```
    1: /* AUTO-GENERATED FILE.  DO NOT MODIFY.
    2:  *
    3:  * This class was automatically generated by the
    4:  * aapt tool from the resource data it found.  It
    5:  * should not be modified by hand.
    6:  */
    7:
    8: package com.iadf.roommapapp;
    9:
   10: public final class R {
   11:     public static final class array {
   12:         public static final int furnitureList=0x7f070001;
   13:         public static final int shapes=0x7f070000;
   14:     }
   15:     public static final class attr {
   16:     }
   17:     public static final class dimen {
   18:         /**  Default screen margins, per the Android Design guidelines.
   19:
   20:          Example customization of dimensions originally defined in res/values/di
mens.xml
   21:          (such as screen margins) for screens with more than 820dp of available
width. This
   22:          would include 7" and 10" devices in landscape (~960dp and ~1280dp respe
ctively).
   23:
   24:          */
   25:         public static final int activity_horizontal_margin=0x7f050000;
   26:         public static final int activity_vertical_margin=0x7f050001;
   27:     }
   28:     public static final class drawable {
   29:         public static final int ic_launcher=0x7f020000;
   30:         public static final int my_divider=0x7f020001;
   31:         public static final int oval=0x7f020002;
   32:         public static final int rectangle=0x7f020003;
   33:     }
   34:     public static final class id {
   35:         public static final int BottomButtons=0x7f0a0005;
   36:         public static final int TopButtons=0x7f0a0003;
   37:         public static final int a_guid=0x7f0a0006;
   38:         public static final int action_settings=0x7f0a000b;
   39:         public static final int checkbox_meat=0x7f0a0002;
   40:         public static final int container=0x7f0a0009;
   41:         public static final int furniture_guid=0x7f0a0007;
   42:         public static final int furniture_list=0x7f0a000a;
   43:         public static final int lengthNP=0x7f0a0000;
   44:         public static final int roomCanvas=0x7f0a0008;
   45:         public static final int roomViewer=0x7f0a0004;
   46:         public static final int widthNP=0x7f0a0001;
   47:     }
   48:     public static final class layout {
   49:         public static final int create_furniture_viewer=0x7f030000;
   50:         public static final int create_room_viewer=0x7f030001;
   51:         public static final int fragment_room_viewer=0x7f030002;
   52:         public static final int furniture_list_item=0x7f030003;
   53:         public static final int lookup_furniture_viewer=0x7f030004;
   54:         public static final int modify_furniture_viewer=0x7f030005;
   55:         public static final int room_canvas=0x7f030006;
   56:         public static final int room_viewer=0x7f030007;
   57:         public static final int view_furniture_list=0x7f030008;
   58:         public static final int view_furniture_list_viewer=0x7f030009;
   59:     }
   60:     public static final class menu {
```

```
   61:         public static final int room_viewer=0x7f090000;
   62:     }
   63:     public static final class raw {
   64:         public static final int sql=0x7f040000;
   65:     }
   66:     public static final class string {
   67:         public static final int action_settings=0x7f060002;
   68:         public static final int app_name=0x7f060000;
   69:         public static final int back=0x7f060003;
   70:         public static final int cancel=0x7f06000e;
   71:         public static final int container=0x7f060013;
   72:         public static final int create_furniture=0x7f060007;
   73:         public static final int create_furniture_dim=0x7f060011;
   74:         public static final int create_room=0x7f060005;
   75:         public static final int delete_room=0x7f060006;
   76:         public static final int furniture_guid=0x7f060014;
   77:         public static final int furniture_lookup=0x7f060017;
   78:         public static final int hello_world=0x7f060001;
   79:         public static final int length=0x7f06000b;
   80:         public static final int load_room=0x7f060004;
   81:         public static final int lookup_furniture=0x7f060008;
   82:         public static final int modify_furniture_dim=0x7f060012;
   83:         public static final int modify_room=0x7f060009;
   84:         public static final int modify_room_dim=0x7f060010;
   85:         public static final int ok=0x7f06000d;
   86:         public static final int room_dim=0x7f06000f;
   87:         public static final int select_furniture=0x7f060015;
   88:         public static final int select_room=0x7f060016;
   89:         public static final int view_furniture=0x7f06000a;
   90:         public static final int width=0x7f06000c;
   91:     }
   92:     public static final class style {
   93:         /**
   94:         Base application theme, dependent on API level. This theme is replaced
   95:         by AppBaseTheme from res/values-vXX/styles.xml on newer devices.
   96:
   97:
   98:             Theme customizations available in newer API levels can go in
   99:             res/values-vXX/styles.xml, while customizations related to
  100:             backward-compatibility can go here.
  101:
  102:
  103:         Base application theme for API 11+. This theme completely replaces
  104:         AppBaseTheme from res/values/styles.xml on API 11+ devices.
  105:
  106:  API 11 theme customizations can go here.
  107:
  108:         Base application theme for API 14+. This theme completely replaces
  109:         AppBaseTheme from BOTH res/values/styles.xml and
  110:         res/values-v11/styles.xml on API 14+ devices.
  111:
  112:  API 14 theme customizations can go here.
  113:          */
  114:         public static final int AppBaseTheme=0x7f080000;
  115:         /**  Application theme.
  116:  All customizations that are NOT specific to a particular API-level can go here.

  117:          */
  118:         public static final int AppTheme=0x7f080001;
  119:     }
  120: }
```

```
1: /** Automatically generated file. DO NOT MODIFY */
2: package com.iadf.roommapapp;
3:
4: public final class BuildConfig {
5:     public final static boolean DEBUG = true;
6: }
```

```
    1: package com.iadf.roommapapp;
    2:
    3: import android.app.Activity;
    4: import android.app.Fragment;
    5: import android.content.SharedPreferences;
    6: import android.database.Cursor;
    7: import android.database.sqlite.SQLiteDatabase;
    8: import android.os.Bundle;
    9: import android.support.v4.app.DialogFragment;
   10: import android.support.v4.app.FragmentActivity;
   11: import android.view.LayoutInflater;
   12: import android.view.Menu;
   13: import android.view.MenuItem;
   14: import android.view.View;
   15: import android.view.ViewGroup;
   16: import android.widget.Toast;
   17:
   18: import com.iadf.FurnitureDatabase.LocalDatabaseConnection.DatabaseHelper;
   19: import com.iadf.SystemController.DatabaseController.Furniture;
   20: import com.iadf.TwoDUserInterface.MenuPackage.CreateFurnitureDialog;
   21: import com.iadf.TwoDUserInterface.MenuPackage.CreateRoomDialog;
   22: import com.iadf.TwoDUserInterface.MenuPackage.CreateRoomDialog.LenghtAndWidthLis
tener;
   23: import com.iadf.TwoDUserInterface.MenuPackage.FurnitureListener;
   24: import com.iadf.TwoDUserInterface.MenuPackage.LookupFurnitureDialog;
   25: import com.iadf.TwoDUserInterface.MenuPackage.ViewFurnitureDialog;
   26: import com.iadf.TwoDUserInterface.MenuPackage.ViewRoomDialog;
   27:
   28: /**
   29:  * The main activity that contains all the buttons and fragments
   30:  *
   31:  * @author CSE324 Spring 2014 Team 4
   32:  */
   33: public class RoomViewer extends FragmentActivity implements LenghtAndWidthListen
er, FurnitureListener {
   34:
   35:         public static SQLiteDatabase db;
   36:         public static DatabaseHelper helper;
   37:         int width;
   38:         int length;
   39:         public static int roomNumber = 1;
   40:         public static Furniture furnitureBuffer = new Furniture(0,0,0,0,0,0,0,0)
;
   41:         Furniture selectedFurniture;
   42:         int operation;
   43:
   44:         SharedPreferences sp;
   45:
   46:         // cases from each dialog
   47:         static final int CREATE = 1;
   48:         static final int LOOKUP_FURNITURE = 2;
   49:         static final int LOAD_ROOM = 8;
   50:         static final int MODIFY_ROOM = 3;
   51:         static final int MODIFY_FURNITURE = 7;
   52:         static final int VIEW = 4;
   53:         static final int DELETE_ROOM = 5;
   54:         static final int DELETE_FURNITURE = 6;
   55:
   56:         @Override
   57:         protected void onCreate(Bundle savedInstanceState) {
   58:                 super.onCreate(savedInstanceState);
   59:                 setContentView(R.layout.room_viewer);
   60:                 db = (new DatabaseHelper(this)).getWritableDatabase();
```

```
   61:                 helper = new DatabaseHelper(this);
   62:
   63:                 if (savedInstanceState == null) {
   64:                         getFragmentManager().beginTransaction()
   65:                                         .add(R.id.container, new PlaceholderFrag
ment()).commit();
   66:                 }
   67:
   68:                 sp = getSharedPreferences("your_prefs", Activity.MODE_PRIVATE);
   69:                 if(sp.contains("room_number")) RoomViewer.roomNumber = sp.getInt
("room_number", -1);
   70:
   71:         }
   72:
   73:         /**
   74:          * saves the currently open room so that when the application is reopene
d it will remember which room to open.
   75:          */
   76:         @Override
   77:         protected void onPause() {
   78:                 super.onPause();
   79:                 sp = getSharedPreferences("your_prefs", Activity.MODE_PRIVATE);
   80:                 SharedPreferences.Editor editor = sp.edit();
   81:                 editor.putInt("room_number", roomNumber);
   82:                 editor.commit();
   83:         }
   84:
   85:         /**
   86:          * opens the room from when the application was last closed
   87:          */
   88:         @Override
   89:         protected void onResume() {
   90:                 super.onResume();
   91:                 sp = getSharedPreferences("your_prefs", Activity.MODE_PRIVATE);
   92:                 if(sp.contains("room_number")) RoomViewer.roomNumber = sp.getInt
("room_number", -1);
   93:         }
   94:
   95:         /**
   96:          * opens the menu when the menu button is pressed
   97:          */
   98:         @Override
   99:         public boolean onCreateOptionsMenu(Menu menu) {
  100:
  101:                 // Inflate the menu; this adds items to the action bar if it is
present.
  102:                 getMenuInflater().inflate(R.menu.room_viewer, menu);
  103:                 return true;
  104:         }
  105:
  106:         /**
  107:          * handles what happens when a menu option is touched
  108:          */
  109:         @Override
  110:         public boolean onOptionsItemSelected(MenuItem item) {
  111:                 // Handle action bar item clicks here. The action bar will
  112:                 // automatically handle clicks on the Home/Up button, so long
  113:                 // as you specify a parent activity in AndroidManifest.xml.
  114:                 int id = item.getItemId();
  115:                 if (id == R.id.action_settings) {
  116:                         return true;
  117:                 }
  118:                 return super.onOptionsItemSelected(item);
```

```
  119:         }
  120:
  121:         /**
  122:          * A placeholder fragment containing a simple view.
  123:          */
  124:         public static class PlaceholderFragment extends Fragment {
  125:
  126:                 public PlaceholderFragment() {
  127:                 }
  128:
  129:                 @Override
  130:                 public View onCreateView(LayoutInflater inflater, ViewGroup cont
ainer,
  131:                                 Bundle savedInstanceState) {
  132:                         View rootView = inflater.inflate(R.layout.fragment_room_
viewer,
  133:                                         container, false);
  134:                         return rootView;
  135:                 }
  136:         }
  137:
  138:         /**
  139:          * shows a list of all rooms in the database
  140:          * @param v
  141:          */
  142:         public void loadRoom(View v) {
  143:
  144:                 operation = RoomViewer.LOAD_ROOM;
  145:
  146:                 DialogFragment d = new ViewRoomDialog();
  147:                 d.show(getSupportFragmentManager(), "ViewRoomDialog");
  148:     }
  149:
  150:         /**
  151:          * opens a dialog to create a new room in the database
  152:          * @param v
  153:          */
  154:         public void createRoom(View v) {
  155:
  156:                 DialogFragment d = new CreateRoomDialog();
  157:                 d.show(getSupportFragmentManager(), "LengthAndWidthAlertDialog")
;
  158:     }
  159:
  160:         /**
  161:          * shows a list of rooms so the user can select one to delete
  162:          * @param v
  163:          */
  164:         public void deleteRoom(View v) {
  165:                 operation = RoomViewer.DELETE_ROOM;
  166:                 DialogFragment d = new ViewRoomDialog();
  167:                 d.show(getSupportFragmentManager(), "ViewRoomDialog");
  168:                 //helper.addFurniture(db, null);
  169:
  170:     }
  171:
  172:         /**
  173:          * opens the create furniture dialog
  174:          * @param v
  175:          */
  176:         public void createFurniture(View v) {
  177:                 DialogFragment d = new CreateFurnitureDialog();
  178:                 operation = RoomViewer.CREATE;
```

```
  179:                 d.show(getSupportFragmentManager(), "CreateFurnitureDialog");
  180:
  181:     }
  182:
  183:         /**
  184:          * opens the lookup furniture dialog
  185:          * @param v
  186:          */
  187:         public void lookupFurniture(View v) {
  188:         DialogFragment d = new LookupFurnitureDialog();
  189:         operation = RoomViewer.LOOKUP_FURNITURE;
  190:         d.show(getSupportFragmentManager(), "LookupFurnitureDialog");
  191:     }
  192:
  193:         /**
  194:          * opens the modify furniture dialog
  195:          * @param v
  196:          */
  197:         public void modifyFurniture(View v) {
  198:                 operation = RoomViewer.MODIFY_ROOM;
  199:                 DialogFragment d = new CreateRoomDialog();
  200:                 d.show(getSupportFragmentManager(), "ViewRoomDialog");
  201:     }
  202:
  203:         /**
  204:          * opens the view furniture dialog with a list of furniture in the curre
ntly open room
  205:          * @param v
  206:          */
  207:         public void viewFurniture(View v) {
  208:             operation = RoomViewer.VIEW;
  209:                 DialogFragment d = new ViewFurnitureDialog();
  210:                 d.show(getSupportFragmentManager(), "ViewFurnitureDialog");
  211:     }
  212:
  213:         /**
  214:          * handles what happens when the "OK" is clicked for a room dialog
  215:          */
  216:         @Override
  217:         public void onDialogPositiveClick(DialogFragment dialog, int length, int
 width) {
  218:                 this.width = width;
  219:                 this.length = length;
  220:                 helper.addRoom(db, width, length);
  221:                 dialog.dismiss();
  222:         }
  223:
  224:         /**
  225:          * closes the dialog when "Cancel" is clicked
  226:          */
  227:         @Override
  228:         public void onDialogNegativeClick(DialogFragment dialog) {
  229:
  230:         }
  231:
  232:         /**
  233:          * handles all cases when "OK" is clicked from a furniture dialog
  234:          */
  235:         @Override
  236:         public void onFurnitureDialogPositiveClick(DialogFragment dialog, Object
 f){
  237:                 switch(operation) {
  238:                         case RoomViewer.DELETE_FURNITURE:{
```

```
  239:                                    helper.deleteFurniture(db, (Furniture) f);
  240:                            }; break;
  241:                            case RoomViewer.DELETE_ROOM:{
  242:                                    helper.deleteRoom(db, (Integer) f);
  243:                                    RoomViewer.roomNumber = 1;
  244:                                    helper.openRoom(db, roomNumber);
  245:                                    finish();
  246:                                    startActivity(getIntent());
  247:                            }; break;
  248:                            case RoomViewer.MODIFY_FURNITURE:{
  249:                                    helper.modifyFurniture(db, (Furniture) f);
  250:                            }; break;
  251:                            //Furniture object is used as a buffer to hold the room
number, room width and room length.
  252:                            case RoomViewer.MODIFY_ROOM: {
  253:                                    Furniture k = (Furniture)f;
  254:                                    helper.modifyRoom(db, k.getRoomNumber(), k.getWi
dth(), k.getLength());
  255:                            } break;
  256:                            case RoomViewer.VIEW: {
  257:                                    RoomViewer.furnitureBuffer.setGUID((Integer) f);
  258:                                    Cursor c = helper.lookupFurniture(db, new Furnit
ure((Integer) f, 0, 0, 0, 0, 0, 0, 0));
  259:                                    c.moveToFirst();
  260:                                    RoomViewer.roomNumber = c.getInt(1);
  261:                                    helper.openRoom(db, roomNumber);
  262:                                    finish();
  263:                                    startActivity(getIntent());
  264:                            };break;
  265:                            case RoomViewer.LOOKUP_FURNITURE: {
  266:                                    Cursor c = helper.lookupFurniture(db, (Furniture
) f);
  267:                                    if(c.moveToFirst()) {
  268:                                            RoomViewer.roomNumber = c.getInt(1);
  269:                                            helper.openRoom(db, roomNumber);
  270:                                            finish();
  271:                                            startActivity(getIntent());
  272:                                    } else {
  273:                                            Toast.makeText(this, "Furniture Not Foun
d", Toast.LENGTH_LONG).show();
  274:                                    }
  275:                            }; break;
  276:                            case RoomViewer.LOAD_ROOM: {
  277:                                    Cursor c = helper.openRoom(db, (Integer) f);
  278:                                    c.moveToFirst();
  279:                                    RoomViewer.roomNumber = (Integer) f;
  280:                                    finish();
  281:                                    startActivity(getIntent());
  282:                            } break;
  283:                            case RoomViewer.CREATE: {
  284:                                    helper.addFurniture(db, (Furniture) f);
  285:                                    finish();
  286:                                    startActivity(getIntent());
  287:                            };  break;
  288:                            default: {System.out.println("Error");} break;
  289:                    }
  290:
  291:                    dialog.dismiss();
  292:            }
  293:
  294:            /**
  295:             * closes the dialog when "Cancel" is clicked
  296:             */
```

```
297:         @Override
298:         public void onFurnitureDialogNegativeClick(DialogFragment dialog) {
299:
300:         }
301:
302:         /**
303:          * sets a furniture object to be a container when the user checks the
304:          * container checkbox
305:          * @param v
306:          */
307:         public void onCheckboxClicked(View v) {
308:
309:         }
310:
311: }
```

```
    1: package com.iadf.roommapapp;
    2:
    3:
    4: import java.util.ArrayList;
    5:
    6: import android.annotation.SuppressLint;
    7: import android.content.Context;
    8: import android.database.Cursor;
    9: import android.graphics.Bitmap;
   10: import android.graphics.BitmapFactory;
   11: import android.graphics.Canvas;
   12: import android.graphics.Color;
   13: import android.graphics.Paint;
   14: import android.graphics.Point;
   15: import android.os.Bundle;
   16: import android.support.v4.app.Fragment;
   17: import android.view.LayoutInflater;
   18: import android.view.MotionEvent;
   19: import android.view.View;
   20: import android.view.ViewGroup;
   21:
   22: import com.iadf.SystemController.DatabaseController.Furniture;
   23:
   24: /**
   25:  * Contains the canvas to draw the room and furniture
   26:  *
   27:  * @author CSE324 Spring 2014 Team 4
   28:  */
   29: public class RoomCanvas extends Fragment {
   30:
   31:         @Override
   32:     public View onCreateView(LayoutInflater inflater, ViewGroup container,
   33:                              Bundle savedInstanceState) {
   34:         // Inflate the layout for this fragment
   35:         return (new RoomView(this.getActivity()));
   36:     }
   37:
   38:     /**
   39:      * This class does the drawing when the room is opened or modified or when f
urniture
   40:      * is moved or modified
   41:      */
   42:     public class RoomView extends View {
   43:
   44:         private ArrayList<Furniture> furnitureItems;
   45:         private int i=-1;
   46:         int xOffset = 25;
   47:         int yOffset = 25;
   48:         int width = 999999;
   49:         int height = 999999;
   50:         Furniture f;
   51:         boolean canvasDrawn = false;
   52:
   53:         public RoomView(Context context) {
   54:                 super(context);
   55:             setFocusable(true); //necessary for getting the touch events
   56:
   57:             furnitureItems = createFurnitureList();
   58:
   59:         }
   60:
   61:         /**
   62:          * returns a list of furniture contained within the room so that they ca
```

```
n be drawn to the canvas
   63:          * @return the list of furniture
   64:          */
   65:         public ArrayList<Furniture> createFurnitureList() {
   66:                 ArrayList<Furniture> furniture = new ArrayList<Furniture>();
   67:
   68:             Cursor c = RoomViewer.helper.getFurnitureList(RoomViewer.db, RoomVie
wer.roomNumber);
   69:
   70:             c.moveToFirst();
   71:             while (c.isAfterLast() == false)
   72:             {
   73:                 Furniture f = new Furniture(c.getInt(0), c.getInt(1), c.getInt(2
), c.getInt(3), c.getInt(4), c.getInt(5), c.getInt(6), c.getInt(7));
   74:                 furniture.add(f);
   75:                 c.moveToNext();
   76:             }
   77:
   78:             return furniture;
   79:         }
   80:
   81:          /**
   82:           * draws all furniture objects that are contained within the furniture
list
   83:           */
   84:          @SuppressLint("DrawAllocation")
   85:                  @Override
   86:          protected void onDraw(Canvas canvas) {
   87:
   88:                 width = canvas.getWidth();
   89:                 height = canvas.getHeight();
   90:
   91:                 Paint paint = new Paint();
   92:
   93:                 paint.setColor(Color.BLACK);
   94:             paint.setStrokeWidth(5);
   95:             canvas.drawRect(95, 95, width - 95, height - 95, paint);
   96:             paint.setColor(Color.WHITE);
   97:             paint.setStrokeWidth(5);
   98:             canvas.drawRect(100, 100, width - 100, height - 100, paint);
   99:
  100:             canvasDrawn = true;
  101:
  102:                 for (Furniture furniture : furnitureItems) {
  103:                         Bitmap bitmap;
  104:                         if(furniture.getShape() == Furniture.OVAL) bitmap = Bitm
apFactory.decodeResource(this.getResources(), R.drawable.oval);
  105:                         else bitmap = BitmapFactory.decodeResource(this.getResou
rces(), R.drawable.rectangle);
  106:                         Bitmap furnitureItem = Bitmap.createScaledBitmap(bitmap,
 furniture.getLength(), furniture.getWidth(), true);
  107:                         Point p = furniture.getCenter();
  108:                 canvas.drawBitmap(furnitureItem, p.x, p.y, null);
  109:             }
  110:
  111:          }
  112:
  113:          /**
  114:           * determines where to draw the item based on X-coordinate in the datab
ase
  115:           */
  116:          public int setX(int x) {
  117:                  if(x < 102) {
```

```
  118:                         return 102;
  119:                 } else if(x > width - 100 - 2*xOffset) {
  120:                         return width - 100 - 2*xOffset;
  121:                 }
  122:                 return x;
  123:         }
  124:
  125:         /**
  126:          * determines where to draw the item based on Y-coordinate in the datab
ase
  127:          */
  128:         public int setY(int y) {
  129:                 if(y < 102) {
  130:                         return 102;
  131:                 } else if (y > height - 100 - 2*yOffset) {
  132:                         return height - 100 - 2*yOffset;
  133:                 }
  134:                 return y;
  135:         }
  136:
  137:         /**
  138:          * handles moving the furniture
  139:          */
  140:         public boolean onTouchEvent(MotionEvent event) {
  141:             int eventaction = event.getAction();
  142:
  143:             int X = (int)event.getX();
  144:             int Y = (int)event.getY();
  145:
  146:             switch (eventaction ) {
  147:
  148:             case MotionEvent.ACTION_DOWN: // checks if the touch is on a furnit
ure item
  149:                i = -1;
  150:                for (Furniture furniture : furnitureItems) {
  151:                        Point p = furniture.getCenter();
  152:
  153:                        if (X > p.x && X < p.x+furniture.getLength() && Y > p.y
&& Y < p.y+furniture.getWidth()){
  154:                        i = furnitureItems.indexOf(furniture);
  155:                        xOffset = furniture.getLength()/2 + 1;
  156:                        yOffset = furniture.getWidth()/2 + 1;
  157:                        break;
  158:                    }
  159:                   }
  160:
  161:                  break;
  162:
  163:
  164:             case MotionEvent.ACTION_MOVE:   // drags the furniture object updat
ing its coordinates
  165:                  if (i >= 0) {
  166:                        f = furnitureItems.get(i);
  167:                        f.setCenter(setX(X-xOffset), setY(Y-yOffset));
  168:                        furnitureItems.set(i, f);
  169:
  170:                  }
  171:
  172:                  break;
  173:
  174:             case MotionEvent.ACTION_UP: // update the database after the finger
 is taken off the screen
  175:                  xOffset = 25;
```

```
176:                    yOffset = 25;
177:                    if(f != null) {
178:                           RoomViewer.helper.modifyFurniture(RoomViewer.db, f);
179:                           furnitureItems = createFurnitureList();
180:                    }
181:                    break;
182:                }
183:
184:                // redraws the canvas
185:                invalidate();
186:                return true;
187:
188:            }
189:        }
190: }
```

```
    1: package com.iadf.TwoDUserInterface.MenuPackage;
    2:
    3: import android.app.Activity;
    4: import android.app.AlertDialog;
    5: import android.app.Dialog;
    6: import android.database.Cursor;
    7: import android.os.Bundle;
    8: import android.support.v4.app.DialogFragment;
    9: import android.support.v4.widget.SimpleCursorAdapter;
   10: import android.view.LayoutInflater;
   11: import android.view.View;
   12: import android.widget.AdapterView;
   13: import android.widget.AdapterView.OnItemClickListener;
   14: import android.widget.ListAdapter;
   15: import android.widget.ListView;
   16:
   17: import com.iadf.FurnitureDatabase.LocalDatabaseConnection.DatabaseHelper;
   18: import com.iadf.roommapapp.R;
   19: import com.iadf.roommapapp.RoomViewer;
   20:
   21: /**
   22:  * Displays a list of furniture in a new dialog allowing the user to select one.
   23:  *
   24:  * @author CSE324 Spring 2014 Team 4
   25:  */
   26: public class ViewFurnitureDialog extends DialogFragment {
   27:
   28:         DatabaseHelper helper = RoomViewer.helper;
   29:         int room = RoomViewer.roomNumber;
   30:
   31:         Cursor c = helper.getFurnitureList(RoomViewer.db, room);
   32:
   33:         FurnitureListener mListener;
   34:         int id;
   35:
   36:     // Override the Fragment.onAttach() method to instantiate the NoticeDialogLi
stener
   37:     @Override
   38:     public void onAttach(Activity activity) {
   39:         super.onAttach(activity);
   40:         // Verify that the host activity implements the callback interface
   41:         try {
   42:             // Instantiate the LookupDialogListener so we can send events to the
 host
   43:             mListener = (FurnitureListener) activity;
   44:         } catch (ClassCastException e) {
   45:             // The activity doesn't implement the interface, throw exception
   46:             throw new ClassCastException(activity.toString()
   47:                     + " must implement NoticeDialogListener");
   48:         }
   49:     }
   50:
   51:         public Dialog onCreateDialog(Bundle savedInstanceState) {
   52:             AlertDialog.Builder builder = new AlertDialog.Builder(this.getActivi
ty());
   53:             // Get the layout inflater
   54:             LayoutInflater inflater = this.getActivity().getLayoutInflater();
   55:
   56:
   57:             // Inflate and set the layout for the dialog
   58:             // Pass null as the parent view because its going in the dialog layo
ut
   59:             final View v = inflater.inflate(R.layout.view_furniture_list_viewer,
```

```
 null);
   60:
   61:             final ListAdapter adapter = new SimpleCursorAdapter(
   62:                 this.getActivity(),
   63:                 R.layout.furniture_list_item,
   64:                 c,
   65:                 new String[] {"_id"},
   66:                 new int[] {R.id.a_guid},
   67:                 0);
   68:             final ListView mFurnitureList = (ListView)v.findViewById(R.id.furnit
ure_list);
   69:             mFurnitureList.setAdapter(adapter);
   70:
   71:             mFurnitureList.setOnItemClickListener(new OnItemClickListener() {
   72:                 @Override
   73:                         public void onItemClick(final AdapterView<?> parentView,
 View view, int position, long id) {
   74:                     ((ListView) parentView).setItemChecked(position, true);
   75:                         Cursor item = (Cursor) ((ListView) parentView).getAdapte
r().getItem(position);
   76:                         if(item.moveToPosition(position)) {
   77:                                 mListener.onFurnitureDialogPositiveClick(ViewFur
nitureDialog.this, (Object) item.getInt(0));
   78:                         }
   79:                 }
   80:             });
   81:
   82:     builder.setView(v);
   83:
   84:     builder.setTitle(R.string.select_furniture);
   85:     return builder.create();
   86:         }
   87: }
```

```
    1: package com.iadf.TwoDUserInterface.MenuPackage;
    2:
    3: import android.app.Activity;
    4: import android.app.AlertDialog;
    5: import android.app.Dialog;
    6: import android.database.Cursor;
    7: import android.os.Bundle;
    8: import android.support.v4.app.DialogFragment;
    9: import android.support.v4.widget.SimpleCursorAdapter;
   10: import android.view.LayoutInflater;
   11: import android.view.View;
   12: import android.widget.AdapterView;
   13: import android.widget.AdapterView.OnItemClickListener;
   14: import android.widget.ListAdapter;
   15: import android.widget.ListView;
   16:
   17: import com.iadf.FurnitureDatabase.LocalDatabaseConnection.DatabaseHelper;
   18: import com.iadf.roommapapp.R;
   19: import com.iadf.roommapapp.RoomViewer;
   20:
   21: /**
   22:  * Displays a list of rooms and allows the user to select one to open
   23:  *
   24:  * @author CSE324 Spring 2014 Team 4
   25:  */
   26: public class ViewRoomDialog extends DialogFragment {
   27:
   28:         DatabaseHelper helper = RoomViewer.helper;
   29:         int room = RoomViewer.roomNumber;
   30:
   31:         Cursor c = helper.viewRooms(RoomViewer.db);
   32:
   33:         FurnitureListener mListener;
   34:         int id;
   35:
   36:     // Override the Fragment.onAttach() method to instantiate the NoticeDialogLi
stener
   37:     @Override
   38:     public void onAttach(Activity activity) {
   39:         super.onAttach(activity);
   40:         // Verify that the host activity implements the callback interface
   41:         try {
   42:             // Instantiate the LookupDialogListener so we can send events to the
 host
   43:             mListener = (FurnitureListener) activity;
   44:         } catch (ClassCastException e) {
   45:             // The activity doesn't implement the interface, throw exception
   46:             throw new ClassCastException(activity.toString()
   47:                     + " must implement NoticeDialogListener");
   48:         }
   49:     }
   50:
   51:         public Dialog onCreateDialog(Bundle savedInstanceState) {
   52:             AlertDialog.Builder builder = new AlertDialog.Builder(this.getActivi
ty());
   53:             // Get the layout inflater
   54:             LayoutInflater inflater = this.getActivity().getLayoutInflater();
   55:
   56:
   57:             // Inflate and set the layout for the dialog
   58:             // Pass null as the parent view because its going in the dialog layo
ut
   59:             final View v = inflater.inflate(R.layout.view_furniture_list_viewer,
```

```
 null);
   60:
   61:             final ListAdapter adapter = new SimpleCursorAdapter(
   62:                 this.getActivity(),
   63:                 R.layout.furniture_list_item,
   64:                 c,
   65:                 new String[] {"_id"},
   66:                 new int[] {R.id.a_guid},
   67:                 0);
   68:             final ListView mFurnitureList = (ListView)v.findViewById(R.id.furnit
ure_list);
   69:             mFurnitureList.setAdapter(adapter);
   70:
   71:             mFurnitureList.setOnItemClickListener(new OnItemClickListener() {
   72:                 @Override
   73:                         public void onItemClick(final AdapterView<?> parentView,
 View view, int position, long id) {
   74:                     ((ListView) parentView).setItemChecked(position, true);
   75:                         Cursor item = (Cursor) ((ListView) parentView).getAdapte
r().getItem(position);
   76:                         if(item.moveToPosition(position)) {
   77:                                 mListener.onFurnitureDialogPositiveClick(ViewRoo
mDialog.this, (Object) item.getInt(0));
   78:                         }
   79:                 }
   80:             });
   81:
   82:     builder.setView(v);
   83:     builder.setTitle(R.string.select_room);
   84:     return builder.create();
   85:         }
   86: }
```

```
    1: package com.iadf.TwoDUserInterface.MenuPackage;
    2:
    3:
    4: import android.app.Activity;
    5: import android.app.AlertDialog;
    6: import android.app.Dialog;
    7: import android.content.DialogInterface;
    8: import android.content.DialogInterface.OnClickListener;
    9: import android.os.Bundle;
   10: import android.support.v4.app.DialogFragment;
   11: import android.view.LayoutInflater;
   12: import android.view.View;
   13: import android.widget.NumberPicker;
   14:
   15: import com.iadf.SystemController.DatabaseController.Furniture;
   16: import com.iadf.roommapapp.R;
   17: import com.iadf.roommapapp.RoomViewer;
   18:
   19:
   20: /**
   21:  * Allows the user to change the attributes of a furniture object in the databas
e
   22:  *
   23:  * @author CSE324 Spring 2014 Team 4
   24:  */
   25: public class ModifyFurnitureDialog extends DialogFragment {
   26:
   27:
   28:         FurnitureListener mListener;
   29:         int shape;
   30:         int checkedItem = 0;
   31:         boolean[] isChecked;
   32:         int type;
   33:
   34:     // Override the Fragment.onAttach() method to instantiate the NoticeDialogLi
stener
   35:     @Override
   36:     public void onAttach(Activity activity) {
   37:         super.onAttach(activity);
   38:         // Verify that the host activity implements the callback interface
   39:         try {
   40:             // Instantiate the NoticeDialogListener so we can send events to the
 host
   41:             mListener = (FurnitureListener) activity;
   42:         } catch (ClassCastException e) {
   43:             // The activity doesn't implement the interface, throw exception
   44:             throw new ClassCastException(activity.toString()
   45:                     + " must implement NoticeDialogListener");
   46:         }
   47:     }
   48:         @Override
   49:         public Dialog onCreateDialog(Bundle savedInstanceState) {
   50:             AlertDialog.Builder builder = new AlertDialog.Builder(this.getActivi
ty());
   51:             // Get the layout inflater
   52:             LayoutInflater inflater = this.getActivity().getLayoutInflater();
   53:
   54:
   55:
   56:             // Inflate and set the layout for the dialog
   57:             // Pass null as the parent view because its going in the dialog layo
ut
   58:             final View v = inflater.inflate(R.layout.create_furniture_viewer, nu
```

```
ll);
   59:
   60:             final NumberPicker widthNP = (NumberPicker) v.findViewById(R.id.widt
hNP);
   61:             widthNP.setMaxValue(100000);
   62:             widthNP.setMinValue(1);
   63:             widthNP.setValue(100);
   64:
   65:             final NumberPicker lengthNP = (NumberPicker) v.findViewById(R.id.len
gthNP);
   66:             lengthNP.setMaxValue(100000);
   67:             lengthNP.setMinValue(1);
   68:             lengthNP.setValue(100);
   69:
   70:             builder.setView(v)
   71:             // Add action buttons
   72:                     .setSingleChoiceItems(R.array.shapes,  checkedItem, new OnCli
ckListener() {
   73:
   74:                                     @Override
   75:                                     public void onClick(DialogInterface dialog, i
nt which) {
   76:                                             shape = which;
   77:
   78:                                     }
   79:                             })
   80:                     .setPositiveButton(R.string.ok, new DialogInterface.OnClickLi
stener() {
   81:                         @Override
   82:                         public void onClick(DialogInterface dialog, int id) {
   83:                             int width;
   84:                             int length;
   85:                             width = widthNP.getValue();
   86:                             length = lengthNP.getValue();
   87:
   88:                             Furniture furniture = new Furniture((int)(Math.random
()*1000000), RoomViewer.roomNumber, width, length, shape, 0, 0, type);
   89:
   90:                             mListener.onFurnitureDialogPositiveClick(ModifyFurnit
ureDialog.this, furniture);
   91:                         }
   92:                     })
   93:                     .setNegativeButton(R.string.cancel, new DialogInterface.OnCli
ckListener() {
   94:                         public void onClick(DialogInterface dialog, int id) {
   95:                             mListener.onFurnitureDialogNegativeClick(ModifyFurnit
ureDialog.this);
   96:                             dialog.cancel();
   97:                         }
   98:
   99:                     });
  100:
  101:             builder.setTitle(R.string.create_furniture_dim);
  102:
  103:
  104:             return builder.create();
  105:         }
  106: }
```

```
    1: package com.iadf.TwoDUserInterface.MenuPackage;
    2:
    3:
    4: import android.support.v4.app.DialogFragment;
    5:
    6: /**
    7:  * This interface is for implementing actions when buttons are pressed on
    8:  * the different dialogs dealing with furniture objects
    9:  *
   10:  * @author CSE324 Spring 2014 Team 4
   11:  */
   12: public interface FurnitureListener {
   13:      public void onFurnitureDialogPositiveClick(DialogFragment dialog, Object f)
;
   14:      public void onFurnitureDialogNegativeClick(DialogFragment dialog);
   15:
   16: }
```

```
    1: package com.iadf.TwoDUserInterface.MenuPackage;
    2:
    3: import android.app.Activity;
    4: import android.app.AlertDialog;
    5: import android.app.Dialog;
    6: import android.content.DialogInterface;
    7: import android.os.Bundle;
    8: import android.support.v4.app.DialogFragment;
    9: import android.view.LayoutInflater;
   10: import android.view.View;
   11: import android.widget.EditText;
   12:
   13: import com.iadf.SystemController.DatabaseController.Furniture;
   14: import com.iadf.roommapapp.R;
   15:
   16: /**
   17:  * Allows the user to input a GUID and open the room containing that furniture i
tem
   18:  *
   19:  * @author CSE324 Spring 2014 Team 4
   20:  */
   21: public class LookupFurnitureDialog extends DialogFragment {
   22:
   23:
   24:
   25:
   26:         FurnitureListener mListener;
   27:
   28:     // Override the Fragment.onAttach() method to instantiate the NoticeDialogLi
stener
   29:     @Override
   30:     public void onAttach(Activity activity) {
   31:         super.onAttach(activity);
   32:         // Verify that the host activity implements the callback interface
   33:         try {
   34:             // Instantiate the LookupDialogListener so we can send events to the
 host
   35:             mListener = (FurnitureListener) activity;
   36:         } catch (ClassCastException e) {
   37:             // The activity doesn't implement the interface, throw exception
   38:             throw new ClassCastException(activity.toString()
   39:                     + " must implement NoticeDialogListener");
   40:         }
   41:     }
   42:
   43:         public Dialog onCreateDialog(Bundle savedInstanceState) {
   44:             AlertDialog.Builder builder = new AlertDialog.Builder(this.getActivi
ty());
   45:             // Get the layout inflater
   46:             LayoutInflater inflater = this.getActivity().getLayoutInflater();
   47:
   48:
   49:
   50:             // Inflate and set the layout for the dialog
   51:             // Pass null as the parent view because its going in the dialog layo
ut
   52:             final View v = inflater.inflate(R.layout.lookup_furniture_viewer, nu
ll);
   53:
   54:     builder.setView(v)
   55:     // Add action buttons
   56:            .setPositiveButton(R.string.ok, new DialogInterface.OnClickListener()
 {
```

```
   57:
   58:                         @Override
   59:                 public void onClick(DialogInterface dialog, int id) {
   60:                     EditText et = (EditText) v.findViewById(R.id.furniture_guid);
   61:                     String s = et.getText().toString();
   62:                     Furniture f = null;
   63:                     if(!s.equals("")) {
   64:                             f = new Furniture(Integer.parseInt(s), 0, 0, 0, 0, 0,
 0, 0);
   65:                             mListener.onFurnitureDialogPositiveClick(LookupFurnit
ureDialog.this, f );
   66:                     }
   67:
   68:                         dialog.dismiss();
   69:
   70:                 }
   71:             })
   72:             .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListen
er() {
   73:                 public void onClick(DialogInterface dialog, int id) {
   74:                     mListener.onFurnitureDialogNegativeClick(LookupFurnitureDialo
g.this);
   75:                     dialog.cancel();
   76:                 }
   77:             });
   78:
   79:     builder.setTitle(R.string.furniture_lookup);
   80:     return builder.create();
   81:         }
   82: }
```

```
    1: package com.iadf.TwoDUserInterface.MenuPackage;
    2:
    3:
    4: import android.app.Activity;
    5: import android.app.AlertDialog;
    6: import android.app.Dialog;
    7: import android.content.DialogInterface;
    8: import android.os.Bundle;
    9: import android.support.v4.app.DialogFragment;
   10: import android.view.LayoutInflater;
   11: import android.view.View;
   12: import android.widget.NumberPicker;
   13:
   14: import com.iadf.roommapapp.R;
   15:
   16:
   17: /**
   18:  * Displays a dialog to the user allowing them to create a new room and specify
its dimensions
   19:  *
   20:  * @author CSE324 Spring 2014 Team 4
   21:  */
   22: public class CreateRoomDialog extends DialogFragment {
   23:
   24:         public interface LenghtAndWidthListener {
   25:         public void onDialogPositiveClick(DialogFragment dialog, int length, int
 width);
   26:         public void onDialogNegativeClick(DialogFragment dialog);
   27:     }
   28:
   29:         LenghtAndWidthListener mListener;
   30:
   31:     // Override the Fragment.onAttach() method to instantiate the NoticeDialogLi
stener
   32:     @Override
   33:     public void onAttach(Activity activity) {
   34:         super.onAttach(activity);
   35:         // Verify that the host activity implements the callback interface
   36:         try {
   37:             // Instantiate the NoticeDialogListener so we can send events to the
 host
   38:             mListener = (LenghtAndWidthListener) activity;
   39:         } catch (ClassCastException e) {
   40:             // The activity doesn't implement the interface, throw exception
   41:             throw new ClassCastException(activity.toString()
   42:                     + " must implement NoticeDialogListener");
   43:         }
   44:     }
   45:         @Override
   46:         public Dialog onCreateDialog(Bundle savedInstanceState) {
   47:             AlertDialog.Builder builder = new AlertDialog.Builder(this.getActivi
ty());
   48:             // Get the layout inflater
   49:             LayoutInflater inflater = this.getActivity().getLayoutInflater();
   50:
   51:
   52:
   53:             // Inflate and set the layout for the dialog
   54:             // Pass null as the parent view because its going in the dialog layo
ut
   55:             final View v = inflater.inflate(R.layout.create_room_viewer, null);
   56:
   57:             final NumberPicker widthNP = (NumberPicker) v.findViewById(R.id.widt
```

```
hNP);
   58:             widthNP.setMaxValue(1000);
   59:             widthNP.setMinValue(1);
   60:             widthNP.setValue(25);
   61:
   62:             final NumberPicker lengthNP = (NumberPicker) v.findViewById(R.id.len
gthNP);
   63:             lengthNP.setMaxValue(1000);
   64:             lengthNP.setMinValue(1);
   65:             lengthNP.setValue(50);
   66:
   67:             builder.setView(v)
   68:             // Add action buttons
   69:                     .setPositiveButton(R.string.ok, new DialogInterface.OnClickLi
stener() {
   70:                         @Override
   71:                         public void onClick(DialogInterface dialog, int id) {
   72:                             int width;
   73:                             int length;
   74:                             width = widthNP.getValue();
   75:                             length = lengthNP.getValue();
   76:                             mListener.onDialogPositiveClick(CreateRoomDialog.this
, length, width);
   77:                         }
   78:                     })
   79:                     .setNegativeButton(R.string.cancel, new DialogInterface.OnCli
ckListener() {
   80:                         public void onClick(DialogInterface dialog, int id) {
   81:                             mListener.onDialogNegativeClick(CreateRoomDialog.this
);
   82:                             dialog.cancel();
   83:                         }
   84:                     });
   85:
   86:             builder.setTitle(R.string.room_dim);
   87:
   88:             return builder.create();
   89:         }
   90: }
```

```
    1: package com.iadf.TwoDUserInterface.MenuPackage;
    2:
    3:
    4: import android.app.Activity;
    5: import android.app.AlertDialog;
    6: import android.app.Dialog;
    7: import android.content.DialogInterface;
    8: import android.os.Bundle;
    9: import android.support.v4.app.DialogFragment;
   10: import android.view.LayoutInflater;
   11: import android.view.View;
   12: import android.widget.NumberPicker;
   13:
   14: import com.iadf.SystemController.DatabaseController.Furniture;
   15: import com.iadf.roommapapp.R;
   16:
   17:
   18: /**
   19:  * Allows the user to change the dimension of a room
   20:  *
   21:  * @author CSE324 Spring 2014 Team 4
   22:  */
   23: public class ModifyRoomDialog extends DialogFragment {
   24:
   25:         FurnitureListener mListener;
   26:
   27:     // Override the Fragment.onAttach() method to instantiate the NoticeDialogLi
stener
   28:     @Override
   29:     public void onAttach(Activity activity) {
   30:         super.onAttach(activity);
   31:         // Verify that the host activity implements the callback interface
   32:         try {
   33:             // Instantiate the NoticeDialogListener so we can send events to the
 host
   34:             mListener = (FurnitureListener) activity;
   35:         } catch (ClassCastException e) {
   36:             // The activity doesn't implement the interface, throw exception
   37:             throw new ClassCastException(activity.toString()
   38:                     + " must implement NoticeDialogListener");
   39:         }
   40:     }
   41:         @Override
   42:         public Dialog onCreateDialog(Bundle savedInstanceState) {
   43:             AlertDialog.Builder builder = new AlertDialog.Builder(this.getActivi
ty());
   44:             // Get the layout inflater
   45:             LayoutInflater inflater = this.getActivity().getLayoutInflater();
   46:
   47:
   48:
   49:             // Inflate and set the layout for the dialog
   50:             // Pass null as the parent view because its going in the dialog layo
ut
   51:             final View v = inflater.inflate(R.layout.create_room_viewer, null);
   52:
   53:             final NumberPicker widthNP = (NumberPicker) v.findViewById(R.id.widt
hNP);
   54:             widthNP.setMaxValue(100000);
   55:             widthNP.setMinValue(1);
   56:
   57:             final NumberPicker lengthNP = (NumberPicker) v.findViewById(R.id.len
gthNP);
```

```
   58:             lengthNP.setMaxValue(100000);
   59:             lengthNP.setMinValue(1);
   60:
   61:             builder.setView(v)
   62:             // Add action buttons
   63:                    .setPositiveButton(R.string.ok, new DialogInterface.OnClickLi
stener() {
   64:                        @Override
   65:                        public void onClick(DialogInterface dialog, int id) {
   66:                            Furniture f = new Furniture(0,0,0,0,0,0,0,0);
   67:                            f.setRoomNumber(1);
   68:                            f.setWidth(widthNP.getValue());
   69:                            f.setLength(lengthNP.getValue());
   70:                            mListener.onFurnitureDialogPositiveClick(ModifyRoomDi
alog.this, (Object) f);
   71:                        }
   72:                    })
   73:                    .setNegativeButton(R.string.cancel, new DialogInterface.OnCli
ckListener() {
   74:                        public void onClick(DialogInterface dialog, int id) {
   75:                            mListener.onFurnitureDialogNegativeClick(ModifyRoomDi
alog.this);
   76:                            dialog.cancel();
   77:                        }
   78:                    });
   79:
   80:             builder.setTitle(R.string.modify_room_dim);
   81:
   82:             return builder.create();
   83:         }
   84:
   85: }
```

```
    1: package com.iadf.TwoDUserInterface.MenuPackage;
    2:
    3:
    4: import android.app.Activity;
    5: import android.app.AlertDialog;
    6: import android.app.Dialog;
    7: import android.content.DialogInterface;
    8: import android.content.DialogInterface.OnClickListener;
    9: import android.os.Bundle;
   10: import android.support.v4.app.DialogFragment;
   11: import android.view.LayoutInflater;
   12: import android.view.View;
   13: import android.widget.NumberPicker;
   14:
   15: import com.iadf.SystemController.DatabaseController.Furniture;
   16: import com.iadf.roommapapp.R;
   17: import com.iadf.roommapapp.RoomViewer;
   18:
   19:
   20: /**
   21:  * This Class creates the fragment that allows the user to input values
   22:  * and create a new furniture object.
   23:  *
   24:  * @author CSE324 Spring 2014 Team 4
   25:  */
   26: public class CreateFurnitureDialog extends DialogFragment {
   27:
   28:
   29:         FurnitureListener mListener;
   30:         int shape;
   31:         int checkedItem = 0;
   32:         boolean[] isChecked;
   33:         int type;
   34:
   35:     // Override the Fragment.onAttach() method to instantiate the NoticeDialogLi
stener
   36:     @Override
   37:     public void onAttach(Activity activity) {
   38:         super.onAttach(activity);
   39:         // Verify that the host activity implements the callback interface
   40:         try {
   41:             // Instantiate the NoticeDialogListener so we can send events to the
 host
   42:             mListener = (FurnitureListener) activity;
   43:         } catch (ClassCastException e) {
   44:             // The activity doesn't implement the interface, throw exception
   45:             throw new ClassCastException(activity.toString()
   46:                     + " must implement NoticeDialogListener");
   47:         }
   48:     }
   49:         @Override
   50:         public Dialog onCreateDialog(Bundle savedInstanceState) {
   51:             AlertDialog.Builder builder = new AlertDialog.Builder(this.getActivi
ty());
   52:             // Get the layout inflater
   53:             LayoutInflater inflater = this.getActivity().getLayoutInflater();
   54:
   55:
   56:
   57:             // Inflate and set the layout for the dialog
   58:             // Pass null as the parent view because its going in the dialog layo
ut
   59:             final View v = inflater.inflate(R.layout.create_furniture_viewer, nu
```

```
ll);
   60:
   61:             final NumberPicker widthNP = (NumberPicker) v.findViewById(R.id.widt
hNP);
   62:             widthNP.setMaxValue(100000);
   63:             widthNP.setMinValue(1);
   64:             widthNP.setValue(100);
   65:
   66:             final NumberPicker lengthNP = (NumberPicker) v.findViewById(R.id.len
gthNP);
   67:             lengthNP.setMaxValue(100000);
   68:             lengthNP.setMinValue(1);
   69:             lengthNP.setValue(100);
   70:
   71:
   72:             builder.setView(v)
   73:             // Add action buttons
   74:                     .setSingleChoiceItems(R.array.shapes,  checkedItem, new OnCli
ckListener() {
   75:
   76:                                     @Override
   77:                                     public void onClick(DialogInterface dialog, i
nt which) {
   78:                                             shape = which;
   79:
   80:                                     }
   81:                             })
   82:                     .setPositiveButton(R.string.ok, new DialogInterface.OnClickLi
stener() {
   83:                         @Override
   84:                         public void onClick(DialogInterface dialog, int id) {
   85:                             int width;
   86:                             int length;
   87:                             width = widthNP.getValue();
   88:                             length = lengthNP.getValue();
   89:
   90:                             Furniture furniture = new Furniture((int)(Math.random
()*1000000), RoomViewer.roomNumber, width, length, shape, 0, 0, type);
   91:
   92:                             mListener.onFurnitureDialogPositiveClick(CreateFurnit
ureDialog.this, furniture);
   93:                         }
   94:                     })
   95:                     .setNegativeButton(R.string.cancel, new DialogInterface.OnCli
ckListener() {
   96:                         public void onClick(DialogInterface dialog, int id) {
   97:                             mListener.onFurnitureDialogNegativeClick(CreateFurnit
ureDialog.this);
   98:                             dialog.cancel();
   99:                         }
  100:
  101:                     });
  102:
  103:             builder.setTitle(R.string.create_furniture_dim);
  104:
  105:
  106:             return builder.create();
  107:         }
  108: }
```

```
    1: package com.iadf.FurnitureDatabase.LocalDatabaseConnection;
    2:
    3: import com.iadf.SystemController.DatabaseController.Furniture;
    4: import android.database.Cursor;
    5:
    6: /**
    7:  * interface containing all necessary methods for a database controller
    8:  *
    9:  * @author CSE324 Spring 2014 Team 4
   10:  */
   11: public interface RoomController {
   12:
   13:         /**
   14:           * returns a room with the given roomNumber
   15:           */
   16:     public Cursor openRoom(Object db, int roomNumber);
   17:
   18:     /**
   19:          * returns the furniture object matching the guid of the furniture param
eter
   20:          */
   21:     public Cursor lookupFurniture(Object db, Furniture furniture);
   22:
   23:     /**
   24:          * adds the furniture object into the table
   25:          */
   26:     public void addFurniture(Object db, Furniture furniture);
   27:
   28:     /**
   29:          * adds a room to the database
   30:          */
   31:     public void addRoom(Object db, int width, int height);
   32:
   33:     /**
   34:          * returns a list of furniture that are inside of a given room
   35:          */
   36:     public Cursor getFurnitureList(Object db, int roomNumber);
   37:
   38:     /**
   39:          * removes a room from the database
   40:          */
   41:     public void deleteRoom(Object db, int roomNumber);
   42:
   43:     /**
   44:          * removes a furniture object from the database
   45:          */
   46:     public void deleteFurniture(Object db, Furniture furniture);
   47:
   48:     /**
   49:          * changes the values for a given room
   50:          */
   51:     public void modifyRoom(Object db, int roomNumber, int width, int length);
   52:
   53:     /**
   54:          * updates the values of a given furniture object in the database
   55:          */
   56:     public void modifyFurniture(Object db, Furniture furniture);
   57:
   58:     /**
   59:          * returns a cursor over all rooms
   60:          */
   61:         public Cursor viewRooms(Object db);
   62:
```

```
   63: }
```

```
    1: package com.iadf.FurnitureDatabase.LocalDatabaseConnection;
    2:
    3: import android.database.Cursor;
    4:
    5: import java.io.InputStream;
    6:
    7: import javax.xml.parsers.DocumentBuilder;
    8: import javax.xml.parsers.DocumentBuilderFactory;
    9:
   10: import org.w3c.dom.Document;
   11: import org.w3c.dom.NodeList;
   12:
   13: import com.iadf.SystemController.DatabaseController.Furniture;
   14: import com.iadf.roommapapp.R;
   15:
   16: import android.content.Context;
   17: import android.database.sqlite.SQLiteDatabase;
   18: import android.database.sqlite.SQLiteOpenHelper;
   19: import android.widget.Toast;
   20:
   21: /**
   22:  * performs all database actions
   23:  *
   24:  * @author CSE324 Spring 2014 Team 4
   25:  */
   26: public class SQLRoomController extends SQLiteOpenHelper implements RoomControlle
r{
   27:
   28:         public static final String DATABASE_NAME = "RoomMapApp";
   29:
   30:     protected Context context;
   31:
   32:     public SQLRoomController(Context context) {
   33:             super(context, DATABASE_NAME, null, 1);
   34:             this.context = context;
   35:     }
   36:
   37:     /**
   38:      * creates the database if it does not already exist
   39:      */
   40:         @Override
   41:     public void onCreate(SQLiteDatabase db) {
   42:              String s;
   43:              try {
   44:                      InputStream in = context.getResources().openRawResource(R.r
aw.sql);
   45:                      DocumentBuilder builder = DocumentBuilderFactory.newInstanc
e().newDocumentBuilder();
   46:                      Document doc = builder.parse(in, null);
   47:                      NodeList statements = doc.getElementsByTagName("statement")
;
   48:                      for (int i=0; i<statements.getLength(); i++) {
   49:                              s = statements.item(i).getChildNodes().item(0).getN
odeValue();
   50:                              db.execSQL(s);
   51:                      }
   52:              } catch (Throwable t) {
   53:                      Toast.makeText(context, t.toString(), Toast.LENGTH_LONG).sh
ow();
   54:              }
   55:      }
   56:
   57:         /**
```

```
   58:          * returns a cursor over all rooms
   59:          */
   60:         @Override
   61:         public Cursor viewRooms(Object db) {
   62:                SQLiteDatabase database = (SQLiteDatabase) db;
   63:        return database.rawQuery("SELECT * FROM rooms ", null);
   64:      }
   65:
   66:          /**
   67:           * returns a room with the given roomNumber
   68:           */
   69:         @Override
   70:         public Cursor openRoom(Object db, int roomNumber) {
   71:                SQLiteDatabase database = (SQLiteDatabase) db;
   72:                return database.rawQuery("SELECT * FROM rooms WHERE room_number
= " + roomNumber , null);
   73:         }
   74:
   75:         /**
   76:          * returns the furniture object matching the guid of the furniture param
eter
   77:          */
   78:         @Override
   79:         public Cursor lookupFurniture(Object db, Furniture furniture) {
   80:                SQLiteDatabase database = (SQLiteDatabase) db;
   81:                return database.rawQuery("SELECT * FROM furniture WHERE _id = "
+ furniture.getGUID() , null);
   82:         }
   83:
   84:         /**
   85:          * adds the furniture object into the table
   86:          */
   87:         @Override
   88:         public void addFurniture(Object db, Furniture furniture) {
   89:                SQLiteDatabase database = (SQLiteDatabase) db;
   90:                database.execSQL("INSERT INTO furniture (_id, room_number, cente
r_x, center_y, width, length, shape, type) VALUES (" + furniture + ")");
   91:
   92:         }
   93:
   94:         /**
   95:          * adds a room to the database
   96:          */
   97:         @Override
   98:         public void addRoom(Object db, int width, int height) {
   99:                SQLiteDatabase database = (SQLiteDatabase) db;
  100:                database.execSQL("INSERT INTO rooms (width, length) VALUES (" +
width + ", " + height + ")");
  101:
  102:         }
  103:
  104:         /**
  105:          * returns a list of furniture that are inside of a given room
  106:          */
  107:         @Override
  108:         public Cursor getFurnitureList(Object db, int roomNumber) {
  109:                SQLiteDatabase database = (SQLiteDatabase) db;
  110:                return database.rawQuery("SELECT * FROM furniture WHERE room_num
ber = " + roomNumber, null);
  111:         }
  112:
  113:         /**
  114:          * removes a room from the database
```

```
  115:          */
  116:         @Override
  117:         public void deleteRoom(Object db, int roomNumber) {
  118:                 SQLiteDatabase database = (SQLiteDatabase) db;
  119:                 database.rawQuery("DELETE FROM rooms WHERE room_number = " + roo
mNumber, null);
  120:
  121:         }
  122:
  123:         /**
  124:          * removes a furniture object from the database
  125:          */
  126:         @Override
  127:         public void deleteFurniture(Object db, Furniture furniture) {
  128:                 SQLiteDatabase database = (SQLiteDatabase) db;
  129:                 database.rawQuery("DELETE FROM furniture WHERE _id = " + furnitu
re.getGUID(), null);
  130:
  131:         }
  132:
  133:         /**
  134:          * changes the values for a given room
  135:          */
  136:         @Override
  137:         public void modifyRoom(Object db, int roomNumber, int width, int length)
 {
  138:                 SQLiteDatabase database = (SQLiteDatabase) db;
  139:                 database.rawQuery("UPDATE rooms SET width = " + width + ", lengt
h = " + length + " WHERE room_number = " , null);
  140:
  141:         }
  142:
  143:         /**
  144:          * updates the values of a given furniture object in the database
  145:          */
  146:         @Override
  147:         public void modifyFurniture(Object db, Furniture furniture) {
  148:                 SQLiteDatabase database = (SQLiteDatabase) db;
  149:                 database.rawQuery("UPDATE furniture SET " + furniture.dbUpdateSt
ring() + "WHERE _id = " + furniture.getGUID(), null);
  150:
  151:         }
  152:
  153:         /**
  154:          * drops the table if it needs to be changed
  155:          */
  156:         @Override
  157:     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
  158:             db.execSQL("DROP TABLE IF EXISTS furniture");
  159:             db.execSQL("DROP TABLE IF EXISTS rooms");
  160:             onCreate(db);
  161:     }
  162:
  163: }
```