedExpParent != null) {
401                selectedExpParent.setMouseTransparent(true);
402            }
403
404            if (deepCopy != null) {
405                deepCopy.setMouseTransparent(true);
406            }
407
408            if (helpMeP != null) {
409                helpMeP.setMouseTransparent(true);
410            }
411
412            if (_rootExpression != null) {
413                _rootExpression.setMouseTransparent(true);
414            }
415
416            if (_pane != null) {
417                _pane.setMouseTransparent(true);
418            }
419
420            if (expressionList != null) {
421                expressionList.setMouseTransparent(true);
422            }
423
424            if (selectedExpNode != null) {
425                selectedExpNode.setMouseTransparent(true);
426            }
427
428            if (selectedExp != null) {
429                selectedExp.setMouseTransparent(false);
430            }
431
432            if (selectedExpParent != null) {
433                selectedExpParent.setMouseTransparent(false);
434            }
435
436            if (deepCopy != null) {
437                deepCopy.setMouseTransparent(false);
438            }
439
440            if (helpMeP != null) {
441                helpMeP.setMouseTransparent(false);
442            }
443
444            if (_rootExpression != null) {
445                _rootExpression.setMouseTransparent(false);
446            }
447
448            if (_pane != null) {
449                _pane.setMouseTransparent(false);
450            }
451
452            if (expressionList != null) {
453                expressionList.setMouseTransparent(false);
454            }
455
456            if (selectedExpNode != null) {
457                selectedExpNode.setMouseTransparent(false);
458            }
459
460            if (selectedExp != null) {
461                selectedExp.setMouseTransparent(true);
462            }
463
464            if (selectedExpParent != null) {
465                selectedExpParent.setMouseTransparent(true);
466            }
467
468            if (deepCopy != null) {
469                deepCopy.setMouseTransparent(true);
470            }
471
472            if (helpMeP != null) {
473                helpMeP.setMouseTransparent(true);
474            }
475
476            if (_rootExpression != null) {
477                _rootExpression.setMouseTransparent(true);
478            }
479
480            if (_pane != null) {
481                _pane.setMouseTransparent(true);
482            }
483
484            if (expressionList != null) {
485                expressionList.setMouseTransparent(true);
486            }
487
488            if (selectedExpNode != null) {
489                selectedExpNode.setMouseTransparent(true);
490            }
491
492            if (selectedExp != null) {
493                selectedExp.setMouseTransparent(false);
494            }
495
496            if (selectedExpParent != null) {
497                selectedExpParent.setMouseTransparent(false);
498            }
499
500            if (deepCopy != null) {
501                deepCopy.setMouseTransparent(false);
502            }
503
504            if (helpMeP != null) {
505                helpMeP.setMouseTransparent(false);
506            }
507
508            if (_rootExpression != null) {
509                _rootExpression.setMouseTransparent(false);
510            }
511
512            if (_pane != null) {
513                _pane.setMouseTransparent(false);
514            }
515
516            if (expressionList != null) {
517                expressionList.setMouseTransparent(false);
518            }
519
520            if (selectedExpNode != null) {
521                selectedExpNode.setMouseTransparent(false);
522            }
523
524            if (selectedExp != null) {
525                selectedExp.setMouseTransparent(true);
526            }
527
528            if (selectedExpParent != null) {
529                selectedExpParent.setMouseTransparent(true);
530            }
531
532            if (deepCopy != null) {
533                deepCopy.setMouseTransparent(true);
534            }
535
536            if (helpMeP != null) {
537                helpMeP.setMouseTransparent(true);
538            }
539
540            if (_rootExpression != null) {
541                _rootExpression.setMouseTransparent(true);
542            }
543
544            if (_pane != null) {
545                _pane.setMouseTransparent(true);
546            }
547
548            if (expressionList != null) {
549                expressionList.setMouseTransparent(true);
550            }
551
552            if (selectedExpNode != null) {
553                selectedExpNode.setMouseTransparent(true);
554            }
555
556            if (selectedExp != null) {
557                selectedExp.setMouseTransparent(false);
558            }
559
560            if (selectedExpParent != null) {
561                selectedExpParent.setMouseTransparent(false);
562            }
563
564            if (deepCopy != null) {
565                deepCopy.setMouseTransparent(false);
566            }
567
568            if (helpMeP != null) {
569                helpMeP.setMouseTransparent(false);
570            }
571
572            if (_rootExpression != null) {
573                _rootExpression.setMouseTransparent(false);
574            }
575
576            if (_pane != null) {
577                _pane.setMouseTransparent(false);
578            }
579
580            if (expressionList != null) {
581                expressionList.setMouseTransparent(false);
582            }
583
584            if (selectedExpNode != null) {
585                selectedExpNode.setMouseTransparent(false);
586            }
587
588            if (selectedExp != null) {
589                selectedExp.setMouseTransparent(true);
590            }
591
592            if (selectedExpParent != null) {
593                selectedExpParent.setMouseTransparent(true);
594            }
595
596            if (deepCopy != null) {
597                deepCopy.setMouseTransparent(true);
598            }
599
600            if (helpMeP != null) {
601                helpMeP.setMouseTransparent(true);
602            }
603
604            if (_rootExpression != null) {
605                _rootExpression.setMouseTransparent(true);
606            }
607
608            if (_pane != null) {
609                _pane.setMouseTransparent(true);
610            }
611
612            if (expressionList != null) {
613                expressionList.setMouseTransparent(true);
614            }
615
616            if (selectedExpNode != null) {
617                selectedExpNode.setMouseTransparent(true);
618            }
619
620            if (selectedExp != null) {
621                selectedExp.setMouseTransparent(false);
622            }
623
624            if (selectedExpParent != null) {
625                selectedExpParent.setMouseTransparent(false);
626            }
627
628            if (deepCopy != null) {
629                deepCopy.setMouseTransparent(false);
630            }
631
632            if (helpMeP != null) {
633                helpMeP.setMouseTransparent(false);
634            }
635
636            if (_rootExpression != null) {
637                _rootExpression.setMouseTransparent(false);
638            }
639
640            if (_pane != null) {
641                _pane.setMouseTransparent(false);
642            }
643
644            if (expressionList != null) {
645                expressionList.setMouseTransparent(false);
646            }
647
648            if (selectedExpNode != null) {
649                selectedExpNode.setMouseTransparent(false);
650            }
651
652            if (selectedExp != null) {
653                selectedExp.setMouseTransparent(true);
654            }
655
656            if (selectedExpParent != null) {
657                selectedExpParent.setMouseTransparent(true);
658            }
659
660            if (deepCopy != null) {
661                deepCopy.setMouseTransparent(true);
662            }
663
664            if (helpMeP != null) {
665                helpMeP.setMouseTransparent(true);
666            }
667
668            if (_rootExpression != null) {
669                _rootExpression.setMouseTransparent(true);
670            }
671
672            if (_pane != null) {
673                _pane.setMouseTransparent(true);
674            }
675
676            if (expressionList != null) {
677                expressionList.setMouseTransparent(true);
678            }
679
680            if (selectedExpNode != null) {
681                selectedExpNode.setMouseTransparent(true);
682            }
683
684            if (selectedExp != null) {
685                selectedExp.setMouseTransparent(false);
686            }
687
688            if (selectedExpParent != null) {
689                selectedExpParent.setMouseTransparent(false);
690            }
691
692            if (deepCopy != null) {
693                deepCopy.setMouseTransparent(false);
694            }
695
696            if (helpMeP != null) {
697                helpMeP.setMouseTransparent(false);
698            }
699
700            if (_rootExpression != null) {
701                _rootExpression.setMouseTransparent(false);
702            }
703
704            if (_pane != null) {
705                _pane.setMouseTransparent(false);
706            }
707
708            if (expressionList != null) {
709                expressionList.setMouseTransparent(false);
710            }
711
712            if (selectedExpNode != null) {
713                selectedExpNode.setMouseTransparent(false);
714            }
715
716            if (selectedExp != null) {
717                selectedExp.setMouseTransparent(true);
718            }
719
720            if (selectedExpParent != null) {
721                selectedExpParent.setMouseTransparent(true);
722            }
723
724            if (deepCopy != null) {
725                deepCopy.setMouseTransparent(true);
726            }
727
728            if (helpMeP != null) {
729                helpMeP.setMouseTransparent(true);
730            }
731
732            if (_rootExpression != null) {
733                _rootExpression.setMouseTransparent(true);
734            }
735
736            if (_pane != null) {
737                _pane.setMouseTransparent(true);
738            }
739
740            if (expressionList != null) {
741                expressionList.setMouseTransparent(true);
742            }
743
744            if (selectedExpNode != null) {
745                selectedExpNode.setMouseTransparent(true);
746            }
747
748            if (selectedExp != null) {
749                selectedExp.setMouseTransparent(false);
750            }
751
752            if (selectedExpParent != null) {
753                selectedExpParent.setMouseTransparent(false);
754            }
755
756            if (deepCopy != null) {
757                deepCopy.setMouseTransparent(false);
758            }
759
760            if (helpMeP != null) {
761                helpMeP.setMouseTransparent(false);
762            }
763
764            if (_rootExpression != null) {
765                _rootExpression.setMouseTransparent(false);
766            }
767
768            if (_pane != null) {
769                _pane.setMouseTransparent(false);
770            }
771
772            if (expressionList != null) {
773                expressionList.setMouseTransparent(false);
774            }
775
776            if (selectedExpNode != null) {
777                selectedExpNode.setMouseTransparent(false);
778            }
779
780            if (selectedExp != null) {
781                selectedExp.setMouseTransparent(true);
782            }
783
784            if (selectedExpParent != null) {
785                selectedExpParent.setMouseTransparent(true);
786            }
787
788            if (deepCopy != null) {
789                deepCopy.setMouseTransparent(true);
790            }
791
792            if (helpMeP != null) {
793                helpMeP.setMouseTransparent(true);
794            }
795
796            if (_rootExpression != null) {
797                _rootExpression.setMouseTransparent(true);
798            }
799
800            if (_pane != null) {
801                _pane.setMouseTransparent(true);
802            }
803
804            if (expressionList != null) {
805                expressionList.setMouseTransparent(true);
806            }
807
808            if (selectedExpNode != null) {
809                selectedExpNode.setMouseTransparent(true);
810            }
811
812            if (selectedExp != null) {
813                selectedExp.setMouseTransparent(false);
814            }
815
816            if (selectedExpParent != null) {
817                selectedExpParent.setMouseTransparent(false);
818            }
819
820            if (deepCopy != null) {
821                deepCopy.setMouseTransparent(false);
822            }
823
824            if (helpMeP != null) {
825                helpMeP.setMouseTransparent(false);
826            }
827
828            if (_rootExpression != null) {
829                _rootExpression.setMouseTransparent(false);
830            }
831
832            if (_pane != null) {
833                _pane.setMouseTransparent(false);
834            }
835
836            if (expressionList != null) {
837                expressionList.setMouseTransparent(false);
838            }
839
840            if (selectedExpNode != null) {
841                selectedExpNode.setMouseTransparent(false);
842            }
843
844            if (selectedExp != null) {
845                selectedExp.setMouseTransparent(true);
846            }
847
848            if (selectedExpParent != null) {
849                selectedExpParent.setMouseTransparent(true);
850            }
851
852            if (deepCopy != null) {
853                deepCopy.setMouseTransparent(true);
854            }
855
856            if (helpMeP != null) {
857                helpMeP.setMouseTransparent(true);
858            }
859
860            if (_rootExpression != null) {
861                _rootExpression.setMouseTransparent(true);
862            }
863
864            if (_pane != null) {
865                _pane.setMouseTransparent(true);
866            }
867
868            if (expressionList != null) {
869                expressionList.setMouseTransparent(true);
870            }
871
872            if (selectedExpNode != null) {
873                selectedExpNode.setMouseTransparent(true);
874            }
875
876            if (selectedExp != null) {
877                selectedExp.setMouseTransparent(false);
878            }
879
880            if (selectedExpParent != null) {
881                selectedExpParent.setMouseTransparent(false);
882            }
883
884            if (deepCopy != null) {
885                deepCopy.setMouseTransparent(false);
886            }
887
888            if (helpMeP != null) {
889                helpMeP.setMouseTransparent(false);
890            }
891
892            if (_rootExpression != null) {
893                _rootExpression.setMouseTransparent(false);
894            }
895
896            if (_pane != null) {
897                _pane.setMouseTransparent(false);
898            }
899
900            if (expressionList != null) {
901                expressionList.setMouseTransparent(false);
902            }
903
904            if (selectedExpNode != null) {
905                selectedExpNode.setMouseTransparent(false);
906            }
907
908            if (selectedExp != null) {
909                selectedExp.setMouseTransparent(true);
910            }
911
912            if (selectedExpParent != null) {
913                selectedExpParent.setMouseTransparent(true);
914            }
915
916            if (deepCopy != null) {
917                deepCopy.setMouseTransparent(true);
918            }
919
920            if (helpMeP != null) {
921                helpMeP.setMouseTransparent(true);
922            }
923
924            if (_rootExpression != null) {
925                _rootExpression.setMouseTransparent(true);
926            }
927
928            if (_pane != null) {
929                _pane.setMouseTransparent(true);
930            }
931
932            if (expressionList != null) {
933                expressionList.setMouseTransparent(true);
934            }
935
936            if (selectedExpNode != null) {
937                selectedExpNode.setMouseTransparent(true);
938            }
939
940            if (selectedExp != null) {
941                selectedExp.setMouseTransparent(false);
942            }
943
944            if (selectedExpParent != null) {
945                selectedExpParent.setMouseTransparent(false);
946            }
947
948            if (deepCopy != null) {
949                deepCopy.setMouseTransparent(false);
950            }
951
952            if (helpMeP != null) {
953                helpMeP.setMouseTransparent(false);
954            }
955
956            if (_rootExpression != null) {
957                _rootExpression.setMouseTransparent(false);
958            }
959
960            if (_pane != null) {
961                _pane.setMouseTransparent(false);
962            }
963
964            if (expressionList != null) {
965                expressionList.setMouseTransparent(false);
966            }
967
968            if (selectedExpNode != null) {
969                selectedExpNode.setMouseTransparent(false);
970            }
971
972            if (selectedExp != null) {
973                selectedExp.setMouseTransparent(true);
974            }
975
976            if (selectedExpParent != null) {
977                selectedExpParent.setMouseTransparent(true);
978            }
979
980            if (deepCopy != null) {
981                deepCopy.setMouseTransparent(true);
982            }
983
984            if (helpMeP != null) {
985                helpMeP.setMouseTransparent(true);
986            }
987
988            if (_rootExpression != null) {
989                _rootExpression.setMouseTransparent(true);
990            }
991
992            if (_pane != null) {
993                _pane.setMouseTransparent(true);
994            }
995
996            if (expressionList != null) {
997                expressionList.setMouseTransparent(true);
998            }
999
1000           if (selectedExpNode != null) {
1001               selectedExpNode.setMouseTransparent(true);
1002           }
1003
1004           if (selectedExp != null) {
1005               selectedExp.setMouseTransparent(false);
1006           }
1007
1008           if (selectedExpParent != null) {
1009               selectedExpParent.setMouseTransparent(false);
1010           }
1011
1012           if (deepCopy != null) {
1013               deepCopy.setMouseTransparent(false);
1014           }
1015
1016           if (helpMeP != null) {
1017               helpMeP.setMouseTransparent(false);
1018           }
1019
1020           if (_rootExpression != null) {
1021               _rootExpression.setMouseTransparent(false);
1022           }
1023
1024           if (_pane != null) {
1025               _pane.setMouseTransparent(false);
1026           }
1027
1028           if (expressionList != null) {
1029               expressionList.setMouseTransparent(false);
1030           }
1031
1032           if (selectedExpNode != null) {
1033               selectedExpNode.setMouseTransparent(false);
1034           }
1035
1036           if (selectedExp != null) {
1037               selectedExp.setMouseTransparent(true);
1038           }
1039
1040           if (selectedExpParent != null) {
1041               selectedExpParent.setMouseTransparent(true);
1042           }
1043
1044           if (deepCopy != null) {
1045               deepCopy.setMouseTransparent(true);
1046           }
1047
1048           if (helpMeP != null) {
1049               helpMeP.setMouseTransparent(true);
1050           }
1051
1052           if (_rootExpression != null) {
1053               _rootExpression.setMouseTransparent(true);
1054           }
1055
1056           if (_pane != null) {
1057               _pane.setMouseTransparent(true);
1058           }
1059
1060           if (expressionList != null) {
1061               expressionList.setMouseTransparent(true);
1062           }
1063
1064           if (selectedExpNode != null) {
1065               selectedExpNode.setMouseTransparent(true);
1066           }
1067
1068           if (selectedExp != null) {
1069               selectedExp.setMouseTransparent(false);
1070           }
1071
1072           if (selectedExpParent != null) {
1073               selectedExpParent.setMouseTransparent(false);
1074           }
1075
1076           if (deepCopy != null) {
1077               deepCopy.setMouseTransparent(false);
1078           }
1079
1080           if (helpMeP != null) {
1081               helpMeP.setMouseTransparent(false);
1082           }
1083
1084           if (_rootExpression != null) {
1085               _rootExpression.setMouseTransparent(false);
1086           }
1087
1088           if (_pane != null) {
1089               _pane.setMouseTransparent(false);
1090           }
1091
1092           if (expressionList != null) {
1093               expressionList.setMouseTransparent(false);
1094           }
1095
1096           if (selectedExpNode != null) {
1097               selectedExpNode.setMouseTransparent(false);
1098           }
1099
1100          if (selectedExp != null) {
1101              selectedExp.setMouseTransparent(true);
1102          }
1103
1104          if (selectedExpParent != null) {
1105              selectedExpParent.setMouseTransparent(true);
1106          }
1107
1108          if (deepCopy != null) {
1109              deepCopy.setMouseTransparent(true);
1110          }
1111
1112          if (helpMeP != null) {
1113              helpMeP.setMouseTransparent(true);
1114          }
1115
1116          if (_rootExpression != null) {
1117              _rootExpression.setMouseTransparent(true);
1118          }
1119
1120          if (_pane != null) {
1121              _pane.setMouseTransparent(true);
1122          }
1123
1124          if (expressionList != null) {
1125              expressionList.setMouseTransparent(true);
1126          }
1127
1128          if (selectedExpNode != null) {
1129              selectedExpNode.setMouseTransparent(true);
1130          }
1131
1132          if (selectedExp != null) {
1133              selectedExp.setMouseTransparent(false);
1134          }
1135
1136          if (selectedExpParent != null) {
1137              selectedExpParent.setMouseTransparent(false);
1138          }
1139
1140          if (deepCopy != null) {
1141              deepCopy.setMouseTransparent(false);
1142          }
1143
1144          if (helpMeP != null) {
1145              helpMeP.setMouseTransparent(false);
1146          }
1147
1148          if (_rootExpression != null) {
1149              _rootExpression.setMouseTransparent(false);
1150          }
1151
1152          if (_pane != null) {
1153              _pane.setMouseTransparent(false);
1154          }
1155
1156          if (expressionList != null) {
1157              expressionList.setMouseTransparent(false);
1158          }
1159
1160          if (selectedExpNode != null) {
1161              selectedExpNode.setMouseTransparent(false);
1162          }
1163
1164          if (selectedExp != null) {
1165              selectedExp.setMouseTransparent(true);
1166          }
1167
1168          if (selectedExpParent != null) {
1169              selectedExpParent.setMouseTransparent(true);
1170          }
1171
1172          if (deepCopy != null) {
1173              deepCopy.setMouseTransparent(true);
1174          }
1175
1176          if (helpMeP != null) {
1177              helpMeP.setMouseTransparent(true);
1178          }
1179
1180          if (_rootExpression != null) {
1181              _rootExpression.setMouseTransparent(true);
1182          }
1183
1184          if (_pane != null) {
1185              _pane.setMouseTransparent(true);
1186          }
1187
1188          if (expressionList != null) {
1189              expressionList.setMouseTransparent(true);
1190          }
1191
1192          if (selectedExpNode != null) {
1193              selectedExpNode.setMouseTransparent(true);
1194          }
1195
1196          if (selectedExp != null) {
1197              selectedExp.setMouseTransparent(false);
1198          }
1199
1200          if (selectedExpParent != null) {
1201              selectedExpParent.setMouseTransparent(false);
1202          }
1203
1204          if (deepCopy != null) {
1205              deepCopy.setMouseTransparent(false);
1206          }
1207
1208          if (helpMeP != null) {
1209              helpMeP.setMouseTransparent(false);
1210          }
1211
1212          if (_rootExpression != null) {
1213              _rootExpression.setMouseTransparent(false);
1214          }
1215
1216          if (_pane != null) {
1217              _pane.setMouseTransparent(false);
1218          }
1219
1220          if (expressionList != null) {
1221              expressionList.setMouseTransparent(false);
1222          }
1223
1224          if (selectedExpNode != null) {
1225              selectedExpNode.setMouseTransparent(false);
1226          }
1227
1228          if (selectedExp != null) {
1229              selectedExp.setMouseTransparent(true);
1230          }
1231
1232          if (selectedExpParent != null) {
1233              selectedExpParent.setMouseTransparent(true);
1234          }
1235
1236          if (deepCopy != null) {
1237              deepCopy.setMouseTransparent(true);
1238          }
1239
1240          if (helpMeP != null) {
1241              helpMeP.setMouseTransparent(true);
1242          }
1243
1244          if (_rootExpression != null) {
1245              _rootExpression.setMouseTransparent(true);
1246          }
1247
1248          if (_pane != null) {
1249              _pane.setMouseTransparent(true);
1250          }
1251
1252          if (expressionList != null) {
1253              expressionList.setMouseTransparent(true);
1254          }
1255
1256          if (selectedExpNode != null) {
1257              selectedExpNode.setMouseTransparent(true);
1258          }
1259
1260          if (selectedExp != null) {
1261              selectedExp.setMouseTransparent(false);
1262          }
1263
1264          if (selectedExpParent != null) {
1265              selectedExpParent.setMouseTransparent(false);
1266          }
1267
1268          if (deepCopy != null) {
1269              deepCopy.setMouseTransparent(false);
1270          }
1271
1272          if (helpMeP != null) {
1273              helpMeP.setMouseTransparent(false);
1274          }
1275
1276          if (_rootExpression != null) {
1277              _rootExpression.setMouseTransparent(false);
1278          }
1279
1280          if (_pane != null) {
1281              _pane.setMouseTransparent(false);
1282          }
1283
1284          if (expressionList != null) {
1285              expressionList.setMouseTransparent(false);
1286          }
1287
1288          if (selectedExpNode != null) {
1289              selectedExpNode.setMouseTransparent(false);
1290          }
1291
1292          if (selectedExp != null) {
1293              selectedExp.setMouseTransparent(true);
1294          }
1295
1296          if (selectedExpParent != null) {
1297              selectedExpParent.setMouseTransparent(true);
1298          }
1299
1300          if (deepCopy != null) {
1301              deepCopy.setMouseTransparent(true);
1302          }
1303
1304          if (helpMeP != null) {
1305              helpMeP.setMouseTransparent(true);
1306          }
1307
1308          if (_rootExpression != null) {
1309              _rootExpression.setMouseTransparent(true);
1310          }
1311
1312          if (_pane != null) {
1313              _pane.setMouseTransparent(true);
1314          }
1315
1316          if (expressionList != null) {
1317              expressionList.setMouseTransparent(true);
1318          }
1319
1320          if (selectedExpNode != null) {
1321              selectedExpNode.setMouseTransparent(true);
1322          }
1323
1324          if (selectedExp != null) {
1325              selectedExp.setMouseTransparent(false);
1326          }
1327
1328          if (selectedExpParent != null) {
1329              selectedExpParent.setMouseTransparent(false);
1330          }
1331
1332          if (deepCopy != null)
```

```

59     } else if (event.getEventType() == MouseEvent.MOUSE_DRAGGED) {
60         if (focusLevel > 0) {
61             selectedExpNode.setOpacity(0.6);
62             if (!_pane.getChildren().contains(deepCopy)) {
63                 _pane.getChildren().add(deepCopy);
64             }
65             moveNode(deepCopy, event.getX(), event.getY());
66             rearrange2(event);
67         }
68     } else if (event.getEventType() == MouseEvent.MOUSE_RELEASED) {
69         if (focusLevel > 0 && wasSwapped) {
70             doTheSwap();
71             _pane.getChildren().remove(deepCopy);
72             selectedExpNode.setOpacity(1); //redundant
73             focusLevel = 0;
74             resetBordersAndOpacity(_pane.getChildren().get(0));
75             System.out.println(_rootExpression.convertToString(0));
76             wasSwapped = false;
77         }
78     }
79 }
80
81 /**
82 * deals with the selecting of the expression
83 * checks first if it is in bounds of the whole thing, then
84 * goes through all the children and trees and tries to find an
expression
85 * that is valid based on the mouse's click location and the current
focus level
86 *
87 * @param event the mouse (click) event
88 */
89 private void highlight(MouseEvent event) {
90     //first child of _pane is one hBox, aka the ROOT NODE
91     //so we want to start with the children of that one node
92     Pane temp = _pane;
93     HBox curHBox = (HBox) temp.getChildren().get(0);
94     resetBordersAndOpacity(curHBox);
95     Point2D mouse = curHBox.sceneToLocal(event.getSceneX(),
event.getSceneY());
96     if (!curHBox.contains(mouse)) {
97         focusLevel = 0;
98         return;
99     } //if out of bounds
100    int currentFocus = 0;
101    CompoundExpression curExp = _rootExpression; //this is the current
expression we are on.
102    //this, and expCounter, are used to know which expression we are
highlighting (keep track of it)
103    while (true) {
104        int expCounter = 0;
105        for (Node n : curHBox.getChildren()) {
106            //sets mouse coordinates equal to n's local coordinates so we can
use contains method
107            mouse = n.sceneToLocal(event.getSceneX(), event.getSceneY());
108            if (n.contains(mouse)) {
109                //if we are on the right focus level and the node is an
expression...
110                if (currentFocus == focusLevel && isValidExpression(n)) {
111                    focusLevel++; //increase focus level (for future selections)

```

```

112
113         final Expression theExpression =
114     curExp.getChildList().get(expCounter); //get the expression
115
116     System.out.println(theExpression.getParent().getChildList().size());
117     if (selectedExp != null) {
118         selectedExp.getNode().setOpacity(1);
119         _pane.getChildren().remove(deepCopy);
120     }
121     selectedExp = theExpression; //based on the tracking that has
122     been going on
123
124     //needed below:
125     CompoundExpression tempCopy = (CompoundExpression)
126     selectedExp.getParent().deepCopy();
127     expressionList = tempCopy.getChildList();
128     System.out.println(expressionList.size()); //this produces 2.
129     should be 4.
130
131     try {
132         Pane pane = (Pane) n;
133         pane.setBorder(Expression.RED_BORDER);
134     } catch (Exception e) {
135         Label label = (Label) n;
136         label.setBorder(Expression.RED_BORDER);
137     }
138     return; //exit loop.
139
140     //ELSE... (aka we need a bigger focus...)
141     currentFocus++; //increase this var
142     if (n.getClass().equals(new HBox().getClass())) { //if n is an
143         hBox, that means
144             //that there are more expressions to check. if it is not hBox,
145             it is the most focused.
146             curHBox = (HBox) n;
147             curExp = (CompoundExpression)
148             curExp.getChildList().get(expCounter); //go to that child
149             expCounter = 0;
150         } else {
151             //if n is NOT hbox, that means that there is no node to find,
152             so we reset focus
153             focusLevel = 0;
154             return; //and exit out of the loop
155         }
156     }
157 }
158
159 /**
160 * removes the border of node n and all children
161 *

```

```

162     * @param n the node to have its border modified
163     */
164     private void resetBordersAndOpacity(Node n) {
165         if (n.getClass().equals(new HBox().getClass())) {
166             Pane hBox = (Pane) n;
167                 hBox.setBorder(Expression.NO_BORDER);
168                 hBox.setOpacity(1);
169             for (Node x : hBox.getChildren()) {
170                 resetBordersAndOpacity(x);
171             }
172         } else {
173             final Label label = (Label) n;
174             label.setBorder(Expression.NO_BORDER);
175             label.setOpacity(1);
176         }
177     }
178 }
179
180 /**
181 * checks if the node is a valid expression to highlight
182 * a single node of +, ., (, or ) is invalid; all else is valid
183 * so only need to check if node n is a label and has one of the four
184 above values.
185 *
186 * @param n the node to be checked
187 * @return true if the node follows above conditions, false otherwise
188 */
189     private boolean isValidExpression(Node n) {
190         if (n.getClass().equals(new Label().getClass())) {
191             final Label label = (Label) n;
192             if (label.getText().equals("+") || label.getText().equals(".")
193                 || label.getText().equals("(") || label.getText().equals(")")) {
194                 return false;
195             }
196         }
197         return true;
198     }
199
200     private void moveNode(Node n, double deltaX, double deltaY) {
201         n.setLayoutY(deltaY);
202         n.setLayoutX(deltaX);
203     }
204
205     private void doTheSwap() {
206         //doesn't work!
207         CompoundExpression parent = selectedExpParent;
208         parent.getChildList().removeAll(parent.getChildList());
209
210         for (Expression e : expressionList) {
211             parent.addSubexpression(e);
212         }
213         /*Pane p = (Pane) deepCopy.getParent();
214         p.getChildren().remove(p);*/
215     }
216
217     //for loop of size of children
218     //create a rearrangement and actually like DO IT (by removing all
219     //children and re-adding them)
220     //check if that distance is closer than the current "closest one"

```

```

220 //if it is, write down a way to get it back (NOT COPY)
221 //then for the one that is closest, set it as it
222
223 //doesn't work with parenthesis (f off)
224 private void rearrange2(MouseEvent event) {
225     //helpMeP is HBox
226     double startingX =
227         helpMeP.localToScene(helpMeP.getBoundsInLocal()).getMinX();
228     double widthOfOperator =
229         helpMeP.getChildren().get(1).getLayoutBounds().getWidth();
230     double expWidth = selectedExpNode.getLayoutBounds().getWidth(); //can
prob delete
231
232     //double targetPoint =
233     selectedExpNode.localToScene(selectedExpNode.getBoundsInLocal()).getMinX() +
234     selectedExpNode.getLayoutBounds().getWidth()/2;
235     //mid point above
236     List<Double> listOfValues = new ArrayList<Double>();
237     List<Node> listOfNodes = new ArrayList<Node>();
238     int theIndex = 0;
239     int pastIndex = 0;
240     for (int i = 0; i < helpMeP.getChildren().size(); i += 2) {
241         //double xCord =
242         helpMeP.getChildren().get(i).localToScene(helpMeP.getBoundsInLocal()).getMinX
());
243         double xCord =
244         helpMeP.getChildren().get(i).getLayoutBounds().getWidth();
245         //so this has 0 + or * labels
246         if (!helpMeP.getChildren().get(i).equals(selectedExpNode)) {
247             listOfValues.add(xCord);
248             listOfNodes.add(helpMeP.getChildren().get(i));
249         } else {
250             theIndex = i;
251             pastIndex = i / 2;
252         }
253     }
254
255     double closestValue = -1;
256     int index = -1;
257     for (int i = 0; i < listOfValues.size() + 1; i++) {
258         final double testValue = calculateX(i, listOfValues, widthOfOperator,
startingX);
259         if (i == 1) {
260             /*System.out.println(closestValue);
261             System.out.println(testValue);
262             System.out.println(deepCopy.getLayoutBounds().getWidth()/2 +
deepCopy.getLayoutX());*/
263         }
264         if (isClosestValue(closestValue, testValue)) {
265             closestValue = testValue;
266             index = i;
267         }
268     }
269     //System.out.println(index);
270     if (curIndex != index && index != -1) {
271         instantiateNewOrdering(index, listOfValues.size() + 1, pastIndex,
theIndex);
272         curIndex = index;
273     }

```

```

269
270 //PROBLEM: THIS IS BASED ON MOUSEX AND NOOOOT ON THE DRAGGED EXPRESSION
271 //WE IGNORE THIS FOR NOW
272
273 //both of them will swap by one.
274
275 //then we calculate the value of the x.
276 }
277
278 private void instantiateNewOrdering(int index, int size, int pastIndex,
279 int theIndex) {
280     wasSwapped = true;
281     List<Node> newList = new ArrayList<Node>();
282     final Label op = (Label) helpMeP.getChildren().get(1);
283     //final Label opCopy = new Label(op.getText());
284     for (int i = 0; i < index - 1; i++) {
285         //add all NON-DRAGGED NON-OPERATION NODES
286         newList.add(helpMeP.getChildren().get(i * 2));
287         newList.add(op);
288     }
289     newList.add(selectedExpNode);
290     for (int i = index + 1; i < size; i++) {
291         newList.add(op);
292         newList.add(helpMeP.getChildren().get(i * 2));
293     }
294     Collection<Node> children = helpMeP.getChildren();
295
296     System.out.println(children.equals(newList));
297     //ADD DRAGGED
298
299     if (pastIndex > index) {
300         //move dragged to the left
301         Node temp = helpMeP.getChildren().get(theIndex - 2);
302         helpMeP.getChildren().set(theIndex, new Label());
303         helpMeP.getChildren().set(theIndex - 2, new Label());
304         helpMeP.getChildren().set(theIndex, temp);
305         helpMeP.getChildren().set(theIndex - 2, selectedExpNode);
306         Expression tempE = expressionList.get(pastIndex);
307         expressionList.set(pastIndex, expressionList.get(pastIndex - 1));
308         expressionList.set(pastIndex - 1, tempE);
309     } else {
310         //move dragged to the right
311         Node temp = helpMeP.getChildren().get(theIndex + 2);
312         helpMeP.getChildren().set(theIndex, new Label());
313         helpMeP.getChildren().set(theIndex + 2, new Label());
314         helpMeP.getChildren().set(theIndex, temp);
315         helpMeP.getChildren().set(theIndex + 2, selectedExpNode);
316         Expression tempE = expressionList.get(pastIndex);
317         expressionList.set(pastIndex, expressionList.get(pastIndex + 1));
318         expressionList.set(pastIndex + 1, tempE);
319     }
320 }
321
322
323
324     private double calculateX(int index, List<Double> list0fValues, double
325     opWidth, double startingPoint) {
326         double sum = startingPoint +
327             selectedExpNode.getLayoutBounds().getWidth() / 2;

```

```
326     for (int i = 0; i < index; i++) {
327         sum += listOfValues.get(i);
328         sum += opWidth;
329     }
330     return sum;
331 }
332
333 private boolean isClosestValue(double oldV, double newV) {
334     if (oldV < 0) {
335         return true;
336     }
337     double oldTwo = Math.abs(deepCopy.getLayoutX() +
338         deepCopy.getLayoutBounds().getWidth() / 2 - oldV);
339     double newTwo = Math.abs(deepCopy.getLayoutX() +
340         deepCopy.getLayoutBounds().getWidth() / 2 - newV);
341     return newTwo < oldTwo;
342     //like if we want to do this based on expression dragged location, we
343     //would have to compare minX and maxX
344     //and see, out of EVERYTHING, which is the smallest. hardest part with
345     //that is calculating
346     //min and max x (JK THIS IS SUPER EASY)
347 }
348 }
349 /**
350 * Size of the GUI
351 */
352 private static final int WINDOW_WIDTH = 500, WINDOW_HEIGHT = 250;
353 /**
354 * Initial expression shown in the textbox
355 */
356 private static final String EXAMPLE_EXPRESSION = "2*x+3*y+4*z+(7+6*z)";
357 /**
358 * Parser used for parsing expressions.
359 */
360 private final ExpressionParser expressionParser = new
361 SimpleExpressionParser();
362
363 @Override
364 public void start (Stage primaryStage) {
365     primaryStage.setTitle("Expression Editor");
366
367     // Add the textbox and Parser button
368     final Pane queryPane = new HBox();
369     final TextField textField = new TextField(EXAMPLE_EXPRESSION);
370     final Button button = new Button("Parse");
371     queryPane.getChildren().add(textField);
372
373     final Pane expressionPane = new Pane();
374
375     // Add the callback to handle when the Parse button is pressed
376     button.setOnMouseClicked(new EventHandler<MouseEvent>() {
377         public void handle (MouseEvent e) {
378             // Try to parse the expression
379             try {
380                 // Success! Add the expression's Node to the expressionPane
```

```

381     final Expression expression =
expressionParser.parse(textField.getText(), true);
382     System.out.println(expression.toString());
383     expressionPane.getChildren().clear();
384         final Node node = expression.getNode();
385     expressionPane.getChildren().add(node);
386     node.setLayoutX(WINDOW_WIDTH/4);
387     node.setLayoutY(WINDOW_HEIGHT/2);
388 // If the parsed expression is a CompoundExpression, then register
some callbacks
389     if (expression instanceof CompoundExpression) {
390         ((Pane) expression.getNode()).setBorder(Expression.NO_BORDER);
391         final MouseEventHandler eventHandler = new
MouseEventHandler(expressionPane, (CompoundExpression) expression);
392         expressionPane.setOnMousePressed(eventHandler);
393         expressionPane.setOnMouseDragged(eventHandler);
394         expressionPane.setOnMouseReleased(eventHandler);
395     }
396 } catch (ExpressionParseException epe) {
397 // If we can't parse the expression, then mark it in red
398     textField.setStyle("-fx-text-fill: red");
399 }
400 }
401 });
402 queryPane.getChildren().add(button);
403
// Reset the color to black whenever the user presses a key
404 textField.setOnKeyPressed(e -> textField.setStyle("-fx-text-fill:
black"));
405
final BorderPane root = new BorderPane();
406 root.setTop(queryPane);
407 root.setCenter(expressionPane);
408
primaryStage.setScene(new Scene(root, WINDOW_WIDTH, WINDOW_HEIGHT));
409 primaryStage.show();
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
```