

A sample implementation of CS349 A2.3 for Fall 2016 at University of Waterloo. This code is not for distribution. Implemented by Byron Weber Becker.

Implements a simple Entity-Relationship editor that supports the following features:

- Display entity relationships graphically.
- Display a list of entities.
- Display a list of their relationships.
- Add a new entity with the mouse.
- Select an entity and press "Delete" to delete it.
- Add a new relationship with the mouse.
- Rename an entity in the entity list.
- Rearrange entities in the graphic display.

Comments in the original code have been deleted before release to students. Present in the code reviewed in class.

```

1 EREdit
2
3 import model.ERModel;
4 import ui.*;
5 import javax.swing.*;
6 import java.awt.*;
7
8
9 /**
10  * Created by bwbecker on 2016-09-30.
11  */
12 public class EREdit {
13     public static void main(String[] args) {
14
15         ERModel model = new ERModel();
16         TopLevelView view = new TopLevelView(model);
17
18         SwingUtilities.invokeLater(new Runnable() {
19             @Override
20             public void run() {
21                 JFrame f = new JFrame("EREdit");
22                 f.setSize(800, 500);
23                 f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24                 f.setContentPane(view);
25                 f.setVisible(true);
26             }
27         });

```

```

28     }
29 }
30
31
32 ERModel
33 package model;
34
35 import java.awt.*;
36 import java.util.*;
37
38 /**
39  * bwbecker 20161103
40  */
41 public class ERModel extends Observable {
42
43     public static final Object ADD_REMOVE_DATA = new Object();
44     private Map<Integer, Entity> entities = new HashMap<Integer, Entity>();
45     private Map<Integer, Arrow> arrows = new HashMap<Integer, Arrow>();
46     private int nextUid = 100;
47
48     public ERModel() {
49         this.entities.put(4, new Entity(4, "Four", new Rectangle(30, 50, 100, 50)));
50         this.entities.put(6, new Entity(6, "Six", new Rectangle(90, 200, 100, 50)));
51         this.entities.put(8, new Entity(8, "Eight", new Rectangle(300, 200, 100, 50)));
52         this.arrows.put(29, new Arrow(29, this.entities.get(4), this.entities.get(6)));
53         this.arrows.put(30, new Arrow(30, this.entities.get(4), this.entities.get(8)));
54     }
55
56     public void addObserver(Observer o) {
57         super.addObserver(o);
58         o.update(this, this.ADD_REMOVE_DATA);
59     }
60
61     public Entity getEntity(int uid) {
62         return this.entities.get(uid);
63     }
64
65     public void selectEntity(int uid) {
66         Entity ent = this.entities.get(uid);
67         for (Entity e : this.entities.values()) {
68             e.setSelected(e.getUid() == uid);
69         }
70     }

```

```

71     for (Arrow a : this.arrows.values()) {
72         a.setSelected(a.fromEntity == ent);
73     }
74     this.setChanged();
75     this.notifyObservers();
76 }
77
78 public Entity getEntity(Point p) {
79     for (Entity e : this.entities.values()) {
80         if (e.contains(p)) {
81             return e;
82         }
83     }
84     return null;
85 }
86
87 public Iterator<Entity> entityIterator() {
88     return this.entities.values().iterator();
89 }
90
91 public int[] getEntityUids() {
92     int[] uids = new int[this.entities.size()];
93
94     int i = 0;
95     for (Entity e : this.entities.values()) {
96         uids[i] = e.getUid();
97         i++;
98     }
99     Arrays.sort(uids);
100    return uids;
101 }
102
103 public void moveEntity(int uid, int dx, int dy) {
104     Entity e = this.getEntity(uid);
105     e.move(dx, dy);
106     this.setChanged();
107     this.notifyObservers();
108 }
109
110 public int addEntity(String name, Point topLeft, Point bottomRight) {
111     int uid = this.nextUid++;
112
113     this.entities.put(uid, new Entity(uid, name, new Rectangle(topLeft.x, topLeft.y,
114         bottomRight.x - topLeft.x, bottomRight.y - topLeft.y)));
115
116     this.setChanged();

```

```

117     this.notifyObservers(this.ADD_REMOVE_DATA);
118     return uid;
119 }
120
121 public void setEntityName(int uid, String name) {
122     System.out.println("Setting entity name to " + name);
123     this.entities.get(uid).setName(name);
124     this.setChanged();
125     this.notifyObservers();
126 }
127
128 public void deleteSelectedEntity() {
129     Iterator<Map.Entry<Integer, Arrow>> alter = this.arrows.entrySet().iterator();
130     while (alter.hasNext()) {
131         Map.Entry<Integer, Arrow> entry = alter.next();
132         if (entry.getValue().fromEntity.isSelected() ||
133             entry.getValue().toEntity.isSelected()) {
134             alter.remove();
135         }
136     }
137
138     Iterator<Map.Entry<Integer, Entity>> elter = this.entities.entrySet().iterator();
139     while (elter.hasNext()) {
140         Map.Entry<Integer, Entity> entry = elter.next();
141         if (entry.getValue().isSelected()) {
142             elter.remove();
143         }
144     }
145
146     this.setChanged();
147     this.notifyObservers(this.ADD_REMOVE_DATA);
148 }
149
150 public int[] getArrowUids() {
151     int[] uids = new int[this.arrows.size()];
152
153     int i = 0;
154     for (Arrow a : this.arrows.values()) {
155         uids[i] = a.getUid();
156         i++;
157     }
158     Arrays.sort(uids);
159     return uids;
160 }
161
162

```

```

163 public Arrow getArrow(int uid) {
164     return this.arrows.get(uid);
165 }
166
167
168 public Arrow[] getArrows() {
169     Arrow[] arrows = new Arrow[this.arrows.size()];
170     this.arrows.values().toArray(arrows);
171     return arrows;
172 }
173
174 public Iterator<Arrow> arrowIterator() {
175     return this.arrows.values().iterator();
176 }
177
178
179 public void selectArrow(int uid) {
180     Arrow arrow = this.arrows.get(uid);
181
182     for (Entity e : this.entities.values()) {
183         e.setSelected(false);
184     }
185
186     for (Arrow a : this.arrows.values()) {
187         if (a.getUid() == uid) {
188             a.fromEntity.setSelected(true);
189             a.toEntity.setSelected(true);
190         }
191         a.setSelected(a.getUid() == uid);
192     }
193
194     this.setChanged();
195     this.notifyObservers();
196 }
197
198 public void addArrow(Entity from, Entity to) {
199     int uid = this.nextUid++;
200     this.arrows.put(uid, new Arrow(uid, from, to));
201     this.setChanged();
202     this.notifyObservers(this.ADD_REMOVE_DATA);
203 }
204
205 }
206

```

## ERModelError

```

207 package model;
208
209
210 public class ERModelError extends Error {
211
212     public ERModelError(String msg) {
213         super(msg);
214     }
215
216     public ERModelError(String msg, Throwable cause) {
217         super(msg, cause);
218     }
219 }
220

```

## DiagramPart

```

221 package model;
222
223
224 /**
225  * Created by bwbecker on 2016-11-03.
226  */
227 public class DiagramPart {
228
229     private int uid;
230     private boolean selected = false;
231
232     public DiagramPart(int uid) {
233         this.uid = uid;
234     }
235
236     public int getUid() {
237         return this.uid;
238     }
239
240     public boolean isSelected() {
241         return this.selected;
242     }
243
244     /* package */ void setSelected(boolean val) {
245         this.selected = val;
246     }
247 }
248

```

```

249 Entity
250 package model;
251
252 import model.DiagramPart;
253 import java.awt.*;
254
255 /**
256  * Created by bwbecker on 2016-11-03.
257  */
258 public class Entity extends DiagramPart {
259
260     private String name;
261     private Rectangle location;
262
263     public Entity(int uid, String name, Rectangle location) {
264         super(uid);
265         this.name = name;
266         this.location = location;
267     }
268
269     public String getName() {
270         return this.name;
271     }
272
273     public Rectangle getRect() {
274         return (Rectangle)this.location.clone();
275     }
276
277     public boolean contains(Point p) {
278         return this.location.contains(p);
279     }
280
281     /* package */ void setName(String name) {
282         this.name = name;
283     }
284
285     /* package */ void move(int dx, int dy) {
286         this.location.translate(dx, dy);
287     }
288
289     /* package */ Rectangle getLocation() {
290         return this.location;
291     }
292
293 }

```

```

294 Arrow
295 package model;
296
297 import java.awt.*;
298
299 /**
300  * Created by bwbecker on 2016-11-03.
301  */
302 public class Arrow extends DiagramPart {
303
304     /* package */ Entity fromEntity;
305     /* package */ Entity toEntity;
306
307     public Arrow(int uid, Entity fromEntity, Entity toEntity) {
308         super(uid);
309
310         this.fromEntity = fromEntity;
311         this.toEntity = toEntity;
312     }
313
314     public String fromName() {
315         return this.fromEntity.getName();
316     }
317
318     public String toName() {
319         return this.toEntity.getName();
320     }
321
322     public Point[] endPoints() {
323         Rectangle from = this.fromEntity.getLocation();
324         Rectangle to = this.toEntity.getLocation();
325         int x1, y1, x2, y2;
326
327         if (from.x + from.width < to.x) {
328             x1 = from.x + from.width;
329             x2 = to.x;
330         } else if (from.x > to.x + to.width) {
331             x1 = from.x;
332             x2 = to.x + to.width;
333         } else {
334             x1 = from.x + from.width/2;
335             x2 = to.x + to.width/2;
336         }
337
338         if (from.y + from.height < to.y) {

```

```

339     y1 = from.y + from.height;
340     y2 = to.y;
341 } else if (to.y + to.height < from.y) {
342     y1 = from.y;
343     y2 = to.y + to.height;
344 } else {
345     y1 = from.y + from.height/2;
346     y2 = to.y + to.height/2;
347 }
348
349 return new Point[]{new Point(x1,y1), new Point(x2,y2)};
350 }
351 }
352
353

```

## 354 TopLevelView

```

355 package ui;
356
357 import model.ERModel;
358 import javax.swing.*.*;
359 import java.awt.*.*;
360
361 /**
362  * Created by bwbecker on 2016-11-03.
363  */
364 public class TopLevelView extends JPanel {
365
366     public TopLevelView(ERModel model) {
367         ToolView toolView = new ToolView();
368         GraphicView graphicView = new GraphicView(model, toolView);
369         EntityView entityView = new EntityView(model);
370         ArrowView arrowView = new ArrowView(model);
371
372         JPanel lists = new JPanel(new GridLayout(2, 1));
373         lists.add(entityView);
374         lists.add(arrowView);
375
376         JPanel left = new JPanel();
377         left.setLayout(new BoxLayout(left, BoxLayout.Y_AXIS));
378         left.add(toolView);
379         left.add(Box.createVerticalStrut(30));
380         left.add(lists);
381

```

```

382         this.setLayout(new BorderLayout());
383         this.add(graphicView, BorderLayout.CENTER);
384         this.add(left, BorderLayout.WEST);
385     }
386 }
387
388

```

## 389 ViewMode

```

390 package ui;
391
392 import java.util.Enumeration;
393
394 /**
395  * Created by bwbecker on 2016-11-03.
396  */
397 public enum ViewMode {
398     NORMAL,
399     ADD_ENTITY,
400     ADD_ARROW,
401     MOVE_ENTITY
402 }
403

```

## 404 ToolView

```

405 package ui;
406
407 import javax.swing.*.*;
408 import java.awt.event.ActionEvent;
409 import java.awt.event.ActionListener;
410 import static ui.ViewMode.*;
411
412 /**
413  * Created by bwbecker on 2016-11-03.
414  */
415 public class ToolView extends JPanel {
416
417     private JButton addEntity = new JButton("Add Entity");
418     private JButton addArrow = new JButton("Add Arrow");
419
420     ViewMode viewMode = NORMAL;
421

```

```

422 public ToolView() {
423     super();
424     this.setLayout(new BorderLayout(this, BorderLayout.X_AXIS));
425
426     this.add(this.addEntity);
427     this.add(this.addArrow);
428
429     this.addEntity.addActionListener(new ActionListener() {
430         @Override
431         public void actionPerformed(ActionEvent e) {
432             viewModel = ADD_ENTITY;
433         }
434     });
435
436     this.addArrow.addActionListener(new ActionListener() {
437         @Override
438         public void actionPerformed(ActionEvent e) {
439             viewModel = ADD_ARROW;
440         }
441     });
442 }
443
444 public ViewMode getViewMode() {
445     return this.viewModel;
446 }
447 }
448 }
449
450

```

## 451 GraphicView

```

452 package ui;
453
454 import model.Arrow;
455 import model.ERModel;
456 import model.Entity;
457
458 import javax.swing.*;
459 import java.awt.*;
460 import java.awt.event.*;
461 import java.util.Iterator;
462 import java.util.Observable;
463 import java.util.Observer;
464

```

```

465 import static ui.ViewMode.*;
466
467 /**
468  * Created by bwbecker on 2016-11-02.
469  */
470
471 public class GraphicView extends JComponent implements Observer {
472
473     private ERModel model;
474     private Point mouseDownPoint = null;
475     private Point currPoint = null;
476     private Entity selectedEntity = null;
477     private ToolView toolView;
478
479     public GraphicView(ERModel model, ToolView toolView) {
480         this.model = model;
481         this.toolView = toolView;
482
483         this.addMouseListener(new MouseAdapter() {
484             @Override
485             public void mousePressed(MouseEvent e) {
486                 super.mousePressed(e);
487                 switch (toolView.getViewMode()) {
488                     case NORMAL:
489                         mouseDownPoint = e.getPoint();
490                         selectedEntity = model.getEntity(mouseDownPoint);
491                         break;
492
493                     case ADD_ENTITY:
494                     case ADD_ARROW:
495                         mouseDownPoint = e.getPoint();
496                         break;
497                 }
498             }
499         })
500
501         @Override
502         public void mouseReleased(MouseEvent e) {
503             super.mouseReleased(e);
504             switch (toolView.getViewMode()) {
505                 case NORMAL:
506                     Point p = e.getPoint();
507                     Entity ent = model.getEntity(p);
508                     if (ent != null) {
509                         setFocusable(true);
510                         requestFocusInWindow();
511                     }
512             }
513         }
514     }
515

```

```

511     model.selectEntity(ent.getUid());
512 }
513 break;
514
515 case ADD_ENTITY:
516     int uid = model.addEntity("", mouseDownPoint, e.getPoint());
517     JTextField name = new JTextField(15);
518     name.setLocation(mouseDownPoint.x, mouseDownPoint.y);
519     name.setSize(e.getPoint().x - mouseDownPoint.x, 25);
520     name.addFocusListener(new FocusAdapter() {
521         @Override
522         public void focusLost(FocusEvent fe) {
523             model.setEntityName(uid, name.getText());
524             remove(name);
525             repaint();
526         }
527     });
528     name.addActionListener(new ActionListener() {
529         @Override
530         public void actionPerformed(ActionEvent ae) {
531             model.setEntityName(uid, name.getText());
532             remove(name);
533             repaint();
534         }
535     });
536     add(name);
537     name.requestFocus();
538     repaint();
539
540     break;
541
542 case ADD_ARROW:
543     Entity endEntity = model.getEntity(e.getPoint());
544     Entity startEntity = model.getEntity(mouseDownPoint);
545     if (endEntity != null && startEntity != null) {
546         model.addArrow(startEntity, endEntity);
547     } else {
548         repaint();
549     }
550     break;
551 }
552 selectedEntity = null;
553 toolView.viewMode = NORMAL;
554 }
555 });
556

```

```

557 this.addMouseMotionListener(new MouseMotionAdapter() {
558     @Override
559     public void mouseDragged(MouseEvent e) {
560         super.mouseDragged(e);
561         switch (toolView.getViewMode()) {
562             case NORMAL:
563                 if (selectedEntity != null) {
564                     Point p = e.getPoint();
565                     int dX = p.x - mouseDownPoint.x;
566                     int dY = p.y - mouseDownPoint.y;
567                     model.moveEntity(selectedEntity.getUid(), dX, dY);
568                     mouseDownPoint = p;
569                 }
570                 break;
571
572             case ADD_ENTITY:
573             case ADD_ARROW:
574                 currPoint = e.getPoint();
575                 repaint();
576                 break;
577         }
578     }
579 });
580
581 this.addKeyListener(new KeyAdapter() {
582     @Override
583     public void keyPressed(KeyEvent e) {
584         if (e.getKeyCode() == KeyEvent.VK_DELETE) {
585             model.deleteSelectedEntity();
586         }
587     }
588 });
589
590 this.model.addObserver(this);
591 }
592
593 public void update(Observable obs, Object obj) {
594     this.repaint();
595 }
596
597
598
599 private Stroke regular = new BasicStroke(1.0F);
600 private Stroke hilite = new BasicStroke(2.0F);
601 private Stroke dashed = new BasicStroke(1, BasicStroke.CAP_BUTT,
602     BasicStroke.JOIN_BEVEL, 0, new float[]{9}, 0);

```



```

603
604 public void paintComponent(Graphics g) {
605     super.paintComponent(g);
606     Graphics2D g2 = (Graphics2D) g;
607
608     g2.drawRect(0, 0, this.getWidth() - 1, this.getHeight() - 1);
609
610     Iterator<Arrow> ait = this.model.arrowIterator();
611     while (ait.hasNext()) {
612         Arrow a = ait.next();
613         Point[] ep = a.endPoints();
614         if (a.isSelected()) {
615             g2.setStroke(hilite);
616         } else {
617             g2.setStroke(regular);
618         }
619         g2.drawLine(ep[0].x, ep[0].y, ep[1].x, ep[1].y);
620         g2.drawOval(ep[1].x - 4, ep[1].y - 4, 8, 8);
621     }
622
623     Iterator<Entity> eit = this.model.entityIterator();
624
625     while (eit.hasNext()){
626         Entity e = eit.next();
627         Rectangle r = e.getRect();
628         g2.setColor(Color.WHITE);
629         g2.fillRoundRect(r.x, r.y, r.width, r.height, 10, 10);
630
631         g2.setColor(Color.BLACK);
632         if (e.isSelected()) {
633             g2.setStroke(hilite);
634             g2.drawRoundRect(r.x, r.y, r.width, r.height, 10, 10);
635             g2.setStroke(regular);
636         } else {
637             g2.drawRoundRect(r.x, r.y, r.width, r.height, 10, 10);
638         }
639         g2.drawString(e.getName(), r.x + 4, r.y + r.height / 2);
640     }
641
642     if (this.toolView.getViewMode() == ADD_ENTITY) {
643         Point pp = this.mouseDownPoint;
644         Point cp = this.currPoint;
645         if (pp != null && cp != null) {
646             g2.setStroke(dashed);
647             g2.drawRoundRect(pp.x, pp.y, cp.x - pp.x, cp.y - pp.y, 10, 10);
648         }
649     }
650 }

```

```

649     } else if (this.toolView.getViewMode() == ADD_ARROW) {
650         Point pp = this.mouseDownPoint;
651         Point cp = this.currPoint;
652         if (pp != null && model.getEntity(pp) != null) {
653             g2.setStroke(dashed);
654             g2.drawLine(pp.x, pp.y, cp.x, cp.y);
655             g2.drawOval(cp.x - 4, cp.y - 4, 8, 8);
656         }
657     }
658 }
659 }

```

## EntityView

```

660 package ui;
661
662 import model.ERModel;
663 import model.Entity;
664 import javax.swing.*;
665 import javax.swing.event.ListSelectionEvent;
666 import javax.swing.event.ListSelectionListener;
667 import javax.swing.table.AbstractTableModel;
668 import java.awt.*;
669 import java.util.Observable;
670 import java.util.Observer;
671
672 /**
673  * Created by bwbecker on 2016-11-03.
674  */
675 public class EntityView extends JPanel implements Observer {
676
677     private ERModel model;
678     private JLabel label = new JLabel("Entities");
679     private EntityTable eTable = new EntityTable();
680     private JTable table = new JTable(eTable);
681
682     private boolean updatingSelections = false;
683
684     public EntityView(ERModel model) {
685         this.model = model;
686
687         table.setPreferredScrollableViewportSize(new Dimension(200, 70));
688         table.setFillsViewportHeight(true);
689
690     }
691 }

```



```

692 this.setLayout(new BorderLayout(this, BorderLayout.Y_AXIS));
693 this.add(this.label);
694 this.add(new JScrollPane(this.table));
695
696
697 this.table.getSelectionModel().addListSelectionListener(new
698 ListSelectionListener() {
699     @Override
700     public void valueChanged(ListSelectionEvent e) {
701         if (!updatingSelections) {
702             if (!e.getValueAdjusting()) {
703                 model.selectEntity(eTable.uids[table.getSelectedRow()]);
704             }
705         }
706     }
707 });
708
709 this.model.addObserver(this);
710 }
711
712 public void update(Observable obs, Object obj) {
713     this.eTable.refreshUids();
714
715     ListSelectionModel lsm = this.table.getSelectionModel();
716     this.updatingSelections = true;
717     lsm.clearSelection();
718     for (int i = 0; i < this.eTable.uids.length; i++) {
719         if (this.model.getEntity(this.eTable.uids[i]).isSelected()) {
720             lsm.addSelectionInterval(i, i);
721         }
722     }
723     this.updatingSelections = false;
724 }
725
726 private class EntityTable extends AbstractTableModel {
727
728     int[] uids = new int[0];
729
730     void refreshUids() {
731         this.uids = model.getEntityUids();
732         updatingSelections = true;
733         this.fireTableDataChanged();
734         updatingSelections = false;
735     }
736
737     @Override

```

```

738     public int getRowCount() {
739         return uids.length;
740     }
741
742     @Override
743     public String getColumnName(int colIndex) {
744         switch (colIndex) {
745             case 0:
746                 return "UID";
747             case 1:
748                 return "Entity Name";
749             default:
750                 return "";
751         }
752     }
753
754     @Override
755     public boolean isCellEditable(int row, int col) {
756         return col == 1;
757     }
758
759     @Override
760     public int getColumnCount() {
761         return 2;
762     }
763
764     @Override
765     public Object getValueAt(int rowIndex, int columnIndex) {
766         Entity ent = model.getEntity(this.uids[rowIndex]);
767         assert (ent != null);
768
769         switch (columnIndex) {
770             case 0:
771                 return ent.getUid();
772             case 1:
773                 return ent.getName();
774         }
775         throw new Error("Fell through switch with columnIndex = " +
776 columnIndex);
777     }
778
779     @Override
780     public void setValueAt(Object val, int rowIndex, int colIndex) {
781
782         if (colIndex == 1) {
783             model.setEntityName(this.uids[rowIndex], val.toString());

```

```

784     }
785     }
786 }
787 }
788
789

```

## 790 ArrowView

```

791 package ui;
792
793 import model.Arrow;
794 import model.ERModel;
795 import javax.swing.*;
796 import javax.swing.event.ListSelectionEvent;
797 import javax.swing.event.ListSelectionListener;
798 import javax.swing.table.AbstractTableModel;
799 import java.awt.*;
800 import java.util.Observable;
801 import java.util.Observer;
802
803 /**
804  * Created by bwbecker on 2016-11-02.
805  */
806 public class ArrowView extends JPanel implements Observer {
807
808     private ERModel model;
809     private JLabel label = new JLabel("Arrows");
810     private ArrowTable aTable = new ArrowTable();
811     private JTable table = new JTable(aTable);
812
813     private boolean updatingSelections = false;
814
815     public ArrowView(ERModel model) {
816         this.model = model;
817
818         table.setPreferredSize(new Dimension(200, 70));
819         table.setFillsViewportHeight(true);
820
821         this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
822         this.add(this.label);
823         this.add(new JScrollPane(this.table));
824
825
826

```

```

827
828     this.table.getSelectionModel().addListSelectionListener(new
829 ListSelectionListener() {
830         @Override
831         public void valueChanged(ListSelectionEvent e) {
832             // See note on updatingSelections
833             if (!updatingSelections) {
834                 if (!e.getValueIsAdjusting()) {
835                     model.selectArrow(aTable.uids[table.getSelectedRow()]);
836                 }
837             }
838         }
839     });
840
841     this.model.addObserver(this);
842 }
843
844 public void update(Observable obs, Object obj) {
845     if (obj == (model.ADD_REMOVE_DATA)) {
846         this.aTable.refreshUids();
847     }
848
849     ListSelectionModel lsm = this.table.getSelectionModel();
850     this.updatingSelections = true;
851     lsm.clearSelection();
852     for (int i = 0; i < this.aTable.uids.length; i++) {
853         if (this.model.getArrow(this.aTable.uids[i]).isSelected()) {
854             lsm.addSelectionInterval(i, i);
855         }
856     }
857     this.updatingSelections = false;
858 }
859
860 private class ArrowTable extends AbstractTableModel {
861
862     private int[] uids = new int[0];
863
864     void refreshUids() {
865         this.uids = model.getArrowUids();
866         updatingSelections = true;
867         this.fireTableDataChanged();
868         updatingSelections = false;
869     }
870
871     @Override
872

```

```

873 public int getRowCount() {
874     return uids.length;
875 }
876
877 @Override
878 public int getColumnCount() {
879     return 3;
880 }
881
882 @Override
883 public String getColumnName(int colIndex) {
884     switch (colIndex) {
885         case 0: return "UID";
886         case 1: return "From Entity";
887         case 2: return "To Entity";
888         default: return "";
889     }
890 }
891
892 @Override
893 public Object getValueAt(int rowIndex, int columnIndex) {
894     Arrow arrow = model.getArrow(this.uids[rowIndex]);
895     assert(arrow != null);
896
897     switch (columnIndex) {
898         case 0:
899             return arrow.getUid();
900         case 1:
901             return arrow.fromName();
902         case 2:
903             return arrow.toName();
904     }
905     throw new Error("Fell through case with columnIndex = " + columnIndex);
906 }
907 }
908
909 }
910

```