```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
 3      package="edu.seaaddicts.brockbutler"
 4      android:versionCode="1"
 5      android:versionName="1.0" >
 6
 7      <uses-sdk
 8          android:minSdkVersion="14"
 9          android:targetSdkVersion="16" />
10
11      <uses-permission android:name="android.permission.INTERNET" />
12      <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
13      <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
14      <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
15      <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
16
17      <application
18          android:allowBackup="true"
19          android:icon="@drawable/ic_launcher"
20          android:label="@string/app_name" >
21          <activity
22              android:name="edu.seaaddicts.brockbutler.LoginActivity"
23              android:label="@string/app_name"
24              android:noHistory="true" >
25          </activity>
26          <activity
27              android:name="edu.seaaddicts.brockbutler.MainActivity"
28              android:icon="@drawable/ic_launcher"
29              android:label="@string/title_activity_main"
30              android:screenOrientation="portrait" >
31              <intent-filter>
32                  <action android:name="android.intent.action.MAIN" />
33
34                  <category android:name="android.intent.category.LAUNCHER" />
35              </intent-filter>
36          </activity>
37          <activity
38              android:name="edu.seaaddicts.brockbutler.maps.MapsActivity"
39              android:icon="@drawable/ic_launcher"
40              android:label="@string/title_activity_map"
41              android:screenOrientation="landscape"
42              android:theme="@android:style/Theme.Holo.Light" >
43          </activity>
44          <activity
45              android:name="edu.seaaddicts.brockbutler.help.HelpActivity"
46              android:icon="@drawable/ic_launcher"
47              android:label="@string/title_activity_help"
48              android:screenOrientation="portrait" >
49          </activity>
50          <activity
51              android:name="edu.seaaddicts.brockbutler.scheduler.SchedulerActivity"
52              android:icon="@drawable/ic_launcher"
53              android:label="@string/title_activity_scheduler"
54              android:screenOrientation="portrait" >
55          </activity>
56          <activity
57              android:name="edu.seaaddicts.brockbutler.scheduler.SchedulerActivityBAK"
58              android:icon="@drawable/ic_launcher"
59              android:label="@string/title_activity_scheduler"
60              android:screenOrientation="portrait" >
61          </activity>
62          <activity
63              android:name="edu.seaaddicts.brockbutler.contacts.ContactsActivity"
64              android:icon="@drawable/ic_launcher"
65              android:label="@string/title_activity_contacts"
66              android:screenOrientation="portrait" >
67          </activity>
68          <activity
69              android:name=
                "edu.seaaddicts.brockbutler.coursemanager.CourseManagerActivity"
70              android:icon="@drawable/ic_launcher"
```

```
71              android:label="@string/activity_coursemanager"
72              android:screenOrientation="portrait" >
73          </activity>
74          <activity
75              android:name="edu.seaaddicts.brockbutler.tour.TourActivity"
76              android:icon="@drawable/ic_launcher"
77              android:label="@string/title_activity_tour"
78              android:screenOrientation="landscape" >
79          </activity>
80          <activity
81              android:name="edu.seaaddicts.brockbutler.scheduler.AddTaskActivity"
82              android:icon="@drawable/ic_launcher"
83              android:label="@string/title_activity_add_task"
84              android:screenOrientation="portrait" >
85          </activity>
86          <activity
87              android:name="edu.seaaddicts.brockbutler.scheduler.ModifyTaskActivity"
88              android:icon="@drawable/ic_launcher"
89              android:label="@string/title_modify_task"
90              android:screenOrientation="portrait" >
91          </activity>
92          <activity
93              android:name="edu.seaaddicts.brockbutler.contacts.AddContactActivity"
94              android:icon="@drawable/ic_launcher"
95              android:label="@string/activity_add_contact"
96              android:screenOrientation="portrait" >
97          </activity>
98          <activity
99              android:name="edu.seaaddicts.brockbutler.coursemanager.AddCourseActivity"
100             android:icon="@drawable/ic_launcher"
101             android:label="@string/title_add_course"
102             android:screenOrientation="portrait" >
103         </activity>
104         <activity
105             android:name=
                "edu.seaaddicts.brockbutler.coursemanager.ModifyCourseActivity"
106             android:icon="@drawable/ic_launcher"
107             android:label="@string/title_modify_course"
108             android:screenOrientation="portrait" >
109         </activity>
110     </application>
111
112 </manifest>
```

```java
package edu.seaaddicts.brockbutler.contacts;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import edu.seaaddicts.brockbutler.R;

public class AddContactActivity extends Activity {

  private Button mSaveButton;
  private Button mCancelButton;
  //private Contact mContact;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add_contact);
    init();
  }

  /*
   * Initialize all views and sets Button OnClickListeners.
   */
  private void init() {
    mSaveButton = (Button) findViewById(R.id.add_contact_save_button);
    mSaveButton.setOnClickListener(new OnClickListener() {

      public void onClick(View v) {
        // Grisdale add contact
        onBackPressed();
      }
    });
    mCancelButton = (Button) findViewById(R.id.add_contact_cancel_button);
    mCancelButton.setOnClickListener(new OnClickListener() {

      public void onClick(View v) {
        onBackPressed();
      }
    });
  }
}

```

```
 1  /**
 2   * Contacts.java
 3   * Brock Butler
 4   * A wrapper class for Contact information
 5   * Created by James Grisdale on 2013-02-24
 6   * Copyright (c) 2013 Sea Addicts. All rights reserved.
 7  **/
 8
 9  package edu.seaaddicts.brockbutler.contacts;
10
11
12  public class Contact {
13    public String mSubj;
14    public String mCode;
15    public int mId;
16    public String mFirstName;
17    public String mLastName;
18    public String mEmail;
19    public Contact(){
20    }
21  }
22
```

```java
package edu.seaaddicts.brockbutler.contacts;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import edu.seaaddicts.brockbutler.R;

public class ContactsActivity extends Activity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_contacts);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.activity_contacts, menu);
    return true;
  }

}

```

```
 1  package edu.seaaddicts.brockbutler.contacts;
 2
 3  import java.util.List;
 4
 5  import android.content.Context;
 6  import android.view.LayoutInflater;
 7  import android.view.View;
 8  import android.view.View.OnClickListener;
 9  import android.view.ViewGroup;
10  import android.widget.BaseAdapter;
11  import android.widget.Button;
12  import android.widget.TextView;
13  import edu.seaaddicts.brockbutler.R;
14
15  public class ContactsAdapter extends BaseAdapter implements OnClickListener {
16      private Context context;
17
18      private List<Contact> mContactsList;
19
20      public ContactsAdapter(Context context, List<Contact> listPhonebook) {
21          this.context = context;
22          this.mContactsList = listPhonebook;
23      }
24
25      public int getCount() {
26          return mContactsList.size();
27      }
28
29      public Object getItem(int position) {
30          return mContactsList.get(position);
31      }
32
33      public long getItemId(int position) {
34          return position;
35      }
36
37      public View getView(int position, View convertView, ViewGroup viewGroup) {
38          Contact c = mContactsList.get(position);
39          if (convertView == null) {
40              LayoutInflater inflater = (LayoutInflater) context
41                      .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
42              convertView = inflater.inflate(R.layout.row_contact_full, null);
43          }
44          TextView tvContact = (TextView) convertView.findViewById(R.id.tv_contact_name
            );
45          tvContact.setText(c.mFirstName + " " + c.mLastName);
46
47          TextView tvEmail = (TextView) convertView.findViewById(R.id.tv_email);
48          tvEmail.setText(c.mEmail);
49
50          TextView tvMail = (TextView) convertView.findViewById(R.id.lv_courses);
51          tvMail.setText(c.mEmail);
52
53          // Set the onClick Listener on this button
54          Button btnRemove = (Button) convertView.findViewById(R.id.btnRemove);
55          btnRemove.setFocusableInTouchMode(false);
56          btnRemove.setFocusable(false);
57          btnRemove.setOnClickListener(this);
58
59          // Set the entry, so that you can capture which item was clicked and
60          // then remove it. As an alternative, you can use the id/position of the
            item to capture
61          // the item that was clicked.
62          btnRemove.setTag(c);
63          // btnRemove.setId(position);
64
65
66          return convertView;
67      }
68
69      public void onClick(View view) {
```

```
70            Contact c = (Contact) view.getTag();
71            mContactsList.remove(c);
72            // listPhonebook.remove(view.getId());
73            notifyDataSetChanged();
74
75        }
76
77        /*private void showDialog(Contact c) {
78            // Create and show your dialog
79            // Depending on the Dialogs button clicks delete it or do nothing
80        }*/
81
82    }
83
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10sp"
    android:orientation="vertical"
    tools:context=".AddTaskActivity" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/activity_add_contact_fname" />

        <EditText
            android:id="@+id/et_contact_fname"
            android:layout_width="180sp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="25sp"
            android:hint="@string/activity_add_contact_hint_fname"
            android:inputType="text" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/activity_add_contact_lname" />

        <EditText
            android:layout_width="180sp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="25sp"
            android:hint="@string/activity_add_contact_hint_lname"
            android:inputType="text" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/activity_add_contact_email" />

        <EditText
            android:layout_width="180sp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="25sp"
            android:hint="@string/activity_add_contact_hint_email"
            android:inputType="text" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10sp"
        android:gravity="center_horizontal"
        android:orientation="vertical" >

        <Button
```

```
72                android:id="@+id/add_contact_add_course_button"
73                android:layout_width="150sp"
74                android:layout_height="wrap_content"
75                android:layout_gravity="center_horizontal"
76                android:text="@string/activity_add_contact_courses" />
77
78            <ListView
79                android:id="@+id/add_contact_course_listview"
80                android:layout_width="wrap_content"
81                android:layout_height="100sp"
82                android:layout_marginTop="10sp"
83                android:text="@string/activity_add_task_contacts"
84                android:textSize="20sp" />
85        </LinearLayout>
86
87        <LinearLayout
88            android:layout_width="fill_parent"
89            android:layout_height="wrap_content"
90            android:layout_margin="10sp"
91            android:gravity="center_horizontal"
92            android:orientation="horizontal" >
93
94            <Button
95                android:id="@+id/add_contact_save_button"
96                android:layout_width="100sp"
97                android:layout_height="wrap_content"
98                android:text="@string/button_save" />
99
100           <Button
101               android:id="@+id/add_contact_cancel_button"
102               android:layout_width="100sp"
103               android:layout_height="wrap_content"
104               android:layout_marginLeft="20sp"
105               android:text="@string/button_cancel" />
106       </LinearLayout>
107
108   </LinearLayout>
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ContactsActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/title_activity_contacts" />

</RelativeLayout>
```

```
 1  package edu.seaaddicts.brockbutler.coursemanager;
 2
 3  import java.util.ArrayList;
 4
 5  import android.app.Activity;
 6  import android.os.Bundle;
 7  import android.util.Log;
 8  import android.util.SparseBooleanArray;
 9  import android.view.View;
10  import android.view.View.OnClickListener;
11  import android.widget.AdapterView;
12  import android.widget.AdapterView.OnItemSelectedListener;
13  import android.widget.ArrayAdapter;
14  import android.widget.Button;
15  import android.widget.LinearLayout;
16  import android.widget.ListView;
17  import android.widget.Spinner;
18  import edu.seaaddicts.brockbutler.R;
19
20  public class AddCourseActivity extends Activity {
21
22    private static final String TAG = "AddCourseActivity";
23
24    //private static final int DATE_DIALOG_ID = 0;
25    private static final int VISIBLE = 0;
26    //private static final int INVISIBLE = 4;
27    private static final int GONE = 8;
28
29    ArrayList<String> mLecs;
30    ArrayList<String> mLabs;
31    ArrayList<String> mTuts;
32    ArrayList<String> mSems;
33
34    ArrayList<Offering> mLecsOfferings;
35    ArrayList<Offering> mLabsOfferings;
36    ArrayList<Offering> mTutsOfferings;
37    ArrayList<Offering> mSemsOfferings;
38
39    private String mSubject;
40    private String mCode;
41
42    private Course mCourse;
43
44    private Button mSaveButton;
45    private Button mCancelButton;
46
47    private CourseHandler mCourseHandle;
48    ArrayList<Offering> mOfferings;
49
50    private Spinner mSubjectSpinner;
51    private Spinner mCodesSpinner;
52
53    private ListView mLecsListView;
54    private ListView mSemsListView;
55    private ListView mTutsListView;
56    private ListView mLabsListView;
57
58    @Override
59    protected void onCreate(Bundle savedInstanceState) {
60      super.onCreate(savedInstanceState);
61      setContentView(R.layout.activity_add_course);
62      mCourseHandle = new CourseHandler(this.getApplicationContext());
63      init();
64    }
65
66    /*
67     * Initialize all views and sets Button OnClickListeners.
68     */
69    private void init() {
70
71      mSaveButton = (Button) findViewById(R.id.add_course_save_button);
```

```
72        mCancelButton = (Button) findViewById(R.id.add_course_cancel_button);
73
74        mSubjectSpinner = (Spinner) findViewById(R.id.add_course_subjects_spinner);
75        try {
76          mSubjectSpinner.setAdapter(new ArrayAdapter<String>(this,
77              android.R.layout.simple_spinner_dropdown_item,
78              mCourseHandle.getSubjects()));
79        } catch (Exception e) {
80          e.printStackTrace();
81        }
82
83        mCodesSpinner = (Spinner) findViewById(R.id.add_course_codes_spinner);
84        mSubjectSpinner.setOnItemSelectedListener(new OnItemSelectedListener() {
85          public void onItemSelected(AdapterView<?> arg0, View arg1,
86              int arg2, long arg3) {
87            try {
88              mSubject = arg0.getItemAtPosition(arg2).toString();
89              mCodesSpinner.setAdapter(new ArrayAdapter<String>(
90                  AddCourseActivity.this,
91                  android.R.layout.simple_spinner_dropdown_item,
92                  mCourseHandle.getCodes(mSubject)));
93            } catch (Exception e) {
94              e.printStackTrace();
95            }
96          }
97
98          public void onNothingSelected(AdapterView<?> arg0) {
99            // Do nothing.
100         }
101       });
102       mCodesSpinner.setOnItemSelectedListener(new OnItemSelectedListener() {
103
104         public void onItemSelected(AdapterView<?> arg0, View arg1,
105             int arg2, long arg3) {
106           mCode = arg0.getItemAtPosition(arg2).toString();
107           mCourse = mCourseHandle.getCourseOfferings(mSubject, mCode);
108           mOfferings = mCourse.mOfferings;
109
110           String s;
111           mLecs = new ArrayList<String>();
112           mLabs = new ArrayList<String>();
113           mTuts = new ArrayList<String>();
114           mSems = new ArrayList<String>();
115
116           mLecsOfferings = new ArrayList<Offering>();
117           mLabsOfferings = new ArrayList<Offering>();
118           mTutsOfferings = new ArrayList<Offering>();
119           mSemsOfferings = new ArrayList<Offering>();
120
121           for (int i = 0; i < mOfferings.size(); i++) {
122             s = mOfferings.get(i).mType;
123
124             // Some offerings don't have any of what we are looking for,
125             // so check length to make sure.
126             if (s.length() > 2) {
127               String ss = s.substring(0, 3).trim();
128               if (ss.equalsIgnoreCase("lec")) {
129                 mLecs.add(s + ", SEC " + mOfferings.get(i).mSection);
130                 mLecsOfferings.add(mOfferings.get(i));
131               } else if (ss.equalsIgnoreCase("lab")) {
132                 mLabs.add(s + ", SEC " + mOfferings.get(i).mSection);
133                 mLabsOfferings.add(mOfferings.get(i));
134               } else if (ss.equalsIgnoreCase("tut")) {
135                 mTuts.add(s + ", SEC " + mOfferings.get(i).mSection);
136                 mTutsOfferings.add(mOfferings.get(i));
137               } else if (ss.equalsIgnoreCase("sem")) {
138                 mSems.add(s + ", SEC " + mOfferings.get(i).mSection);
139                 mSemsOfferings.add(mOfferings.get(i));
140               }
141             }
142           }
```

```
143
144           // Check if offerings available, and add ListView to display if
145           // so.
146           LinearLayout lec_lay = (LinearLayout) findViewById(R.id.layout_add_lecs);
147           LinearLayout lab_lay = (LinearLayout) findViewById(R.id.layout_add_labs);
148           LinearLayout tut_lay = (LinearLayout) findViewById(R.id.layout_add_tuts);
149           LinearLayout sem_lay = (LinearLayout) findViewById(R.id.layout_add_sems);
150
151           if (mLecs.size() > 0) {
152             lec_lay.setVisibility(VISIBLE);
153             mLecsListView = (ListView) findViewById(R.id.add_course_add_lecs);
154             mLecsListView.setAdapter(new ArrayAdapter<String>(
155                 getApplicationContext(),
156                 android.R.layout.simple_list_item_multiple_choice,
157                 mLecs));
158           } else {
159             lec_lay.setVisibility(GONE);
160           }
161           if (mLabs.size() > 0) {
162             lab_lay.setVisibility(VISIBLE);
163             mLabsListView = (ListView) findViewById(R.id.add_course_add_labs);
164             mLabsListView.setAdapter(new ArrayAdapter<String>(
165                 getApplicationContext(),
166                 android.R.layout.simple_list_item_multiple_choice,
167                 mLabs));
168           } else {
169             lab_lay.setVisibility(GONE);
170           }
171           if (mTuts.size() > 0) {
172             tut_lay.setVisibility(VISIBLE);
173             mTutsListView = (ListView) findViewById(R.id.add_course_add_tuts);
174             mTutsListView.setAdapter(new ArrayAdapter<String>(
175                 getApplicationContext(),
176                 android.R.layout.simple_list_item_multiple_choice,
177                 mTuts));
178           } else {
179             tut_lay.setVisibility(GONE);
180           }
181           if (mSemsOfferings.size() > 0) {
182             sem_lay.setVisibility(VISIBLE);
183             mSemsListView = (ListView) findViewById(R.id.add_course_add_sems);
184             mSemsListView.setAdapter(new ArrayAdapter<String>(
185                 getApplicationContext(),
186                 android.R.layout.simple_list_item_multiple_choice,
187                 mSems));
188           } else {
189             sem_lay.setVisibility(GONE);
190           }
191         }
192
193         public void onNothingSelected(AdapterView<?> arg0) {
194         }
195       });
196
197       mSaveButton.setOnClickListener(new OnClickListener() {
198         public void onClick(View v) {
199           Course c = new Course();
200           c.mSubject = mSubject;
201           c.mCode = mCode;
202           c.mInstructor = mCourseHandle.getCourseOfferings(mSubject,
203               mCode).mInstructor;
204           try {
205             c.mInstructor_email = ""+c.mInstructor.charAt(c.mInstructor.length()-1);
206             c.mInstructor_email += c.mInstructor.substring(0,c.mInstructor.length()-2);
207             c.mInstructor_email += "@brocku.ca";
208           } catch (Exception e){}
209
210           SparseBooleanArray sba1, sba2, sba3, sba4;
211           if (mLecsListView != null) {
212             sba1 = mLecsListView.getCheckedItemPositions();
213             for (int i = 0; i < mLecsListView.getCount(); i++) {
```

```
214                 if (sba1.get(i) == true) {
215                   c.mOfferings.add(mLecsOfferings.get(i));
216                   Log.d(TAG, "Added: " + mLecsOfferings.get(i).mSubj
217                       + " " + mOfferings.get(i).mCode + ", Type "
218                       + mOfferings.get(i).mType + ", Section "
219                       + mOfferings.get(i).mSection
220                       + " to Offerings");
221                 }
222               }
223             }
224
225             if (mLabsListView != null) {
226               sba2 = mLabsListView.getCheckedItemPositions();
227
228               for (int i = 0; i < mLabsListView.getCount(); i++) {
229                 if (sba2.get(i) == true) {
230                   c.mOfferings.add(mLabsOfferings.get(i));
231                   Log.d(TAG, "Added: " + mLabsOfferings.get(i).mSubj
232                       + " " + mOfferings.get(i).mCode + ", Type "
233                       + mOfferings.get(i).mType + ", Section "
234                       + mOfferings.get(i).mSection
235                       + " to Offerings");
236                 }
237               }
238             }
239
240             if (mTutsListView != null) {
241               sba3 = mTutsListView.getCheckedItemPositions();
242
243               for (int i = 0; i < mTutsListView.getCount(); i++) {
244                 if (sba3.get(i) == true) {
245                   c.mOfferings.add(mTutsOfferings.get(i));
246                   Log.d(TAG, "Added: " + mTutsOfferings.get(i).mSubj
247                       + " " + mOfferings.get(i).mCode + ", Type "
248                       + mOfferings.get(i).mType + ", Section "
249                       + mOfferings.get(i).mSection
250                       + " to Offerings");
251                 }
252               }
253             }
254
255             if (mSemsListView != null) {
256               sba4 = mSemsListView.getCheckedItemPositions();
257
258               for (int i = 0; i < mSemsListView.getCount(); i++) {
259                 if (sba4.get(i) == true) {
260                   c.mOfferings.add(mSemsOfferings.get(i));
261                   Log.d(TAG, "Added: " + mSemsOfferings.get(i).mSubj
262                       + " " + mOfferings.get(i).mCode + ", Type "
263                       + mOfferings.get(i).mType + ", Section "
264                       + mOfferings.get(i).mSection
265                       + " to Offerings");
266                 }
267               }
268             }
269
270             if (c.mSubject != null) {
271               try {
272                 mCourseHandle.addCourse(c);
273                 Log.d("# ADDED OFFERINGS:", "" + c.mOfferings.size());
274                 for (int i = 0; i < c.mOfferings.size(); i++) {
275                   Log.d(TAG, "Added: " + c.mOfferings.get(i).mSubj
276                       + " " + mOfferings.get(i).mCode + ", Type "
277                       + mOfferings.get(i).mType + ", Section "
278                       + mOfferings.get(i).mSection
279                       + " to Offerings");
280                 }
281                 onBackPressed();
282               } catch (Exception e) {
283                 e.printStackTrace();
284               }
```

```
285           }
286         }
287       });
288
289       mCancelButton.setOnClickListener(new OnClickListener() {
290         public void onClick(View v) {
291           onBackPressed();
292         }
293       });
294     }
295   }
296
```

```
1   /**
2    * Brocku.java
3    * Brock Butler
4    * Connects to the Brock University registrars' website to obtain course
5    * information from the current timetable
6    * Created by James Grisdale on 2013-02-24
7    * * Copyright (c) 2013 Sea Addicts. All rights reserved.
8   **/
9
10  package edu.seaaddicts.brockbutler.coursemanager;
11
12  import java.io.BufferedReader;
13  import java.io.InputStreamReader;
14  import java.net.URI;
15  import java.util.ArrayList;
16  import org.apache.http.client.HttpClient;
17  import org.apache.http.client.methods.HttpGet;
18  import org.apache.http.impl.client.*;
19  import org.apache.http.HttpResponse;
20  import android.os.AsyncTask;
21  //Using AsyncTask to do operations off the main thread
22  public class Brocku extends AsyncTask<Void, Void, ArrayList<MasterCourse>> {
23
24    /* doInBackground - connects to Brock University's registrar's office website to
      gather
25     * information on courses being offered, then returns an arraylist of MasterCourse
26     * objects which hold the data for all offerings at Brock.
27     */
28    protected ArrayList<MasterCourse> doInBackground(Void... Void) {
29      String codes[] = new String[74];
30      BufferedReader in = null;
31      String info = new String();
32      String substring = new String();
33      MasterCourse course;
34      boolean done;
35      ArrayList<MasterCourse> courseList = new ArrayList<MasterCourse>();
36      courseList.ensureCapacity(8000);
37      //test = "working";
38      try{
39        HttpClient client = new DefaultHttpClient();
40        HttpGet request = new HttpGet();
41        URI BTimeTable = new URI(
          "http://www.brocku.ca/registrar/guides/returning/timetable/a_get_subj.php?subj=C
          OSC");
42        request.setURI(BTimeTable);
43        HttpResponse response = client.execute(request);
44        in = new BufferedReader(new InputStreamReader(response.getEntity().getContent
          ()));
45        for (int i=0; i<3; i++) in.readLine();
46        for (int i=0; i<74; i++){
47          info = in.readLine();
48          codes[i] = info.substring(19,23);
49        }//retrieving all subjects
50        in.close();
51        for (int h=0; h<codes.length ; h++){
52          done = false;
53          BTimeTable = new URI(
            "http://www.brocku.ca/registrar/guides/returning/timetable/a_get_subj.php?subj
            ="+codes[h]);
54          request.setURI(BTimeTable);
55          response = client.execute(request);
56          in = new BufferedReader(new InputStreamReader(response.getEntity().getContent
            ()));
57          for (int i=0; i<98; i++) {in.readLine();}
58          info = in.readLine();
59          //for each subjects get all course offering information
60          if (info.length()<50){
61            while(!done){
62              course = new MasterCourse();
63              substring = info.substring(24, info.length() - 5);
64              course.id = substring;
```

```
65              for (int i=0; i<2; i++) {in.readLine();}
66              info = in.readLine();
67              substring = info.substring(123, 127);
68              course.subj = substring;
69              substring = info.substring(128, 132);
70              course.code = substring;
71              in.readLine();
72              info = in.readLine();
73              substring = info.substring(26,info.length() - 7);
74              course.desc = substring;
75              for (int i=0; i<4; i++) {in.readLine();}
76              info = in.readLine();
77              substring = info.substring(24,26);
78              course.dur = substring;
79              info = in.readLine();
80              substring = info.substring(24,info.length() - 5);
81              course.type = substring;
82              info = in.readLine();
83              substring = info.substring(24,info.length() - 5);
84              course.sec = substring;
85              info = in.readLine();
86              substring = info.substring(24,30);
87              course.days = substring;
88              info = in.readLine();
89              substring = info.substring(24,info.length() - 5);
90              course.time = substring;
91              info = in.readLine();
92              substring = info.substring(24,info.length() - 5);
93              if (info.substring(24,26).equals("<a"))
94                substring = info.substring(94, info.length()-9);
95              course.location = substring;
96              info = in.readLine();
97              if (info.length()>16)
98                substring = info.substring(9,info.length() - 5);
99              else substring = " ";
100             course.instructor = substring;
101             in.readLine();
102             in.readLine();
103             info = in.readLine();
104             courseList.add(course);
105             if (info.length() <20){
106               info = in.readLine();
107               done = true;
108             }
109           }
110         }
111         in.close();
112       }
113     }
114     //if there's an error return the error information in a course object
115     catch (Exception e){
116     info = e.toString();
117     course = new MasterCourse();
118     course.id = info;
119     courseList.add(course);
120     }
121     return courseList;
122   }
123
124   //not used
125   protected void onPostExecute(MasterCourse course){
126     posttest(course);
127   }
128
129   //not used
130   public MasterCourse posttest(MasterCourse course){
131     return course;
132   }
133 }
134
```

```java
/**
 * Course.java
 * Brock Butler
 * A wrapper class for Course information
 * Created by James Grisdale on 2013-02-24
 * Copyright (c) 2013 Sea Addicts. All rights reserved.
 **/

package edu.seaaddicts.brockbutler.coursemanager;

import java.util.ArrayList;

import edu.seaaddicts.brockbutler.contacts.Contact;
import edu.seaaddicts.brockbutler.scheduler.Task;

public class Course {
  public int mId; //course ID
  public String mSubject;  //subject name
  public String mCode;  //course code
  public String mDesc;  //course description
  public String mInstructor;  //instructor name
  public String mInstructor_email;  //instructor's email
  public ArrayList<Offering> mOfferings;  //list of offerings for this course
  public ArrayList<Task> mTasks; //list of tasks associated with this course
  public ArrayList<Contact> mContacts;  //contacts for this course

  //Constructor - initializes the arraylists for offerings, tasks and contacts
  public Course() {
    mOfferings = new ArrayList<Offering>();
    mTasks = new ArrayList<Task>();
    mContacts = new ArrayList<Contact>();
  }
}
```

```java
/**
 * CourseHandler.java
 * Brock Butler
 * A class to allow easy access to database functions
 * Created by James Grisdale on 2013-02-24
 * Copyright (c) 2013 Sea Addicts. All rights reserved.
 **/

package edu.seaaddicts.brockbutler.coursemanager;
import java.util.ArrayList;

import android.content.Context;
import android.database.Cursor;
import edu.seaaddicts.brockbutler.contacts.Contact;
import edu.seaaddicts.brockbutler.scheduler.Task;

public class CourseHandler {
  // Context context;
  CurrentCoursesHandler CH;
  CourseListHandler courseList;

  /* Constructor - opens and closes database to ensure the database exists
   *    and if not, it copies over the installed database of course offerings
   * @param context - application context
   */
  public CourseHandler(Context context) {
    // this.context = context;
    CH = new CurrentCoursesHandler(context);
    courseList = new CourseListHandler(context);
    try{
    courseList.createDataBase();
    courseList.openDataBase();
    courseList.close();
    }
    catch(Exception e){};
    //SQLiteDatabase db = courseList.getWritableDatabase();
    //db.close();
  }

  /* updateAll - updates all course information from the Brock University registrar's
   * office website
   */
  public void updateAll(){
    courseList.addCourse();
  }

  /* Depreciated - getAllCourses - grabs course data from the registrar's timetable
  and
   * inserts data into the masterlist table.
   * Depreciated due to information no longer being available on website
   */
  public void getAllCourses() {
    courseList.addCourse();
  }

  /* getCourse - gets all information for a given course subject and code
   * @param subj - subject name to get
   * @param code - course code to get
  */
  public Course getCourse(final String subj, final String code) {
    return CH.getCourse(subj, code);
  }

  /* updateCourse - updates all the information for a given course
   * @param course - course information to update
   */
  public void updateCourse(Course course) {
    CH.addCourse(course);
  }

  /* getSubjects - gets a list of subjects available from the master list
```

```
71       * returns an arraylist of subject offerings
72       */
73      public ArrayList<String> getSubjects() throws Exception {
74        return courseList.getSubjects();
75      }
76
77      /* getCodes - gets a list of codes for a given subject from the master list
78       * returns an arraylist of subject codes
79       * @param subj - return codes for this subject
80       */
81      public ArrayList<String> getCodes(String subj) {
82        return courseList.getCodes(subj);
83      }
84
85      /* getCourseOfferings - returns all offerings offered for a given course
86       * Converts the offerings from MasterCourse format to Course format
87       * @param subj - subject name
88       * @param code - course code
89       */
90      public Course getCourseOfferings(String subj, String code) {
91        //get list of offerings as a list of MasterCourse objects
92        ArrayList<MasterCourse> list = courseList.getCourses(subj, code);
93        Course course = new Course(); //create a new course object
94        ArrayList<OfferingTime> offeringtimes;
95        course.mSubject = list.get(0).subj;
96        course.mCode = list.get(0).code;
97        course.mInstructor = list.get(0).instructor;
98        course.mDesc = list.get(0).desc;
99        Offering offering;
100       int tindex = 0;
101       OfferingTime otime;
102       //add all offerings for a particular course
103       ArrayList<Offering> offerings = new ArrayList<Offering>();
104       for (int i = 0; i < list.size(); i++) {
105         offering = new Offering();
106         offering.mSubj = list.get(i).subj;
107         offering.mCode = list.get(i).code;
108         offering.mType = list.get(i).type;
109         offering.mSection = Integer.parseInt(list.get(i).sec);
110         //add all the offeringtimes associated with all the offerings
111         offeringtimes = new ArrayList<OfferingTime>();
112         for (int j = 0; j < 5; j++) {
113           if (list.get(i).days.charAt(j) != ' ') {
114             otime = new OfferingTime();
115             otime.mDay = list.get(i).days.substring(j, j+1);
116             otime.mLocation = list.get(i).location;
117             for (int h = 0; h < list.get(i).time.length(); h++) {
118               if (list.get(i).time.charAt(h) == '-') {
119                 tindex = h;
120                 break;
121               }
122             }
123             //get the times for each offering
124             otime.mStartTime = list.get(i).time.substring(0, tindex);
125             otime.mEndTime = list.get(i).time.substring(tindex + 1,
126                 list.get(i).time.length());
127             offeringtimes.add(otime);
128           }
129         }
130         offering.mOfferingTimes = offeringtimes;
131         offerings.add(offering);
132
133       }
134       course.mOfferings = offerings;
135
136       return course;//return the course object
137     }
138
139     /* addCourse - adds information for a course into the database
140      * returns a 0 if sucessful, 1 if the add failed
141      * @param course - the course object to be added
```

```
142       */
143      public int addCourse(Course course) throws Exception {
144        try {
145          CH.addCourse(course);
146          return 0;
147        } catch (Exception e) {
148          return 1;
149        }
150      }
151
152      /* removeCourse - deletes all information from the database for a course
153       * returns a 0 on success, returns 1 if failure
154       * @param course - the course information to be deleted
155       */
156      public int removeCourse(Course course) {
157        try {
158          CH.deleteCourse(course);
159          return 0;
160        } catch (Exception e) {
161          return 1;
162        }
163      }
164
165      /*
166       * getRegisteredCourses - returns all information for all courses in the
167       * current courses database
168       */
169      public ArrayList<Course> getRegisteredCourses() {
170        return CH.getRegCourses();
171      }
172
173      /* getOfferings - get all offerings for a certain course
174       * @param subj - course name
175       * @param code - course code
176       */
177      public ArrayList<Offering> getOfferings(String subj, String code) {
178        return CH.getOfferings(subj, code);
179      }
180
181      /* getTasks - gets all tasks from the database
182       */
183      public ArrayList<Task> getTasks() {
184        return CH.getTasks();
185      }
186
187      /* addTask - adds a given task to the task table in the database
188       * returns 0 if sucessful, returns 1 if it fails
189       * @param task - the task information to be added to the database
190       */
191      public int addTask(Task task) {
192        try {
193          CH.addTasks(task);
194          return 0;
195        } catch (Exception e) {
196          return 1;
197        }
198      }
199
200      // addTask - adds the tasks for a given course to the task table in the
201        // database
202      public int addTask(Course course) {
203        try {
204          CH.addTasks(course);
205          return 0;
206        } catch (Exception e) {
207          return 1;
208        }
209      }
210
211      /* removeTask - deletes task information from the database for a given task
212       * returns 0 if sucessful, 1 if failure
```

```
213       * @param task -  the task information to be removed from the database
214       */
215      public int removeTask(Task task) {
216        try {
217          CH.removeTask(task);
218          return 0;
219        } catch (Exception e) {
220          return 1;
221        }
222      }
223
224      /* getBase - returns the total base mark for a particular course
225       * given base information from the course
226       * @param course - the course to calculate the total base for
227       */
228      public float getBase(Course course) {
229        float base = 0;
230        for (int i = 0; i < course.mTasks.size(); i++)
231          base +=course.mTasks.get(i).mWeight;
232         return base;
233      }
234
235      /* getMark - returns the calculated progress mark for a course
236       * given mark information from the course, the mark is calculated and returned
237       * as a float
238       * @param course - the course to calculate the marks for
239       */
240      public float getMark(Course course) {
241        float mark = 0;
242        for (int i = 0; i < course.mTasks.size(); i++){
243          if(course.mTasks.get(i).mBase !=0){
244          mark += (course.mTasks.get(i).mMark / course.mTasks.get(i).mBase)
245              * course.mTasks.get(i).mWeight;}
246          else mark+=0;
247        }
248
249         return mark;
250      }
251
252      /* getSize - returns the number of courses added to the course database
253       */
254      public int getSize() {
255        return courseList.size();
256      }
257
258      /* removeContact - removes contact information from the database
259       * @param contact - the contact information to be removed
260       */
261      public void removeContact(Contact contact){
262        CH.removeContact(contact);
263      }
264
265      /* Query - returns a cursor with results for a custom query
266       * @param query - a string with a sqlite query
267       */
268      public Cursor Query(String query) {
269        return CH.Query(query);
270      }
271    }
272
```

```java
/**
 * CourseListHandler.java
 * Brock Butler
 * Creates a database table for a full list of offerings on the registrar's
 * timetable and allows the table to have inserts or be read
 * Created by James Grisdale on 2013-02-24
 * Copyright (c) 2013 Sea Addicts. All rights reserved.
 **/

package edu.seaaddicts.brockbutler.coursemanager;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;
import android.os.Looper;

public class CourseListHandler extends SQLiteOpenHelper {

  // All Static variables
  // Database Version
  private static final int DATABASE_VERSION = 1;
  private static final String DATABASE_NAME = "Database";
  private static String DB_PATH =
  "/data/data/edu.seaddicts.brockbutler.cousemanager/databases";
    private SQLiteDatabase myDataBase;
  // Database Name


  // Full course list table name
  private static final String TABLE_MCOURSES = "MasterList";
  //current courses table names
  //private static final String TABLE_COURSES = "courses";
  //private static final String TABLE_TASKS = "tasks";
  //private static final String TABLE_OFFERINGS = "offerings";
  //private static final String TABLE_OFFERING_TIMES = "offering_times";
  //private static final String TABLE_CONTACTS = "contacts";
  // All field names used in the database
  private static final String KEY_SUBJ = "subj";
  private static final String KEY_CODE = "code";
  private static final String KEY_DESC = "desc";
  private static final String KEY_INSTRUCTOR = "instructor";
  private static final String KEY_ID = "id";
  private static final String KEY_TYPE = "type";
  private static final String KEY_SEC = "sec";
  //private static final String KEY_DAY = "day";
  //private static final String KEY_TIMES = "time_start";
  //private static final String KEY_TIMEE = "time_end";
  private static final String KEY_LOCATION = "location";
  private static final String KEY_DUR = "dur";
  //private static final String KEY_ASSIGN = "assign";
  //private static final String KEY_NAME = "name";
  //private static final String KEY_MARK = "mark";
  //private static final String KEY_BASE = "base";
  //private static final String KEY_WEIGHT = "weight";
  //private static final String KEY_DUE = "due";
  //private static final String KEY_CREATE_DATE = "create_date";
  //private static final String KEY_CID = "cid";
  //private static final String KEY_FNAME = "fname";
  //private static final String KEY_LNAME = "lname";
  //private static final String KEY_EMAIL = "email";
  //private static final String KEY_PRIORITY = "priority";
```

```
71      //private static final String KEY_INSTREMAIL = "instructor_email";
72      private static final String KEY_DAYS = "days";
73      private static final String KEY_TIME = "time";
74      Context context;//holds the application context
75
76      /* CourseListHandler - constructor.  Takes the application context and initializes
        the
77       * database, creates the database if it does not exist and creates the tables
78       * @param context - the application context
79       */
80      public CourseListHandler(Context context) {
81        super(context, DATABASE_NAME, null, DATABASE_VERSION);//initialize database
82        this.context = context;//get context
83        DB_PATH = this.context.getDatabasePath(DATABASE_NAME).getAbsolutePath();//get
          database path
84      }
85
86      /* onCreate - creates the tables for the database if they do not exist in
87       * the database. This method is depreciated since the database is being added
88       * from a prebuilt database in the assets folder
89       * @param db - reference to the database
90       */
91      @Override
92      public void onCreate(SQLiteDatabase db) {
93        /* All tables are not being built by the app since BrockU is no longer
94         * being used.  The tables in the database are now preloaded from the
95         * database file in the assests folder
96         *
97         * Saved here when new courses are available for the new year of school
98         *
99        String CREATE_COURSES_TABLE = "CREATE TABLE " + TABLE_MCOURSES + "("
100           + KEY_ID + " TEXT," + KEY_SUBJ + " TEXT," + KEY_CODE + " TEXT,"
101           + KEY_DESC + " TEXT," + KEY_TYPE + " TEXT," + KEY_SEC
102           + " TEXT," + KEY_DUR + " TEXT," + KEY_DAYS + " TEXT,"
103           + KEY_TIME + " TEXT," + KEY_LOCATION + " TEXT,"
104           + KEY_INSTRUCTOR + " TEXT" + ")";
105       db.execSQL(CREATE_COURSES_TABLE);
106
107       String CREATE_COURSES = "CREATE TABLE " + TABLE_COURSES + "("
108           + KEY_SUBJ + " TEXT," + KEY_CODE + " TEXT," + KEY_DESC
109           + " TEXT," + KEY_INSTRUCTOR + " TEXT," + KEY_INSTREMAIL
110           + " TEXT," + "PRIMARY KEY(" + KEY_SUBJ + "," + KEY_CODE + ")"
111           + ")";
112
113       String CREATE_TASKS = "CREATE TABLE " + TABLE_TASKS + "(" + KEY_SUBJ
114           + " TEXT," + KEY_CODE + " TEXT," + KEY_ASSIGN + " INTEGER,"
115           + KEY_NAME + " TEXT," + KEY_MARK + " INTEGER," + KEY_BASE
116           + " INTEGER," + KEY_WEIGHT + " REAL," + KEY_DUE + " TEXT,"
117           + KEY_CREATE_DATE + " TEXT," + KEY_PRIORITY + " INTEGER,"
118           + "PRIMARY KEY(" + KEY_SUBJ + "," + KEY_CODE + "," + KEY_ASSIGN
119           + ")"+ ")"; //+ "FOREIGN KEY(" + KEY_SUBJ + "," + KEY_CODE
120           //+ ") REFERENCES " + TABLE_COURSES + "(" + KEY_SUBJ + ","
121           //+ KEY_CODE + ")" + ")";
122
123       String CREATE_OFFERINGS = "CREATE TABLE " + TABLE_OFFERINGS + "("
124           + KEY_ID + " INTEGER," + KEY_SUBJ + " TEXT ," + KEY_CODE
125           + " TEXT ," + KEY_TYPE + " TEXT," + KEY_SEC + " INTEGER,"
126           + "PRIMARY KEY(" + KEY_ID + ")"+ ")";// + "FOREIGN KEY(" + KEY_SUBJ
127           //+ "," + KEY_CODE + ") REFERENCES " + TABLE_COURSES + "("
128           //+ KEY_SUBJ + "," + KEY_CODE + ")" + ")";
129
130       String CREATE_OFFERING_TIMES = "CREATE TABLE " + TABLE_OFFERING_TIMES
131           + "(" + KEY_ID + " INTEGER," + KEY_DAY + " TEXT," + KEY_TIMES
132           + " TEXT ," + KEY_TIMEE + " TEXT," + KEY_LOCATION + " TEXT,"
133           + "PRIMARY KEY(" + KEY_ID + "," + KEY_DAY + ")"+ ")";
134           //+ "FOREIGN KEY(" + KEY_ID + ") REFERENCES " + TABLE_OFFERINGS
135           //+ "(" + KEY_ID + ")" + ")";
136
137       String CREATE_CONTACTS = "CREATE TABLE " + TABLE_CONTACTS + "("
138           + KEY_SUBJ + " TEXT," + KEY_CODE + " TEXT," + KEY_CID
139           + " INTEGER," + KEY_FNAME + " TEXT," + KEY_LNAME + " TEXT,"
```

```
140           + KEY_EMAIL + " TEXT," + "PRIMARY KEY(" + KEY_CID + ")"+ ")";
141           //+ "FOREIGN KEY(" + KEY_SUBJ + "," + KEY_CODE + ") REFERENCES "
142           //+ TABLE_COURSES + "(" + KEY_SUBJ + "," + KEY_CODE + ")" + ")";
143
144       db.execSQL(CREATE_COURSES);
145       db.execSQL(CREATE_TASKS);
146       db.execSQL(CREATE_OFFERINGS);
147       db.execSQL(CREATE_OFFERING_TIMES);
148       db.execSQL(CREATE_CONTACTS);
149       */
150     }
151
152     /* createDataBase - if the database does not currently exist then the database
153      * we read data from the included database to copy to a newly created one
154      */
155     public void createDataBase() throws IOException{
156
157        boolean dbExist = checkDataBase();
158
159        if(dbExist){
160          //do nothing - database already exist
161        }else{
162        //By calling this method an empty database will be created into the default
           system path
163        //of the application so that it can be overwritten by the included database.
164          this.getReadableDatabase();
165          try {
166            copyDataBase();
167          } catch (IOException e) {
168            throw new Error("Error copying database");
169          }
170        }
171     }
172
173     /* checkDataBase - checks if the database for the app currently exists
174      */
175     private boolean checkDataBase(){
176
177        SQLiteDatabase checkDB = null;
178
179        try{
180          String myPath = DB_PATH;// + DATABASE_NAME;
181          checkDB = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.
           OPEN_READONLY);
182
183        }catch(SQLiteException e){
184          //database does't exist yet.
185        }
186        if(checkDB != null){
187          checkDB.close();
188        }
189         return checkDB != null ? true : false;
190     }
191
192
193      /* copyDataBase - copies all the data from the included database in the assests
194       * folder and copies that information to the newly created application
195       * database
196       */
197     private void copyDataBase() throws IOException{
198       //Open the asset db as the input stream
199       InputStream myInput = this.context.getAssets().open(DATABASE_NAME);
200       // Path to the just created empty db
201       String outFileName = DB_PATH;
202       //Open the empty db as the output stream
203       OutputStream myOutput = new FileOutputStream(outFileName);
204       //transfer bytes from the inputfile to the outputfile
205       byte[] buffer = new byte[1024];
206       int length;
207       while ((length = myInput.read(buffer))>0){
208         myOutput.write(buffer, 0, length);
```

```
209       }
210       //Close the streams
211       myOutput.flush();
212       myOutput.close();
213       myInput.close();
214     }
215
216     /* openDataBase - open the database from the set database path
217      */
218     public void openDataBase() throws SQLException{
219       //Open the database
220       String myPath = DB_PATH;// + DATABASE_NAME;
221       myDataBase = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.
          OPEN_READONLY);
222     }
223
224     /* close - closes the streams for the database. checks if the database is open */
225       @Override
226     public synchronized void close() {
227       if(myDataBase != null)
228       myDataBase.close();
229       super.close();
230     }
231
232     /* onUpgrade - upgrading the database will drop the table and recreate */
233     @Override
234     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
235       // Drop older table if existed
236       db.execSQL("DROP TABLE IF EXISTS " + TABLE_MCOURSES);
237       // Create tables again
238       onCreate(db);
239     }
240
241     /* addCourse - initializes Brocku which get all course information from the
242      * Brock Univeristy registrar's webiste.  Gets a list of all offerings and
243      * stores the information into the MasterList table in the database
244      */
245     public void addCourse() {
246       Looper myLooper;
247       Brocku list = new Brocku();
248       myLooper = Looper.myLooper();
249           Looper.loop();
250           myLooper.quit();
251       ArrayList<MasterCourse> course = new ArrayList<MasterCourse>();
252       try {
253         course = list.execute().get();
254         SQLiteDatabase db = this.getWritableDatabase();
255         //start a bulk transaction to the database
256         db.beginTransaction();
257         for (int i = 0; i < course.size(); i++) {
258           ContentValues values = new ContentValues();
259           values.put(KEY_ID, course.get(i).id); // Course id
260           values.put(KEY_SUBJ, course.get(i).subj); // subject code
261           values.put(KEY_CODE, course.get(i).code);
262           values.put(KEY_DESC, course.get(i).desc);
263           values.put(KEY_TYPE, course.get(i).type);
264           values.put(KEY_SEC, course.get(i).sec);
265           values.put(KEY_DUR, course.get(i).dur);
266           values.put(KEY_DAYS, course.get(i).days);
267           values.put(KEY_TIME, course.get(i).time);
268           values.put(KEY_LOCATION, course.get(i).location);
269           values.put(KEY_INSTRUCTOR, course.get(i).instructor);
270           // Inserting Row to the table
271           db.insert(TABLE_MCOURSES, null, values);
272         }
273         //bulk transaction is successful
274         db.setTransactionSuccessful();
275         db.endTransaction();
276         //bulk transaction is complete
277         db.close(); // Closing database connection
278       } catch (Exception e) {}
```

```
279     }
280
281     /* getCourses - returns a list of offerings for a particular subject and
282      * code, returns an arraylist of courses
283      * @param subj - subject name
284      * @param code - subject code
285     */
286     public ArrayList<MasterCourse> getCourses(String subj, String code) {
287       SQLiteDatabase db = this.getReadableDatabase();
288       ArrayList<MasterCourse> courseList = new ArrayList<MasterCourse>();
289       courseList.ensureCapacity(50);
290       MasterCourse course;
291       //search the db for all items with subj and code
292       Cursor c = db.rawQuery("SELECT * FROM " + TABLE_MCOURSES + " where "
293           + KEY_SUBJ + "= '" + subj + "' and " + KEY_CODE + " = '" + code
294           + "'", null);
295       if (c != null) {
296         //start at the first element
297         if (c.moveToFirst()) {
298           do {
299             //enter the data from the query into a MasterCourse object
300             course = new MasterCourse();
301             course.id = c.getString(c.getColumnIndex(KEY_ID));
302             course.subj = c.getString(c.getColumnIndex(KEY_SUBJ));
303             course.code = c.getString(c.getColumnIndex(KEY_CODE));
304             course.desc = c.getString(c.getColumnIndex(KEY_DESC));
305             course.type = c.getString(c.getColumnIndex(KEY_TYPE));
306             course.sec = c.getString(c.getColumnIndex(KEY_SEC));
307             course.dur = c.getString(c.getColumnIndex(KEY_DUR));
308             course.days = c.getString(c.getColumnIndex(KEY_DAYS));
309             course.time = c.getString(c.getColumnIndex(KEY_TIME));
310             course.location = c.getString(c
311                 .getColumnIndex(KEY_LOCATION));
312             course.instructor = c.getString(c
313                 .getColumnIndex(KEY_INSTRUCTOR));
314             courseList.add(course);//add this offering to the list
315           } while (c.moveToNext());
316         }
317       }
318       c.close();
319       db.close();
320       return courseList;//return the list of offerings
321     }
322
323     /* getSubjects - returns a list of all subjects from the database */
324     public ArrayList<String> getSubjects() {
325       // String subjects;
326       ArrayList<String> subj = new ArrayList<String>();
327       try {
328         SQLiteDatabase db = this.getReadableDatabase();
329         //query the database for distinct subjects
330         Cursor c = db.rawQuery("SELECT DISTINCT " + KEY_SUBJ + " FROM "
331             + TABLE_MCOURSES + " ORDER BY " + KEY_SUBJ + " ASC", null);
332         if (c != null) {
333           //start at the first entry
334           if (c.moveToFirst()) {
335             do {//add the subjects to an arraylist
336               subj.add(c.getString(c.getColumnIndex(KEY_SUBJ)));
337             } while (c.moveToNext());
338           }
339         }
340         db.close();//close the db
341         c.close();//close the cursor
342       } catch (Exception e) {
343         subj.add(e.toString());//add an error to the list
344       }
345       return subj; //return the subject list
346     }
347
348     /* getCodes - returns a list of codes for a subject from the database
349      * @param subj - the subject to get all the codes for
```

```
350      */
351     public ArrayList<String> getCodes(String subj) {
352       ArrayList<String> codes = new ArrayList<String>();
353       try {
354         SQLiteDatabase db = this.getReadableDatabase();
355         //query for all distict subject codes given the subject
356         Cursor c = db.rawQuery("SELECT DISTINCT " + KEY_CODE + " FROM "
357             + TABLE_MCOURSES + " WHERE " + KEY_SUBJ + "='" + subj
358             + "' ORDER BY " + KEY_SUBJ + " ASC", null);
359         if (c != null) {
360           //start at the first element
361           if (c.moveToFirst()) {
362             do {
363               //add the code to the code list
364               codes.add(c.getString(c.getColumnIndex(KEY_CODE)));
365             } while (c.moveToNext());
366           }
367         }
368         db.close();//close the db
369         c.close();//close the cursor
370       } catch (Exception e) {
371         codes.add(e.toString());//add error to list if the query fails
372       }
373       return codes; //return the list of codes
374     }
375 
376     /* size - returns the total number of entries in the masterList table */
377     public int size() {
378       int i = 0;
379       try {
380         SQLiteDatabase db = this.getReadableDatabase();
381         Cursor c = db.rawQuery("SELECT COUNT(*) FROM " + TABLE_MCOURSES,
382             null);
383         if (c != null) {
384           //move to first entry which will be the count we want
385           c.moveToFirst();
386           i = c.getInt(0);
387         }
388         db.close();//close the db
389         c.close();//close the cursor
390       } catch (Exception e) {
391         i = 0;//return 0 if there are no entries in the table
392       }
393       return i; //return the number of entries in the table
394     }
395   }
```

```
 1  package edu.seaaddicts.brockbutler.coursemanager;
 2
 3  import java.util.ArrayList;
 4
 5  import android.app.Activity;
 6  import android.app.ProgressDialog;
 7  import android.content.Context;
 8  import android.content.Intent;
 9  import android.net.Uri;
10  import android.os.Bundle;
11  import android.os.Handler;
12  import android.util.Log;
13  import android.view.ContextMenu;
14  import android.view.ContextMenu.ContextMenuInfo;
15  import android.view.View.OnClickListener;
16  import android.view.Gravity;
17  import android.view.Menu;
18  import android.view.MenuItem;
19  import android.view.View;
20  import android.view.ViewGroup;
21  import android.widget.AdapterView;
22  import android.widget.ArrayAdapter;
23  import android.widget.Button;
24  import android.widget.LinearLayout;
25  import android.widget.ListView;
26  import android.widget.TextView;
27  import android.widget.Toast;
28  import edu.seaaddicts.brockbutler.R;
29  import edu.seaaddicts.brockbutler.animation.ExpandAnimation;
30  import edu.seaaddicts.brockbutler.help.HelpActivity;
31
32  public class CourseManagerActivity extends Activity {
33    public static final int CODE_COURSE_MODIFIED = 0;
34    public static final int CODE_COURSE_UNMODIFIED = 1;
35    public static final int CODE_ADD_COURSE = 2;
36
37    public static final String CODE_COURSE_SUBJECT = "csubj";
38    public static final String CODE_COURSE_CODE = "ccode";
39    public static final String CODE_COURSE_DESC = "cdesc";
40    public static final String CODE_COURSE_INSTRUCTOR = "cinstruct";
41    public static final String CODE_COURSE_INSTRUCTOR_EMAIL = "cinstructemail";
42    public static final String CODE_COURSE_OFFERINGS = "coffs";
43
44    private static final String TAG = "CourseManagerActivity";
45
46    private static final int VISIBLE = 0;
47    private static final int GONE = 8;
48
49    private ArrayList<Course> mRegisteredCoursesList;
50    private CourseHandler mCourseHandle = null;
51    private ListView mRegisteredCoursesListView = null;
52
53    @Override
54    protected void onCreate(Bundle savedInstanceState) {
55      super.onCreate(savedInstanceState);
56      setContentView(R.layout.activity_coursemanager);
57      mCourseHandle = new CourseHandler(this.getApplicationContext());
58
59      if (mCourseHandle.getSize() < 1) {
60        updateCourseDatabaseFromRegistrar();
61      }
62    }
63
64    @Override
65    protected void onResume() {
66      super.onResume();
67      populateCoursesLayout();
68    }
69
70    /**
71     * Populates the ListView with registered classes and brief details, i.e.
```

```
72       * instructor name and class times.
73       */
74      private void populateCoursesLayout() {
75        TextView tvNoCourses = (TextView) findViewById(R.id.tv_no_courses);
76        mRegisteredCoursesList = mCourseHandle.getRegisteredCourses();
77        mRegisteredCoursesListView = (ListView) findViewById(R.id.course_manager_list);
78        if (mRegisteredCoursesList.size() == 0) {
79          // There are no registered courses so set message.
80          mRegisteredCoursesListView.setVisibility(GONE);
81          tvNoCourses.setVisibility(VISIBLE);
82        } else {
83          // We have registered courses so populate ListView.
84          // Creating the list adapter and populating the list
85          ArrayAdapter<String> listAdapter = new CustomListAdapter(this,
86              R.layout.course_list_item);
87
88          for (int i = 0; i < mRegisteredCoursesList.size(); i++) {
89            listAdapter.add(mRegisteredCoursesList.get(i).mSubject + " "
90                + mRegisteredCoursesList.get(i).mCode);
91
92            for (int j = 0; j < mRegisteredCoursesList.get(i).mOfferings
93                .size(); j++)
94              Log.d(TAG, "Offerings: "
95                  + mRegisteredCoursesList.get(i).mOfferings.get(j));
96
97            Log.d(TAG, "# Offerings: "
98                + mRegisteredCoursesList.get(i).mOfferings.size());
99          }
100         mRegisteredCoursesListView.setAdapter(listAdapter);
101         tvNoCourses.setVisibility(GONE);
102         mRegisteredCoursesListView.setVisibility(VISIBLE);
103         mRegisteredCoursesListView
104             .setOnItemClickListener(new AdapterView.OnItemClickListener() {
105               public void onItemClick(AdapterView<?> parent,
106                   final View view, int position, long id) {
107                 showHideToolbar(view, position);
108               }
109             });
110         registerForContextMenu(mRegisteredCoursesListView);
111       }
112     }
113
114     public void showHelp(MenuItem item) {
115       Intent intent = new Intent(CourseManagerActivity.this,
116           HelpActivity.class);
117       Bundle bundle = new Bundle();
118       bundle.putString("activity", "coursemanager");
119       intent.putExtras(bundle);
120       startActivity(intent);
121     }
122
123     @Override
124     public void onCreateContextMenu(ContextMenu menu, View v,
125         ContextMenuInfo menuInfo) {
126       if (v.getId() == R.id.course_manager_list) {
127         //AdapterView.AdapterContextMenuInfo info =
          (AdapterView.AdapterContextMenuInfo) menuInfo;
128         String[] menuItems = getResources().getStringArray(
129             R.array.course_manager_context_menu);
130         for (int i = 0; i < menuItems.length; i++) {
131           menu.add(Menu.NONE, i, i, menuItems[i]);
132         }
133       }
134     }
135
136     /**
137      * Determines which MenuItem was selected and acts appropriately depending
138      * on choice.
139      */
140     @Override
141     public boolean onContextItemSelected(MenuItem item) {
```

```
142       AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
          item
143           .getMenuInfo();
144       int menuItemIndex = item.getItemId();
145       Log.d(TAG, "" + menuItemIndex);
146       Course thisCourse = mRegisteredCoursesList.get(info.position);
147       switch (menuItemIndex) {
148       case 0:
149
150         // Start Intent with Course as Extra
151         Intent i = new Intent(CourseManagerActivity.this,
152             ModifyCourseActivity.class);
153         i.putExtra(CODE_COURSE_SUBJECT, thisCourse.mSubject);
154         i.putExtra(CODE_COURSE_CODE, thisCourse.mCode);
155         startActivity(i);
156         break;
157       case 1:
158           Course c = mRegisteredCoursesList.get(info.position);
159           mCourseHandle.removeCourse(c);
160       }
161       populateCoursesLayout();
162       return true;
163     }
164
165     @Override
166     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
167       if (resultCode == RESULT_OK) {
168         switch (requestCode) {
169         case (CODE_ADD_COURSE):
170           // Course c = (Course)
171           // data.getSerializableExtra(CODE_COURSE_OBJECT);
172           // Toast.makeText(getApplicationContext(),
173           // c.mSubject + " " + c.mCode + " added.",
174           // Toast.LENGTH_LONG).show();
175           break;
176         default:
177           break;
178         }
179       }
180       if (resultCode == RESULT_OK && requestCode == CODE_COURSE_MODIFIED) {
181         if (data.hasExtra("returnKey1")) {
182           Toast.makeText(this, data.getExtras().getString("returnKey1"),
183               Toast.LENGTH_SHORT).show();
184         }
185       }
186     }
187
188     /**
189      * Shows/hides verbose description of course.
190      *
191      * @param view
192      *            The selected view to hide/show.
193      */
194     private void showHideToolbar(View view, int position) {
195       View toolbar = view.findViewById(R.id.course_manager_toolbar);
196
197       // Get current course info.
198       String subj = mRegisteredCoursesList.get(position).mSubject;
199       String code = mRegisteredCoursesList.get(position).mCode;
200       String offering = "";
201       ArrayList<Offering> offs = mRegisteredCoursesList.get(position).mOfferings;
202       ArrayList<OfferingTime> offTimes;
203
204       // Creating the expand animation for the item
205       ExpandAnimation expandAni = new ExpandAnimation(toolbar,
206           ExpandAnimation.ANIMATE_SHORT);
207
208       // Start the animation on the toolbar
209       toolbar.startAnimation(expandAni);
210
211       ((TextView) view.findViewById(R.id.tv_prof_name))
```

```
212           .setText(mRegisteredCoursesList.get(position).mInstructor);
213       final String e= mRegisteredCoursesList.get(position).mInstructor_email;
214       ((Button) view.findViewById(R.id.prof_email_button))
215       .setOnClickListener(new OnClickListener() {
216         public void onClick(View v) {
217           sendEmail(e);
218         }
219       });
220
221       Log.d(TAG,"Number of Offerings for " + subj + " " + code + ": " + offs.size());
222
223       // Add Offerings registered for.
224       for (int i = 0; i < offs.size(); i++) {
225         String what = offs.get(i).mType.substring(0, 3).trim();
226
227         offTimes = offs.get(i).mOfferingTimes;
228
229         Log.d(TAG, "Offering Type: " + what + ", # " + offTimes.size());
230
231         offering = "";
232
233         // Loop through OfferingTimes for each Offering to populate
234         for (int j = 0; j < offTimes.size(); j++) {
235           offering += offTimes.get(j).mDay + " "
236               + offTimes.get(j).mStartTime + " - "
237               + offTimes.get(j).mEndTime + " @ "
238               + offTimes.get(j).mLocation + "\n";
239
240           // Check for type of Offering and add as appropriate
241           if (what.equalsIgnoreCase("lec")) {
242             TextView tv = ((TextView) view
243                 .findViewById(R.id.tv_lecture));
244             tv.setText(offering);
245           }
246
247           else if (what.equalsIgnoreCase("lab"))
248             ((TextView) view.findViewById(R.id.tv_lab))
249                 .setText(offering);
250
251           else if (what.equalsIgnoreCase("tut"))
252             ((TextView) view.findViewById(R.id.tv_tutorial))
253                 .setText(offering);
254
255           else if (what.equalsIgnoreCase("sem"))
256             ((TextView) view.findViewById(R.id.tv_seminar))
257                 .setText(offering);
258         }
259       }
260
261       float mark  = mCourseHandle.getMark(mRegisteredCoursesList.get(position));
262       float base  = mCourseHandle.getBase(mRegisteredCoursesList.get(position));
263       float total=0;
264       if(base != 0){
265         total= (mark/base)*100;
266       }
267
268       String grade= ""+(int)mark+"/"+(int)base+" = "+ (int)total+"%";
269       ((TextView) view.findViewById(R.id.course_grade_grade))
270       .setText(grade);
271
272       /*
273        * Hide class type if none available
274        */
275       if (((TextView) view.findViewById(R.id.tv_lecture)).getText()
276           .toString().equalsIgnoreCase("none"))
277         view.findViewById(R.id.row_lec).setVisibility(GONE);
278       if (((TextView) view.findViewById(R.id.tv_lab)).getText()
279           .toString().equalsIgnoreCase("none"))
280         view.findViewById(R.id.row_lab).setVisibility(GONE);
281       if (((TextView) view.findViewById(R.id.tv_tutorial)).getText()
282           .toString().equalsIgnoreCase("none"))
```

```
283         view.findViewById(R.id.row_tut).setVisibility(GONE);
284       if (((TextView) view.findViewById(R.id.tv_seminar)).getText()
285           .toString().equalsIgnoreCase("none"))
286         view.findViewById(R.id.row_sem).setVisibility(GONE);
287     }
288
289     @Override
290     public boolean onCreateOptionsMenu(Menu menu) {
291       // Inflate the menu; this adds items to the action bar if it is present.
292       getMenuInflater().inflate(R.menu.activity_coursemanager, menu);
293       return true;
294     }
295
296     /**
297      * A simple implementation of list adapter to populate ListView with
298      * courses.
299      */
300     class CustomListAdapter extends ArrayAdapter<String> {
301
302       public CustomListAdapter(Context context, int textViewResourceId) {
303         super(context, textViewResourceId);
304       }
305
306       @Override
307       public View getView(int position, View convertView, ViewGroup parent) {
308
309         if (convertView == null) {
310           convertView = getLayoutInflater().inflate(
311               R.layout.course_list_item, null);
312         }
313
314         ((TextView) convertView.findViewById(R.id.course_list_item_title))
315             .setText(getItem(position));
316
317         // Resets the toolbar to be closed
318         View toolbar = convertView
319             .findViewById(R.id.course_manager_toolbar);
320         ((LinearLayout.LayoutParams) toolbar.getLayoutParams()).bottomMargin = -50;
321         toolbar.setVisibility(View.GONE);
322         return convertView;
323       }
324     }
325
326     /**
327      * Launches AddCourseActivity as intent.
328      *
329      * @param menu
330      *            MenuItem selected.
331      */
332     public void addCourse(MenuItem menu) {
333       Intent i = new Intent(CourseManagerActivity.this,
334           AddCourseActivity.class);
335       startActivity(i);
336     }
337
338     /**
339      * Allows user to manually fetch course calendar offerings from Registrar
340      *
341      * @param item
342      */
343     public void updateMaster(MenuItem item) {
344       updateCourseDatabaseFromRegistrar();
345     }
346
347     private void sendEmail(String instr_email) {
348       Intent i = new Intent(Intent.ACTION_SENDTO);
349       i.setType("message/rfc822");
350       i.setData(Uri.parse("mailto:"+instr_email));
351       try {
352         startActivity(Intent.createChooser(i, "Send mail..."));
353       } catch (android.content.ActivityNotFoundException ex) {
```

```
354         Toast.makeText(CourseManagerActivity.this,
355             "There are no email clients installed.", Toast.LENGTH_SHORT)
356             .show();
357       }
358     }
359
360     /**
361      * Updates the course calendar offerings master table. Is called at first
362      * run (if table does not exist) and manually when user wishes to check for
363      * updates. Progress bar to prevent hanging on main thread.
364      */
365     private void updateCourseDatabaseFromRegistrar() {
366       final Handler handler = new Handler();
367       final ProgressDialog progressDialog;
368
369       TextView title = new TextView(CourseManagerActivity.this);
370       title.setText(R.string.loading_courses_registrar);
371       title.setGravity(Gravity.FILL);
372
373       TextView msg = new TextView(CourseManagerActivity.this);
374       msg.setText(R.string.loading_courses_registrar_msg);
375       msg.setGravity(Gravity.FILL);
376
377       progressDialog = ProgressDialog.show(this, "Course Timetable Update",
378           "Updating course timetable from registrar. Please be patient.");
379
380       Thread thread = new Thread() {
381         public void run() {
382           mCourseHandle.updateAll();
383           // this will handle the post task.
384           // it will run when the time consuming task get finished
385           handler.post(new Runnable() {
386             public void run() {
387
388               // Update your UI or
389               // do any Post job after the time consuming task
390               // remember to dismiss the progress dialog here.
391               // updateUI();
392               progressDialog.dismiss();
393               populateCoursesLayout();
394             }
395           });
396         }
397       };
398       thread.start();
399     }
400   }
401
```

```java
/**
 * CurrentCoursesHandler.java
 * Brock Butler
 * Handles database table creation and queries for course information
 * Created by James Grisdale on 2013-02-24
 * Copyright (c) 2013 Sea Addicts. All rights reserved.
 **/

package edu.seaaddicts.brockbutler.coursemanager;

import java.util.ArrayList;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.DatabaseUtils;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import edu.seaaddicts.brockbutler.contacts.Contact;
import edu.seaaddicts.brockbutler.scheduler.Task;

public class CurrentCoursesHandler extends SQLiteOpenHelper {
  private static final int DATABASE_VERSION = 1;
  // Database Name
  private static final String DATABASE_NAME = "Database";
  // table names
  private static final String TABLE_COURSES = "courses";
  private static final String TABLE_TASKS = "tasks";
  private static final String TABLE_OFFERINGS = "offerings";
  private static final String TABLE_OFFERING_TIMES = "offering_times";
  private static final String TABLE_CONTACTS = "contacts";
  // field names
  private static final String KEY_SUBJ = "subj";
  private static final String KEY_CODE = "code";
  private static final String KEY_DESC = "desc";
  private static final String KEY_INSTRUCTOR = "instructor";
  private static final String KEY_ID = "id";
  private static final String KEY_TYPE = "type";
  private static final String KEY_SEC = "sec";
  private static final String KEY_DAY = "day";
  private static final String KEY_TIMES = "time_start";
  private static final String KEY_TIMEE = "time_end";
  private static final String KEY_LOCATION = "location";
  private static final String KEY_ASSIGN = "assign";
  private static final String KEY_NAME = "name";
  private static final String KEY_MARK = "mark";
  private static final String KEY_BASE = "base";
  private static final String KEY_WEIGHT = "weight";
  private static final String KEY_DUE = "due";
  private static final String KEY_CREATE_DATE = "create_date";
  private static final String KEY_IS_DONE = "is_done";
  private static final String KEY_CID = "cid";
  private static final String KEY_FNAME = "fname";
  private static final String KEY_LNAME = "lname";
  private static final String KEY_EMAIL = "email";
  private static final String KEY_PRIORITY = "priority";
  private static final String KEY_INSTREMAIL = "instructor_email";

  /* Constructor for the database helper */
  public CurrentCoursesHandler(Context context) {
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
  }

  /* Create tables for courses, tasks, offerings, offering times, and contacts
   * in the database if they do not exist when the database helper is first
   * called
   */
  @Override
  public void onCreate(SQLiteDatabase db) {
  }

```

```
72      /* on an upgrade drop tables and recreate */
73      @Override
74      public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
75        // Drop older table if existed
76        db.execSQL("DROP TABLE IF EXISTS " + TABLE_COURSES);
77        db.execSQL("DROP TABLE IF EXISTS " + TABLE_TASKS);
78        db.execSQL("DROP TABLE IF EXISTS " + TABLE_OFFERINGS);
79        db.execSQL("DROP TABLE IF EXISTS " + TABLE_OFFERING_TIMES);
80        db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);
81        // Create tables again
82        onCreate(db);
83      }
84
85      /* addCourse - adds all information for a course to the database adding
86       * course, offerings, tasks and contacts information, if information exists
87       * for the course then an update is done, otherwise and insert is done
88       * @param course - the course information to add to the course table
89       */
90      public void addCourse(Course course) {
91        SQLiteDatabase db = this.getWritableDatabase();
92        ContentValues values = new ContentValues();
93        long num = 0;
94        boolean update = false;
95        //check if the course already exists in the table
96        num = DatabaseUtils.queryNumEntries(db, TABLE_COURSES, KEY_SUBJ + " ='"
97            + course.mSubject + "' AND " + KEY_CODE + "='" + course.mCode
98            + "'");
99        if (num > 0)//if it exists then do an update
100         update = true;
101       // values to be added to the table
102       values.put(KEY_SUBJ, course.mSubject); // subject code
103       values.put(KEY_CODE, course.mCode);
104       values.put(KEY_DESC, course.mDesc);
105       values.put(KEY_INSTRUCTOR, course.mInstructor);
106       values.put(KEY_INSTREMAIL, course.mInstructor_email);
107       // Inserting or updating Row
108       if (update)
109         db.update(TABLE_COURSES, values, KEY_SUBJ + " ='" + course.mSubject
110             + "' AND " + KEY_CODE + "='" + course.mCode + "'", null);
111       else
112         db.insert(TABLE_COURSES, null, values);
113       db.close(); // Closing database connection
114       values.clear();
115       addOfferings(course);//add the offerings for the course
116       addTasks(course);//add the tasks for the course
117       addContacts(course.mContacts);//add the contacts for the course
118       db.close();//close the database
119     }
120
121     /* deleteCourse - removes all information for the given course from the
122      * database
123      * @param course - the course data to be removed from the course table
124      */
125     public void deleteCourse(Course course) {
126       SQLiteDatabase db = this.getWritableDatabase();
127       //delete the row for the selected course
128       db.delete(TABLE_COURSES, KEY_SUBJ + "='" + course.mSubject + "' AND "
129           + KEY_CODE + "='" + course.mCode + "'", null);
130       db.close(); //close the db
131       //delete all the offerings
132       for (int i=0; i<course.mOfferings.size(); i++){
133         deleteOffering(course.mOfferings.get(i));
134       }
135       //delete all the tasks
136       for (int i=0; i<course.mTasks.size(); i++){
137         removeTask(course.mTasks.get(i));
138       }
139     }
140
141     /* getCourse - retrieves all information for the given course
142      * @param subj - the course name
```

```
143       * @param code - the course code
144       */
145      public Course getCourse(String subj, String code) {
146        SQLiteDatabase db = this.getReadableDatabase();
147        Course course = new Course();
148        //query to retrieve all information for the given subj and code
149        Cursor c = db.rawQuery("SELECT * FROM " + TABLE_COURSES + " where "
150            + KEY_SUBJ + "= '" + subj + "' and " + KEY_CODE + " = '" + code
151            + "'", null);
152        if (c != null) {
153          //start at the first record
154          if (c.moveToFirst()) {
155            do {
156              //set values in the course object from the table
157              course.mSubject = c.getString(c.getColumnIndex(KEY_SUBJ));
158              course.mCode = c.getString(c.getColumnIndex(KEY_CODE));
159              course.mDesc = c.getString(c.getColumnIndex(KEY_DESC));
160              course.mInstructor = c.getString(c
161                  .getColumnIndex(KEY_INSTRUCTOR));
162              course.mInstructor_email = c.getString(c
163                  .getColumnIndex(KEY_INSTREMAIL));
164              course.mOfferings = getOfferings(course.mSubject,
165                  course.mCode);
166              course.mTasks = getTasks(course);//get the tasks for the course
167              course.mContacts = getContacts(course);//get the contacts for the course
168            } while (c.moveToNext());
169          }
170        }
171        c.close();//close the cursor
172        db.close();//close the database
173        return course;//return the course object
174      }
175
176       /* addOfferings - adds all offerings offered by a particular course as well
177       * as their offering times
178       * @param course - the course to add the offerings from
179       */
180      public void addOfferings(Course course) {
181        Offering offering;
182        OfferingTime offeringtime;
183        ContentValues values = new ContentValues();
184        SQLiteDatabase db = this.getWritableDatabase();
185        long num = 0;
186        boolean update = false;
187        for (int i = 0; i < course.mOfferings.size(); i++) {
188          offering = course.mOfferings.get(i);
189          num = 0;
190          update = false;
191          //find if the offering already exists
192          num = DatabaseUtils.queryNumEntries(db, TABLE_OFFERINGS, KEY_SUBJ
193              + " ='" + offering.mSubj + "' AND " + KEY_CODE + "='"
194              + offering.mCode + "' AND " + KEY_TYPE + "='"
195              + offering.mType + "' AND " + KEY_SEC + "="
196              + offering.mSection);
197          if (num > 0)//if the offering exists then do an update
198            update = true;
199          //set the feilds and values
200          values.put(KEY_SUBJ, course.mSubject);
201          values.put(KEY_CODE, course.mCode);
202          values.put(KEY_TYPE, offering.mType);
203          values.put(KEY_SEC, offering.mSection);
204          if (update)//update the offering information
205            db.update(TABLE_OFFERINGS, values, KEY_SUBJ + " ='"
206                + offering.mSubj + "' AND " + KEY_CODE + "='"
207                + offering.mCode + "' AND " + KEY_TYPE + "='"
208                + offering.mType + "' AND " + KEY_SEC + "="
209                + offering.mSection, null);
210          else//insert the offering information
211            db.insert(TABLE_OFFERINGS, null, values);
212          values.clear();
213        }
```

```
214       SQLiteDatabase rdb = this.getReadableDatabase();
215       for (int i=0; i<course.mOfferings.size(); i++){
216         offering = course.mOfferings.get(i);
217         //now adding the offering times for each offering
218         for (int j = 0; j < offering.mOfferingTimes.size(); j++) {
219           offeringtime = offering.mOfferingTimes.get(j);
220           num = 0;
221           update = false;
222           //see if the offering time exists
223           num = DatabaseUtils.queryNumEntries(db, TABLE_OFFERING_TIMES,
224               KEY_ID + " =" + offering.mId+ " AND "+KEY_DAY+"='"+offeringtime.mDay+"'");
225           if (num > 1)//if exists then update
226             update = true;
227           //query for offering id for each offering time
228           Cursor c = rdb.rawQuery("SELECT " + KEY_ID + " FROM "
229               + TABLE_OFFERINGS + " WHERE " + KEY_SUBJ + "='"
230               + offering.mSubj + "' AND " + KEY_CODE + "='"
231               + offering.mCode + "' AND " + KEY_TYPE + "='"
232               + offering.mType + "' AND " + KEY_SEC + "="
233               + offering.mSection, null);
234           c.moveToFirst();
235           //set fields and values to be added
236           offering.mId = c.getInt(c.getColumnIndex(KEY_ID));
237           values.put(KEY_ID, offering.mId);
238           values.put(KEY_DAY, offeringtime.mDay);
239           values.put(KEY_TIMES, offeringtime.mStartTime);
240           values.put(KEY_TIMEE, offeringtime.mEndTime);
241           values.put(KEY_LOCATION, offeringtime.mLocation);
242           if (update)//update the record
243             db.update(TABLE_OFFERING_TIMES, values, KEY_ID + " ="
244                 + offering.mId, null);
245           else//insert the record
246             db.insert(TABLE_OFFERING_TIMES, null, values);
247           values.clear();
248           c.close();//close the cursor
249         }
250       }
251       rdb.close();//close database connection
252       db.close();//close database connection
253     }
254
255
256     /* deleteOffering - removes all information from the databse for the given
257      * offering
258      * @param offering - the offering to be removed from the offerings table
259      */
260     public void deleteOffering(Offering offering) {
261       int id;
262       SQLiteDatabase rdb = this.getReadableDatabase();
263       //query to get the id of the offering to be deleted
264       Cursor c = rdb.rawQuery("SELECT " + KEY_ID + " FROM "
265           + TABLE_OFFERINGS + " WHERE " + KEY_SUBJ + "='"
266           + offering.mSubj + "' AND " + KEY_CODE + "='"
267           + offering.mCode + "' AND " + KEY_TYPE + "='"
268           + offering.mType + "' AND " + KEY_SEC + "="
269           + offering.mSection, null);
270       c.moveToFirst();
271       id = c.getInt(c.getColumnIndex(KEY_ID));
272       c.close();//close cursor
273       SQLiteDatabase db = this.getWritableDatabase();
274       //delete the offering times associated to the offering
275       db.delete(TABLE_OFFERING_TIMES, KEY_ID +"="+id, null);
276       //delete the offering from the offerings table
277       db.delete(TABLE_OFFERINGS, KEY_SUBJ
278           + " ='" + offering.mSubj + "' AND " + KEY_CODE + "='"
279           + offering.mCode + "' AND " + KEY_TYPE + "='"
280           + offering.mType + "' AND " + KEY_SEC + "="
281           + offering.mSection, null);
282       db.close();//close the database
283     }
284
```

```
285     /* addTasks - adds all tasks associated with a given course
286      * @param course - the course object with the tasks to be added
287      */
288     public void addTasks(Course course) {
289       Task task;
290       ContentValues values = new ContentValues();
291       SQLiteDatabase db = this.getWritableDatabase();
292       long num = 0;
293       boolean update = false;
294       for (int i = 0; i < course.mTasks.size(); i++) {
295         task = course.mTasks.get(i);
296         num = 0;
297         update = false;
298         //see if the task exists already
299         num = DatabaseUtils.queryNumEntries(db, TABLE_TASKS, KEY_ASSIGN
300             + " ='" + task.mAssign + "' AND " + KEY_SUBJ + " ='"
301             + task.mSubj + "' AND " + KEY_CODE + "='" + task.mCode
302             + "'");
303         if (num > 0)//if exists then update
304           update = true;
305         values.put(KEY_SUBJ, task.mSubj);
306         values.put(KEY_CODE, task.mCode);
307         //if the task number is not 0 then use that value
308         try {
309           if (task.mAssign != 0)
310             values.put(KEY_ASSIGN, task.mAssign);
311         } catch (NullPointerException e){}
312         //set all values to be added
313         values.put(KEY_NAME, task.mName);
314         values.put(KEY_MARK, task.mMark);
315         values.put(KEY_BASE, task.mBase);
316         values.put(KEY_WEIGHT, task.mWeight);
317         values.put(KEY_DUE, task.mDueDate);
318         values.put(KEY_CREATE_DATE, task.mCreationDate);
319         values.put(KEY_PRIORITY, task.mPriority);
320         values.put(KEY_IS_DONE, task.mIsDone);
321         if (update)//update the row
322           db.update(TABLE_TASKS, values, KEY_ASSIGN + " ='" + task.mAssign
323               + "' AND " + KEY_SUBJ + " ='" + task.mSubj + "' AND "
324               + KEY_CODE + "='" + task.mCode + "'", null);
325         else//insert the row
326           db.insert(TABLE_TASKS, null, values);
327         values.clear();
328       }
329       db.close();//close the database
330     }
331
332     /* addContacts - add contacts to the contacts table in the database for the
333      * given list of contacts
334      * @param contacts - the list of contacts to be added to the contacts table
335      */
336     public void addContacts(ArrayList<Contact> contacts) {
337       Contact contact;
338       ContentValues values = new ContentValues();
339       SQLiteDatabase db = this.getWritableDatabase();
340       long num = 0;
341       boolean update = false;
342       for (int j = 0; j < contacts.size(); j++) {
343         contact = contacts.get(j);
344         num = 0;
345         update = false;
346         //check if the contact exists
347         num = DatabaseUtils.queryNumEntries(db, TABLE_CONTACTS, KEY_SUBJ
348             + " ='" + contact.mSubj + "' AND " + KEY_CODE + "='"
349             + contact.mCode + "' AND " + KEY_FNAME + "='" + contact.mFirstName
350             + "' AND " + KEY_LNAME + "='"+ contact.mLastName+"'");
351         if (num > 0)//if exits then update
352           update = true;
353         //set the fields and the values
354         values.put(KEY_SUBJ, contact.mSubj);
355         values.put(KEY_CODE, contact.mCode);
```

```
356         values.put(KEY_FNAME, contact.mFirstName);
357         values.put(KEY_LNAME, contact.mLastName);
358         values.put(KEY_EMAIL, contact.mEmail);
359         if (update)//update the record
360           db.update(TABLE_CONTACTS, values, KEY_SUBJ + " ='"
361               + contact.mSubj + "' AND " + KEY_CODE + "='"
362               + contact.mCode + "' AND " + KEY_FNAME + "='"
363               + contact.mFirstName + "' AND " + KEY_LNAME + "='"
364               + contact.mLastName + "' AND " + KEY_EMAIL + "='"
365               + contact.mEmail + "'", null);
366         else//insert the record
367           db.insert(TABLE_CONTACTS, null, values);
368         values.clear();
369       }
370       db.close();//close the database
371     }
372
373     /* addTasks - adds a task for a certain course using the addTasks(course) method
374      * @param task - the task to be added
375      */
376     public void addTasks(Task task) {
377       Course course = new Course();
378       course.mTasks.add(task);
379       addTasks(course);//add the tasks for the course object
380     }
381
382     /* getOfferings - gets all offerings for a given subject and code
383      * @param subj - the course subject
384      * @param code - the course code
385      */
386     public ArrayList<Offering> getOfferings(String subj, String code) {
387       ArrayList<Offering> offerings = new ArrayList<Offering>();
388       ArrayList<OfferingTime> offtimes;
389       Offering offering;
390       OfferingTime otime;
391       SQLiteDatabase db = this.getReadableDatabase();
392       //get all offerings for the subj and code
393       Cursor c = db.rawQuery("SELECT *  FROM " + TABLE_OFFERINGS + " WHERE "
394           + KEY_SUBJ + "='" + subj + "' and " + KEY_CODE + "='" + code
395           + "'", null);
396       try {
397         if (c != null) {
398           if (c.moveToFirst()) {//start at the first record
399             do {
400               offering = new Offering();
401               //add the data into a new offering object
402               offering.mId = c.getInt(c.getColumnIndex(KEY_ID));
403               offering.mSubj = c
404                   .getString(c.getColumnIndex(KEY_SUBJ));
405               offering.mCode = c
406                   .getString(c.getColumnIndex(KEY_CODE));
407               offering.mType = c
408                   .getString(c.getColumnIndex(KEY_TYPE));
409               offering.mSection = c.getInt(c.getColumnIndex(KEY_SEC));
410               offerings.add(offering);
411             }while (c.moveToNext());//get next record
412             c.close();//close cursor
413             //get all the offering times for each offering
414             for (int i=0; i<offerings.size(); i++){
415               offtimes = new ArrayList<OfferingTime>();
416               //get the id for the offering
417               Cursor o = db.rawQuery("SELECT *  FROM "
418                   + TABLE_OFFERING_TIMES + " WHERE " + KEY_ID
419                   + "=" + offerings.get(i).mId, null);
420               if (o != null) {
421                 if (o.moveToFirst()) {//move to first offering time
422                   do {
423                     otime = new OfferingTime();
424                     //insert data from table to OfferingTime oject
425                     otime.mOid = o.getInt(o
426                         .getColumnIndex(KEY_ID));
```

```
427                     otime.mDay = o.getString(o
428                         .getColumnIndex(KEY_DAY));
429                     otime.mStartTime = o.getString(o
430                         .getColumnIndex(KEY_TIMES));
431                     otime.mEndTime = o.getString(o
432                         .getColumnIndex(KEY_TIMEE));
433                     otime.mLocation = o.getString(o
434                         .getColumnIndex(KEY_LOCATION));
435                     offtimes.add(otime);
436                   } while (o.moveToNext());//get next time
437                 }
438               }
439               offerings.get(i).mOfferingTimes = offtimes;
440               o.close();//close cursor
441             }
442           }
443         }
444         db.close();//close database
445       } catch (Exception e) {}
446       return offerings;//return the offerings
447     }
448
449     /* getTasks - gets all tasks a person may have from the database */
450     public ArrayList<Task> getTasks() {
451       ArrayList<Task> tasks = new ArrayList<Task>();
452       SQLiteDatabase db = this.getReadableDatabase();
453       Task task;
454       //get all tasks from the tasks table
455       Cursor c = db.rawQuery("SELECT * FROM " + TABLE_TASKS, null);
456       if (c != null) {
457         if (c.moveToFirst()) {//start at the first record
458           do {
459             task = new Task();
460             //insert data from the table into a new task object
461             task.mSubj = c.getString(c.getColumnIndex(KEY_SUBJ));
462             task.mCode = c.getString(c.getColumnIndex(KEY_CODE));
463             task.mAssign = c.getInt(c.getColumnIndex(KEY_ASSIGN));
464             task.mName = c.getString(c.getColumnIndex(KEY_NAME));
465             task.mMark = c.getInt(c.getColumnIndex(KEY_MARK));
466             task.mBase = c.getInt(c.getColumnIndex(KEY_BASE));
467             task.mWeight = c.getFloat(c.getColumnIndex(KEY_WEIGHT));
468             task.mDueDate = c.getString(c.getColumnIndex(KEY_DUE));
469             task.mIsDone = c.getInt(c.getColumnIndex(KEY_IS_DONE));
470             task.mCreationDate = c.getString(c.getColumnIndex(KEY_CREATE_DATE));
471             task.mPriority = c.getInt(c.getColumnIndex(KEY_PRIORITY));
472             tasks.add(task);//add the task to the list
473           } while (c.moveToNext());
474         }
475       }
476       c.close();//close cursor
477       db.close();//close database
478       return tasks;//return the list of tasks
479     }
480
481     /* getTasks - gets all tasks for a particular course
482      * @param course - the course to get the tasks for
483      */
484     private ArrayList<Task> getTasks(Course course) {
485       ArrayList<Task> tasks = new ArrayList<Task>();
486       SQLiteDatabase db = this.getReadableDatabase();
487       Task task;
488       // get all task information for the choosen course
489       Cursor c = db.rawQuery("SELECT * FROM " + TABLE_TASKS + " WHERE "
490           + KEY_SUBJ + "='" + course.mSubject + "' AND " + KEY_CODE
491           + "='" + course.mCode + "'", null);
492       if (c != null) {
493         if (c.moveToFirst()) {//start at the first record
494           do {
495             task = new Task();
496             //insert data from the table to a new task object
497             task.mSubj = c.getString(c.getColumnIndex(KEY_SUBJ));
```

```
498             task.mCode = c.getString(c.getColumnIndex(KEY_CODE));
499             task.mAssign = c.getInt(c.getColumnIndex(KEY_ASSIGN));
500             task.mName = c.getString(c.getColumnIndex(KEY_NAME));
501             task.mMark = c.getInt(c.getColumnIndex(KEY_MARK));
502             task.mBase = c.getInt(c.getColumnIndex(KEY_BASE));
503             task.mWeight = c.getFloat(c.getColumnIndex(KEY_WEIGHT));
504             task.mDueDate = c.getString(c.getColumnIndex(KEY_DUE));
505             task.mIsDone = c.getInt(c.getColumnIndex(KEY_IS_DONE));
506             task.mCreationDate = c.getString(c.getColumnIndex(KEY_CREATE_DATE));
507             task.mPriority = c.getInt(c.getColumnIndex(KEY_PRIORITY));
508             task.mContacts = getContacts(course);
509             tasks.add(task);//add task to the list of tasks
510           } while (c.moveToNext());//get next record
511         }
512       }
513       c.close();//close cursor
514       db.close();//close database
515       return tasks;//return the list of tasks
516     }
517
518     /* getContacts - get all contacts for a specified course
519      * @param course - the course object to get contacts for
520      */
521     private ArrayList<Contact> getContacts(Course course) {
522       ArrayList<Contact> contacts = new ArrayList<Contact>();
523       SQLiteDatabase db = this.getReadableDatabase();
524       Contact contact;
525       //get all contacts from the contacts table for the specified course
526       Cursor c = db.rawQuery("SELECT * FROM " + TABLE_CONTACTS + " WHERE "
527           + KEY_SUBJ + "='" + course.mSubject + "' AND " + KEY_CODE
528           + "='" + course.mCode + "'", null);
529       if (c != null) {
530         if (c.moveToFirst()) {//get first record
531           do {
532             contact = new Contact();
533             //insert data from the contacts table to a new contact object
534             contact.mSubj = c.getString(c.getColumnIndex(KEY_SUBJ));
535             contact.mCode = c.getString(c.getColumnIndex(KEY_CODE));
536             contact.mId = c.getInt(c.getColumnIndex(KEY_CID));
537             contact.mFirstName = c.getString(c
538                 .getColumnIndex(KEY_FNAME));
539             contact.mLastName = c
540                 .getString(c.getColumnIndex(KEY_LNAME));
541             contact.mEmail = c.getString(c.getColumnIndex(KEY_EMAIL));
542             contacts.add(contact);//add contact to list of contacts
543           } while (c.moveToNext());//get next record
544         }
545       }
546       c.close(); //close cursor
547       db.close(); //close database
548       return contacts; //return list of contacts
549     }
550
551     /* removeTask - deletes a given task from the tasks table of the database
552      * @param task - the task to be removed from the tasks table
553      */
554     public void removeTask(Task task) {
555       SQLiteDatabase db = this.getWritableDatabase();
556       //delete the record for the given task
557       db.delete(TABLE_TASKS, KEY_SUBJ
558           + " ='" + task.mSubj + "' AND " + KEY_CODE + "='"
559           + task.mCode + "' AND " + KEY_ASSIGN + "="
560           + task.mAssign, null);
561       db.close();//close database
562     }
563
564     /* removeContact - deletes a contact from the contacts table from the database
565      * @param contact - the contact information to be deleted from the table
566      */
567     public void removeContact(Contact contact){
568       SQLiteDatabase db = this.getWritableDatabase();
```

```
569       //delete the record associated with the contact id
570       db.delete(TABLE_CONTACTS, KEY_CID +"="+contact.mId, null );
571     }
572
573     /* getRegCourses - gets all courses added to the courses table of the
574      * database and all of it's components
575      */
576     public ArrayList<Course> getRegCourses() {
577       ArrayList<Course> courses = new ArrayList<Course>();
578       SQLiteDatabase db = this.getReadableDatabase();
579       try {
580         //get all courses from the course table
581         Cursor c = db.rawQuery("SELECT * FROM " + TABLE_COURSES, null);
582         if (c != null) {
583           if (c.moveToFirst()) {//start at the first record
584             do {
585               //get course information for each course found in courses table
586               courses.add(getCourse(
587                   c.getString(c.getColumnIndex(KEY_SUBJ)),
588                   c.getString(c.getColumnIndex(KEY_CODE))));
589             } while (c.moveToNext());//get next record
590           }
591         }
592         c.close();//close cursor
593       } catch (Exception e) {}
594       db.close();//close database
595       return courses; //return list of current courses
596     }
597
598     /* Query - a general method to allow a query to be done that has not been
599      * specified. it returns a cursor object to allow the data to be read
600      * @param s - a query sent as a string to perform the query on the database
601      */
602     public Cursor Query(String s) {
603       SQLiteDatabase db = this.getReadableDatabase();
604       Cursor c = db.rawQuery(s, null);//perform query
605       db.close();
606       return c;//return cursor object
607     }
608   }
609
```

```
1   /**
2    * MasterCourse.java
3    * Brock Butler
4    * A wrapper class for course timetable information
5    * Created by James Grisdale on 2013-02-24
6    * Copyright (c) 2013 Sea Addicts. All rights reserved.
7   **/
8
9   package edu.seaaddicts.brockbutler.coursemanager;
10
11  public class MasterCourse {
12    public String id;  //course id
13    public String code;  //course code
14    public String subj;  //faculty name
15    public String desc;  //course description
16    public String type;  //course type
17    public String sec;  //section
18    public String dur;  //duration
19    public String days;  //days offered
20    public String time;  //time offered
21    public String location;  //location
22    public String instructor;  //instructor
23  }
24
```

```
 1  package edu.seaaddicts.brockbutler.coursemanager;
 2
 3  import java.util.ArrayList;
 4
 5  import android.app.Activity;
 6  import android.os.Bundle;
 7  import android.util.Log;
 8  import android.util.SparseBooleanArray;
 9  import android.view.View;
10  import android.view.View.OnClickListener;
11  import android.widget.ArrayAdapter;
12  import android.widget.Button;
13  import android.widget.LinearLayout;
14  import android.widget.ListView;
15  import android.widget.TextView;
16  import edu.seaaddicts.brockbutler.R;
17  import edu.seaaddicts.brockbutler.contacts.Contact;
18  import edu.seaaddicts.brockbutler.scheduler.Task;
19
20  public class ModifyCourseActivity extends Activity {
21
22    private static final String TAG = "ModifyCourseActivity";
23
24    private static final int VISIBLE = 0;
25    private static final int GONE = 8;
26
27    ArrayList<String> mLecs;
28    ArrayList<String> mLabs;
29    ArrayList<String> mTuts;
30    ArrayList<String> mSems;
31
32    ArrayList<Offering> mLecsOfferings;
33    ArrayList<Offering> mLabsOfferings;
34    ArrayList<Offering> mTutsOfferings;
35    ArrayList<Offering> mSemsOfferings;
36
37    public String mSubject;
38    public String mCode;
39    public String mDesc;
40    public String mInstructor;
41    public String mInstructor_email;
42    public ArrayList<Offering> mOfferings;
43    public ArrayList<Task> mTasks;
44    public ArrayList<Contact> mContacts;
45    public Course course;
46    public Course brock;
47
48    private Button mSaveButton;
49    private Button mCancelButton;
50
51    private ListView mLecsListView;
52    private ListView mSemsListView;
53    private ListView mTutsListView;
54    private ListView mLabsListView;
55
56    private CurrentCoursesHandler mCourseHandle;
57
58    private TextView mSubjectTextView;
59
60    @Override
61    protected void onCreate(Bundle savedInstanceState) {
62      super.onCreate(savedInstanceState);
63      setContentView(R.layout.activity_modify_course);
64      Bundle bundle = this.getIntent().getExtras();
65
66      brock = new CourseHandler(getApplicationContext()).getCourseOfferings(
67          bundle.getString(CourseManagerActivity.CODE_COURSE_SUBJECT),
68          bundle.getString(CourseManagerActivity.CODE_COURSE_CODE));
69      mCourseHandle = new CurrentCoursesHandler(this.getApplicationContext());
70      course = mCourseHandle.getCourse(bundle.getString(CourseManagerActivity.
        CODE_COURSE_SUBJECT),
```

```
71            bundle.getString(CourseManagerActivity.CODE_COURSE_CODE));
72
73        init();
74      }
75
76      /*
77       * Initialize all views and sets Button OnClickListeners.
78       */
79      private void init() {
80        // Set Buttons
81        mSaveButton = (Button) findViewById(R.id.add_course_save_button);
82        mCancelButton = (Button) findViewById(R.id.add_course_cancel_button);
83
84        // Instantiate TextView
85        mSubjectTextView = (TextView) findViewById(R.id.modify_course_subject_textview);
86
87        // Set TextView
88        mSubjectTextView.setText(course.mSubject+" "+course.mCode);
89
90        mCode = course.mCode;
91        mOfferings = brock.mOfferings;
92
93        String s;
94        mLecs = new ArrayList<String>();
95        mLabs = new ArrayList<String>();
96        mTuts = new ArrayList<String>();
97        mSems = new ArrayList<String>();
98
99        mLecsOfferings = new ArrayList<Offering>();
100       mLabsOfferings = new ArrayList<Offering>();
101       mTutsOfferings = new ArrayList<Offering>();
102       mSemsOfferings = new ArrayList<Offering>();
103
104       for (int i = 0; i < mOfferings.size(); i++) {
105         s = mOfferings.get(i).mType;
106
107         // Some offerings don't have any of what we are looking for,
108         // so check length to make sure.
109         if (s.length() > 2) {
110           String ss = s.substring(0, 3).trim();
111           if (ss.equalsIgnoreCase("lec")) {
112             mLecs.add(s + ", SEC " + mOfferings.get(i).mSection);
113             mLecsOfferings.add(mOfferings.get(i));
114           } else if (ss.equalsIgnoreCase("lab")) {
115             mLabs.add(s + ", SEC " + mOfferings.get(i).mSection);
116             mLabsOfferings.add(mOfferings.get(i));
117           } else if (ss.equalsIgnoreCase("tut")) {
118             mTuts.add(s + ", SEC " + mOfferings.get(i).mSection);
119             mTutsOfferings.add(mOfferings.get(i));
120           } else if (ss.equalsIgnoreCase("sem")) {
121             mSems.add(s + ", SEC " + mOfferings.get(i).mSection);
122             mSemsOfferings.add(mOfferings.get(i));
123           }
124         }
125       }
126
127       // Check if offerings available, and add ListView to display if so.
128       LinearLayout lec_lay = (LinearLayout) findViewById(R.id.layout_add_lecs);
129       LinearLayout lab_lay = (LinearLayout) findViewById(R.id.layout_add_labs);
130       LinearLayout tut_lay = (LinearLayout) findViewById(R.id.layout_add_tuts);
131       LinearLayout sem_lay = (LinearLayout) findViewById(R.id.layout_add_sems);
132
133       if (mLecs.size() > 0) {
134         lec_lay.setVisibility(VISIBLE);
135         mLecsListView = (ListView) findViewById(R.id.add_course_add_lecs);
136         mLecsListView.setAdapter(new ArrayAdapter<String>(
137             getApplicationContext(),
138             android.R.layout.simple_list_item_multiple_choice,
139             mLecs));
140       } else {
141         lec_lay.setVisibility(GONE);
```

```
142       }
143       if (mLabs.size() > 0) {
144         lab_lay.setVisibility(VISIBLE);
145         mLabsListView = (ListView) findViewById(R.id.add_course_add_labs);
146         mLabsListView.setAdapter(new ArrayAdapter<String>(
147             getApplicationContext(),
148             android.R.layout.simple_list_item_multiple_choice,
149             mLabs));
150       } else {
151         lab_lay.setVisibility(GONE);
152       }
153       if (mTuts.size() > 0) {
154         tut_lay.setVisibility(VISIBLE);
155         mTutsListView = (ListView) findViewById(R.id.add_course_add_tuts);
156         mTutsListView.setAdapter(new ArrayAdapter<String>(
157             getApplicationContext(),
158             android.R.layout.simple_list_item_multiple_choice,
159             mTuts));
160       } else {
161         tut_lay.setVisibility(GONE);
162       }
163       if (mSemsOfferings.size() > 0) {
164         sem_lay.setVisibility(VISIBLE);
165         mSemsListView = (ListView) findViewById(R.id.add_course_add_sems);
166         mSemsListView.setAdapter(new ArrayAdapter<String>(
167             getApplicationContext(),
168             android.R.layout.simple_list_item_multiple_choice,
169             mSems));
170       } else {
171         sem_lay.setVisibility(GONE);
172       }
173
174       // Set OnClickListener
175       mSaveButton.setOnClickListener(new OnClickListener() {
176         public void onClick(View v) {
177           Course c = new Course();
178           c.mSubject          = course.mSubject;
179           c.mCode             = course.mCode;
180           c.mDesc             = course.mDesc;
181           c.mInstructor       = course.mInstructor;
182           c.mInstructor_email = course.mInstructor_email;
183           for (Task t : course.mTasks)
184             c.mTasks.add(t);
185
186           for (Offering o : course.mOfferings)
187             mCourseHandle.deleteOffering(o);
188
189           SparseBooleanArray sba1, sba2, sba3, sba4;
190           if (mLecsListView != null) {
191             sba1 = mLecsListView.getCheckedItemPositions();
192             for (int i = 0; i < mLecsListView.getCount(); i++) {
193               if (sba1.get(i) == true) {
194                 c.mOfferings.add(mLecsOfferings.get(i));
195                 Log.d(TAG, "Added: " + mLecsOfferings.get(i).mSubj
196                     + " " + mOfferings.get(i).mCode + ", Type "
197                     + mOfferings.get(i).mType + ", Section "
198                     + mOfferings.get(i).mSection
199                     + " to Offerings");
200               }
201             }
202           }
203
204           if (mLabsListView != null) {
205             sba2 = mLabsListView.getCheckedItemPositions();
206
207             for (int i = 0; i < mLabsListView.getCount(); i++) {
208               if (sba2.get(i) == true) {
209                 c.mOfferings.add(mLabsOfferings.get(i));
210                 Log.d(TAG, "Added: " + mLabsOfferings.get(i).mSubj
211                     + " " + mOfferings.get(i).mCode + ", Type "
212                     + mOfferings.get(i).mType + ", Section "
```

```
213                     + mOfferings.get(i).mSection
214                     + " to Offerings");
215               }
216             }
217           }
218
219           if (mTutsListView != null) {
220             sba3 = mTutsListView.getCheckedItemPositions();
221
222             for (int i = 0; i < mTutsListView.getCount(); i++) {
223               if (sba3.get(i) == true) {
224                 c.mOfferings.add(mTutsOfferings.get(i));
225                 Log.d(TAG, "Added: " + mTutsOfferings.get(i).mSubj
226                     + " " + mOfferings.get(i).mCode + ", Type "
227                     + mOfferings.get(i).mType + ", Section "
228                     + mOfferings.get(i).mSection
229                     + " to Offerings");
230               }
231             }
232           }
233
234           if (mSemsListView != null) {
235             sba4 = mSemsListView.getCheckedItemPositions();
236
237             for (int i = 0; i < mSemsListView.getCount(); i++) {
238               if (sba4.get(i) == true) {
239                 c.mOfferings.add(mSemsOfferings.get(i));
240                 Log.d(TAG, "Added: " + mSemsOfferings.get(i).mSubj
241                     + " " + mOfferings.get(i).mCode + ", Type "
242                     + mOfferings.get(i).mType + ", Section "
243                     + mOfferings.get(i).mSection
244                     + " to Offerings");
245               }
246             }
247           }
248           if (c.mSubject != null) {
249             try {
250               mCourseHandle.addCourse(c);
251               onBackPressed();
252             } catch (Exception e) {
253               // TODO Auto-generated catch block
254               e.printStackTrace();
255             }
256           }
257         }
258       });
259
260       mCancelButton.setOnClickListener(new OnClickListener() {
261         public void onClick(View v) {
262           // Do nothing.
263           onBackPressed();
264         }
265       });
266     }
267   }
268
```

```java
/**
 * Offering.java
 * Brock Butler
 * A wrapper class for Offering information
 * Created by James Grisdale on 2013-02-24
 * Copyright (c) 2013 Sea Addicts. All rights reserved.
**/

package edu.seaaddicts.brockbutler.coursemanager;

import java.util.ArrayList;

public class Offering {
  public int mId;     //offering id
  public String mSubj;//faculty name
  public String mCode;//course code
  public int mSection;//section
  public String mType;//type
  public ArrayList<OfferingTime> mOfferingTimes;//list of times offered
  public int mOid; //extra id if needed

  //constructor - initializes arraylist of offering times
  public Offering(){
    mOfferingTimes = new ArrayList<OfferingTime>();
  }
}
```

```
 1  /**
 2   * OfferingTime.java
 3   * Brock Butler
 4   * A wrapper class for OfferingTime information
 5   * Created by James Grisdale on 2013-02-24
 6   * Copyright (c) 2013 Sea Addicts. All rights reserved.
 7  **/
 8
 9  package edu.seaaddicts.brockbutler.coursemanager;
10
11  public class OfferingTime {
12    public int mOid;  //id associated with offering
13    public String mStartTime; //start time of offering
14    public String mEndTime; //offering end time
15    public String mDay;  //day available
16    public String mLocation; //location
17  }
18
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10sp"
    android:orientation="vertical"
    android:scrollbars="vertical"
    tools:context=".AddTaskActivity" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/add_course_subject" />

        <Spinner
            android:id="@+id/add_course_subjects_spinner"
            android:layout_width="180sp"
            android:layout_height="40sp"
            android:layout_marginLeft="25sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/add_course_code" />

        <Spinner
            android:id="@+id/add_course_codes_spinner"
            android:layout_width="180sp"
            android:layout_height="40sp"
            android:layout_marginLeft="25sp" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/layout_add_lecs"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20sp"
        android:orientation="horizontal"
        android:visibility="gone" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/add_course_add_lecs" />

        <ListView
            android:id="@+id/add_course_add_lecs"
            android:layout_width="fill_parent"
            android:layout_height="100sp"
            android:layout_marginLeft="25sp"
            android:choiceMode="singleChoice" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/layout_add_labs"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20sp"
        android:orientation="horizontal"
        android:visibility="gone" >
```

```
72
73          <TextView
74              android:layout_width="85sp"
75              android:layout_height="wrap_content"
76              android:text="@string/add_course_add_labs" />
77
78          <ListView
79              android:id="@+id/add_course_add_labs"
80              android:layout_width="fill_parent"
81              android:layout_height="100sp"
82              android:layout_marginLeft="25sp"
83              android:choiceMode="singleChoice"
84              android:text="@string/add_course_add_labs" />
85      </LinearLayout>
86
87      <LinearLayout
88          android:id="@+id/layout_add_tuts"
89          android:layout_width="fill_parent"
90          android:layout_height="wrap_content"
91          android:layout_marginTop="20sp"
92          android:orientation="horizontal"
93          android:visibility="gone" >
94
95          <TextView
96              android:layout_width="85sp"
97              android:layout_height="wrap_content"
98              android:text="@string/add_course_add_tuts" />
99
100         <ListView
101             android:id="@+id/add_course_add_tuts"
102             android:layout_width="fill_parent"
103             android:layout_height="100sp"
104             android:layout_marginLeft="25sp"
105             android:choiceMode="singleChoice"
106             android:text="@string/add_course_add_tuts" />
107     </LinearLayout>
108
109     <LinearLayout
110         android:id="@+id/layout_add_sems"
111         android:layout_width="fill_parent"
112         android:layout_height="wrap_content"
113         android:layout_marginTop="20sp"
114         android:orientation="horizontal"
115         android:visibility="gone" >
116
117         <TextView
118             android:layout_width="85sp"
119             android:layout_height="wrap_content"
120             android:text="@string/add_course_add_sems" />
121
122         <ListView
123             android:id="@+id/add_course_add_sems"
124             android:layout_width="fill_parent"
125             android:layout_height="100sp"
126             android:layout_marginLeft="25sp"
127             android:choiceMode="singleChoice"
128             android:text="@string/add_course_add_sems" />
129     </LinearLayout>
130
131     <LinearLayout
132         android:layout_width="fill_parent"
133         android:layout_height="wrap_content"
134         android:gravity="center_horizontal"
135         android:orientation="horizontal"
136         android:layout_marginTop="10sp" >
137
138         <Button
139             android:id="@+id/add_course_save_button"
140             android:layout_width="100sp"
141             android:layout_height="wrap_content"
142             android:text="@string/button_save" />
```

```
143
144             <Button
145                 android:id="@+id/add_course_cancel_button"
146                 android:layout_width="100sp"
147                 android:layout_height="wrap_content"
148                 android:layout_marginLeft="20sp"
149                 android:text="@string/button_cancel" />
150         </LinearLayout>
151
152     </LinearLayout>
```

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/add_course"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:onClick="addCourse"
        android:title="@string/add_course_add"/>

    <item
        android:id="@+id/update_master_list"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:onClick="updateMaster"
        android:title="@string/add_course_update_master_list"/>
    <item
        android:id="@+id/menu_show_coursemanager_help"
         android:orderInCategory="100"
         android:showAsAction="never"
        android:onClick="showHelp"
        android:title="@string/menu_show_coursemanager_help"/>

</menu>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/course_manager_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ListView
        android:id="@+id/course_manager_list"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:visibility="gone"
        android:divider="@drawable/grad_course_title" >
    </ListView>

    <TextView
        android:id="@+id/tv_no_courses"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:text="@string/no_courses"
        android:textColor="#ddd"
        android:textStyle="italic"
        android:textSize="20sp"
        android:gravity="center"
        android:visibility="gone" />
</LinearLayout>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10sp"
    android:orientation="vertical"
    android:scrollbars="vertical"
    tools:context=".ModifyTaskActivity" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <TextView
            android:id="@+id/modify_course_subject_textview"
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/add_course_subject" />

    </LinearLayout>

    <LinearLayout
        android:id="@+id/layout_add_lecs"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20sp"
        android:orientation="horizontal"
        android:visibility="gone" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/add_course_add_lecs" />

        <ListView
            android:id="@+id/add_course_add_lecs"
            android:layout_width="fill_parent"
            android:layout_height="100sp"
            android:layout_marginLeft="25sp"
            android:choiceMode="singleChoice" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/layout_add_labs"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20sp"
        android:orientation="horizontal"
        android:visibility="gone" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/add_course_add_labs" />

        <ListView
            android:id="@+id/add_course_add_labs"
            android:layout_width="fill_parent"
            android:layout_height="100sp"
            android:layout_marginLeft="25sp"
            android:choiceMode="singleChoice"
            android:text="@string/add_course_add_labs" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/layout_add_tuts"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20sp"
        android:orientation="horizontal"
```

```
72          android:visibility="gone" >
73
74          <TextView
75              android:layout_width="85sp"
76              android:layout_height="wrap_content"
77              android:text="@string/add_course_add_tuts" />
78
79          <ListView
80              android:id="@+id/add_course_add_tuts"
81              android:layout_width="fill_parent"
82              android:layout_height="100sp"
83              android:layout_marginLeft="25sp"
84              android:choiceMode="singleChoice"
85              android:text="@string/add_course_add_tuts" />
86      </LinearLayout>
87
88      <LinearLayout
89          android:id="@+id/layout_add_sems"
90          android:layout_width="fill_parent"
91          android:layout_height="wrap_content"
92          android:layout_marginTop="20sp"
93          android:orientation="horizontal"
94          android:visibility="gone" >
95
96          <TextView
97              android:layout_width="85sp"
98              android:layout_height="wrap_content"
99              android:text="@string/add_course_add_sems" />
100
101         <ListView
102             android:id="@+id/add_course_add_sems"
103             android:layout_width="fill_parent"
104             android:layout_height="100sp"
105             android:layout_marginLeft="25sp"
106             android:choiceMode="singleChoice"
107             android:text="@string/add_course_add_sems" />
108     </LinearLayout>
109
110     <LinearLayout
111         android:layout_width="fill_parent"
112         android:layout_height="wrap_content"
113         android:gravity="center_horizontal"
114         android:orientation="horizontal"
115         android:layout_marginTop="10sp" >
116
117         <Button
118             android:id="@+id/add_course_save_button"
119             android:layout_width="100sp"
120             android:layout_height="wrap_content"
121             android:text="@string/button_save" />
122
123         <Button
124             android:id="@+id/add_course_cancel_button"
125             android:layout_width="100sp"
126             android:layout_height="wrap_content"
127             android:layout_marginLeft="20sp"
128             android:text="@string/button_cancel" />
129     </LinearLayout>
130
131 </LinearLayout>
```

```
 1  package edu.seaaddicts.brockbutler.help;
 2
 3  import android.annotation.SuppressLint;
 4  import android.app.Activity;
 5  import android.os.Bundle;
 6  import android.util.DisplayMetrics;
 7  import android.view.Menu;
 8  import android.webkit.WebSettings.PluginState;
 9  import android.webkit.WebView;
10  import edu.seaaddicts.brockbutler.R;
11  public class HelpActivity extends Activity {
12
13    @SuppressLint("SetJavaScriptEnabled")
14    @Override
15    protected void onCreate(Bundle savedInstanceState) {
16      super.onCreate(savedInstanceState);
17      WebView wb = new WebView(this);
18      wb.getSettings().setJavaScriptEnabled(true);
19      wb.getSettings().setPluginState(PluginState.ON);
20      //this does not work contrary to popular belief
21      //wb.getSettings().setBuiltInZoomControls(false);
22      //wb.getSettings().setUseWideViewPort(true);
23      //wb.getSettings().setLoadWithOverviewMode(true);
24      float scaling = 100;
25      int display_width;
26      DisplayMetrics dm = new DisplayMetrics();
27      getWindowManager().getDefaultDisplay().getMetrics(dm);
28      display_width = dm.widthPixels;
29      scaling = (((float)display_width/400)*100); // 400 here is my container div width
30      scaling = (int) Math.floor(scaling);
31      wb.setInitialScale((int)scaling);
32      Bundle  bundle = this.getIntent().getExtras();
33      String actName = bundle.getString("activity");
34      String url;
35      if(actName.equalsIgnoreCase("scheduler"))
36        url ="file:///android_asset/help_schedule.htm";
37      else if(actName.equalsIgnoreCase("coursemanager"))
38        url ="file:///android_asset/help_coursemanager.htm";
39      else if(actName.equalsIgnoreCase("maps"))
40        url ="file:///android_asset/help_gps.htm";
41      else if(actName.equalsIgnoreCase("contacts"))
42        url="file:///android_asset/contacts.htm";
43      else
44        url ="file:///android_asset/helpMain.html";
45
46
47          wb.loadUrl(url);
48          setContentView(wb);
49    }
50
51    @Override
52    public boolean onCreateOptionsMenu(Menu menu) {
53      // Inflate the menu; this adds items to the action bar if it is present.
54      getMenuInflater().inflate(R.menu.activity_help, menu);
55      return true;
56    }
57
58  }
59
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HelpActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/title_activity_help" />

</RelativeLayout>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Help v1.0</title>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  $(".toggle_container").hide();
  $("h2.expand_heading").toggle(function(){
    $(this).addClass("active");
    }, function () {
    $(this).removeClass("active");
  });
  $("h2.expand_heading").click(function(){
    $(this).next(".toggle_container").slideToggle("slow");
  });
  $(".expand_all").toggle(function(){
    $(this).addClass("expanded");
    }, function () {
    $(this).removeClass("expanded");
  });
  $(".expand_all").click(function(){
    $(".toggle_container").slideToggle("slow");
  });
});
</script>
<style type="text/css">
html {
  overflow-Y: scroll;
}
*{
outline:none;
}
body {
  font: 10px normal Georgia,Arial, Helvetica, sans-serif;
  margin: 0;
  padding: 0;
  line-height: 1.7em;
  background:#313131;
}

.wrapper {
  width: 400px;
  margin: 0 auto;
}
.expand_top,.expand_wrapper
{
  width: 400px;
  padding:0px;
  margin:0px 0px 5px 0px;
  float:left;
}
h1 {
  font: 4em normal Georgia, 'Times New Roman', Times, serif;
  text-align:center;
  padding: 20px 0;
  color: #ffffff;
}

h2.expand_heading {
  padding: 0 0 0 20px;
  margin: 0 0 5px 0;
  background: url(expand_collapse.png) no-repeat;
  height: 38px;
  line-height: 38px;
  width: 400px;
  font-size: 2em;
```

```
    font-weight: normal;
    float: left;
  }
  .expand_all
  {
    cursor:default;
  }
  h2.expand_heading a {
    color: #fff;
    text-decoration: none;
    display: block;
  }
  h2.expand_heading a:hover {
    color: #ccc;
  }

  h2.active {background-position: left bottom;}
  .toggle_container {
    margin: 0 0 5px;
    padding: 0;
    border-top: 1px solid #d6d6d6;
    background: #ffffff;
    overflow: hidden;
    font-size: 1.2em;
    width: 400px;
    clear: both;
  }
  .toggle_container .box {
    padding: 20px;
  }
  .toggle_container .box p {
    padding: 5px 0;
    margin: 5px 0;
  }
  .toggle_container h3 {
    font: 2.0em normal Georgia, "Times New Roman", Times, serif;
    margin: 0 0 5px;
    padding: 0 0 5px 0;
    color:#000000;
    border-bottom: 1px dotted #ccc;
  }
  .toggle_container img {
    float: left;
    margin: 10px 15px 15px 0;
    padding: 5px;
    background: #ddd;
    border: 1px solid #ccc;
  }
  .expand_all
  {
  width:116px;
  height:29px;
  background:url(extra_buttons.png) no-repeat top left;
  float:center;
  cursor: pointer;
  }
  .expanded
  {
  background-position:bottom left;
  }
  .footer
  {
  margin:200px 0px 0px 0px;
  text-align:center;
  float:left;
  color:#ffffff;
  font-size:15px;
  overflow:hidden;
  }
```

```
138   .footer a
139   {
140   color:#ffffff;
141   font-weight:bold;
142   font-size:15px;
143   line-height:20px;
144   text-decoration:none;
145   }
146   </style>
147
148
149
150   </head>
151   <body>
152   <h1>BrockButler Help</h1>
153
154   <div class="wrapper">
155   <div class="expand_top"><div class="expand_all"></div></div>
156   <div class="expand_wrapper">
157     <h2 class="expand_heading"><a href="#">Overview</a></h2>
158
159     <div class="toggle_container">
160       <div class="box">
161         <h3>Overview</h3>
162
163         <p>BrockButler is a general purpose Course Management and Navigation System
          for Brock Students that can help you manage your courses, plan your day, give
          you a tour and find your classes.
164         </p>
165       </div>
166     </div>
167
168     <h2 class="expand_heading"><a href="#">Feature List</a></h2>
169
170     <div class="toggle_container">
171       <div class="box">
172         <h3>Feature List</h3>
173
174         <p><ul>
175       <li>Course Management</li>
176       <li>Schedule/Task Management</li>
177       <li>Tour</li>
178       <li>Navigation</li>
179       </ul></p>
180       <p></p>
181       </div>
182
183     </div>
184
185     <h2 class="expand_heading"><a href="#">Help Topics</a></h2>
186     <div class="toggle_container">
187       <div class="box">
188       <h3>Help Topics</h3>
189
190       <p>
191         <ul>
192       <li><a href="file:///android_asset/help_gps.htm">How can I find my classes?
          </a></li>
193       <li><a href="file:///android_asset/help_schedule.htm">How can I plan my day?
          </a></li>
194       <li><a href="file:///android_asset/help_coursemanager.htm">How can I manage my
          courses</a></li>
195
196
197       </ul>
198       </p>
199       </div>
200     </div>
201
```

```
202
203   </div>
204
205
206   </div>
207   </body>
208   </html>
209
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>HTML viewer</title>
<style type="text/css">
html {
  overflow-Y: scroll;
}
*{
outline:none;
}
body {
  font: 10px normal Georgia,Arial, Helvetica, sans-serif;
  margin: 0;
  padding: 0;
  line-height: 1.7em;
  background:#313131;
}

.wrapper {
  width: 400px;
  margin: 0 auto;
}
.expand_top,.expand_wrapper
{
  width: 400px;
  padding:0px;
  margin:0px 0px 5px 0px;
  float:left;
}
h1 {
  font: 4em normal Georgia, 'Times New Roman', Times, serif;
  text-align:center;
  padding: 20px 0;
  color: #ffffff;
}

h2.expand_heading {
  padding: 0 0 0 20px;
  margin: 0 0 5px 0;
  background: url(expand_collapse.png) no-repeat;
  height: 38px;
  line-height: 38px;
  width: 400px;
  font-size: 2em;
  font-weight: normal;
  float: left;
}
.expand_all
{
cursor:default;
}
h2.expand_heading a {
  color: #fff;
  text-decoration: none;
  display: block;
}
h2.expand_heading a:hover {
  color: #ccc;
}

h2.active {background-position: left bottom;}
.toggle_container {
  margin: 0 0 5px;
  padding: 0;
  border-top: 1px solid #d6d6d6;
  background: #ffffff;
  overflow: hidden;
```

```
    font-size: 1.2em;
    width: 400px;
    clear: both;
  }
  .toggle_container .box {
    padding: 20px;
  }
  .toggle_container .box p {
    padding: 5px 0;
    margin: 5px 0;
  }
  .toggle_container h3 {
    font: 2.0em normal Georgia, "Times New Roman", Times, serif;
    margin: 0 0 5px;
    padding: 0 0 5px 0;
    color:#000000;
    border-bottom: 1px dotted #ccc;
  }
  .toggle_container img {
    float: left;
    margin: 10px 15px 15px 0;
    padding: 5px;
    background: #ddd;
    border: 1px solid #ccc;
  }
  .expand_all
  {
  width:116px;
  height:29px;
  background:url(extra_buttons.png) no-repeat top left;
  float:right;
  cursor: pointer;
  }
  .expanded
  {
  background-position:bottom left;
  }
  .footer
  {
  margin:200px 0px 0px 0px;
  text-align:center;
  float:left;
  color:#ffffff;
  font-size:15px;
  overflow:hidden;
  }
  .footer a
  {
  color:#ffffff;
  font-weight:bold;
  font-size:15px;
  line-height:20px;
  text-decoration:none;
  }
  </style>
  </head>
  <body>



    <div class="toggle_container"><h3>How can I manage my courses?</h3>
      <div class="box">


      <p>
      <h5>To add a course</h5>
      Click on the menu and select add a course menu item and select the courses you
      want to add and select the save button.
          </p>
```

```
137       <p>
138       <h5>To modify/remove a course</h5>
139       Press and hold on the course you wish to modify or remove. A menu will appear,
          select the desired option.
140       </p>
141       <p>
142       <h5>View course details</h5>
143       Tap the course name, and the course information will be expanded and displayed.
144       </p>
145     <p>
146       <h5>Update Course Timetable</h5>
147       Click on the menu and select update master list item to update the course
          timetable.
148     </p>
149       </div>
150     </div>
151   </body>
152   </html>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>HTML viewer</title>
<style type="text/css">
html {
  overflow-Y: scroll;
}
*{
outline:none;
}
body {
  font: 10px normal Georgia,Arial, Helvetica, sans-serif;
  margin: 0;
  padding: 0;
  line-height: 1.7em;
  background:#313131;
}

.wrapper {
  width: 400px;
  margin: 0 auto;
}
.expand_top,.expand_wrapper
{
  width: 400px;
  padding:0px;
  margin:0px 0px 5px 0px;
  float:left;
}
h1 {
  font: 4em normal Georgia, 'Times New Roman', Times, serif;
  text-align:center;
  padding: 20px 0;
  color: #ffffff;
}

h2.expand_heading {
  padding: 0 0 0 20px;
  margin: 0 0 5px 0;
  background: url(expand_collapse.png) no-repeat;
  height: 38px;
  line-height: 38px;
  width: 400px;
  font-size: 2em;
  font-weight: normal;
  float: left;
}
.expand_all
{
cursor:default;
}
h2.expand_heading a {
  color: #fff;
  text-decoration: none;
  display: block;
}
h2.expand_heading a:hover {
  color: #ccc;
}

h2.active {background-position: left bottom;}
.toggle_container {
  margin: 0 0 5px;
  padding: 0;
  border-top: 1px solid #d6d6d6;
  background: #ffffff;
  overflow: hidden;
```

```
    font-size: 1.2em;
    width: 400px;
    clear: both;
  }
  .toggle_container .box {
    padding: 20px;
  }
  .toggle_container .box p {
    padding: 5px 0;
    margin: 5px 0;
  }
  .toggle_container h3 {
    font: 2.0em normal Georgia, "Times New Roman", Times, serif;
    margin: 0 0 5px;
    padding: 0 0 5px 0;
    color:#000000;
    border-bottom: 1px dotted #ccc;
  }
  .toggle_container img {
    float: left;
    margin: 10px 15px 15px 0;
    padding: 5px;
    background: #ddd;
    border: 1px solid #ccc;
  }
  .expand_all
  {
  width:116px;
  height:29px;
  background:url(extra_buttons.png) no-repeat top left;
  float:right;
  cursor: pointer;
  }
  .expanded
  {
  background-position:bottom left;
  }
  .footer
  {
  margin:200px 0px 0px 0px;
  text-align:center;
  float:left;
  color:#ffffff;
  font-size:15px;
  overflow:hidden;
  }
  .footer a
  {
  color:#ffffff;
  font-weight:bold;
  font-size:15px;
  line-height:20px;
  text-decoration:none;
  }
  </style>
  </head>
  <body>



    <div class="toggle_container"><h3>How can I find my classes?</h3>
      <div class="box">


      <p>
        <h5>To adjust the zoom level<h5>
        Pinch the screen to zoom in or out.
      </p>
      <p>
```

```
138         <h5>To search for a location<h5>
139         Open the menu and select "Search For Location." Then enter the room number you
            wish to search for. The location
140         will then be displayed on the map in bright blue.
141       </p>
142       <p>
143         <h5>To get directions to a location<h5>
144         Open the menu and select "Get Directions." Enter a room number near you, and
            then enter the
145         location to wish to get directions to. The route will then be displayed on the
            map in bright blue.
146       </p>
147       <p>
148        <h5>To search for location<h5>
149       Click on maps and then click on the menu button and select the search for
          location menu item and then select the search location to be added to the map
150     </p>
151   <!--  <p>
152       <h5>To update position<h5>
153       Click on the maps and then click on the menu button and then select the update
          position
154     </p> -->
155       </div>
156     </div>
157   </body>
158   </html>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>HTML viewer</title>
    <style type="text/css">
html {
  overflow-Y: scroll;
}
*{
outline:none;
}
body {
  font: 10px normal Georgia,Arial, Helvetica, sans-serif;
  margin: 0;
  padding: 0;
  line-height: 1.7em;
  background:#313131;
}

.wrapper {
  width: 400px;
  margin: 0 auto;
}
.expand_top,.expand_wrapper
{
  width: 400px;
  padding:0px;
  margin:0px 0px 5px 0px;
  float:left;
}
h1 {
  font: 4em normal Georgia, 'Times New Roman', Times, serif;
  text-align:center;
  padding: 20px 0;
  color: #ffffff;
}

h2.expand_heading {
  padding: 0 0 0 20px;
  margin: 0 0 5px 0;
  background: url(expand_collapse.png) no-repeat;
  height: 38px;
  line-height: 38px;
  width: 400px;
  font-size: 2em;
  font-weight: normal;
  float: left;
}
.expand_all
{
cursor:default;
}
h2.expand_heading a {
  color: #fff;
  text-decoration: none;
  display: block;
}
h2.expand_heading a:hover {
  color: #ccc;
}

h2.active {background-position: left bottom;}
.toggle_container {
  margin: 0 0 5px;
  padding: 0;
  border-top: 1px solid #d6d6d6;
  background: #ffffff;
  overflow: hidden;
```

```
    font-size: 1.2em;
    width: 400px;
    clear: both;
  }
  .toggle_container .box {
    padding: 20px;
  }
  .toggle_container .box p {
    padding: 5px 0;
    margin: 5px 0;
  }
  .toggle_container h3 {
    font: 2.0em normal Georgia, "Times New Roman", Times, serif;
    margin: 0 0 5px;
    padding: 0 0 5px 0;
    color:#000000;
    border-bottom: 1px dotted #ccc;
  }
  .toggle_container img {
    float: left;
    margin: 10px 15px 15px 0;
    padding: 5px;
    background: #ddd;
    border: 1px solid #ccc;
  }
  .expand_all
  {
  width:116px;
  height:29px;
  background:url(extra_buttons.png) no-repeat top left;
  float:right;
  cursor: pointer;
  }
  .expanded
  {
  background-position:bottom left;
  }
  .footer
  {
  margin:200px 0px 0px 0px;
  text-align:center;
  float:left;
  color:#ffffff;
  font-size:15px;
  overflow:hidden;
  }
  .footer a
  {
  color:#ffffff;
  font-weight:bold;
  font-size:15px;
  line-height:20px;
  text-decoration:none;
  }
  </style>
  </head>
  <body>
    <div class="toggle_container"> <h3>How can I plan my day?</h3>
      <div class="box">

          <p>
              <h5>To add a task</h5>
      Click on the menu button and select Add Task. In the Add Task window, select the
      course you wish to add
      the task to from the drop down menu.

          </p>
          <p>
              <h5>To modify/remove a task</h5>
```

```
137       Expand the course that contains the task you wish to remove. Then click the
          white i to modify, or the red x to
138       delete the task. To mark a task as completed, click the checkbox next to the task.
139           </p>
140     <!--<p>
141       <h5>To see completed task</h5>
142       Click on the scheduler from the main menu and then click on the menu button and
          select show/hide completed tasks
143     </p>
144     <p>
145       <h5>To Expand or Collapse all tasks</h5>
146       Click on the scheduler from the main menu and then click on the menu button to
          expand or collapse all tasks
147     </p>-->
148   <!--  <p>
149       <h5>To add/remove/modify contacts</h5>
150       Click on the scheduler from the main menu and then select a task which expands
          to display the modify tasks where contacts can
151       be changed
152     </p> -->
153     <p>
154       <h5>To set task priority</h5>
155       You can set a task's priority during creation, or you can change its priority by
          modifying the task.
156     </p>
157       </div>
158     </div>
159   </body>
160   </html>
```

```java
/**
 * TourActivity.java
 * Brock Butler
 * Main activity for the tour portion of Brock Butler.
 * Created by Taras Mychaskiw 2013-02-20
 * Copyright (c) 2013 Sea Addicts. All rights reserveR.drawable.
 */
package edu.seaaddicts.brockbutler.tour;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ImageButton;
import android.widget.RelativeLayout;
import android.widget.Toast;
import edu.seaaddicts.brockbutler.R;

public class TourActivity extends Activity {
  private TourNode[] nodes;
  private final int numImages = R.drawable.j328 - R.drawable._a301b + 1;
  private TourInfo info;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_tour);

    RelativeLayout rl = (RelativeLayout)findViewById(R.id.screen);
    ImageButton[] buttons = new ImageButton[5];
    buttons[0] = (ImageButton)findViewById(R.id.outerleft);
    buttons[1] = (ImageButton)findViewById(R.id.innerleft);
    buttons[2] = (ImageButton)findViewById(R.id.center);
    buttons[3] = (ImageButton)findViewById(R.id.innerright);
    buttons[4] = (ImageButton)findViewById(R.id.outerright);
    info = new TourInfo(rl,buttons,getApplicationContext());
    initNodes();
    nodes[idx(R.drawable._a301f)].paint(info);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.activity_tour, menu);
    return true;
  }

  /**
   * Pops the previous TourNode from the stack and goes to that node.
   */
  public void goBack(MenuItem item){
    if (!info.history.empty())
      info.history.pop().paint(info);
  }

  /**
   * Goes to the node in the tour which is logically turning around.
   */
  public void turnAround(MenuItem item){
    if (info.current.canTurnAround()){
      info.history.push(info.current);
      info.current.turnAroundNode.paint(info);
    }
    else
      Toast.makeText(info.context,"No data available",Toast.LENGTH_SHORT).show();
  }

  /**
   * Teleports the user to the specified block.
   */
  public void teleport(MenuItem item){
```

```
72        info.history.push(info.current);
73        switch (item.getItemId()){
74        case R.id.teleport_a_block: nodes[idx(R.drawable._a301f)].paint(info); return;
75        case R.id.teleport_b_block: nodes[idx(R.drawable._b301f)].paint(info); return;
76        case R.id.teleport_c_block: nodes[idx(R.drawable._c301f)].paint(info); return;
77        case R.id.teleport_d_block: nodes[idx(R.drawable._d301b)].paint(info); return;
78        case R.id.teleport_e_block: nodes[idx(R.drawable._e301f)].paint(info); return;
79        case R.id.teleport_f_block: nodes[idx(R.drawable._f301f)].paint(info); return;
80        case R.id.teleport_g_block: nodes[idx(R.drawable._h338f)].paint(info); return;
81        case R.id.teleport_h_block: nodes[idx(R.drawable._f301f)].paint(info); return;
82        case R.id.teleport_j_block: nodes[idx(R.drawable._j301f)].paint(info); return;
83        }
84      }
85
86      /**
87       * Ends the tour and destroys the Activity
88       */
89      public void endTour(MenuItem item){
90        super.onBackPressed();
91      }
92
93      /**
94       * Overrides the back button to pop previous TourNode from the stack
95       * and goes to that node.
96       */
97      @Override
98      public void onBackPressed(){
99        goBack(null);
100     }
101
102     /**
103      * @param r - image resource value
104      * @return index in `nodes` of TourNode which shows the image `r`
105      */
106     private int idx(int r){
107       return(r - R.drawable._a301b);
108     }
109
110     /**
111      * Adds a new TourRoom to nodes.
112      * @param r - the image resource value
113      * @param roomNumber - the title of the room, such as A323
114      */
115     private void room(int r, String roomNumber){
116       nodes[idx(r)] = new TourRoom(r,roomNumber);
117     }
118
119     /**
120      * Adds a new TourHall to nodes.
121      * @param r  - the image resource value
122      * @param ol - image resource value of outer left button
123      * @param ul - resource of inner left
124      * @param c  - resource of center
125      * @param ur - resource of inner right
126      * @param or - resource of outer right
127      * @param ta - resource of turn around node
128      */
129     private void hall(int r, int ll, int ul, int c, int ur, int lr, int ta){
130       if (ta != -1)
131         nodes[idx(r)] = new TourHall(r,
132           (ll == -1)? null : nodes[idx(ll)],
133           (ul == -1)? null : nodes[idx(ul)],
134           (c == -1) ? null : nodes[idx(c)],
135           (ur == -1)? null : nodes[idx(ur)],
136           (lr == -1)? null : nodes[idx(lr)],
137           /*(ta == -1)?*/nodes[idx(ta)]);
138       else
139         nodes[idx(r)] = new TourHall(r,
140           (ll == -1)? null : nodes[idx(ll)],
141           (ul == -1)? null : nodes[idx(ul)],
142           (c == -1) ? null : nodes[idx(c)],
```

```
143           (ur == -1)? null : nodes[idx(ur)],
144           (lr == -1)? null : nodes[idx(lr)]);
145       }
146       private void hall(int r, int c){ hall(r,-1,-1,c,-1,-1,-1); }
147       private void hall(int r, int c, int ta){ hall(r,-1,-1,c,-1,-1,ta); }
148       private void hall (int r, int ll, int ul, int c, int ur, int lr){ hall(r,ll,ul,c,ur
          ,lr,-1); }
149
150       /**
151        * Giant method to link all of the images together. The fun stuff. Right here.
152        */
153       private void initNodes(){
154         nodes = new TourNode[numImages];
155
156         /** A BLOCK ROOMS **/
157         room(R.drawable.a323,"A323");
158
159         /** B BLOCK ROOMS **/
160         room(R.drawable.b303a,"B303A");
161         room(R.drawable.b303b,"B303B");
162         hall(R.drawable.b303,-1,-1,R.drawable.b303b,R.drawable.b303a,-1);
163
164         /** C BLOCK ROOMS **/
165         room(R.drawable.c300,"C300");
166         room(R.drawable.c303,"C303");
167         room(R.drawable.c304,"C304");
168         room(R.drawable.c306_1,"C306");
169         room(R.drawable.c306_2,"C306");
170         hall(R.drawable.c306_3,R.drawable.c306_2,-1,-1,-1,R.drawable.c306_1);
171         room(R.drawable.c308,"C308");
172         room(R.drawable.c310,"C310");
173
174         /** D BLOCK ROOMS **/
175         room(R.drawable.d300,"D300");
176         room(R.drawable.d301,"D301");
177         room(R.drawable.d303,"D303");
178         room(R.drawable.d304,"D304");
179         room(R.drawable.d308,"D308");
180         room(R.drawable.d309_1,"D309");
181         room(R.drawable.d309_2,"D309");
182         room(R.drawable.d310,"D310");
183         room(R.drawable.d314,"D314");
184         room(R.drawable.d315,"D315");
185         room(R.drawable.d316,"D316");
186         room(R.drawable.d317,"D317");
187         room(R.drawable.d318,"D318");
188         room(R.drawable.d319,"D319");
189         room(R.drawable.d350l,"D350L");
190
191         /** E BLOCK ROOMS **/
192         room(R.drawable.e302,"E302");
193         room(R.drawable.e303,"E303");
194         room(R.drawable.e304,"E304");
195
196         /** F BLOCK ROOMS **/
197         /* N/A */
198         /** G BLOCK ROOMS **/
199         room(R.drawable.g301_2,"G301");
200         room(R.drawable.g301_3,"G301");
201         hall(R.drawable.g301_1,R.drawable.g301_2,-1,-1,-1,R.drawable.g301_3);
202         room(R.drawable.g305,"G305");
203
204         /** H BLOCK ROOMS **/
205         room(R.drawable.h300,"H300");
206         room(R.drawable.h302,"H302");
207         room(R.drawable.h303,"H303");
208         room(R.drawable.h304a,"H304A");
209         hall(R.drawable.h304,-1,-1,-1,-1,R.drawable.h304a);
210         hall(R.drawable.h306,/*R.drawable.h309a6*/-1,-1,-1,/*R.drawable.h306a6*/-1,-1);
211         hall(R.drawable.h309,-1,/*R.drawable.h306a9*/-1,-1,-1,/*R.drawable.h309a9*/-1);
212         hall(R.drawable.h306a6,-1,-1,-1,R.drawable.h309,-1);
```

```
213         hall(R.drawable.h306a9,-1,R.drawable.h306,-1,-1,-1,R.drawable.h306a6);
214         hall(R.drawable.h309a6,R.drawable.h309,-1,-1,-1,-1);
215         hall(R.drawable.h309a9,R.drawable.h306,-1,-1,-1,-1,R.drawable.h309a6);
216         nodes[idx(R.drawable.h306)].setOuterLeftNode(nodes[idx(R.drawable.h309a6)]);
217         nodes[idx(R.drawable.h306)].setInnerRightNode(nodes[idx(R.drawable.h306a6)]);
218         nodes[idx(R.drawable.h309)].setInnerLeftNode(nodes[idx(R.drawable.h309a9)]);
219         nodes[idx(R.drawable.h309)].setOuterRightNode(nodes[idx(R.drawable.h306a9)]);
220         room(R.drawable.h310,"H310");
221         room(R.drawable.h313,"H313");
222         room(R.drawable.h315,"H315");
223         room(R.drawable.h316,"H316");
224         room(R.drawable.h317,"H317");
225         room(R.drawable.h318a,"H318A");
226         hall(R.drawable.h318,-1,-1,-1,R.drawable.h318a,-1);
227         room(R.drawable.h320,"H320");
228         room(R.drawable.h321,"H321");
229         room(R.drawable.h322,"H322");
230         room(R.drawable.h323,"H323");
231         room(R.drawable.h324,"H324");
232
233         /** J BLOCK ROOMS **/
234         room(R.drawable.j301,"J301");
235         room(R.drawable.j310,"J310");
236         room(R.drawable.j327,"J327");
237         room(R.drawable.j328,"J328");
238
239         /** J BLOCK FORWARD PASS **/
240         hall(R.drawable._j315f,-1,-1,-1,-1,R.drawable.j301);
241         hall(R.drawable._j314f,R.drawable._j315f);
242         hall(R.drawable._j313f,R.drawable._j314f);
243         hall(R.drawable._j312f,-1,-1,-1,R.drawable._j313f,-1);
244         hall(R.drawable._j311f,R.drawable._j312f);
245         hall(R.drawable._j310f,R.drawable._j311f);
246         hall(R.drawable._j309f,R.drawable._j310f);
247         hall(R.drawable._j308f,R.drawable._j309f);
248         hall(R.drawable._j307f,-1);
249         hall(R.drawable._j306f,R.drawable._j307f,-1,-1,-1,R.drawable._j308f);
250         hall(R.drawable._j305f,R.drawable._j306f);
251         hall(R.drawable._j304f,-1,R.drawable.j327,R.drawable._j305f,R.drawable.j328,-1);
252         hall(R.drawable._j303f,R.drawable._j304f);
253         hall(R.drawable._j302f,-1,R.drawable.j310,-1,R.drawable.j301,-1);
254         hall(R.drawable._j301f,-1,-1,R.drawable._j303f,R.drawable._j302f,-1);
255
256         /** D BLOCK FORWARD PASS **/
257         hall(R.drawable._d343f,/*R.drawable._d304b*/-1,-1,-1,/*R.drawable._j312b*/-1,-1);
258         hall(R.drawable._d342f,R.drawable._d343f);
259         hall(R.drawable._d341f,-1,R.drawable.g310,R.drawable._j301f,R.drawable._d342f,-1);
260         hall(R.drawable._d340f,-1,-1,R.drawable._d341f,-1,/*R.drawable._d307b*/-1);
261         hall(R.drawable._d339f,-1,-1,R.drawable._d340f,/*R.drawable._d307b*/-1,R.drawable
            .d309_2);
262         hall(R.drawable._d338f,R.drawable._d339f);
263         hall(R.drawable._d337f,-1,-1,R.drawable._d338f,R.drawable.d309_1,-1);
264         hall(R.drawable._d336f,R.drawable._d337f);
265         hall(R.drawable._d335f,/*R.drawable._d306b*/-1,-1,-1,-1,R.drawable._d341f);
266         hall(R.drawable._d334f,-1,-1,R.drawable._d335f,-1,R.drawable.d310);
267         hall(R.drawable._d333f,-1,-1,R.drawable._d334f,-1,R.drawable.d314);
268         hall(R.drawable._d332f,R.drawable._d333f);
269         hall(R.drawable._d331f,-1,-1,R.drawable._d332f,-1,R.drawable.d315);
270         hall(R.drawable._d330f,-1,-1,R.drawable._d331f,-1,R.drawable.d316);
271         hall(R.drawable._d329f,-1,-1,R.drawable._d330f,-1,R.drawable.d317);
272         hall(R.drawable._d328f,-1,R.drawable._d329f,R.drawable.d318,-1,-1);
273         hall(R.drawable._d327f,R.drawable._d328f);
274         hall(R.drawable._d326f,R.drawable.d319,-1,R.drawable._d327f,-1,-1);
275         hall(R.drawable._d325f,R.drawable._d326f);
276         hall(R.drawable._d324f,R.drawable._d325f);
277         hall(R.drawable._d323f,R.drawable._d324f);
278         hall(R.drawable._d322f,R.drawable._d323f);
279         hall(R.drawable._d321f,R.drawable._d322f);
280         hall(R.drawable._d320f,R.drawable._d321f);
281         hall(R.drawable._d319f,R.drawable._d320f);
282         hall(R.drawable._d318f,-1,-1,R.drawable._d319f,/*R.drawable._d326b*/-1,-1);
```

```
283         hall(R.drawable._d317f,-1,R.drawable._d336f,R.drawable.d308,R.drawable._d318f,-1);
284         hall(R.drawable._d316f,R.drawable.d304,-1,R.drawable._d317f,-1,-1);
285         hall(R.drawable._d315f,R.drawable.d303,-1,R.drawable._d316f,-1,-1);
286         hall(R.drawable._d314f,R.drawable._d315f);
287         hall(R.drawable._d313f,R.drawable.d301,-1,R.drawable._d314f,-1,-1);
288         hall(R.drawable._d312f,/*R.drawable._d319b*/-1,-1,-1,R.drawable._d319f);
289         hall(R.drawable._d311f,R.drawable._d312f);
290         hall(R.drawable._d310f,-1,R.drawable._d311f,-1,-1,-1);
291         hall(R.drawable._d309f,R.drawable._d310f);
292         hall(R.drawable._d308f,R.drawable._d309f);
293         hall(R.drawable._d307f,R.drawable._d308f);
294         hall(R.drawable._d306f,R.drawable._d307f);
295         hall(R.drawable._d305f,R.drawable._d306f);
296         hall(R.drawable._d304f,-1,R.drawable.d350l,-1,-1,-1);
297         hall(R.drawable._d303f,-1,R.drawable._d305f,-1,R.drawable._d304f,-1);
298         hall(R.drawable._d302f,R.drawable.d300,-1,R.drawable._d313f,-1,R.drawable._d303f);
299         hall(R.drawable._d301f,R.drawable._d302f);
300
301         /** C BLOCK FORWARD PASS **/
302         hall(R.drawable._c312f,-1,-1,-1,R.drawable._d301f,-1);
303         hall(R.drawable._c311f,-1,R.drawable.c310,R.drawable._c312f,-1,-1);
304         hall(R.drawable._c310f,R.drawable._c311f);
305         hall(R.drawable._c309f,R.drawable.c306_3,-1,R.drawable._c310f,-1,-1);
306         hall(R.drawable._c308f,R.drawable._c309f);
307         hall(R.drawable._c307f,-1,R.drawable._c308f,-1,-1,-1);
308         hall(R.drawable._c306f,-1,-1,/*R.drawable._c306b*/-1,R.drawable._c307f,-1);
309         hall(R.drawable._c305f,R.drawable.c304,-1,R.drawable._c306f,-1,-1);
310         hall(R.drawable._c304f,R.drawable.c303,-1,R.drawable._c305f,-1,-1);
311         hall(R.drawable._c303f,R.drawable._c304f);
312         hall(R.drawable._c302f,R.drawable._c303f);
313         hall(R.drawable._c301f,-1,-1,-1,R.drawable._c302f,-1);
314         hall(R.drawable._c316f,-1,R.drawable._c306f,/*R.drawable._c310b*/-1,-1,-1);
315         hall(R.drawable._c315f,R.drawable._c316f);
316         hall(R.drawable._c314f,R.drawable.c308,-1,R.drawable._c315f,-1,-1);
317         hall(R.drawable._c313f,-1,-1,-1,R.drawable._c314f,-1);
318
319         /** F BLOCK FORWARD PASS **/
320         hall(R.drawable._f308f,R.drawable._c313f);
321         hall(R.drawable._f307f,R.drawable._f308f);
322         hall(R.drawable._f306f,R.drawable._f307f);
323         hall(R.drawable._f305f,R.drawable._f306f);
324         hall(R.drawable._f304f,R.drawable._f305f);
325         hall(R.drawable._f303f,R.drawable._f304f);
326         hall(R.drawable._f302f,R.drawable._f303f);
327         hall(R.drawable._f301f,-1,R.drawable._f302f,-1,-1,/*R.drawable._h301b*/-1);
328
329         /** G BLOCK FORWARD PASS **/
330         hall(R.drawable._g313f,-1);
331         hall(R.drawable._g312f,R.drawable._g313f);
332         hall(R.drawable._g311f,R.drawable.g305,-1,R.drawable._g312f,-1,-1);
333         hall(R.drawable._g310f,R.drawable._g311f);
334         hall(R.drawable._g309f,R.drawable._g310f);
335         hall(R.drawable._g308f,R.drawable._g309f);
336         hall(R.drawable._g307f,R.drawable._g308f);
337         hall(R.drawable._g306f,R.drawable._g307f);
338         hall(R.drawable._g305f,R.drawable.g301_1,-1,R.drawable._g306f,-1,-1);
339         hall(R.drawable._g304f,R.drawable._g305f);
340         hall(R.drawable._g303f,R.drawable._g304f);
341         hall(R.drawable._g302f,-1,-1,R.drawable._g303f,-1,-1);
342         hall(R.drawable._g301f,-1,R.drawable._g302f,-1,-1,-1,R.drawable._f301f);
343
344         /** H BLOCK FORWARD PASS **/
345         hall(R.drawable._h338f,-1,R.drawable._g301f,-1,R.drawable._f302f,-1);
346         hall(R.drawable._h337f,R.drawable._h338f);
347         hall(R.drawable._h336f,R.drawable._h337f);
348         hall(R.drawable._h335f,R.drawable._h336f);
349         hall(R.drawable._h334f,R.drawable._h335f);
350         hall(R.drawable._h333f,-1,/*R.drawable._h303b*/-1,R.drawable._h334f,-1,-1);
351         hall(R.drawable._h332f,R.drawable._h333f);
352         hall(R.drawable._h331f,R.drawable._h332f);
353         hall(R.drawable._h330f,R.drawable._h331f);
```

```
354         hall(R.drawable._h329f,-1,R.drawable._h334f,/*R.drawable._h303b*/-1,-1,
            /*R.drawable._h322b*/-1);
355         hall(R.drawable._h328f,-1,R.drawable.h324,R.drawable._h329f,-1,-1);
356         hall(R.drawable._h327f,R.drawable.h323,-1,R.drawable._h328f,-1,-1);
357         hall(R.drawable._h326f,R.drawable._h327f);
358         hall(R.drawable._h325f,R.drawable._h326f);
359         hall(R.drawable._h324f,R.drawable.h320,-1,R.drawable._h325f,-1,-1);
360         hall(R.drawable._h323f,-1,-1,R.drawable._h324f,R.drawable.h322,R.drawable.h321);
361         hall(R.drawable._h322f,R.drawable.h318,-1,R.drawable._h323f,-1,-1);
362         hall(R.drawable._h321f,-1,R.drawable.h316,R.drawable._h322f,-1,R.drawable.h317);
363         hall(R.drawable._h320f,R.drawable._h321f);
364         hall(R.drawable._h319f,-1,-1,-1,-1,R.drawable._h320f);
365         hall(R.drawable._h318f,-1,R.drawable.h315,R.drawable._h319f,-1,-1);
366         hall(R.drawable._h317f,R.drawable.h313,-1,R.drawable._h318f,-1,-1);
367         hall(R.drawable._h316f,R.drawable._h317f);
368         hall(R.drawable._h315f,R.drawable._h316f);
369         hall(R.drawable._h314f,-1);
370         hall(R.drawable._h313f,R.drawable._h314f);
371         hall(R.drawable._h312f,-1,R.drawable.h310,R.drawable._h313f,R.drawable._h315f,-1);
372         hall(R.drawable._h311f,-1,-1,-1,R.drawable.h310,R.drawable._h312f);
373         hall(R.drawable._h310f,-1,-1,R.drawable._h311f,R.drawable.h309,-1);
374         hall(R.drawable._h309f,-1,-1,R.drawable._h310f,-1,R.drawable.h306);
375         hall(R.drawable._h308f,-1,-1,R.drawable._h309f,-1,R.drawable.h304);
376         hall(R.drawable._h307f,R.drawable.h303,-1,R.drawable._h308f,-1,-1);
377         hall(R.drawable._h306f,R.drawable._h307f);
378         hall(R.drawable._h305f,-1,-1,R.drawable._h306f,-1,R.drawable.h302);
379         hall(R.drawable._h304f,-1,R.drawable.h300,R.drawable._h305f,-1,-1);
380         hall(R.drawable._h303f,-1,R.drawable._h304f,-1,R.drawable._h330f,-1);
381         hall(R.drawable._h302f,R.drawable._h303f);
382         hall(R.drawable._h301f,R.drawable._h302f);
383
384         /** E BLOCK FORWARD PASS **/
385         hall(R.drawable._e311f,-1);
386         hall(R.drawable._e310f,-1,R.drawable._h301f,-1,R.drawable._e311f,-1);
387         hall(R.drawable._e309f,R.drawable._e310f);
388         hall(R.drawable._e308f,R.drawable._e309f);
389         hall(R.drawable._e307f,-1,-1,R.drawable._e308f,-1,-1);
390         hall(R.drawable._e306f,R.drawable.e304,-1,R.drawable._e307f,-1,-1);
391         hall(R.drawable._e305f,R.drawable.e303,-1,R.drawable._e306f,-1,-1);
392         hall(R.drawable._e304f,R.drawable._e305f);
393         hall(R.drawable._e303f,-1,R.drawable.e302,R.drawable._e304f,-1,-1);
394         hall(R.drawable._e302f,R.drawable._e303f);
395         hall(R.drawable._e301f,-1,-1,-1,R.drawable._e302f,-1);
396
397         /** B BLOCK FORWARD PASS **/
398         hall(R.drawable._b314f,R.drawable._c301f);
399         hall(R.drawable._b313f,R.drawable._b314f);
400         hall(R.drawable._b312f,R.drawable._b314f);
401         hall(R.drawable._b311f,-1,-1,-1,R.drawable._b312f,-1);
402         hall(R.drawable._b310f,-1,R.drawable._b311f,R.drawable._b313f,-1,-1);
403         hall(R.drawable._b309f,R.drawable._b310f);
404         hall(R.drawable._b308f,-1,R.drawable._b309f,-1,-1,-1);
405         hall(R.drawable._b307f,R.drawable._e301f);
406         hall(R.drawable._b306f,R.drawable._b307f);
407         hall(R.drawable._b305f,R.drawable._b306f);
408         hall(R.drawable._b304f,-1,-1,R.drawable._b305f,R.drawable.b303,-1);
409         hall(R.drawable._b303f,R.drawable._b304f);
410         hall(R.drawable._b302f,-1,R.drawable._b303f,-1,R.drawable._b308f,-1);
411         hall(R.drawable._b301f,-1,-1,-1,R.drawable._b302f,-1);
412
413         /** A BLOCK FORWARD PASS **/
414         hall(R.drawable._a309f,-1,R.drawable._b301f,-1,-1,-1);
415         hall(R.drawable._a308f,R.drawable._a309f);
416         hall(R.drawable._a307f,R.drawable._a308f);
417         hall(R.drawable._a306f,R.drawable._a307f);
418         hall(R.drawable._a305f,-1,-1,R.drawable._a306f,-1,R.drawable.a323);
419         hall(R.drawable._a304f,R.drawable._a305f);
420         hall(R.drawable._a303f,R.drawable._a304f);
421         hall(R.drawable._a302f,R.drawable._a303f);
422         hall(R.drawable._a301f,R.drawable._a302f);
423
```

```
424         /** A BLOCK BACK PASS **/
425         hall(R.drawable._a307b,-1,R.drawable._a302f);
426         hall(R.drawable._a306b,R.drawable._a307b,R.drawable._a303f);
427         hall(R.drawable._a305b,R.drawable._a306b,R.drawable._a304f);
428         hall(R.drawable._a304b,-1,R.drawable.a323,R.drawable._a305b,-1,-1,R.drawable.
            _a307f);
429         hall(R.drawable._a303b,R.drawable._a304b,R.drawable._a308f);
430         hall(R.drawable._a302b,-1,-1,-1,R.drawable._a303b,-1,R.drawable._a309f);
431         hall(R.drawable._a301b,R.drawable._a302b,R.drawable._b301f);
432
433         /** B BLOCK BACK PASS **/
434         hall(R.drawable._b317b,R.drawable._a301b,R.drawable._b301f);
435         hall(R.drawable._b316b,-1,R.drawable._b317b,-1,R.drawable._b303f,-1,R.drawable.
            _b302f);
436         hall(R.drawable._b313b,-1,-1,-1,R.drawable._b316b,-1,R.drawable._b309f);
437         hall(R.drawable._b312b,R.drawable._b313b,R.drawable._b313f);
438         hall(R.drawable._b311b,R.drawable._b313b,R.drawable._b311f);
439         hall(R.drawable._b310b,R.drawable._b311b,R.drawable._b312f);
440         hall(R.drawable._b309b,-1,-1,R.drawable._b312b,R.drawable._b310b,-1,R.drawable.
            _b314f);
441         hall(R.drawable._b308b,-1,R.drawable._b309b,-1,-1,-1,R.drawable._b314f);
442         hall(R.drawable._b306b,-1,-1,-1,R.drawable._b308b,-1,R.drawable._c301f);
443         hall(R.drawable._b305b,-1,R.drawable._b308f,-1,R.drawable._b317b,-1,R.drawable.
            _b303f);
444         hall(R.drawable._b304b,R.drawable._b305b,R.drawable._b304f);
445         hall(R.drawable._b303b,R.drawable.b303,-1,R.drawable._b304b,-1,-1,R.drawable.
            _b305f);
446         hall(R.drawable._b302b,R.drawable._b303b,R.drawable._b306f);
447         hall(R.drawable._b301b,R.drawable._b302b,R.drawable._e301f);
448
449         /** E BLOCK BACK PASS **/
450         hall(R.drawable._e309b,-1,-1,R.drawable._b301b,-1,R.drawable.e302,R.drawable.
            _e303f);
451         hall(R.drawable._e308b,R.drawable._e309b,R.drawable._e304f);
452         hall(R.drawable._e307b,-1,-1,R.drawable._e308b,-1,R.drawable.e303,R.drawable.
            _e305f);
453         hall(R.drawable._e306b,-1,-1,R.drawable._e307b,-1,R.drawable.e304,R.drawable.
            _e306f);
454         hall(R.drawable._e305b,R.drawable._e306b,R.drawable._e307f);
455         hall(R.drawable._e304b,R.drawable._e305b,R.drawable._e308f);
456         hall(R.drawable._e303b,R.drawable._e304b,R.drawable._e309f);
457         hall(R.drawable._e302b,R.drawable._e303b,R.drawable._e311f);
458         hall(R.drawable._e301b,R.drawable._e311f,-1,R.drawable._e303b,-1,-1,R.drawable.
            _h301f);
459
460         /**  H BLOCK BACK PASS **/
461         hall(R.drawable._h326b,R.drawable._e301b,R.drawable._h302f);
462         hall(R.drawable._h325b,-1,-1,R.drawable._h326b,-1,R.drawable._h304f,R.drawable.
            _h330f);
463         hall(R.drawable._h324b,R.drawable._h325b,R.drawable._h331f);
464         hall(R.drawable._h323b,R.drawable._h324b,R.drawable._h332f);
465         hall(R.drawable._h322b,R.drawable._h323b,R.drawable._h333f);
466         hall(R.drawable._h321b,R.drawable._h330f,-1,-1,-1,R.drawable._h326b,R.drawable.
            _h304f);
467         hall(R.drawable._h320b,R.drawable.h302,-1,R.drawable._h321b,R.drawable.h300,-1,R.
            drawable._h305f);
468         hall(R.drawable._h319b,-1,-1,R.drawable._h320b,-1,R.drawable.h303,R.drawable.
            _h307f);
469         hall(R.drawable._h318b,-1,R.drawable.h304,R.drawable._h319b,-1,-1,R.drawable.
            _h308f);
470         hall(R.drawable._h317b,-1,R.drawable.h306,R.drawable._h318b,-1,-1,R.drawable.
            _h309f);
471         hall(R.drawable._h316b,-1,R.drawable.h309,R.drawable._h317b,-1,-1,R.drawable.
            _h311f);
472         hall(R.drawable._h315b,-1,R.drawable._h316b,-1,R.drawable.h310,R.drawable._h313f,
            R.drawable._h315f);
473         hall(R.drawable._h314b,R.drawable._h315b,R.drawable._h316f);
474         hall(R.drawable._h313b,-1,-1,R.drawable._h314b,R.drawable.h313,-1,R.drawable.
            _h317f);
475         hall(R.drawable._h312b,-1,-1,R.drawable._h313b,R.drawable.h315,-1,R.drawable.
            _h319f);
```

```
476         hall(R.drawable._h311b,-1,R.drawable.h317,R.drawable._h312b,R.drawable.h316,R.
            drawable.h318,R.drawable._h322f);
477         hall(R.drawable._h310b,R.drawable._h311b,R.drawable._h323f);
478         hall(R.drawable._h309b,-1,-1,R.drawable._h310b,-1,R.drawable.h320,R.drawable.
            _h324f);
479         hall(R.drawable._h308b,R.drawable._h309b,R.drawable._h325f);
480         hall(R.drawable._h307b,R.drawable.h322,R.drawable.h321,R.drawable._h308b,-1,-1,R.
            drawable._h326f);
481         hall(R.drawable._h306b,-1,-1,R.drawable._h307b,R.drawable.h323,R.drawable.h324,R.
            drawable._h327f);
482         hall(R.drawable._h305b,R.drawable._h306b,R.drawable._h328f);
483         hall(R.drawable._h304b,R.drawable._h305b,R.drawable._h329f);
484         hall(R.drawable._h303b,-1,R.drawable._h322b,-1,R.drawable._h304b,-1,R.drawable.
            _h334f);
485         hall(R.drawable._h302b,R.drawable._h303b,R.drawable._h335f);
486         hall(R.drawable._h301b,R.drawable._h302b,R.drawable._h337f);
487
488         /** G BLOCK BACK PASS **/
489         hall(R.drawable._g309b,-1,-1,-1,R.drawable._f301f,-1,R.drawable._g302f);
490         hall(R.drawable._g308b,-1,R.drawable._g309b,-1,-1,-1,R.drawable._g303f);
491         hall(R.drawable._g307b,R.drawable._g308b,R.drawable._g304f);
492         hall(R.drawable._g306b,-1,-1,R.drawable._g307b,-1,R.drawable.g301_1,R.drawable.
            _g305f);
493         hall(R.drawable._g305b,R.drawable._g306b,R.drawable._g306f);
494         hall(R.drawable._g304b,R.drawable._g305b,R.drawable._g309f);
495         hall(R.drawable._g303b,R.drawable._g304b,R.drawable._g310f);
496         hall(R.drawable._g302b,-1,-1,R.drawable._g303b,-1,R.drawable.g305,R.drawable.
            _g311f);
497         hall(R.drawable._g301b,-1,-1,R.drawable._g302b,R.drawable._g313f,-1,R.drawable.
            _g312f);
498
499         /** F BLOCK BACK PASS **/
500         hall(R.drawable._f305b,-1,-1,R.drawable._f301f,R.drawable._g301f,-1,R.drawable.
            _f303f);
501         hall(R.drawable._f304b,R.drawable._f305b,R.drawable._f304f);
502         hall(R.drawable._f303b,R.drawable._f304b,R.drawable._f305f);
503         hall(R.drawable._f302b,R.drawable._f303b,R.drawable._f306f);
504         hall(R.drawable._f301b,-1,-1,-1,R.drawable._f302b,-1,R.drawable._f308f);
505
506         /** C BLOCK BACK PASS **/
507         hall(R.drawable._c315b,R.drawable._b306b,R.drawable._c302f);
508         hall(R.drawable._c314b,-1,-1,R.drawable._c315b,-1,R.drawable.c300,R.drawable.
            _c303f);
509         hall(R.drawable._c313b,R.drawable._c314b,R.drawable._c304f);
510         hall(R.drawable._c312b,-1,-1,R.drawable._c313b,-1,R.drawable.c303,R.drawable.
            _c305f);
511         hall(R.drawable._c311b,-1,-1,R.drawable._c312b,-1,R.drawable.c304,R.drawable.
            _c306f);
512         hall(R.drawable._c310b,R.drawable._c311b,R.drawable._c306f);
513         hall(R.drawable._c309b,-1,R.drawable._f301b,-1,-1,-1,R.drawable._c313f);
514         hall(R.drawable._c308b,-1,-1,R.drawable._c309b,-1,R.drawable.c308,R.drawable.
            _c314f);
515         hall(R.drawable._c307b,R.drawable._c308b,R.drawable._c315f);
516         hall(R.drawable._c306b,R.drawable._c307b,R.drawable._c316f);
517         hall(R.drawable._c305b,-1,-1,-1,R.drawable._c310b,R.drawable._c306b,R.drawable.
            _c308f);
518         hall(R.drawable._c304b,-1,-1,R.drawable._c305b,-1,R.drawable.c306_3,R.drawable.
            _c309f);
519         hall(R.drawable._c303b,R.drawable._c304b,R.drawable._c310f);
520         hall(R.drawable._c302b,-1,-1,-1,R.drawable._c303b,-1,R.drawable._c311f);
521         hall(R.drawable._c301b,-1,R.drawable._c302b,R.drawable.c310,-1,-1,R.drawable.
            _c312f);
522
523         /** D BLOCK BACK PASS **/
524         hall(R.drawable._d334b,R.drawable._c301b,R.drawable._d301f);
525         hall(R.drawable._d333b,R.drawable._d334b,-1,R.drawable.d300,-1,R.drawable._d313f,
            R.drawable._d303f);
526         hall(R.drawable._d332b,R.drawable._d333b,R.drawable._d303f);
527         hall(R.drawable._d331b,-1,-1,R.drawable._d304f,R.drawable._d332b,-1,R.drawable.
            _d305f);
528         hall(R.drawable._d330b,R.drawable._d331b,R.drawable._d306f);
```

```
529         hall(R.drawable._d329b,R.drawable._d330b,R.drawable._d307f);
530         hall(R.drawable._d328b,R.drawable._d329b,R.drawable._d308f);
531         hall(R.drawable._d327b,R.drawable._d328b,R.drawable._d309f);
532         hall(R.drawable._d326b,-1,-1,-1,R.drawable._d327b,-1,R.drawable._d310f);
533         hall(R.drawable._d325b,-1,R.drawable._d303f,R.drawable._d334b,R.drawable.d300,R.
            drawable.d301,R.drawable._d314f);
534         hall(R.drawable._d324b,-1,-1,R.drawable._d325b,-1,R.drawable.d303,R.drawable.
            _d315f);
535         hall(R.drawable._d323b,-1,-1,R.drawable._d324b,-1,R.drawable.d304,R.drawable.
            _d317f);
536         hall(R.drawable._d322b,R.drawable.d308,R.drawable._d318f,R.drawable._d323b,-1,-1,
            R.drawable._d337f);
537         hall(R.drawable._d321b,R.drawable.d309_1,-1,R.drawable._d322b,-1,-1,R.drawable.
            _d338f);
538         hall(R.drawable._d320b,R.drawable.d309_2,-1,R.drawable._d321b,-1,-1,R.drawable.
            _d340f);
539         hall(R.drawable._d319b,-1,R.drawable._d326b,R.drawable._d323b,R.drawable._d336f,-
            1,R.drawable._d319f);
540         hall(R.drawable._d318b,R.drawable._d319b,R.drawable._d320f);
541         hall(R.drawable._d317b,R.drawable._d318b,R.drawable._d321f);
542         hall(R.drawable._d316b,R.drawable._d317b,R.drawable._d322f);
543         hall(R.drawable._d315b,R.drawable._d316b,R.drawable._d323f);
544         hall(R.drawable._d314b,R.drawable._d315b,R.drawable._d323f);
545         hall(R.drawable._d313b,-1,-1,R.drawable._d314b,-1,R.drawable.d319,R.drawable.
            _d325f);
546         hall(R.drawable._d312b,R.drawable._d313b,R.drawable._d327f);
547         hall(R.drawable._d311b,R.drawable.d317,R.drawable.d318,-1,R.drawable._d312b,-1,R.
            drawable._d329f);
548         hall(R.drawable._d310b,R.drawable.d316,-1,R.drawable._d311b,-1,-1,R.drawable.
            _d330f);
549         hall(R.drawable._d309b,R.drawable.d315,-1,R.drawable._d310b,-1,-1,R.drawable.
            _d331f);
550         hall(R.drawable._d308b,R.drawable.d314,-1,R.drawable._d309b,-1,-1,R.drawable.
            _d333f);
551         hall(R.drawable._d307b,R.drawable.d310,-1,R.drawable._d308b,-1,-1,R.drawable.
            _d334f);
552         hall(R.drawable._d306b,R.drawable._d307b,-1,R.drawable._d320b,-1,-1,R.drawable.
            _d340f);
553         hall(R.drawable._d305b,-1,R.drawable._d307b,R.drawable._d306b,-1,-1,R.drawable.
            _d341f);
554         hall(R.drawable._d304b,-1,R.drawable._d305b,-1,R.drawable._d341f,-1,R.drawable.
            _d342f);
555         hall(R.drawable._d303b,R.drawable._d343f,R.drawable._d343f,-1,-1,-1);
556         hall(R.drawable._d302b,R.drawable._d342f,R.drawable._d306b,-1,-1,R.drawable.g310,
            R.drawable._j301f);
557         hall(R.drawable._d301b,R.drawable._d302b,R.drawable._j301f);
558
559         /** J BLOCK BACK PASS **/
560         hall(R.drawable._j312b,-1,R.drawable._d303b,R.drawable._d301b,R.drawable._j303f,-
            1,R.drawable._j302f);
561         hall(R.drawable._j311b,-1,R.drawable._j302f,-1,R.drawable._d301b,-1,R.drawable.
            _j304f);
562         hall(R.drawable._j310b,-1,R.drawable.j328,R.drawable._j311b,-1,R.drawable.j327,R.
            drawable._j305f);
563         hall(R.drawable._j309b,R.drawable._j310b,R.drawable._j306f);
564         hall(R.drawable._j308b,-1,-1,R.drawable._j308f,R.drawable._j309b,-1,R.drawable.
            _j307f);
565         hall(R.drawable._j307b,-1,R.drawable._j309b,R.drawable._j307f,-1,-1,R.drawable.
            _j308f);
566         hall(R.drawable._j306b,R.drawable._j307b,R.drawable._j309f);
567         hall(R.drawable._j305b,-1,R.drawable._j306b,-1,-1,-1,R.drawable._j312f);
568         hall(R.drawable._j304b,R.drawable._j305b,R.drawable._j313f);
569         hall(R.drawable._j303b,R.drawable.j310,-1,R.drawable._j304b,-1,-1,R.drawable.
            _j313f);
570         hall(R.drawable._j302b,R.drawable._j303b,R.drawable._j314f);
571         hall(R.drawable._j301b,R.drawable._j302b,R.drawable._j315f);
572
573         /** BUTTON FIXES **/
574
            //hall(R.drawable._d343f,/*R.drawable._d304b*/-1,-1,-1,/*R.drawable._j312b*/-1,-1)
            ;
```

```
575         nodes[idx(R.drawable._d343f)].setOuterLeftNode(nodes[idx(R.drawable._d304b)]);
576         nodes[idx(R.drawable._d343f)].setInnerRightNode(nodes[idx(R.drawable._j312b)]);
577         //hall(R.drawable._d340f,-1,-1,R.drawable._d341f,-1,/*R.drawable._d307b*/-1);
578         nodes[idx(R.drawable._d340f)].setOuterRightNode(nodes[idx(R.drawable._d307b)]);
579
            //hall(R.drawable._d339f,-1,-1,R.drawable._d340f,/*R.drawable._d307b*/-1,R.drawabl
            e.d309_2);
580         nodes[idx(R.drawable._d339f)].setInnerRightNode(nodes[idx(R.drawable._d307b)]);
581         //hall(R.drawable._d335f,/*R.drawable._d306b*/-1,-1,-1,-1,R.drawable._d341f);
582         nodes[idx(R.drawable._d335f)].setOuterLeftNode(nodes[idx(R.drawable._d306b)]);
583         //hall(R.drawable._d318f,-1,-1,R.drawable._d319f,/*R.drawable._d326b*/-1,-1);
584         nodes[idx(R.drawable._d318f)].setInnerRightNode(nodes[idx(R.drawable._d326b)]);
585         //hall(R.drawable._d312f,/*R.drawable._d319b*/-1,-1,-1,-1,R.drawable._d319f);
586         nodes[idx(R.drawable._d312f)].setOuterLeftNode(nodes[idx(R.drawable._d319b)]);
587         //hall(R.drawable._c306f,-1,-1,/*R.drawable._c306b*/-1,R.drawable._c307f,-1);
588         nodes[idx(R.drawable._c306f)].setCenterNode(nodes[idx(R.drawable._c306b)]);
589         //hall(R.drawable._c316f,-1,R.drawable._c306f,/*R.drawable._c310b*/-1,-1,-1);
590         nodes[idx(R.drawable._c316f)].setCenterNode(nodes[idx(R.drawable._c310b)]);
591         //hall(R.drawable._f301f,-1,R.drawable._f302f,-1,-1,/*R.drawable._h301b*/-1);
592         nodes[idx(R.drawable._f301f)].setInnerRightNode(nodes[idx(R.drawable._h301b)]);
593         //hall(R.drawable._h333f,-1,/*R.drawable._h303b*/-1,R.drawable._h334f,-1,-1);
594         nodes[idx(R.drawable._h333f)].setInnerLeftNode(nodes[idx(R.drawable._h303b)]);
595
            //hall(R.drawable._h329f,-1,R.drawable._h334f,/*R.drawable._h303b*/-1,-1,/*R.drawa
            ble._h322b*/-1);
596         nodes[idx(R.drawable._h329f)].setCenterNode(nodes[idx(R.drawable._h303b)]);
597         nodes[idx(R.drawable._h329f)].setOuterRightNode(nodes[idx(R.drawable._h322b)]);
598
599         /** TURN AROUND FIXES **/
600         nodes[idx(R.drawable._a301f)].setTurnAroundNode(nodes[idx(R.drawable._a307b)]);
601         nodes[idx(R.drawable._a305f)].setTurnAroundNode(nodes[idx(R.drawable._a305b)]);
602         nodes[idx(R.drawable._a306f)].setTurnAroundNode(nodes[idx(R.drawable._a305b)]);
603
604         nodes[idx(R.drawable._b307f)].setTurnAroundNode(nodes[idx(R.drawable._b302b)]);
605         nodes[idx(R.drawable._b308f)].setTurnAroundNode(nodes[idx(R.drawable._b317b)]);
606         nodes[idx(R.drawable._b310f)].setTurnAroundNode(nodes[idx(R.drawable._b313b)]);
607
608         nodes[idx(R.drawable._c307f)].setTurnAroundNode(nodes[idx(R.drawable._c310b)]);
609
610         nodes[idx(R.drawable._d302f)].setTurnAroundNode(nodes[idx(R.drawable._d334b)]);
611         nodes[idx(R.drawable._d311f)].setTurnAroundNode(nodes[idx(R.drawable._d326b)]);
612         nodes[idx(R.drawable._d312f)].setTurnAroundNode(nodes[idx(R.drawable._d326b)]);
613         nodes[idx(R.drawable._d313f)].setTurnAroundNode(nodes[idx(R.drawable._d325b)]);
614         nodes[idx(R.drawable._d316f)].setTurnAroundNode(nodes[idx(R.drawable._d324b)]);
615         nodes[idx(R.drawable._d318f)].setTurnAroundNode(nodes[idx(R.drawable._d319b)]);
616         nodes[idx(R.drawable._d324f)].setTurnAroundNode(nodes[idx(R.drawable._d313b)]);
617         nodes[idx(R.drawable._d326f)].setTurnAroundNode(nodes[idx(R.drawable._d313b)]);
618         nodes[idx(R.drawable._d328f)].setTurnAroundNode(nodes[idx(R.drawable._d312b)]);
619         nodes[idx(R.drawable._d332f)].setTurnAroundNode(nodes[idx(R.drawable._d309b)]);
620         nodes[idx(R.drawable._d335f)].setTurnAroundNode(nodes[idx(R.drawable._d307b)]);
621         nodes[idx(R.drawable._d336f)].setTurnAroundNode(nodes[idx(R.drawable._d322b)]);
622         nodes[idx(R.drawable._d339f)].setTurnAroundNode(nodes[idx(R.drawable._d306b)]);
623
624         nodes[idx(R.drawable._e302f)].setTurnAroundNode(nodes[idx(R.drawable._e309b)]);
625         nodes[idx(R.drawable._e310f)].setTurnAroundNode(nodes[idx(R.drawable._e303b)]);
626
627         nodes[idx(R.drawable._f302f)].setTurnAroundNode(nodes[idx(R.drawable._f305b)]);
628         nodes[idx(R.drawable._f307f)].setTurnAroundNode(nodes[idx(R.drawable._f302b)]);
629
630         nodes[idx(R.drawable._g307f)].setTurnAroundNode(nodes[idx(R.drawable._g305b)]);
631         nodes[idx(R.drawable._g308f)].setTurnAroundNode(nodes[idx(R.drawable._g304b)]);
632         nodes[idx(R.drawable._g313f)].setTurnAroundNode(nodes[idx(R.drawable._g301b)]);
633
634         nodes[idx(R.drawable._h303f)].setTurnAroundNode(nodes[idx(R.drawable._h326b)]);
635         nodes[idx(R.drawable._h306f)].setTurnAroundNode(nodes[idx(R.drawable._h320b)]);
636         nodes[idx(R.drawable._h310f)].setTurnAroundNode(nodes[idx(R.drawable._h317b)]);
637         nodes[idx(R.drawable._h312f)].setTurnAroundNode(nodes[idx(R.drawable._h316b)]);
638         nodes[idx(R.drawable._h318f)].setTurnAroundNode(nodes[idx(R.drawable._h313b)]);
639         nodes[idx(R.drawable._h320f)].setTurnAroundNode(nodes[idx(R.drawable._h311b)]);
640         nodes[idx(R.drawable._h321f)].setTurnAroundNode(nodes[idx(R.drawable._h311b)]);
641         nodes[idx(R.drawable._h336f)].setTurnAroundNode(nodes[idx(R.drawable._h302b)]);
```

```
642         nodes[idx(R.drawable._h338f)].setTurnAroundNode(nodes[idx(R.drawable._h301b)]);
643
644         nodes[idx(R.drawable._j303f)].setTurnAroundNode(nodes[idx(R.drawable._j311b)]);
645         nodes[idx(R.drawable._j310f)].setTurnAroundNode(nodes[idx(R.drawable._j306b)]);
646         nodes[idx(R.drawable._j311f)].setTurnAroundNode(nodes[idx(R.drawable._j305b)]);
647       }
648     }
649
```

```
 1  /**
 2   * TourHall.java
 3   * Brock Butler
 4   * A hallway node in the tour. Hallways have multiple branching points,
 5   * and can turn around. Each hallway knows about all of it's branching points.
 6   * Created by Taras Mychaskiw 2013-02-20
 7   * Copyright (c) 2013 Sea Addicts. All rights reserved.
 8   */
 9  package edu.seaaddicts.brockbutler.tour;
10
11  import android.view.View;
12  import android.view.View.OnClickListener;
13  import android.widget.Toast;
14  import edu.seaaddicts.brockbutler.R;
15
16  public class TourHall extends TourNode {
17    protected TourNode[] nodes; //each of the nodes this node branches off to
18
19    /**
20     * Forward pass constructor. Defines each button, and inits the turn around
21     * location to null.
22     * @param img - the image resource value
23     * @param ll - outer left node
24     * @param ul - upper left node
25     * @param c - center node
26     * @param ur - upper right node
27     * @param lr - outer right node
28     */
29    public TourHall(int img, TourNode ll, TourNode ul, TourNode c, TourNode ur,
      TourNode lr){
30      image = img;
31      nodes = new TourNode[5];
32      nodes[0] = ll;
33      nodes[1] = ul;
34      nodes[2] = c;
35      nodes[3] = ur;
36      nodes[4] = lr;
37      turnAroundNode = null;
38      title = makeTitle(img);
39    }
40
41    /**
42     * Second pass constructor. Defines each button, and also defines the turn
43     * around node. Links `turnAroundNode` back to this node via turning around.
44     * @param ta - turn around node
45     */
46    public TourHall(int img, TourNode ll, TourNode ul, TourNode c, TourNode ur,
      TourNode lr, TourNode ta){
47      this(img,ll,ul,c,ur,lr);
48      turnAroundNode = ta;
49      turnAroundNode.setTurnAroundNode(this);
50    }
51
52    public void setOuterLeftNode(TourNode node){  nodes[0] = node; }
53    public void setInnerLeftNode(TourNode node){  nodes[1] = node; }
54    public void setCenterNode(TourNode node){     nodes[2] = node; }
55    public void setInnerRightNode(TourNode node){ nodes[3] = node; }
56    public void setOuterRightNode(TourNode node){ nodes[4] = node; }
57
58    /**
59     * @param img - the resource value of the image of this node
60     * @return - the value for `title`
61     */
62    private String makeTitle(int img){
63      switch (img){
64      case R.drawable.b303: return("B303");
65      case R.drawable.c306_3: return("C306");
66      case R.drawable.g301_1: return("G301");
67      case R.drawable.h304: return("H304");
68      case R.drawable.h306: return("H306");
69      case R.drawable.h306a6: case R.drawable.h306a9: return("H306A");
```

```
70        case R.drawable.h309: return("H309");
71        case R.drawable.h309a6: case R.drawable.h309a9: return("H309A");
72        case R.drawable.h318: return("H318");
73        case R.drawable._a301f: case R.drawable._a301b:
74          return("A Block");
75        case R.drawable._b301f: case R.drawable._b301b: case R.drawable._b306b:
76          return("B Block");
77        case R.drawable._c301f: case R.drawable._c301b: case R.drawable._c313f:
78          return("C Block");
79        case R.drawable._d301f: case R.drawable._d301b: case R.drawable._d304b:
80          return("D Block");
81        case R.drawable._e301f: case R.drawable._e301b:
82          return("E Block");
83        case R.drawable._f301b: case R.drawable._f302f:
84          return("F Block");
85        case R.drawable._g301f:
86          return("G Block");
87        case R.drawable._h301f: case R.drawable._h301b:
88          return("H Block");
89        case R.drawable._j301f: case R.drawable._j312b:
90          return("J Block");
91        }
92        return(null);
93      }
94
95      /**
96       * Changes the image displayed on the screen and redefines where the buttons lead
          us to.
97       * Also pushes this node onto the TourInfo's history.
98       */
99      @Override
100     public void paint(final TourInfo info){
101       info.rl.setBackgroundResource(image);
102       info.current = this;
103       if (title != null)
104         Toast.makeText(info.context,title,Toast.LENGTH_SHORT).show();
105       for (int i = 0; i < nodes.length; i++){
106         final int idx = i;
107         info.buttons[idx].setOnClickListener(new OnClickListener(){
108           public void onClick(View v){
109             if (nodes[idx] != null){
110               info.history.push(TourHall.this);
111               nodes[idx].paint(info);
112             }
113           }
114         });
115         int resID = (nodes[idx] == null)? R.drawable.qb_empty : info.arrows[idx];
116         info.buttons[idx].setImageResource(resID);
117       }
118     }
119   }
120
```

```java
/**
 * TourInfo.java
 * Brock Butler
 * Wrapper class for all passing required information around throughout the tour.
 * Created by Taras Mychaskiw 2013-02-20
 * Copyright (c) 2013 Sea Addicts. All rights reserved.
 */
package edu.seaaddicts.brockbutler.tour;

import java.util.Stack;

import android.content.Context;
import android.widget.ImageButton;
import android.widget.RelativeLayout;
import edu.seaaddicts.brockbutler.R;

public class TourInfo {
  public RelativeLayout rl;
  public ImageButton[] buttons;
  public int[] arrows;
  public Context context;
  public TourNode current;
  public Stack<TourNode> history;

  /**
   * @param r - the layout of which the background is changed from node to node
   * @param b - array of ImageButtons where one would tap to change nodes
   * @param c - getApplicationContext(), for toasts
   */
  public TourInfo(RelativeLayout r, ImageButton[] b, Context c){
    rl = r;
    buttons = b;
    context = c;
    current = null;
    history = new Stack<TourNode>();
    arrows = new int[5];
    arrows[0] = R.drawable.qb_outerleft;
    arrows[1] = R.drawable.qb_innerleft;
    arrows[2] = R.drawable.qb_center;
    arrows[3] = R.drawable.qb_innerright;
    arrows[4] = R.drawable.qb_outerright;
  }
}

```

```
1   /**
2    * TourNode.java
3    * Brock Butler
4    * Simple abstract wrapper for TourRoom and TourHall.
5    * Created by Taras Mychaskiw 2013-02-20
6    * Copyright (c) 2013 Sea Addicts. All rights reserved.
7    */
8   package edu.seaaddicts.brockbutler.tour;
9
10  public abstract class TourNode {
11    protected int image;    //resource for the image on this node
12    protected TourNode turnAroundNode;
13    protected String title;   //text for Toast to display, if any
14
15    public boolean canTurnAround(){ return(turnAroundNode != null); }
16    public TourNode getTurnAroundNode(){ return(turnAroundNode); }
17
18    public void setTurnAroundNode(TourNode node){ turnAroundNode = node; }
19
20    public void setOuterLeftNode(TourNode node){}
21    public void setInnerLeftNode(TourNode node){}
22    public void setCenterNode(TourNode node){}
23    public void setInnerRightNode(TourNode node){}
24    public void setOuterRightNode(TourNode node){}
25
26    /**
27     * Changes the background image and redefines what the buttons do,
28     * ie where the buttons will take you.
29     * @param info - current tour state
30     */
31    public abstract void paint(TourInfo info);
32  }
33
```

```java
/**
 * TourRoom.java
 * Brock Butler
 * A room node in the tour. Rooms are simple, they don't go anywhere.
 * Created by Taras Mychaskiw 2013-02-20
 * Copyright (c) 2013 Sea Addicts. All rights reserved.
 */
package edu.seaaddicts.brockbutler.tour;

import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Toast;
import edu.seaaddicts.brockbutler.R;

public class TourRoom extends TourNode {

  /**
   * Initializes this room, rooms only need the image resource value
   * @param img - the resource value of the image of this room
   */
  public TourRoom(int img, String s){
    image = img;
    title = s;
    turnAroundNode = null;
  }

  @Override
  public void paint(TourInfo info) {
    info.rl.setBackgroundResource(image);
    info.current = this;
    if (title != null)  //should never be null, but just in case
      Toast.makeText(info.context,title,Toast.LENGTH_SHORT).show();
    for (int i = 0; i < info.buttons.length; i++){
      info.buttons[i].setOnClickListener(new OnClickListener(){
        public void onClick(View v){}
      });
      info.buttons[i].setImageResource(R.drawable.qb_empty);
    }
  }
}

```

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/menu_go_back"
        android:onClick="goBack"
        android:title="@string/menu_go_back"
        android:titleCondensed="@string/menu_go_back">
    </item>
    <item
        android:id="@+id/menu_turn_around"
        android:onClick="turnAround"
        android:title="@string/menu_turn_around"
        android:titleCondensed="@string/menu_turn_around">
    </item>
    <item
        android:id="@+id/menu_teleport"
        android:title="@string/menu_teleport"
        android:titleCondensed="@string/menu_teleport">
        <menu>
            <item
                android:id="@+id/teleport_a_block"
                android:onClick="teleport"
                android:title="@string/a_block"
                android:titleCondensed="@string/a_block"/>
            <item
                android:id="@+id/teleport_b_block"
                android:onClick="teleport"
                android:title="@string/b_block"
                android:titleCondensed="@string/b_block"/>
            <item
                android:id="@+id/teleport_c_block"
                android:onClick="teleport"
                android:title="@string/c_block"
                android:titleCondensed="@string/c_block"/>
            <item
                android:id="@+id/teleport_d_block"
                android:onClick="teleport"
                android:title="@string/d_block"
                android:titleCondensed="@string/d_block"/>
            <item
                android:id="@+id/teleport_e_block"
                android:onClick="teleport"
                android:title="@string/e_block"
                android:titleCondensed="@string/e_block"/>
            <item
                android:id="@+id/teleport_f_block"
                android:onClick="teleport"
                android:title="@string/f_block"
                android:titleCondensed="@string/f_block"/>
            <item
                android:id="@+id/teleport_g_block"
                android:onClick="teleport"
                android:title="@string/g_block"
                android:titleCondensed="@string/g_block"/>
            <item
                android:id="@+id/teleport_h_block"
                android:onClick="teleport"
                android:title="@string/h_block"
                android:titleCondensed="@string/h_block"/>
            <item
                android:id="@+id/teleport_j_block"
                android:onClick="teleport"
                android:title="@string/j_block"
                android:titleCondensed="@string/j_block"/>
        </menu>
    </item>
    <item
        android:id="@+id/menu_end_tour"
        android:onClick="endTour"
        android:title="@string/menu_end_tour"
        android:titleCondensed="@string/menu_end_tour">
```

```
72        </item>
73
74    </menu>
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/screen"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    tools:context=".TourHandler" >

    <ImageButton
        android:id="@+id/outerleft"
        android:contentDescription="@string/go_left"
        android:background="@android:color/transparent"
        android:src="@drawable/qb_outerleft"
        android:padding="5dp"
        android:scaleType="fitEnd"
        android:layout_width="75dp"
        android:layout_height="400dp"
        android:layout_alignParentLeft="true"
        android:layout_marginLeft="10dp"
        android:layout_centerVertical="true" />

    <ImageButton
        android:id="@+id/innerleft"
        android:contentDescription="@string/go_left"
        android:background="@android:color/transparent"
        android:src="@drawable/qb_innerleft"
        android:padding="5dp"
        android:scaleType="fitEnd"
        android:layout_width="75dp"
        android:layout_height="400dp"
        android:layout_marginLeft="20dp"
        android:layout_toRightOf="@+id/outerleft"
        android:layout_centerVertical="true" />

    <ImageButton
        android:id="@+id/center"
        android:contentDescription="@string/go_forward"
        android:background="@android:color/transparent"
        android:src="@drawable/qb_center"
        android:padding="5dp"
        android:scaleType="fitEnd"
        android:layout_width="75dp"
        android:layout_height="400dp"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true" />

    <ImageButton
        android:id="@+id/innerright"
        android:contentDescription="@string/go_right"
        android:background="@android:color/transparent"
        android:src="@drawable/qb_innerright"
        android:padding="5dp"
        android:scaleType="fitEnd"
        android:layout_width="75dp"
        android:layout_height="400dp"
        android:layout_marginRight="20dp"
        android:layout_toLeftOf="@+id/outerright"
        android:layout_centerVertical="true" />

    <ImageButton
        android:id="@+id/outerright"
        android:contentDescription="@string/go_right"
        android:background="@android:color/transparent"
        android:src="@drawable/qb_outerright"
        android:padding="5dp"
        android:scaleType="fitEnd"
        android:layout_width="75dp"
        android:layout_height="400dp"
        android:layout_alignParentRight="true"
        android:layout_marginRight="10dp"
        android:layout_centerVertical="true" />
```

```
72
73    </RelativeLayout>
74
```

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape=
    "rectangle">
3       <solid android:color="#BB000000" />
4   </shape>
```

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
<item>
    <shape>
        <gradient
            android:angle="0"
            android:startColor="#B40404"
            android:endColor="#000"
            android:type="linear" />
    </shape>
</item>
</selector>
```

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <selector xmlns:android="http://schemas.android.com/apk/res/android">
3
4       <item><shape>
5               <gradient android:angle="0" android:endColor="#000" android:startColor=
                "#2E2E2E" android:type="linear" />
6           </shape></item>
7
8   </selector>
```

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

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package edu.seaaddicts.brockbutler;

public final class R {
    public static final class array {
        public static final int course_manager_context_menu=0x7f050000;
        public static final int empty=0x7f050002;
        public static final int task_priority_array=0x7f050001;
    }
    public static final class attr {
    }
    public static final class drawable {
        public static final int _a301b=0x7f020000;
        public static final int _a301f=0x7f020001;
        public static final int _a302b=0x7f020002;
        public static final int _a302f=0x7f020003;
        public static final int _a303b=0x7f020004;
        public static final int _a303f=0x7f020005;
        public static final int _a304b=0x7f020006;
        public static final int _a304f=0x7f020007;
        public static final int _a305b=0x7f020008;
        public static final int _a305f=0x7f020009;
        public static final int _a306b=0x7f02000a;
        public static final int _a306f=0x7f02000b;
        public static final int _a307b=0x7f02000c;
        public static final int _a307f=0x7f02000d;
        public static final int _a308f=0x7f02000e;
        public static final int _a309f=0x7f02000f;
        public static final int _b301b=0x7f020010;
        public static final int _b301f=0x7f020011;
        public static final int _b302b=0x7f020012;
        public static final int _b302f=0x7f020013;
        public static final int _b303b=0x7f020014;
        public static final int _b303f=0x7f020015;
        public static final int _b304b=0x7f020016;
        public static final int _b304f=0x7f020017;
        public static final int _b305b=0x7f020018;
        public static final int _b305f=0x7f020019;
        public static final int _b306b=0x7f02001a;
        public static final int _b306f=0x7f02001b;
        public static final int _b307f=0x7f02001c;
        public static final int _b308b=0x7f02001d;
        public static final int _b308f=0x7f02001e;
        public static final int _b309b=0x7f02001f;
        public static final int _b309f=0x7f020020;
        public static final int _b310b=0x7f020021;
        public static final int _b310f=0x7f020022;
        public static final int _b311b=0x7f020023;
        public static final int _b311f=0x7f020024;
        public static final int _b312b=0x7f020025;
        public static final int _b312f=0x7f020026;
        public static final int _b313b=0x7f020027;
        public static final int _b313f=0x7f020028;
        public static final int _b314f=0x7f020029;
        public static final int _b316b=0x7f02002a;
        public static final int _b317b=0x7f02002b;
        public static final int _c301b=0x7f02002c;
        public static final int _c301f=0x7f02002d;
        public static final int _c302b=0x7f02002e;
        public static final int _c302f=0x7f02002f;
        public static final int _c303b=0x7f020030;
        public static final int _c303f=0x7f020031;
        public static final int _c304b=0x7f020032;
        public static final int _c304f=0x7f020033;
        public static final int _c305b=0x7f020034;
```

```
72          public static final int _c305f=0x7f020035;
73          public static final int _c306b=0x7f020036;
74          public static final int _c306f=0x7f020037;
75          public static final int _c307b=0x7f020038;
76          public static final int _c307f=0x7f020039;
77          public static final int _c308b=0x7f02003a;
78          public static final int _c308f=0x7f02003b;
79          public static final int _c309b=0x7f02003c;
80          public static final int _c309f=0x7f02003d;
81          public static final int _c310b=0x7f02003e;
82          public static final int _c310f=0x7f02003f;
83          public static final int _c311b=0x7f020040;
84          public static final int _c311f=0x7f020041;
85          public static final int _c312b=0x7f020042;
86          public static final int _c312f=0x7f020043;
87          public static final int _c313b=0x7f020044;
88          public static final int _c313f=0x7f020045;
89          public static final int _c314b=0x7f020046;
90          public static final int _c314f=0x7f020047;
91          public static final int _c315b=0x7f020048;
92          public static final int _c315f=0x7f020049;
93          public static final int _c316f=0x7f02004a;
94          public static final int _d301b=0x7f02004b;
95          public static final int _d301f=0x7f02004c;
96          public static final int _d302b=0x7f02004d;
97          public static final int _d302f=0x7f02004e;
98          public static final int _d303b=0x7f02004f;
99          public static final int _d303f=0x7f020050;
100         public static final int _d304b=0x7f020051;
101         public static final int _d304f=0x7f020052;
102         public static final int _d305b=0x7f020053;
103         public static final int _d305f=0x7f020054;
104         public static final int _d306b=0x7f020055;
105         public static final int _d306f=0x7f020056;
106         public static final int _d307b=0x7f020057;
107         public static final int _d307f=0x7f020058;
108         public static final int _d308b=0x7f020059;
109         public static final int _d308f=0x7f02005a;
110         public static final int _d309b=0x7f02005b;
111         public static final int _d309f=0x7f02005c;
112         public static final int _d310b=0x7f02005d;
113         public static final int _d310f=0x7f02005e;
114         public static final int _d311b=0x7f02005f;
115         public static final int _d311f=0x7f020060;
116         public static final int _d312b=0x7f020061;
117         public static final int _d312f=0x7f020062;
118         public static final int _d313b=0x7f020063;
119         public static final int _d313f=0x7f020064;
120         public static final int _d314b=0x7f020065;
121         public static final int _d314f=0x7f020066;
122         public static final int _d315b=0x7f020067;
123         public static final int _d315f=0x7f020068;
124         public static final int _d316b=0x7f020069;
125         public static final int _d316f=0x7f02006a;
126         public static final int _d317b=0x7f02006b;
127         public static final int _d317f=0x7f02006c;
128         public static final int _d318b=0x7f02006d;
129         public static final int _d318f=0x7f02006e;
130         public static final int _d319b=0x7f02006f;
131         public static final int _d319f=0x7f020070;
132         public static final int _d320b=0x7f020071;
133         public static final int _d320f=0x7f020072;
134         public static final int _d321b=0x7f020073;
135         public static final int _d321f=0x7f020074;
136         public static final int _d322b=0x7f020075;
137         public static final int _d322f=0x7f020076;
138         public static final int _d323b=0x7f020077;
139         public static final int _d323f=0x7f020078;
140         public static final int _d324b=0x7f020079;
141         public static final int _d324f=0x7f02007a;
142         public static final int _d325b=0x7f02007b;
```

```
143         public static final int _d325f=0x7f02007c;
144         public static final int _d326b=0x7f02007d;
145         public static final int _d326f=0x7f02007e;
146         public static final int _d327b=0x7f02007f;
147         public static final int _d327f=0x7f020080;
148         public static final int _d328b=0x7f020081;
149         public static final int _d328f=0x7f020082;
150         public static final int _d329b=0x7f020083;
151         public static final int _d329f=0x7f020084;
152         public static final int _d330b=0x7f020085;
153         public static final int _d330f=0x7f020086;
154         public static final int _d331b=0x7f020087;
155         public static final int _d331f=0x7f020088;
156         public static final int _d332b=0x7f020089;
157         public static final int _d332f=0x7f02008a;
158         public static final int _d333b=0x7f02008b;
159         public static final int _d333f=0x7f02008c;
160         public static final int _d334b=0x7f02008d;
161         public static final int _d334f=0x7f02008e;
162         public static final int _d335f=0x7f02008f;
163         public static final int _d336f=0x7f020090;
164         public static final int _d337f=0x7f020091;
165         public static final int _d338f=0x7f020092;
166         public static final int _d339f=0x7f020093;
167         public static final int _d340f=0x7f020094;
168         public static final int _d341f=0x7f020095;
169         public static final int _d342f=0x7f020096;
170         public static final int _d343f=0x7f020097;
171         public static final int _e301b=0x7f020098;
172         public static final int _e301f=0x7f020099;
173         public static final int _e302b=0x7f02009a;
174         public static final int _e302f=0x7f02009b;
175         public static final int _e303b=0x7f02009c;
176         public static final int _e303f=0x7f02009d;
177         public static final int _e304b=0x7f02009e;
178         public static final int _e304f=0x7f02009f;
179         public static final int _e305b=0x7f0200a0;
180         public static final int _e305f=0x7f0200a1;
181         public static final int _e306b=0x7f0200a2;
182         public static final int _e306f=0x7f0200a3;
183         public static final int _e307b=0x7f0200a4;
184         public static final int _e307f=0x7f0200a5;
185         public static final int _e308b=0x7f0200a6;
186         public static final int _e308f=0x7f0200a7;
187         public static final int _e309b=0x7f0200a8;
188         public static final int _e309f=0x7f0200a9;
189         public static final int _e310f=0x7f0200aa;
190         public static final int _e311f=0x7f0200ab;
191         public static final int _f301b=0x7f0200ac;
192         public static final int _f301f=0x7f0200ad;
193         public static final int _f302b=0x7f0200ae;
194         public static final int _f302f=0x7f0200af;
195         public static final int _f303b=0x7f0200b0;
196         public static final int _f303f=0x7f0200b1;
197         public static final int _f304b=0x7f0200b2;
198         public static final int _f304f=0x7f0200b3;
199         public static final int _f305b=0x7f0200b4;
200         public static final int _f305f=0x7f0200b5;
201         public static final int _f306f=0x7f0200b6;
202         public static final int _f307f=0x7f0200b7;
203         public static final int _f308f=0x7f0200b8;
204         public static final int _g301b=0x7f0200b9;
205         public static final int _g301f=0x7f0200ba;
206         public static final int _g302b=0x7f0200bb;
207         public static final int _g302f=0x7f0200bc;
208         public static final int _g303b=0x7f0200bd;
209         public static final int _g303f=0x7f0200be;
210         public static final int _g304b=0x7f0200bf;
211         public static final int _g304f=0x7f0200c0;
212         public static final int _g305b=0x7f0200c1;
213         public static final int _g305f=0x7f0200c2;
```

```
214         public static final int _g306b=0x7f0200c3;
215         public static final int _g306f=0x7f0200c4;
216         public static final int _g307b=0x7f0200c5;
217         public static final int _g307f=0x7f0200c6;
218         public static final int _g308b=0x7f0200c7;
219         public static final int _g308f=0x7f0200c8;
220         public static final int _g309b=0x7f0200c9;
221         public static final int _g309f=0x7f0200ca;
222         public static final int _g310f=0x7f0200cb;
223         public static final int _g311f=0x7f0200cc;
224         public static final int _g312f=0x7f0200cd;
225         public static final int _g313f=0x7f0200ce;
226         public static final int _h301b=0x7f0200cf;
227         public static final int _h301f=0x7f0200d0;
228         public static final int _h302b=0x7f0200d1;
229         public static final int _h302f=0x7f0200d2;
230         public static final int _h303b=0x7f0200d3;
231         public static final int _h303f=0x7f0200d4;
232         public static final int _h304b=0x7f0200d5;
233         public static final int _h304f=0x7f0200d6;
234         public static final int _h305b=0x7f0200d7;
235         public static final int _h305f=0x7f0200d8;
236         public static final int _h306b=0x7f0200d9;
237         public static final int _h306f=0x7f0200da;
238         public static final int _h307b=0x7f0200db;
239         public static final int _h307f=0x7f0200dc;
240         public static final int _h308b=0x7f0200dd;
241         public static final int _h308f=0x7f0200de;
242         public static final int _h309b=0x7f0200df;
243         public static final int _h309f=0x7f0200e0;
244         public static final int _h310b=0x7f0200e1;
245         public static final int _h310f=0x7f0200e2;
246         public static final int _h311b=0x7f0200e3;
247         public static final int _h311f=0x7f0200e4;
248         public static final int _h312b=0x7f0200e5;
249         public static final int _h312f=0x7f0200e6;
250         public static final int _h313b=0x7f0200e7;
251         public static final int _h313f=0x7f0200e8;
252         public static final int _h314b=0x7f0200e9;
253         public static final int _h314f=0x7f0200ea;
254         public static final int _h315b=0x7f0200eb;
255         public static final int _h315f=0x7f0200ec;
256         public static final int _h316b=0x7f0200ed;
257         public static final int _h316f=0x7f0200ee;
258         public static final int _h317b=0x7f0200ef;
259         public static final int _h317f=0x7f0200f0;
260         public static final int _h318b=0x7f0200f1;
261         public static final int _h318f=0x7f0200f2;
262         public static final int _h319b=0x7f0200f3;
263         public static final int _h319f=0x7f0200f4;
264         public static final int _h320b=0x7f0200f5;
265         public static final int _h320f=0x7f0200f6;
266         public static final int _h321b=0x7f0200f7;
267         public static final int _h321f=0x7f0200f8;
268         public static final int _h322b=0x7f0200f9;
269         public static final int _h322f=0x7f0200fa;
270         public static final int _h323b=0x7f0200fb;
271         public static final int _h323f=0x7f0200fc;
272         public static final int _h324b=0x7f0200fd;
273         public static final int _h324f=0x7f0200fe;
274         public static final int _h325b=0x7f0200ff;
275         public static final int _h325f=0x7f020100;
276         public static final int _h326b=0x7f020101;
277         public static final int _h326f=0x7f020102;
278         public static final int _h327f=0x7f020103;
279         public static final int _h328f=0x7f020104;
280         public static final int _h329f=0x7f020105;
281         public static final int _h330f=0x7f020106;
282         public static final int _h331f=0x7f020107;
283         public static final int _h332f=0x7f020108;
284         public static final int _h333f=0x7f020109;
```

```
285         public static final int _h334f=0x7f02010a;
286         public static final int _h335f=0x7f02010b;
287         public static final int _h336f=0x7f02010c;
288         public static final int _h337f=0x7f02010d;
289         public static final int _h338f=0x7f02010e;
290         public static final int _j301b=0x7f02010f;
291         public static final int _j301f=0x7f020110;
292         public static final int _j302b=0x7f020111;
293         public static final int _j302f=0x7f020112;
294         public static final int _j303b=0x7f020113;
295         public static final int _j303f=0x7f020114;
296         public static final int _j304b=0x7f020115;
297         public static final int _j304f=0x7f020116;
298         public static final int _j305b=0x7f020117;
299         public static final int _j305f=0x7f020118;
300         public static final int _j306b=0x7f020119;
301         public static final int _j306f=0x7f02011a;
302         public static final int _j307b=0x7f02011b;
303         public static final int _j307f=0x7f02011c;
304         public static final int _j308b=0x7f02011d;
305         public static final int _j308f=0x7f02011e;
306         public static final int _j309b=0x7f02011f;
307         public static final int _j309f=0x7f020120;
308         public static final int _j310b=0x7f020121;
309         public static final int _j310f=0x7f020122;
310         public static final int _j311b=0x7f020123;
311         public static final int _j311f=0x7f020124;
312         public static final int _j312b=0x7f020125;
313         public static final int _j312f=0x7f020126;
314         public static final int _j313f=0x7f020127;
315         public static final int _j314f=0x7f020128;
316         public static final int _j315f=0x7f020129;
317         public static final int a323=0x7f02012a;
318         public static final int actionbar_bg=0x7f02012b;
319         public static final int b303=0x7f02012c;
320         public static final int b303a=0x7f02012d;
321         public static final int b303b=0x7f02012e;
322         public static final int c300=0x7f02012f;
323         public static final int c303=0x7f020130;
324         public static final int c304=0x7f020131;
325         public static final int c306_1=0x7f020132;
326         public static final int c306_2=0x7f020133;
327         public static final int c306_3=0x7f020134;
328         public static final int c308=0x7f020135;
329         public static final int c310=0x7f020136;
330         public static final int compass_icon=0x7f020137;
331         public static final int compass_needle=0x7f020138;
332         public static final int d300=0x7f020139;
333         public static final int d301=0x7f02013a;
334         public static final int d303=0x7f02013b;
335         public static final int d304=0x7f02013c;
336         public static final int d308=0x7f02013d;
337         public static final int d309_1=0x7f02013e;
338         public static final int d309_2=0x7f02013f;
339         public static final int d310=0x7f020140;
340         public static final int d314=0x7f020141;
341         public static final int d315=0x7f020142;
342         public static final int d316=0x7f020143;
343         public static final int d317=0x7f020144;
344         public static final int d318=0x7f020145;
345         public static final int d319=0x7f020146;
346         public static final int d350l=0x7f020147;
347         public static final int e302=0x7f020148;
348         public static final int e303=0x7f020149;
349         public static final int e304=0x7f02014a;
350         public static final int earth_orbit_elliptic=0x7f02014b;
351         public static final int g301_1=0x7f02014c;
352         public static final int g301_2=0x7f02014d;
353         public static final int g301_3=0x7f02014e;
354         public static final int g305=0x7f02014f;
355         public static final int g310=0x7f020150;
```

```
356             public static final int grad_course_title=0x7f020151;
357             public static final int grad_task_title=0x7f020152;
358             public static final int h300=0x7f020153;
359             public static final int h302=0x7f020154;
360             public static final int h303=0x7f020155;
361             public static final int h304=0x7f020156;
362             public static final int h304a=0x7f020157;
363             public static final int h306=0x7f020158;
364             public static final int h306a6=0x7f020159;
365             public static final int h306a9=0x7f02015a;
366             public static final int h309=0x7f02015b;
367             public static final int h309a6=0x7f02015c;
368             public static final int h309a9=0x7f02015d;
369             public static final int h310=0x7f02015e;
370             public static final int h313=0x7f02015f;
371             public static final int h315=0x7f020160;
372             public static final int h316=0x7f020161;
373             public static final int h317=0x7f020162;
374             public static final int h318=0x7f020163;
375             public static final int h318a=0x7f020164;
376             public static final int h320=0x7f020165;
377             public static final int h321=0x7f020166;
378             public static final int h322=0x7f020167;
379             public static final int h323=0x7f020168;
380             public static final int h324=0x7f020169;
381             public static final int ic_course=0x7f02016a;
382             public static final int ic_email=0x7f02016b;
383             public static final int ic_help=0x7f02016c;
384             public static final int ic_launcher=0x7f02016d;
385             public static final int ic_maps=0x7f02016e;
386             public static final int ic_scheduler=0x7f02016f;
387             public static final int ic_tour=0x7f020170;
388             public static final int j301=0x7f020171;
389             public static final int j310=0x7f020172;
390             public static final int j327=0x7f020173;
391             public static final int j328=0x7f020174;
392             public static final int mch_maps=0x7f020175;
393             public static final int mch_maps_whitebg=0x7f020176;
394             public static final int qb_center=0x7f020177;
395             public static final int qb_empty=0x7f020178;
396             public static final int qb_innerleft=0x7f020179;
397             public static final int qb_innerright=0x7f02017a;
398             public static final int qb_outerleft=0x7f02017b;
399             public static final int qb_outerright=0x7f02017c;
400         }
401         public static final class id {
402             public static final int activity_scheduler=0x7f080035;
403             public static final int add_contact_add_course_button=0x7f080001;
404             public static final int add_contact_cancel_button=0x7f080004;
405             public static final int add_contact_course_listview=0x7f080002;
406             public static final int add_contact_save_button=0x7f080003;
407             public static final int add_course=0x7f08005c;
408             public static final int add_course_add_labs=0x7f08000a;
409             public static final int add_course_add_lecs=0x7f080008;
410             public static final int add_course_add_sems=0x7f08000e;
411             public static final int add_course_add_tuts=0x7f08000c;
412             public static final int add_course_cancel_button=0x7f080010;
413             public static final int add_course_codes_spinner=0x7f080006;
414             public static final int add_course_save_button=0x7f08000f;
415             public static final int add_course_subjects_spinner=0x7f080005;
416             public static final int add_task_cancel_button=0x7f08001a;
417             public static final int add_task_course_spinner=0x7f080011;
418             public static final int add_task_date_textview=0x7f080018;
419             public static final int add_task_due_date_button=0x7f080017;
420             public static final int add_task_priority_spinner=0x7f080013;
421             public static final int add_task_save_button=0x7f080019;
422             public static final int add_task_title=0x7f080012;
423             public static final int app_logo=0x7f08001e;
424             public static final int app_logo2=0x7f08001f;
425             public static final int btnRemove=0x7f080048;
426             public static final int btn_contacts=0x7f08002b;
```

```
427         public static final int btn_help=0x7f080030;
428         public static final int btn_login_login=0x7f080023;
429         public static final int btn_login_skip=0x7f080024;
430         public static final int btn_maps=0x7f08002d;
431         public static final int btn_sched=0x7f08002a;
432         public static final int btn_tour=0x7f08002e;
433         public static final int center=0x7f08003a;
434         public static final int chk_login_remember=0x7f080026;
435         public static final int chk_login_skip=0x7f080028;
436         public static final int course_grade_grade=0x7f08004a;
437         public static final int course_grade_title=0x7f080049;
438         public static final int course_list_item_title=0x7f08003d;
439         public static final int course_manager_layout=0x7f08001b;
440         public static final int course_manager_list=0x7f08001c;
441         public static final int course_manager_toolbar=0x7f08003e;
442         public static final int et_contact_fname=0x7f080000;
443         public static final int imgv_map=0x7f080032;
444         public static final int innerleft=0x7f080039;
445         public static final int innerright=0x7f08003b;
446         public static final int layout_add_labs=0x7f080009;
447         public static final int layout_add_lecs=0x7f080007;
448         public static final int layout_add_sems=0x7f08000d;
449         public static final int layout_add_tuts=0x7f08000b;
450         public static final int layout_row_prof_simple=0x7f08004c;
451         public static final int login_email=0x7f080021;
452         public static final int login_password=0x7f080022;
453         public static final int login_remember=0x7f080025;
454         public static final int login_skip=0x7f080027;
455         public static final int login_table=0x7f080020;
456         public static final int lv_courses=0x7f080047;
457         public static final int main_icon_row1=0x7f080029;
458         public static final int main_icon_row2=0x7f08002c;
459         public static final int main_icon_row3=0x7f08002f;
460         public static final int maps_layout=0x7f080031;
461         public static final int menu_add_task=0x7f080063;
462         public static final int menu_end_tour=0x7f080071;
463         public static final int menu_exit_maps=0x7f080061;
464         public static final int menu_get_directions=0x7f080060;
465         public static final int menu_go_back=0x7f080065;
466         public static final int menu_search=0x7f08005f;
467         public static final int menu_show_coursemanager_help=0x7f08005e;
468         public static final int menu_show_maps_help=0x7f080062;
469         public static final int menu_show_scheduler_help=0x7f080064;
470         public static final int menu_teleport=0x7f080067;
471         public static final int menu_turn_around=0x7f080066;
472         public static final int modify_course_subject_textview=0x7f080034;
473         public static final int outerleft=0x7f080038;
474         public static final int outerright=0x7f08003c;
475         public static final int prof_email_button=0x7f08004d;
476         public static final int row_grade=0x7f080044;
477         public static final int row_lab=0x7f080041;
478         public static final int row_lec=0x7f080040;
479         public static final int row_prof=0x7f08003f;
480         public static final int row_sem=0x7f080043;
481         public static final int row_task_info=0x7f08004e;
482         public static final int row_tut=0x7f080042;
483         public static final int sched_list=0x7f080036;
484         public static final int sched_list_item=0x7f080057;
485         public static final int sched_list_item_title=0x7f080058;
486         public static final int sched_tasks=0x7f08005b;
487         public static final int sched_tasks_title=0x7f08005a;
488         public static final int sched_toolbar=0x7f080059;
489         public static final int screen=0x7f080037;
490         public static final int task_base=0x7f080015;
491         public static final int task_complete_checkbox=0x7f080051;
492         public static final int task_info=0x7f080050;
493         public static final int task_mark=0x7f080016;
494         public static final int task_weight=0x7f080014;
495         public static final int teleport_a_block=0x7f080068;
496         public static final int teleport_b_block=0x7f080069;
497         public static final int teleport_c_block=0x7f08006a;
```

```
498             public static final int teleport_d_block=0x7f08006b;
499             public static final int teleport_e_block=0x7f08006c;
500             public static final int teleport_f_block=0x7f08006d;
501             public static final int teleport_g_block=0x7f08006e;
502             public static final int teleport_h_block=0x7f08006f;
503             public static final int teleport_j_block=0x7f080070;
504             public static final int tv_assignment=0x7f08004f;
505             public static final int tv_contact_name=0x7f080045;
506             public static final int tv_email=0x7f080046;
507             public static final int tv_lab=0x7f080052;
508             public static final int tv_lecture=0x7f080053;
509             public static final int tv_no_courses=0x7f08001d;
510             public static final int tv_prof_name=0x7f08004b;
511             public static final int tv_seminar=0x7f080054;
512             public static final int tv_test=0x7f080055;
513             public static final int tv_tutorial=0x7f080056;
514             public static final int txtv_count=0x7f080033;
515             public static final int update_master_list=0x7f08005d;
516         }
517         public static final class layout {
518             public static final int activity_add_contact=0x7f030000;
519             public static final int activity_add_course=0x7f030001;
520             public static final int activity_add_task=0x7f030002;
521             public static final int activity_contacts=0x7f030003;
522             public static final int activity_coursemanager=0x7f030004;
523             public static final int activity_help=0x7f030005;
524             public static final int activity_login=0x7f030006;
525             public static final int activity_main=0x7f030007;
526             public static final int activity_map=0x7f030008;
527             public static final int activity_modify_course=0x7f030009;
528             public static final int activity_modify_task=0x7f03000a;
529             public static final int activity_scheduler=0x7f03000b;
530             public static final int activity_tour=0x7f03000c;
531             public static final int course_list_item=0x7f03000d;
532             public static final int row_contact_full=0x7f03000e;
533             public static final int row_contact_simple=0x7f03000f;
534             public static final int row_grade=0x7f030010;
535             public static final int row_prof_full=0x7f030011;
536             public static final int row_prof_simple=0x7f030012;
537             public static final int row_task=0x7f030013;
538             public static final int row_task_assignment=0x7f030014;
539             public static final int row_task_info=0x7f030015;
540             public static final int row_task_lab=0x7f030016;
541             public static final int row_task_lecture=0x7f030017;
542             public static final int row_task_seminar=0x7f030018;
543             public static final int row_task_test=0x7f030019;
544             public static final int row_task_tutorial=0x7f03001a;
545             public static final int sched_list_item=0x7f03001b;
546         }
547         public static final class menu {
548             public static final int activity_add_task=0x7f070000;
549             public static final int activity_contacts=0x7f070001;
550             public static final int activity_coursemanager=0x7f070002;
551             public static final int activity_help=0x7f070003;
552             public static final int activity_login=0x7f070004;
553             public static final int activity_main=0x7f070005;
554             public static final int activity_map=0x7f070006;
555             public static final int activity_scheduler=0x7f070007;
556             public static final int activity_tour=0x7f070008;
557         }
558         public static final class string {
559             public static final int a_block=0x7f040046;
560             /**  AddContactActivity
561              */
562             public static final int activity_add_contact=0x7f04001a;
563             public static final int activity_add_contact_add_courses=0x7f040022;
564             public static final int activity_add_contact_courses=0x7f040021;
565             public static final int activity_add_contact_email=0x7f04001f;
566             public static final int activity_add_contact_fname=0x7f04001b;
567             public static final int activity_add_contact_hint_email=0x7f040020;
568             public static final int activity_add_contact_hint_fname=0x7f04001c;
```

```
569         public static final int activity_add_contact_hint_lname=0x7f04001e;
570         public static final int activity_add_contact_lname=0x7f04001d;
571         public static final int activity_add_contact_remove=0x7f040023;
572         /**  AddTaskActivity
573          */
574         public static final int activity_add_task=0x7f040001;
575         public static final int activity_add_task_base=0x7f04000a;
576         public static final int activity_add_task_contact=0x7f04000f;
577         public static final int activity_add_task_contacts=0x7f040011;
578         public static final int activity_add_task_course=0x7f040004;
579         public static final int activity_add_task_date=0x7f040010;
580         public static final int activity_add_task_description=0x7f040008;
581         public static final int activity_add_task_due_date=0x7f04000e;
582         public static final int activity_add_task_hint_base=0x7f04000b;
583         public static final int activity_add_task_hint_description=0x7f040009;
584         public static final int activity_add_task_hint_mark=0x7f04000d;
585         public static final int activity_add_task_hint_task_title=0x7f040003;
586         public static final int activity_add_task_hint_weight=0x7f040007;
587         public static final int activity_add_task_mark=0x7f04000c;
588         public static final int activity_add_task_priority=0x7f040005;
589         public static final int activity_add_task_task_title=0x7f040002;
590         public static final int activity_add_task_weight=0x7f040006;
591         /**  CourseManagerActivity
592          */
593         public static final int activity_coursemanager=0x7f040012;
594         public static final int add_course_add=0x7f040026;
595         public static final int add_course_add_labs=0x7f040029;
596         public static final int add_course_add_lecs=0x7f040028;
597         public static final int add_course_add_sems=0x7f04002a;
598         public static final int add_course_add_tuts=0x7f04002b;
599         public static final int add_course_code=0x7f040027;
600         public static final int add_course_remove=0x7f040013;
601         public static final int add_course_subject=0x7f040025;
602         public static final int add_course_update_master_list=0x7f040014;
603         public static final int app_name=0x7f040000;
604         public static final int b_block=0x7f040047;
605         public static final int button_cancel=0x7f04006e;
606         public static final int button_save=0x7f04006d;
607         public static final int c_block=0x7f040048;
608         /**  Miscellaneous
609          */
610         public static final int contentdesc=0x7f04006c;
611         public static final int course_grade=0x7f040019;
612         public static final int d_block=0x7f040049;
613         public static final int e_block=0x7f04004a;
614         public static final int f_block=0x7f04004b;
615         public static final int g_block=0x7f04004c;
616         public static final int go_forward=0x7f040050;
617         public static final int go_left=0x7f04004f;
618         public static final int go_right=0x7f040051;
619         public static final int h_block=0x7f04004d;
620         public static final int j_block=0x7f04004e;
621         public static final int loading_courses_registrar=0x7f040015;
622         public static final int loading_courses_registrar_msg=0x7f040017;
623         public static final int login=0x7f040059;
624         public static final int login_email=0x7f040054;
625         public static final int login_password=0x7f040055;
626         public static final int login_remember=0x7f040057;
627         public static final int login_skip=0x7f040056;
628         public static final int login_skip_msg=0x7f040058;
629         /**  SchedulerActivity
630          */
631         public static final int menu_add_task=0x7f04002f;
632         public static final int menu_end_tour=0x7f040045;
633         public static final int menu_exit_maps=0x7f040040;
634         public static final int menu_expand_collapse=0x7f040032;
635         public static final int menu_get_directions=0x7f04003e;
636         /**  TourActivity
637          */
638         public static final int menu_go_back=0x7f040042;
639         public static final int menu_modify_task=0x7f040031;
```

```
640           public static final int menu_remove_task=0x7f040030;
641           /**  MapsActivity
642            */
643           public static final int menu_search=0x7f04003d;
644           public static final int menu_settings=0x7f040052;
645           public static final int menu_show_coursemanager_help=0x7f040018;
646           public static final int menu_show_maps_help=0x7f040041;
647           public static final int menu_show_scheduler_help=0x7f040034;
648           public static final int menu_teleport=0x7f040044;
649           public static final int menu_turn_around=0x7f040043;
650           public static final int menu_update_position=0x7f04003f;
651           public static final int msg_skip=0x7f04005b;
652           public static final int no_courses=0x7f040016;
653           public static final int none=0x7f04006f;
654           public static final int prof_email=0x7f04002e;
655           /**  Professors
656            */
657           public static final int prof_name=0x7f04002c;
658           public static final int prof_office=0x7f04002d;
659           public static final int sched_no_courses=0x7f040033;
660           public static final int scheduler_menu_showhide_completed=0x7f040035;
661           public static final int task_priority_prompt=0x7f04006b;
662           public static final int temp=0x7f04005a;
663           public static final int test_image=0x7f04005e;
664           public static final int title_activity_add_task=0x7f040068;
665           public static final int title_activity_contacts=0x7f040066;
666           public static final int title_activity_help=0x7f040064;
667           /**  LoginActivity
668            */
669           public static final int title_activity_login=0x7f040053;
670           /**  MainActivity
671            */
672           public static final int title_activity_main=0x7f04005c;
673           public static final int title_activity_map=0x7f04005d;
674           public static final int title_activity_scheduler=0x7f040065;
675           public static final int title_activity_tour=0x7f040067;
676           /**  AddCourseActivity
677            */
678           public static final int title_add_course=0x7f040024;
679           public static final int title_assignment=0x7f04003b;
680           public static final int title_contacts=0x7f040061;
681           public static final int title_help=0x7f040063;
682           public static final int title_lab=0x7f040038;
683           public static final int title_lecture=0x7f040037;
684           public static final int title_maps=0x7f040060;
685           public static final int title_modify_course=0x7f04006a;
686           public static final int title_modify_task=0x7f040069;
687           public static final int title_scheduler=0x7f04005f;
688           public static final int title_seminar=0x7f04003a;
689           /**  row_task_*
690            */
691           public static final int title_tasks=0x7f040036;
692           public static final int title_test=0x7f04003c;
693           public static final int title_tour=0x7f040062;
694           public static final int title_tutorial=0x7f040039;
695       }
696       public static final class style {
697           /**
698           Base application theme, dependent on API level. This theme is replaced
699           by AppBaseTheme from res/values-vXX/styles.xml on newer devices.
700
701
702               Theme customizations available in newer API levels can go in
703               res/values-vXX/styles.xml, while customizations related to
704               backward-compatibility can go here.
705
706
707           Base application theme for API 11+. This theme completely replaces
708           AppBaseTheme from res/values/styles.xml on API 11+ devices.
709
710       API 11 theme customizations can go here.
```

```
711
712             Base application theme for API 14+. This theme completely replaces
713             AppBaseTheme from BOTH res/values/styles.xml and
714             res/values-v11/styles.xml on API 14+ devices.
715
716         API 14 theme customizations can go here.
717                 */
718             public static final int AppBaseTheme=0x7f060000;
719             /**  Application theme.
720         All customizations that are NOT specific to a particular API-level can go here.
721                 */
722             public static final int AppTheme=0x7f060001;
723         }
724     }
725
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/course_list_item_title"
        android:layout_width="fill_parent"
        android:layout_height="60sp"
        android:focusable="false"
        android:focusableInTouchMode="false"
        android:padding="10dip"
        android:textSize="30sp"
        android:textStyle="bold"
        android:layout_marginBottom="5dip"
        android:background="@drawable/grad_course_title" />

    <!-- Toolbar -->
    <LinearLayout
        android:id="@+id/course_manager_toolbar"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="10dip"
        android:visibility="gone" >

        <include
            android:id="@+id/row_prof"
            layout="@layout/row_prof_simple" />

        <include
            android:id="@+id/row_lec"
            layout="@layout/row_task_lecture" />

        <include
            android:id="@+id/row_lab"
            layout="@layout/row_task_lab" />

        <include
            android:id="@+id/row_tut"
            layout="@layout/row_task_tutorial" />

        <include
            android:id="@+id/row_sem"
            layout="@layout/row_task_seminar" />

        <include
            android:id="@+id/row_grade"
            layout="@layout/row_grade" />
    </LinearLayout>

</LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:paddingLeft="10sp"
    android:paddingRight="10sp"
    android:paddingTop="2sp" >

    <TextView
        android:id="@+id/tv_contact_name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp"
        android:textStyle="bold"
        android:typeface="sans"
        android:text="@string/activity_add_contact_fname" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/activity_add_contact_email"
            android:textSize="13sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/tv_email"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="13sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/activity_add_contact_courses"
            android:textSize="13sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/lv_courses"
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:layout_weight="1" />

        <Button
            android:id="@+id/btnRemove"
            android:layout_width="wrap_content"
            android:layout_height="30sp"
            android:layout_gravity="right"
            android:text="@string/activity_add_contact_remove"
            android:textSize="10sp" />
    </LinearLayout>

</LinearLayout>
```

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:layout_width="fill_parent"
4       android:layout_height="wrap_content"
5       android:orientation="horizontal"
6       android:paddingLeft="10sp"
7       android:paddingRight="10sp"
8       android:paddingTop="2sp" >
9
10      <TextView
11          android:id="@+id/tv_contact_name"
12          android:layout_width="0dip"
13          android:layout_height="wrap_content"
14          android:layout_weight="1"
15          android:text="@string/activity_add_contact_fname"
16          android:textSize="16sp"
17          android:textStyle="bold"
18          android:typeface="sans" />
19
20      <Button
21          android:id="@+id/btnRemove"
22          android:layout_width="wrap_content"
23          android:layout_height="30sp"
24          android:layout_gravity="right"
25          android:text="@string/activity_add_contact_remove"
26          android:textSize="10sp" />
27
28  </LinearLayout>
```

```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 3      android:layout_width="fill_parent"
 4      android:layout_height="wrap_content"
 5      android:layout_margin="10dip"
 6      android:layout_marginLeft="15dip"
 7      android:orientation="vertical"
 8      android:paddingLeft="10sp"
 9      android:paddingRight="10sp"
10      android:paddingTop="2sp" >
11
12      <TextView
13          android:id="@+id/course_grade_title"
14          android:layout_width="fill_parent"
15          android:layout_height="wrap_content"
16          android:background="@drawable/grad_task_title"
17          android:paddingLeft="5dip"
18          android:text="@string/course_grade"
19          android:textSize="16sp"
20          android:textStyle="bold"
21          android:typeface="sans" />
22
23      <TextView
24          android:id="@+id/course_grade_grade"
25          android:layout_width="fill_parent"
26          android:layout_height="wrap_content"
27          android:layout_marginLeft="5dip"
28          android:paddingLeft="10dip"
29          android:text="@string/none"
30          android:textColor="#ddd" >
31      </TextView>
32
33  </LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:paddingLeft="10sp"
    android:paddingRight="10sp"
    android:paddingTop="2sp" >

    <TextView
        android:id="@+id/tv_prof_name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp"
        android:textStyle="bold"
        android:typeface="sans"
        android:text="@string/prof_name" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/activity_add_contact_email"
            android:textSize="13sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/tv_email"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="13sp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:layout_width="85sp"
            android:layout_height="wrap_content"
            android:text="@string/prof_office"
            android:textSize="13sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/lv_courses"
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:layout_weight="1" />
    </LinearLayout>

</LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/layout_row_prof_simple"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10sp" >

    <TextView
        android:id="@+id/tv_prof_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:text="@string/prof_name"
        android:textSize="16sp"
        android:textStyle="bold"
        android:typeface="sans" />

    <Button
        android:id="@+id/prof_email_button"
        android:layout_width="40dip"
        android:layout_height="40dip"
        android:padding="20dip"
        android:hapticFeedbackEnabled="true"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="20sp"
        android:background="@drawable/ic_email"
        android:contentDescription="@string/contentdesc"
        android:focusable="false"
        android:focusableInTouchMode="false" />

</LinearLayout>
```

```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 3      android:layout_width="fill_parent"
 4      android:layout_height="wrap_content"
 5      android:layout_margin="10dip"
 6      android:layout_marginLeft="15dip"
 7      android:orientation="vertical"
 8      android:paddingLeft="10sp"
 9      android:paddingRight="10sp"
10      android:paddingTop="2sp" >
11
12      <TextView
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:layout_gravity="center_vertical"
16          android:text="@string/title_tasks"
17          android:textSize="16sp"
18          android:textStyle="bold"
19          android:typeface="sans" />
20
21          <include android:id="@+id/row_task_info" layout="@layout/row_task_info"/>
22
23  </LinearLayout>
```

```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 3      android:layout_width="fill_parent"
 4      android:layout_height="wrap_content"
 5      android:layout_margin="10dip"
 6      android:layout_marginLeft="15dip"
 7      android:orientation="vertical"
 8      android:paddingLeft="10sp"
 9      android:paddingRight="10sp"
10      android:paddingTop="2sp" >
11
12      <TextView
13          android:layout_width="fill_parent"
14          android:layout_height="wrap_content"
15          android:text="@string/title_assignment"
16          android:textSize="16sp"
17          android:textStyle="bold"
18          android:typeface="sans" />
19
20      <TextView
21          android:id="@+id/tv_assignment"
22          android:layout_width="fill_parent"
23          android:layout_height="wrap_content"
24          android:layout_marginLeft="5dip"
25          android:textColor="#ddd"
26          android:text="@string/none" >
27      </TextView>
28
29  </LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10sp" >

    <TextView
        android:id="@+id/task_info"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:textSize="16sp"
        android:textStyle="bold"
        android:typeface="sans" />

    <CheckBox
        android:id="@+id/task_complete_checkbox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:focusable="false"
        android:focusableInTouchMode="false" />
</RelativeLayout>
```

```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 3      android:layout_width="fill_parent"
 4      android:layout_height="wrap_content"
 5      android:layout_margin="10dip"
 6      android:layout_marginLeft="15dip"
 7      android:orientation="vertical"
 8      android:paddingLeft="10sp"
 9      android:paddingRight="10sp"
10      android:paddingTop="2sp" >
11
12      <TextView
13          android:layout_width="fill_parent"
14          android:layout_height="wrap_content"
15          android:background="@drawable/grad_task_title"
16          android:paddingLeft="5dip"
17          android:text="@string/title_lab"
18          android:textSize="16sp"
19          android:textStyle="bold"
20          android:typeface="sans" />
21
22      <TextView
23          android:id="@+id/tv_lab"
24          android:layout_width="fill_parent"
25          android:layout_height="wrap_content"
26          android:layout_marginLeft="5dip"
27          android:paddingLeft="10dip"
28          android:text="@string/none"
29          android:textColor="#ddd" >
30      </TextView>
31
32  </LinearLayout>
```

```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 3      android:layout_width="fill_parent"
 4      android:layout_height="wrap_content"
 5      android:layout_margin="10dip"
 6      android:layout_marginLeft="15dip"
 7      android:orientation="vertical"
 8      android:paddingLeft="10sp"
 9      android:paddingRight="10sp"
10      android:paddingTop="2sp" >
11
12      <TextView
13          android:layout_width="fill_parent"
14          android:layout_height="wrap_content"
15          android:background="@drawable/grad_task_title"
16          android:paddingLeft="5dip"
17          android:text="@string/title_lecture"
18          android:textSize="16sp"
19          android:textStyle="bold"
20          android:typeface="sans" />
21
22      <TextView
23          android:id="@+id/tv_lecture"
24          android:layout_width="fill_parent"
25          android:layout_height="wrap_content"
26          android:layout_marginLeft="5dip"
27          android:paddingLeft="10dip"
28          android:text="@string/none"
29          android:textColor="#ddd" >
30      </TextView>
31
32  </LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dip"
    android:layout_marginLeft="15dip"
    android:orientation="vertical"
    android:paddingLeft="10sp"
    android:paddingRight="10sp"
    android:paddingTop="2sp" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/grad_task_title"
        android:paddingLeft="5dip"
        android:text="@string/title_seminar"
        android:textSize="16sp"
        android:textStyle="bold"
        android:typeface="sans" />

    <TextView
        android:id="@+id/tv_seminar"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"
        android:paddingLeft="10dip"
        android:text="@string/none"
        android:textColor="#ddd" >
    </TextView>

</LinearLayout>
```

```
 1  <?xml version="1.0" encoding="utf-8"?>
 2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 3      android:layout_width="fill_parent"
 4      android:layout_height="wrap_content"
 5      android:layout_margin="10dip"
 6      android:layout_marginLeft="15dip"
 7      android:orientation="vertical"
 8      android:paddingLeft="10sp"
 9      android:paddingRight="10sp"
10      android:paddingTop="2sp" >
11
12      <TextView
13          android:layout_width="fill_parent"
14          android:layout_height="wrap_content"
15          android:text="@string/title_test"
16          android:textSize="16sp"
17          android:textStyle="bold"
18          android:typeface="sans" />
19
20      <TextView
21          android:id="@+id/tv_test"
22          android:layout_width="fill_parent"
23          android:layout_height="wrap_content"
24          android:layout_marginLeft="5dip"
25          android:textColor="#ddd"
26          android:text="@string/none" >
27      </TextView>
28
29  </LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dip"
    android:layout_marginLeft="15dip"
    android:orientation="vertical"
    android:paddingLeft="10sp"
    android:paddingRight="10sp"
    android:paddingTop="2sp" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/grad_task_title"
        android:paddingLeft="5dip"
        android:text="@string/title_tutorial"
        android:textSize="16sp"
        android:textStyle="bold"
        android:typeface="sans" />

    <TextView
        android:id="@+id/tv_tutorial"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"
        android:paddingLeft="10dip"
        android:text="@string/none"
        android:textColor="#ddd" >
    </TextView>

</LinearLayout>
```

```
 1  package edu.seaaddicts.brockbutler;
 2
 3  import android.app.Activity;
 4  import android.app.AlertDialog;
 5  import android.content.DialogInterface;
 6  import android.content.Intent;
 7  import android.os.Bundle;
 8  import android.view.Gravity;
 9  import android.view.View;
10  import android.view.View.OnClickListener;
11  import android.widget.Button;
12  import android.widget.TextView;
13
14  public class LoginActivity extends Activity {
15    private Button mBtnLogin;
16    private Button mBtnSkip;
17    //private CheckBox mChkRemember;
18    //private CheckBox mChkSkip;
19
20    @Override
21    protected void onCreate(Bundle savedInstanceState) {
22      super.onCreate(savedInstanceState);
23      setContentView(R.layout.activity_login);
24      init();
25    }
26
27    @Override
28    public void onBackPressed() {
29      AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
30      alertDialogBuilder.setTitle("Exit application?");
31      alertDialogBuilder
32          .setMessage(
33              "Are you sure you want to exit and close BrockButler?")
34          .setCancelable(false)
35          .setPositiveButton("Yes",
36              new DialogInterface.OnClickListener() {
37                public void onClick(DialogInterface dialog, int id) {
38                  LoginActivity.this.finish();
39                }
40              })
41          .setNegativeButton("No", new DialogInterface.OnClickListener() {
42            public void onClick(DialogInterface dialog, int id) {
43              dialog.cancel();
44            }
45          });
46      AlertDialog alertDialog = alertDialogBuilder.create();
47      alertDialog.show();
48    }
49
50
51    // Assigns Buttons and Checkboxes, as well as their functionalities, i.e.
52    // OnClickListeners.
53    public void init() {
54
55      mBtnLogin = (Button) findViewById(R.id.btn_login_login);
56      mBtnSkip = (Button) findViewById(R.id.btn_login_skip);
57
58      //mChkRemember = (CheckBox) findViewById(R.id.chk_login_remember);
59      //mChkSkip = (CheckBox) findViewById(R.id.chk_login_skip);
60
61      mBtnLogin.setOnClickListener(new OnClickListener() {
62
63        public void onClick(View v) {
64
65          Intent i = new Intent(LoginActivity.this, MainActivity.class);
66                  startActivity(i);
67        }
68      });
69      mBtnSkip.setOnClickListener(new OnClickListener() {
70
71        public void onClick(View v) {
```

```
72            TextView messageText = new TextView(LoginActivity.this);
73            messageText.setText(R.string.msg_skip);
74            messageText.setGravity(Gravity.FILL);
75            AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(
76                LoginActivity.this);
77            alertDialogBuilder.setTitle("Skip login?");
78            alertDialogBuilder
79                .setView(messageText)
80                .setCancelable(false)
81                .setPositiveButton("Yes",
82                    new DialogInterface.OnClickListener() {
83                      public void onClick(DialogInterface dialog,
84                          int id) {
85                        Intent i = new Intent(LoginActivity.this, MainActivity.class);
86                                  startActivity(i);
87                      }
88                    })
89                .setNegativeButton("No",
90                    new DialogInterface.OnClickListener() {
91                      public void onClick(DialogInterface dialog,
92                          int id) {
93                        dialog.cancel();
94                      }
95                    });
96            messageText.setGravity(Gravity.CENTER);
97            AlertDialog alertDialog = alertDialogBuilder.create();
98            alertDialog.show();
99          }
100       });
101     }
102
103   }
104
```

```
 1  package edu.seaaddicts.brockbutler;
 2
 3  import android.app.Activity;
 4  import android.content.Intent;
 5  import android.os.Bundle;
 6  import android.view.View;
 7  import android.view.View.OnClickListener;
 8  import android.widget.Button;
 9  import edu.seaaddicts.brockbutler.coursemanager.CourseManagerActivity;
10  import edu.seaaddicts.brockbutler.help.HelpActivity;
11  import edu.seaaddicts.brockbutler.maps.MapsActivity;
12  import edu.seaaddicts.brockbutler.scheduler.SchedulerActivity;
13  import edu.seaaddicts.brockbutler.tour.TourActivity;
14
15  public class MainActivity extends Activity {
16
17    private Button mSchedulerButton;
18    private Button mCourseManager;
19    private Button mMapsButton;
20    private Button mTourButton;
21    private Button mHelpButton;
22    View v[];
23
24    @Override
25    protected void onCreate(Bundle savedInstanceState) {
26      super.onCreate(savedInstanceState);
27      setContentView(R.layout.activity_main);
28      init();
29    }
30
31    private void init()
32    {
33      // Instantiate Buttons
34      mSchedulerButton = (Button) findViewById(R.id.btn_sched);
35      mCourseManager = (Button) findViewById(R.id.btn_contacts);
36      mMapsButton   = (Button) findViewById(R.id.btn_maps);
37      mTourButton   = (Button) findViewById(R.id.btn_tour);
38      mHelpButton   = (Button) findViewById(R.id.btn_help);
39
40      // Set OnClickListeners
41      mSchedulerButton.setOnClickListener(new OnClickListener() {
42        public void onClick(View v) {
43          Intent i = new Intent(MainActivity.this, SchedulerActivity.class);
44                  startActivity(i);
45        }
46      });
47      mCourseManager.setOnClickListener(new OnClickListener() {
48        public void onClick(View v) {
49          Intent i = new Intent(MainActivity.this, CourseManagerActivity.class);
50                  startActivity(i);
51        }
52      });
53      mMapsButton.setOnClickListener(new OnClickListener() {
54        public void onClick(View v) {
55          Intent i = new Intent(MainActivity.this, MapsActivity.class);
56                  startActivity(i);
57        }
58      });
59      mTourButton.setOnClickListener(new OnClickListener() {
60        public void onClick(View v) {
61          Intent i = new Intent(MainActivity.this, TourActivity.class);
62                  startActivity(i);
63        }
64      });
65      mHelpButton.setOnClickListener(new OnClickListener() {
66        public void onClick(View v) {
67          Intent i = new Intent(MainActivity.this, HelpActivity.class);
68          Bundle bundle = new Bundle();
69          bundle.putString("activity","main");
70          i.putExtras(bundle);
71                  startActivity(i);
```

```
72          }
73        });
74
75      }
76    }
77
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000"
    tools:context=".LoginActivity" >

    <TextView
        android:id="@+id/app_logo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dip"
        android:text="@string/app_name"
        android:textSize="30sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/app_logo2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dip"
        android:layout_below="@id/app_logo"
        android:text="@string/temp"
        android:textSize="20sp"
        android:textStyle="bold" />

    <TableLayout
        android:id="@+id/login_table"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true" >

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:padding="10dip" >

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/login_email"
                android:textSize="20sp" />

            <EditText
                android:id="@+id/login_email"
                android:layout_width="200dip"
                android:layout_height="wrap_content"
                android:layout_marginLeft="45dip"
                android:inputType="textEmailAddress" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:padding="10dip" >

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/login_password"
                android:textSize="20sp" />

            <EditText
                android:id="@+id/login_password"
                android:layout_width="200dip"
                android:layout_height="wrap_content"
                android:layout_marginLeft="5dip"
                android:inputType="textPassword" />
        </LinearLayout>

        <LinearLayout
```

```
72              android:layout_width="wrap_content"
73              android:layout_height="wrap_content"
74              android:padding="10dip" >
75
76              <Button
77                  android:id="@+id/btn_login_login"
78                  android:layout_width="150dip"
79                  android:layout_height="wrap_content"
80                  android:text="@string/login"
81                  android:textSize="20sp" >
82              </Button>
83
84              <Button
85                  android:id="@+id/btn_login_skip"
86                  android:layout_width="150dip"
87                  android:layout_height="wrap_content"
88                  android:text="@string/login_skip"
89                  android:textSize="20sp" >
90              </Button>
91          </LinearLayout>
92      </TableLayout>
93
94
95      <TextView
96          android:id="@+id/login_remember"
97          android:layout_width="wrap_content"
98          android:layout_height="wrap_content"
99          android:layout_below="@+id/login_table"
100         android:layout_marginLeft="16dp"
101         android:layout_marginTop="10dip"
102         android:layout_toRightOf="@+id/app_logo"
103         android:text="@string/login_remember" />
104
105     <CheckBox
106         android:id="@+id/chk_login_remember"
107         android:layout_width="wrap_content"
108         android:layout_height="wrap_content"
109         android:layout_alignBaseline="@+id/login_remember"
110         android:layout_alignBottom="@+id/login_remember"
111         android:layout_toRightOf="@+id/login_remember" />
112
113     <TextView
114         android:id="@+id/login_skip"
115         android:layout_width="wrap_content"
116         android:layout_height="wrap_content"
117         android:layout_below="@id/login_remember"
118         android:layout_marginTop="30dip"
119         android:layout_toRightOf="@+id/app_logo"
120
121         android:text="@string/login_skip_msg" />
122
123     <CheckBox
124         android:id="@+id/chk_login_skip"
125         android:layout_width="wrap_content"
126         android:layout_height="wrap_content"
127         android:layout_alignBaseline="@+id/login_skip"
128         android:layout_alignBottom="@+id/login_skip"
129         android:layout_toRightOf="@+id/login_remember" />
130
131 </RelativeLayout>
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <LinearLayout
        android:id="@+id/main_icon_row1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="5sp"
        android:layout_centerHorizontal="true"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btn_sched"
            android:layout_width="135sp"
            android:layout_height="135sp"
            android:layout_margin="15sp"
            android:contentDescription="@string/contentdesc"
            android:background="@drawable/ic_scheduler" />

        <Button
            android:id="@+id/btn_contacts"
            android:layout_width="135sp"
            android:layout_height="135sp"
            android:layout_margin="15sp"
            android:contentDescription="@string/contentdesc"
            android:background="@drawable/ic_course" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/main_icon_row2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btn_maps"
            android:layout_width="135sp"
            android:layout_height="135sp"
            android:layout_margin="15sp"
            android:contentDescription="@string/contentdesc"
            android:background="@drawable/ic_maps" />

        <Button
            android:id="@+id/btn_tour"
            android:layout_width="135sp"
            android:layout_height="135sp"
            android:layout_margin="15sp"
            android:contentDescription="@string/contentdesc"
            android:background="@drawable/ic_tour" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/main_icon_row3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btn_help"
            android:layout_width="135sp"
            android:layout_height="135sp"
            android:layout_marginBottom="15sp"
            android:contentDescription="@string/contentdesc"
            android:background="@drawable/ic_help" />
    </LinearLayout>
```

```
72
73    </RelativeLayout>
```

```
 1  package edu.seaaddicts.brockbutler.maps;
 2
 3  /**
 4   * Locate.java
 5   * Brock Butler
 6   * portion of Brock Butler.
 7   * Created by Thomas Nelson 2013-03-10
 8   * Copyright (c) 2013 Sea Addicts. All rights reserved.
 9   */
10
11  import java.util.List;
12  import android.content.Context;
13  import android.net.wifi.ScanResult;
14  import android.net.wifi.WifiManager;
15  import android.util.Log;
16
17  public class Locate {
18
19    /**
20     * Class variables
21     */
22    private static WifiManager      wifiMgr;
23    private static List<ScanResult> scanResults;
24    static Context parentContext;
25    int[] answer = new int[10];
26    /**
27     * Wireless information containers
28     */
29    private static int[] sigStr  = new int[10];
30    private static String[] address = new String[10];
31    private static double[] addIn = new double[10];
32    /**
33     * Layers
34     */
35    private static final int inputs = 2;
36    private static final int hidden = 8;
37    private static final int output = 5;
38    /**
39     * Weights
40     */
41    private static double[][] W = new double[inputs][hidden];
42    private static double[][] V = new double[hidden][output];
43    private static double[] HB = new double[hidden];
44    private static double[] OB = new double[output];
45    /**
46     * Neurons
47     */
48    private static double[] hiddenVal = new double[hidden];
49    private static double[] outputVal = new double[output];
50    private static double[][] inputVal = new double[10][inputs];
51
52    /**
53     * Constructor for the Locate class sets context and initializes weights
54     * only once.
55     * @param pc
56     */
57    public Locate (Context pc) {
58      parentContext = pc;
59      initWeights();
60    }
61
62    /**
63     * Used by the mapping thread to get current user position, returns null if no
       position is found.
64     * @return
65     */
66    public String getUserPosition ( ) {
67      try {
68        //getWirelessData(); // Use when you are testing on device
69        initTestData(); // Use when testing on simulator testData1, testData2, or
          testData3
```

```
70          //initData(); // Use when you are testing on device
71
72          for(int i=0; i<10; i++) {
73            calcNetwork(i);
74            answer[i] = (int) (outputVal[0]*16 + outputVal[1]*8 + outputVal[2]*4 +
              outputVal[3]*2 + outputVal[4]*1);
75          }
76
77          int location = mode(answer);
78          Log.i("LOCATE", "Node: " + location);
79          switch(location) {
80            case 1:
81              return "J01";
82            case 2:
83              return "J02";
84            case 3:
85              return "J03";
86            case 4:
87              return "J04";
88            case 5:
89              return "J05";
90            case 6:
91              return "J06";
92            case 7:
93              return "J07";
94            case 8:
95              return "J08";
96            case 9:
97              return "J09";
98            case 10:
99              return "J10";
100           case 11:
101             return "J11";
102           case 12:
103             return "J12";
104           case 13:
105             return "J13";
106           case 14:
107             return "J14";
108           case 15:
109             return "J15";
110           case 16:
111             return "J16";
112           case 17:
113             return "J17";
114           case 18:
115             return "J18";
116           case 19:
117             return "J19";
118           case 20:
119             return "J20";
120           case 21:
121             return "J21";
122           case 123:
123             return "J22";
124           case 23:
125             return "J23";
126           default:
127             return "";
128         }
129       } catch (Exception err) {
130         Log.e("LOCATE", err.getMessage());
131       }
132       return null;
133     }
134
135     /**
136      * Gathers wireless information from the device for 10 wireless access points
137      * currently in range. Gathers MAC address and received signal strength
138      */
139     @SuppressWarnings("unused")
```

```
140     private static void getWirelessData() {
141       wifiMgr = (WifiManager)parentContext.getSystemService(Context.WIFI_SERVICE);
142         int x = 0;
143 
144         for(int num=0; num<10; num++) {
145           wifiMgr.startScan();
146           scanResults = wifiMgr.getScanResults();
147 
148           x = 0;
149           sigStr  = new int[10];
150           address = new String[10];
151 
152           for(ScanResult scanRes : scanResults) {
153             if(x < 10) {
154               address[x] = scanRes.BSSID;
155               sigStr[x] = scanRes.level;
156               x++;
157             }
158           }
159         }
160     }
161 
162     /**
163      * This Method will return the sigmoid value of an argument
164      * for the final node value.
165      * @param x
166      * @return
167      */
168     private static double sigmoid(double x) {
169       return 1 / (1 + Math.exp(-x));
170     }
171 
172     /**
173      * The beans of this class, uses normalized wireless data to predict
174      * user location based on an inputed input pattern.
175      * @param pat
176      */
177     private static void calcNetwork(int pat) {
178       for(int h=0; h<hidden; h++) {
179         hiddenVal[h] = -HB[h];
180         for(int i=0; i<inputs; i++) {
181           hiddenVal[h] += (inputVal[pat][i] * W[i][h]);
182         }
183         hiddenVal[h] = sigmoid(hiddenVal[h]);
184       }
185 
186       for(int o=0; o<output; o++) {
187         outputVal[o] = -OB[o];
188         for(int h=0; h<hidden; h++) {
189           outputVal[o] += (hiddenVal[h] * V[h][o]);
190         }
191         outputVal[o] = sigmoid(outputVal[o]);
192 
193         if(outputVal[o] >= 0.5)
194           outputVal[o] = 1;
195           else if(outputVal[o] < 0.5)
196             outputVal[o] = 0;
197       }
198     }
199 
200     /**
201      * Searches through network output to find the most likely
202      * user position.
203      * @param a
204      * @return
205      */
206     private static int mode(int a[]) {
207       int maxValue=0, maxCount=0;
208 
209       for (int i = 0; i < a.length; ++i) {
210         int count = 0;
```

```
211         for (int j = 0; j < a.length; ++j) {
212           if (a[j] == a[i]) ++count;
213         }
214         if (count > maxCount) {
215           maxCount = count;
216           maxValue = a[i];
217         }
218       }
219
220       return maxValue;
221     }
222
223     private void initTestData() {
224       inputVal[0][0] = 1; inputVal[0][1] = -65;
225       inputVal[1][0] = 2; inputVal[1][1] = -60;
226       inputVal[2][0] = 3; inputVal[2][1] = -64;
227       inputVal[3][0] = 4; inputVal[3][1] = -64;
228       inputVal[4][0] = 5; inputVal[4][1] = -68;
229       inputVal[5][0] = 6; inputVal[5][1] = -69;
230       inputVal[6][0] = 7; inputVal[6][1] = -69;
231       inputVal[7][0] = 8; inputVal[7][1] = -72;
232       inputVal[8][0] = 9; inputVal[8][1] = -74;
233       inputVal[9][0] = 10; inputVal[9][1] = -74;
234     }
235
236     @SuppressWarnings("unused")
237     private void initTestData2() {
238       inputVal[0][0] = 1; inputVal[0][1] = -65;
239       inputVal[1][0] = 1; inputVal[1][1] = -89;
240       inputVal[2][0] = 6; inputVal[2][1] = -64;
241       inputVal[3][0] = 8; inputVal[3][1] = -64;
242       inputVal[4][0] = 11; inputVal[4][1] = -66;
243       inputVal[5][0] = 12; inputVal[5][1] = -66;
244       inputVal[6][0] = 13; inputVal[6][1] = -71;
245       inputVal[7][0] = 14; inputVal[7][1] = -72;
246       inputVal[8][0] = 15; inputVal[8][1] = -72;
247       inputVal[9][0] = 16; inputVal[9][1] = -72;
248     }
249
250     @SuppressWarnings("unused")
251     private void initTestData3() {
252       inputVal[0][0] = 6; inputVal[0][1] = -58;
253       inputVal[1][0] = 8; inputVal[1][1] = -58;
254       inputVal[2][0] = 11; inputVal[2][1] = -72;
255       inputVal[3][0] = 12; inputVal[3][1] = -72;
256       inputVal[4][0] = 13; inputVal[4][1] = -75;
257       inputVal[5][0] = 14; inputVal[5][1] = -66;
258       inputVal[6][0] = 15; inputVal[6][1] = -77;
259       inputVal[7][0] = 16; inputVal[7][1] = -75;
260       inputVal[8][0] = 9; inputVal[8][1] = -69;
261       inputVal[9][0] = 16; inputVal[9][1] = -72;
262     }
263
264     /**
265      * Makes the wireless data usable and sets it up for
266      * the network to use by putting the values between 0 and 1
267      * with min/max normalization.
268      */
269     @SuppressWarnings("unused")
270     private void initData() {
271       for (int x=0; x<10; x++) {
272         if (address[x].equalsIgnoreCase("00:0b:86:91:ce:a1"))
273           addIn[x] = 1;
274         else if (address[x].equalsIgnoreCase("00:0b:86:8a:8c:02"))
275           addIn[x] = 2;
276         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:af:21"))
277           addIn[x] = 3;
278         else if (address[x].equalsIgnoreCase("00:0b:86:91:ce:a2"))
279           addIn[x] = 4;
280         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:b0:e2"))
281           addIn[x] = 5;
```

```
282         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:b0:e1"))
283           addIn[x] = 6;
284         else if (address[x].equalsIgnoreCase("00:0b:86:89:f6:e1"))
285           addIn[x] = 7;
286         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:af:22"))
287           addIn[x] = 8;
288         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:b2:62"))
289           addIn[x] = 9;
290         else if (address[x].equalsIgnoreCase("00:0b:86:4d:8f:21"))
291           addIn[x] = 10;
292         else if (address[x].equalsIgnoreCase("00:0b:86:4d:8f:22"))
293           addIn[x] = 11;
294         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:b0:21"))
295           addIn[x] = 12;
296         else if (address[x].equalsIgnoreCase("00:1a:1e:a7:dc:22"))
297           addIn[x] = 13;
298         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:b0:22"))
299           addIn[x] = 14;
300         else if (address[x].equalsIgnoreCase("00:1a:1e:a7:dc:21"))
301           addIn[x] = 15;
302         else if (address[x].equalsIgnoreCase("00:0b:86:8a:8c:01"))
303           addIn[x] = 16;
304         else if (address[x].equalsIgnoreCase("00:1a:1e:a7:e4:c2"))
305           addIn[x] = 17;
306         else if (address[x].equalsIgnoreCase("00:1a:1e:a7:e4:c1"))
307           addIn[x] = 18;
308         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:ac:82"))
309           addIn[x] = 19;
310         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:ac:81"))
311           addIn[x] = 20;
312         else if (address[x].equalsIgnoreCase("00:0b:86:91:ce:a0"))
313           addIn[x] = 21;
314         else if (address[x].equalsIgnoreCase("00:0b:86:8a:8c:00"))
315           addIn[x] = 22;
316         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:ac:80"))
317           addIn[x] = 23;
318         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:b2:61"))
319           addIn[x] = 24;
320         else if (address[x].equalsIgnoreCase("00:0b:86:42:de:80"))
321           addIn[x] = 25;
322         else if (address[x].equalsIgnoreCase("00:0b:86:42:de:82"))
323           addIn[x] = 26;
324         else if (address[x].equalsIgnoreCase("00:1a:1e:a7:e4:c0"))
325           addIn[x] = 27;
326         else if (address[x].equalsIgnoreCase("00:1a:1e:fc:b2:60"))
327           addIn[x] = 28;
328         else
329           addIn[x] = 0;
330       }
331
332       for (int x=0; x<10; x++) {
333         inputVal[x][0] = (addIn[x] - 1) / 28;
334         inputVal[x][1] = (sigStr[x] - -97) / 58;
335       }
336     }
337
338     /**
339      * Initializes the network with pre-defined weights, currently will only find a
340      * position in JBlock.
341      */
342     private void initWeights ( ) {
343       W[0][0] = -5.191953555370145;
344       W[1][0] = 8.311119623052747;
345       HB[0] = 13.645070679100112;
346       V[0][0] = -1.4188817448241078;
347       OB[0] = 8.485931304116875;
348       V[0][1] = -0.12644482595532947;
349       OB[1] = 7.676266130312892;
350       V[0][2] = -0.8742897792429708;
351       OB[2] = 7.71163044603109;
352       V[0][3] = 8.174930730213324;
```

```
353         OB[3] = 5.050964297399435;
354         V[0][4] = -1.483533992384484;
355         OB[4] = -0.9537318879960314;
356         W[0][1] = 1.2747639658533194;
357         W[1][1] = 35.85241447388153;
358         HB[1] = 22.187068222811735;
359         V[1][0] = 0.08863280595533382;
360         OB[0] = 8.485931304116875;
361         V[1][1] = -0.711381178420192;
362         OB[1] = 7.676266130312892;
363         V[1][2] = 0.47717648845370575;
364         OB[2] = 7.71163044603109;
365         V[1][3] = -0.9106351920878477;
366         OB[3] = 5.050964297399435;
367         V[1][4] = -21.858140121995646;
368         OB[4] = -0.9537318879960314;
369         W[0][2] = 72.6159523077509;
370         W[1][2] = 62.83113577555713;
371         HB[2] = 48.81891128647234;
372         V[2][0] = -0.2564556186462736;
373         OB[0] = 8.485931304116875;
374         V[2][1] = -1.3189848572962326;
375         OB[1] = 7.676266130312892;
376         V[2][2] = -1.1323714524773l2;
377         OB[2] = 7.71163044603109;
378         V[2][3] = 6.23102460219137;
379         OB[3] = 5.050964297399435;
380         V[2][4] = 0.9547440817617039;
381         OB[4] = -0.9537318879960314;
382         W[0][3] = -12.93760218402065;
383         W[1][3] = -27.44465213223537;
384         HB[3] = -12.785991014176767;
385         V[3][0] = -0.3900489003340711;
386         OB[0] = 8.485931304116875;
387         V[3][1] = -0.4526342230399839;
388         OB[1] = 7.676266130312892;
389         V[3][2] = -1.1900195982331039;
390         OB[2] = 7.71163044603109;
391         V[3][3] = 32.266073798358136;
392         OB[3] = 5.050964297399435;
393         V[3][4] = 0.8870702490085969;
394         OB[4] = -0.9537318879960314;
395         W[0][4] = 20.564192571900584;
396         W[1][4] = 64.556122443447;
397         HB[4] = 51.94209137066473;
398         V[4][0] = -0.14542636219949362;
399         OB[0] = 8.485931304116875;
400         V[4][1] = -0.18725148477033168;
401         OB[1] = 7.676266130312892;
402         V[4][2] = -1.3082405929431982;
403         OB[2] = 7.71163044603109;
404         V[4][3] = 7.750490464388548;
405         OB[3] = 5.050964297399435;
406         V[4][4] = 7.381930098882;
407         OB[4] = -0.9537318879960314;
408         W[0][5] = 26.203506852804754;
409         W[1][5] = 30.430410744252843;
410         HB[5] = 30.471677989844203;
411         V[5][0] = 0.6785829497339613;
412         OB[0] = 8.485931304116875;
413         V[5][1] = -0.7290230440401178;
414         OB[1] = 7.676266130312892;
415         V[5][2] = 0.01863788955350768;
416         OB[2] = 7.71163044603109;
417         V[5][3] = 13.290562812876107;
418         OB[3] = 5.050964297399435;
419         V[5][4] = -3.191226914646271;
420         OB[4] = -0.9537318879960314;
421         W[0][6] = -4.992559685945157;
422         W[1][6] = 88.04743967482369;
423         HB[6] = 50.39739242687603;
```

```
424         V[6][0] = 0.0746839836828444;
425         OB[0] = 8.485931304116875;
426         V[6][1] = -0.5879731640198912;
427         OB[1] = 7.676266130312892;
428         V[6][2] = -0.25250658203018556;
429         OB[2] = 7.71163044603109;
430         V[6][3] = 1.4602581997464505;
431         OB[3] = 5.050964297399435;
432         V[6][4] = 18.463904409636967;
433         OB[4] = -0.9537318879960314;
434         W[0][7] = -4.74928985780168;
435         W[1][7] = 40.78597102067532;
436         HB[7] = 35.79898233956728;
437         V[7][0] = -0.09439880889024627;
438         OB[0] = 8.485931304116875;
439         V[7][1] = -1.1230801185236317;
440         OB[1] = 7.676266130312892;
441         V[7][2] = -0.2674705623236563;
442         OB[2] = 7.71163044603109;
443         V[7][3] = 8.441254951069187;
444         OB[3] = 5.050964297399435;
445         V[7][4] = -20.04591503798929;
446         OB[4] = -0.9537318879960314;
447       }
448     }
449
450
```

```java
package edu.seaaddicts.brockbutler.maps;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.Toast;
import edu.seaaddicts.brockbutler.R;
import edu.seaaddicts.brockbutler.help.HelpActivity;

public class MapsActivity extends Activity {
  private static final String TAG = "MapsActivity";

  private EditText mSearchEditText;

  private Handler mHandler;
  private MapsTouchImageView mMapImage;
  private MapsHandler mMapsHandler;

  private Position mStartPosition;
  private Position mGoalPosition;
  private Astar school;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    getWindow().requestFeature(Window.FEATURE_ACTION_BAR_OVERLAY);
    setContentView(R.layout.activity_map);

    RelativeLayout layout = (RelativeLayout) findViewById(R.id.maps_layout);
    layout.setSystemUiVisibility(View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN);
    getActionBar().setBackgroundDrawable(
        getResources().getDrawable(R.drawable.actionbar_bg));
    init();

    mHandler = new Handler() {

      // Handles Messages sent from Thomas.
      @Override
      public void handleMessage(Message msg) {
        switch (msg.what) {
        case MapsHandler.MAPS_REQUEST_UPDATE:
          Log.d("MAIN HANDLER", "YAYAAA!!!");
          break;
        case MapsHandler.THREAD_UPDATE_POSITION:
          Log.d("TEST",msg.getData().getString("pos"));
          break;
        default:
          Log.d(TAG,"-----+++++ Got THREAD_UPDATE_POSITION message. +++++-----");
          break;
        }
      }
    };
    mMapsHandler = new MapsHandler(mHandler,this);
        school = new Astar();
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
```

```
72        getMenuInflater().inflate(R.menu.activity_map, menu);
73        return true;
74      }
75
76      private void init() {
77        mMapImage = (MapsTouchImageView) findViewById(R.id.imgv_map);
78        // mMapImage.setOnClickListener(new OnClickListener() {
79        //
80        // public void onClick(View v) {
81        // mMapsHandler.sendEmptyMessage(MapsHandler.MAPS_REQUEST_UPDATE);
82        // }
83        // });
84        // start = (Button) findViewById(R.id.btnstart);
85        // start.setOnClickListener(new OnClickListener() {
86        //
87        // public void onClick(View v) {
88        // mMapsHandler.sendEmptyMessage(MapsHandler.THREAD_REQUEST_START);
89        // }
90        // });
91        // stop = (Button) findViewById(R.id.btnstop);
92        // stop.setOnClickListener(new OnClickListener() {
93        //
94        // public void onClick(View v) {
95        // mMapsHandler.sendEmptyMessage(MapsHandler.THREAD_REQUEST_PAUSE);
96        // }
97        // });
98        //
99        // resume = (Button) findViewById(R.id.btnresume);
100       // resume.setOnClickListener(new OnClickListener() {
101       //
102       // public void onClick(View v) {
103       // mMapsHandler
104       // .sendEmptyMessage(MapsHandler.THREAD_REQUEST_RESUME);
105       // }
106       // });
107     }
108
109     @Override
110     public void onBackPressed() {
111       mMapsHandler.sendEmptyMessage(MapsHandler.THREAD_REQUEST_PAUSE);
112       mMapsHandler = null;
113       super.onBackPressed();
114     }
115
116     /**
117      *
118      * @param item
119      */
120     public void exitMaps(MenuItem item) {
121       onBackPressed();
122     }
123
124     /**
125      * Prompts user to search for existence of a location.
126      *
127      * @param item
128      */
129     public void displaySearchDialog(MenuItem item) {
130       AlertDialog.Builder editalert = new AlertDialog.Builder(this);
131
132       editalert.setTitle("MChown Location Search");
133       editalert.setMessage("Enter block or room name (i.e. B314)");
134
135       mSearchEditText = new EditText(this);
136       mSearchEditText.setSingleLine(true);
137       LinearLayout.LayoutParams lp = new LinearLayout.LayoutParams(
138           LinearLayout.LayoutParams.MATCH_PARENT,
139           LinearLayout.LayoutParams.MATCH_PARENT);
140       mSearchEditText.setLayoutParams(lp);
141       editalert.setView(mSearchEditText);
142
```

```
143       editalert.setPositiveButton("Search",
144           new DialogInterface.OnClickListener() {
145             public void onClick(DialogInterface dialog, int whichButton) {
146               try {
147                 Position p1, p2=new Position(), p3=new Position(), p4=new Position(),
                    p5=new Position();
148
149                 p1 = school.findPosition(mSearchEditText.getText().toString());
150
151                 p2.xPosition = p1.xPosition - 1;
152                 p2.yPosition = p1.yPosition - 1;
153
154                 p3.xPosition = p1.xPosition + 1;
155                 p3.yPosition = p1.yPosition + 1;
156
157                 p4.xPosition = p1.xPosition + 1;
158                 p4.yPosition = p1.yPosition - 1;
159
160                 p5.xPosition = p1.xPosition - 1;
161                 p5.yPosition = p1.yPosition + 1;
162
163                 Position[] pTest = {p5,p3,p4,p2,p5,p4,p3,p2};
164
165                 if(p1 != null && p2 != null)
166                   mMapImage.drawPosition(pTest,30);
167                 }
168               catch (NullPointerException e) {
169                 Toast.makeText(getApplicationContext(), "Location not found",Toast.
                    LENGTH_LONG).show();
170               }
171             }
172           });
173       editalert.setNegativeButton("Cancel",
174           new DialogInterface.OnClickListener() {
175             public void onClick(DialogInterface dialog, int whichButton) {
176               // do nothing
177             }
178           });
179
180       editalert.show();
181     }
182
183     /**
184      * Displays AlertDialog for user to enter destination. First the location is
185      * determined to exist, then if true path is drawn on map.
186      *
187      * @param item
188      */
189     public void displayGetDirectionsDialog(MenuItem item) {
190       school = new Astar();
191       AlertDialog.Builder alert = new AlertDialog.Builder(this);
192
193       alert.setTitle("Get Directions");
194       alert.setMessage("Enter Start and End Points");
195
196       LinearLayout lila1= new LinearLayout(this);
197         lila1.setOrientation(1); //1 is for vertical orientation
198         final EditText input = new EditText(this);
199         final EditText input1 = new EditText(this);
200         lila1.addView(input);
201         lila1.addView(input1);
202         alert.setView(lila1);
203
204       alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
205         public void onClick(DialogInterface dialog, int whichButton) {
206           try {
207             mStartPosition = school.findPosition(input.getText().toString());
208             mGoalPosition = school.findPosition(input1.getText().toString());
209
210             Position[] p = school.pathGeneration(mStartPosition, mGoalPosition);
211             if(mStartPosition != null && mGoalPosition != null)
```

```
212                 mMapImage.drawPosition(p,8);
213             }
214             catch (NullPointerException e){
215               Toast.makeText(getApplicationContext(), "Location not found",Toast.
                  LENGTH_LONG).show();
216             }
217           }
218         });
219
220         alert.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
221           public void onClick(DialogInterface dialog, int whichButton) {
222             // Canceled.
223           }
224         });
225
226         alert.show();
227       }
228
229       public void showHelp(MenuItem item)
230       {
231         Intent intent = new Intent(MapsActivity.this,HelpActivity.class);
232         Bundle bundle = new Bundle();
233         bundle.putString("activity", "maps");
234         intent.putExtras(bundle);
235         startActivity(intent);
236       }
237
238       /**
239        * Menu item onClick that calls mapping function to manual update user
240        * location in case first reported location isnot accurate.
241        * @param item
242        */
243       public void updatePosition(MenuItem item) {
244         Toast.makeText(getApplicationContext(), "Thomas' method goes here.",
245             Toast.LENGTH_LONG).show();
246         // call Thomas' method to update current position.
247       }
248     }
249
```

```
 1  package edu.seaaddicts.brockbutler.maps;
 2
 3  import android.content.Context;
 4  import android.os.Handler;
 5  import android.os.Looper;
 6  import android.os.Message;
 7  import android.util.Log;
 8
 9  public class MapsHandler extends Handler {
10
11    public static final int MAPS_REQUEST_UPDATE = 0x001;
12    public static final int MAPS_REQUEST_LOCATION_EXISTS = 0x002;
13    public static final int MAPS_REQUEST_DIRECTION = 0x003;
14
15    public static final int MAPS_SEND_POSITION = 0x004;
16    public static final int MAPS_SEND_DIRECTIONS = 0x005;
17
18    public static final int MAPS_ERROR_NO_LOCATION = 0x006;
19    public static final int MAPS_ERROR_NO_WIFI = 0x007;
20
21    public static final int THREAD_REQUEST_START = 0x008;
22    public static final int THREAD_REQUEST_STOP = 0x009;
23    public static final int THREAD_REQUEST_PAUSE = 0x010;
24    public static final int THREAD_REQUEST_RESUME = 0x011;
25
26    public static final int THREAD_UPDATE_POSITION = 0x012;
27
28    private static final String tag = "MapsHandler";
29    private Handler mMainHandler;
30    private Thread mMapsThread;
31
32    private Object mPauseLock;
33    private boolean mIsPaused;
34    private boolean mIsFinished;
35
36    public MapsHandler(Looper main, Context c) {
37      super(main);
38      Log.d(tag, "-----++++++ Creating Handler from Looper ++++++------");
39      mMainHandler = new Handler(main);
40      mIsPaused = true;
41      init();
42    }
43
44    public MapsHandler(Handler main, Context c) {
45      Log.d(tag, "-----++++++ Creating Handler. ++++++------");
46      mMainHandler = main;
47      mIsPaused = true;
48      init();
49    }
50
51    private void init() {
52      mPauseLock = new Object();
53
54      // Set up Maps Thread
55      mMapsThread = new Thread() {
56
57        int count = 20;
58
59
60        @Override
61        public void run() {
62          Log.d(tag, "Thread started.");
63
64          while (!mIsFinished) {
65            // do your stuff here
66
67            mMainHandler.sendEmptyMessage(count);
68            try {
69              Thread.sleep(3000);
70            } catch (InterruptedException e) {
71              // TODO Auto-generated catch block
```

```
72              e.printStackTrace();
73            }
74            count += 1;
75            synchronized (mPauseLock) {
76              while (mIsPaused) {
77                try {
78                  mPauseLock.wait();
79                  Log.d(tag,
80                      "-----++++++ Thread paused. ++++++------");
81                } catch (InterruptedException e) {
82                  e.printStackTrace();
83                }
84              }
85            }
86          }
87        }
88      };
89    }
90
91    @Override
92    public void handleMessage(Message msg) {
93      switch (msg.what) {
94
95      case MAPS_REQUEST_UPDATE:
96        Log.d(tag, "-----+++++ Sending update to MapsActivity. +++++-----");
97        mMainHandler.sendEmptyMessage(MAPS_SEND_POSITION);
98        break;
99
100     case MAPS_REQUEST_LOCATION_EXISTS:
101       Log.d(tag, "-----+++++ Checking for location. +++++-----");
102
103       // Runs Thomas' code to check for existence.
104
105       // if (mDoesExist)
106       // send information
107       // else
108       // mMainHandler.sendEmptyMessage(MAPS_ERROR_NO_LOCATION);
109       break;
110
111     case MAPS_REQUEST_DIRECTION:
112       Log.d(tag, "-----+++++ Getting directions. +++++-----");
113       mMainHandler.sendEmptyMessage(MAPS_SEND_DIRECTIONS);
114       break;
115     case THREAD_REQUEST_START:
116       if (!mMapsThread.isAlive()) {
117         Log.d(tag, "-----+++++ Starting thread. +++++-----");
118         mMapsThread.start();
119         mIsPaused = false;
120       }
121       break;
122     case THREAD_REQUEST_PAUSE:
123       if (!mIsPaused) {
124         synchronized (mPauseLock) {
125           Log.d(tag, "-----+++++ Pausing thread. +++++-----");
126           mIsPaused = true;
127         }
128       }
129       break;
130     case THREAD_REQUEST_RESUME:
131       if (mIsPaused) {
132         synchronized (mPauseLock) {
133           Log.d(tag, "-----+++++ Resuming thread. +++++-----");
134           mIsPaused = false;
135           mPauseLock.notifyAll();
136         }
137       }
138       break;
139     default:
140       break;
141     }
142   }
```

```
143     }
144
```

```java
/**
 * Android: TouchImageView.java
 * Created by: Mike Ortiz
 * Updated by: Vince Pascuzzi
 * Date: 3/14/2013
 *
 * Allows pinching, zooming, translating, and drawing on an ImageView.
 */

package edu.seaaddicts.brockbutler.maps;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.graphics.PointF;
import android.graphics.drawable.Drawable;
import android.util.AttributeSet;
import android.util.Log;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.widget.ImageView;

public class MapsTouchImageView extends ImageView {
  //private static final String TAG = "MapsTouchImageView";

  @SuppressWarnings("unused")
  private static final int MAP_WIDTH = 2000;
  private static final int MAP_HEIGHT = 1100;
  private static final int CLICK = 3;

  private Matrix mMatrixMap;

  // States of touch.
  private static final int NONE = 0;
  private static final int DRAG = 1;
  private static final int ZOOM = 2;
  private int mode = NONE;
  private int stroke = 8;

  // Zooming variables.
  private PointF last = new PointF();
  private PointF start = new PointF();
  private float minScale = 1f;
  private float maxScale = 8f;
  private float[] m;

  // Ratio of screen resolution to map image resolution
  private double mMapRatio;

  private int viewWidth, viewHeight;
  @SuppressWarnings("unused")
  private int oldMeasuredWidth, oldMeasuredHeight;

  private float scaleFactor = 1f;
  private float origWidth, origHeight;

  private final Paint mPathPaint = new Paint();

  private ScaleGestureDetector mScaleDetector;

  //private Context mContext;
  int actionBarHeight;

  public Position[] mPosition = null;

  public MapsTouchImageView(Context context) {
    super(context);
    sharedConstructing(context);
```

```
72      }
73
74      public MapsTouchImageView(Context context, AttributeSet attrs) {
75        super(context, attrs);
76        sharedConstructing(context);
77      }
78
79      @Override
80      protected void onDraw(Canvas canvas) {
81        super.onDraw(canvas);
82        mPathPaint.setColor(Color.CYAN);
83        mPathPaint.setStrokeWidth(stroke);
84        canvas.setMatrix(mMatrixMap);
85
86        if (mPosition != null) {
87          for (int i = 0; i < mPosition.length - 1; i++) {
88            Position p = mPosition[i];
89            Position q = mPosition[i + 1];
90            int x1 = p.xPosition;
91            int y1 = p.yPosition;
92            float[] f1 = convertDimensions(x1, y1);
93            int x2 = q.xPosition;
94            int y2 = q.yPosition;
95            float[] f2 = convertDimensions(x2, y2);
96            canvas.drawLine(f1[0], f1[1], f2[0], f2[1], mPathPaint);
97          }
98        }
99      }
100
101     private void sharedConstructing(Context context) {
102       super.setClickable(true);
103       //this.mContext = context;
104       mScaleDetector = new ScaleGestureDetector(context, new ScaleListener());
105       mMatrixMap = new Matrix();
106       m = new float[9];
107       setImageMatrix(mMatrixMap);
108       setScaleType(ScaleType.MATRIX);
109
110       setOnTouchListener(new OnTouchListener() {
111
112         public boolean onTouch(View v, MotionEvent event) {
113           mScaleDetector.onTouchEvent(event);
114           PointF curr = new PointF(event.getX(), event.getY());
115
116           switch (event.getAction()) {
117           case MotionEvent.ACTION_DOWN:
118             last.set(curr);
119             start.set(last);
120             mode = DRAG;
121             break;
122
123           case MotionEvent.ACTION_MOVE:
124             float fixTransX;
125             float fixTransY;
126             if (mode == DRAG) {
127               float deltaX = curr.x - last.x;
128               float deltaY = curr.y - last.y;
129               fixTransX = getFixDragTrans(deltaX, viewWidth,
130                   origWidth * scaleFactor);
131               fixTransY = getFixDragTrans(deltaY, viewHeight,
132                   origHeight * scaleFactor);
133               mMatrixMap.postTranslate(fixTransX, fixTransY);
134               fixTrans();
135               last.set(curr.x, curr.y);
136             }
137             break;
138
139           case MotionEvent.ACTION_UP:
140             mode = NONE;
141             int xDiff = (int) Math.abs(curr.x - start.x);
142             int yDiff = (int) Math.abs(curr.y - start.y);
```

```
143
144             if (xDiff < CLICK && yDiff < CLICK)
145               performClick();
146             break;
147
148           case MotionEvent.ACTION_POINTER_UP:
149             mode = NONE;
150             break;
151           }
152
153           setImageMatrix(mMatrixMap);
154           invalidate();
155           return true; // indicate event was handled
156         }
157
158       });
159     }
160
161     public void setMaxZoom(float x) {
162       maxScale = x;
163     }
164
165     private class ScaleListener extends
166         ScaleGestureDetector.SimpleOnScaleGestureListener {
167       @Override
168       public boolean onScaleBegin(ScaleGestureDetector detector) {
169         mode = ZOOM;
170         return true;
171       }
172
173       @Override
174       public boolean onScale(ScaleGestureDetector detector) {
175         float mScaleFactor = detector.getScaleFactor();
176         float origScale = scaleFactor;
177         scaleFactor *= mScaleFactor;
178         if (scaleFactor > maxScale) {
179           scaleFactor = maxScale;
180           mScaleFactor = maxScale / origScale;
181         } else if (scaleFactor < minScale) {
182           scaleFactor = minScale;
183           mScaleFactor = minScale / origScale;
184         }
185
186         if (origWidth * scaleFactor <= viewWidth
187             || origHeight * scaleFactor <= viewHeight)
188           mMatrixMap.postScale(mScaleFactor, mScaleFactor, viewWidth / 2,
189               viewHeight / 2);
190         else
191           mMatrixMap.postScale(mScaleFactor, mScaleFactor,
192               detector.getFocusX(), detector.getFocusY());
193         fixTrans();
194         return true;
195       }
196     }
197
198     void fixTrans() {
199       mMatrixMap.getValues(m);
200       float fixTransX;
201       float fixTransY;
202       float transX = m[Matrix.MTRANS_X];
203       float transY = m[Matrix.MTRANS_Y];
204
205       fixTransX = getFixTrans(transX, viewWidth, origWidth * scaleFactor);
206       fixTransY = getFixTrans(transY, viewHeight, origHeight * scaleFactor);
207
208       if (fixTransX != 0 || fixTransY != 0)
209         mMatrixMap.postTranslate(fixTransX, fixTransY);
210     }
211
212     /*
213      * Fixes (when required) the translation matrix.
```

```
214       */
215      float getFixTrans(float trans, float viewSize, float contentSize) {
216        float minTrans, maxTrans;
217
218        if (contentSize <= viewSize) {
219          minTrans = 0;
220          maxTrans = viewSize - contentSize;
221        } else {
222          minTrans = viewSize - contentSize;
223          maxTrans = 0;
224        }
225
226        if (trans < minTrans)
227          return -trans + minTrans;
228        if (trans > maxTrans)
229          return -trans + maxTrans;
230        return 0;
231      }
232
233      /*
234       * Adjusts the translation when dragging so that this stays in the correct
235       * location on screen.
236       */
237      float getFixDragTrans(float delta, float viewSize, float contentSize) {
238        if (contentSize <= viewSize) {
239          return 0;
240        }
241        return delta;
242      }
243
244      @Override
245      protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
246        super.onMeasure(widthMeasureSpec, heightMeasureSpec);
247        viewWidth = MeasureSpec.getSize(widthMeasureSpec);
248        viewHeight = MeasureSpec.getSize(heightMeasureSpec);
249
250        // Does image rescaling on rotation. Not necessary since our orientation
251        // is fixed in landscape.
252        if (oldMeasuredHeight == viewWidth && oldMeasuredHeight == viewHeight
253            || viewWidth == 0 || viewHeight == 0)
254          return;
255        oldMeasuredHeight = viewHeight;
256        oldMeasuredWidth = viewWidth;
257
258        if (scaleFactor == 1) {
259          // Fit to screen.
260          float scale;
261
262          Drawable drawable = getDrawable();
263          if (drawable == null || drawable.getIntrinsicWidth() == 0
264              || drawable.getIntrinsicHeight() == 0)
265            return;
266          int bmWidth = drawable.getIntrinsicWidth();
267          int bmHeight = drawable.getIntrinsicHeight();
268
269          mMapRatio = (double) (bmHeight) / (double) MAP_HEIGHT;
270
271          Log.d("bmSize", "bmWidth: " + bmWidth + " bmHeight : " + bmHeight
272              + "ratio" + mMapRatio);
273
274          float scaleX = (float) viewWidth / (float) bmWidth;
275          float scaleY = (float) viewHeight / (float) bmHeight;
276          scale = Math.min(scaleX, scaleY);
277          mMatrixMap.setScale(scale, scale);
278
279          // Center the image
280          float redundantYSpace = (float) viewHeight
281              - (scale * (float) bmHeight);
282          float redundantXSpace = (float) viewWidth
283              - (scale * (float) bmWidth);
284          redundantYSpace /= (float) 2;
```

```
285         redundantXSpace /= (float) 2;
286
287         mMatrixMap.postTranslate(redundantXSpace, redundantYSpace);
288
289         origWidth = viewWidth - 2 * redundantXSpace;
290         origHeight = viewHeight - 2 * redundantYSpace;
291         setImageMatrix(mMatrixMap);
292       }
293       fixTrans();
294     }
295
296     float[] convertDimensions(float x, float y) {
297       float f[] = new float[2];
298       f[0] = (float) mMapRatio * x;
299       f[1] = (float) mMapRatio * y;
300       return f;
301     }
302
303     public void drawPosition(Position[] p, int n) {
304       stroke = n;
305       mPosition = p;
306       invalidate();
307     }
308   }
```

```
 1  package edu.seaaddicts.brockbutler.maps;
 2
 3  /**
 4   * Position.java
 5   * Brock Butler
 6   * Type for holding Position node
 7   * portion of Brock Butler.
 8   * Created by Thomas Nelson 2013-03-05
 9   * Copyright (c) 2013 Sea Addicts. All rights reserved.
10   */
11
12  import android.util.Log;
13
14  public class Position implements Comparable<Object> {
15
16    /**
17     * Class variable for the POSITION class. All are public
18     * to avoid using get/set variables to increase performance
19     */
20    public int      xPosition;
21    public int      yPosition;
22    public double   fScore;
23    public double   gScore;
24    public double   hScore;
25    public String   nodeNumber;
26    public String   nodeName;
27    public boolean  visited;
28      public Position from;
29      public Position accesible[];
30      public Position nonaccesible[];
31
32    /**
33     * Constructor methods for no arguments
34     */
35    public Position ( ) {
36      xPosition = 0;
37      yPosition = 0;
38
39      nodeNumber = "";
40      nodeName   = "";
41
42      fScore = Double.MAX_VALUE;
43          gScore = Double.MAX_VALUE;
44          hScore = -1;
45
46          visited = false;
47          from    = null;
48    }
49
50    /**
51     * Constructor with coordinates set
52     * @param inputX
53     * @param inputY
54     */
55    public Position (int inputX, int inputY) {
56      xPosition = inputX;
57      yPosition = inputY;
58
59      nodeNumber = "";
60      nodeName   = "";
61
62      fScore = Double.MAX_VALUE;
63      gScore = Double.MAX_VALUE;
64      hScore = Double.MAX_VALUE;
65
66      visited = false;
67          from    = null;
68    }
69
70    /**
71     * Constructor with all position information set
```

```
72       * @param inputX
73       * @param inputY
74       * @param inputName
75       * @param inputNumber
76       */
77      public Position (int inputX, int inputY, String inputName, String inputNumber) {
78        xPosition = inputX;
79        yPosition = inputY;
80
81        nodeNumber = inputNumber;
82        nodeName   = inputName;
83
84        fScore = Double.MAX_VALUE;
85        gScore = Double.MAX_VALUE;
86        hScore = Double.MAX_VALUE;
87
88        visited = false;
89            from    = null;
90      }
91
92      /**
93       * Set coordinates
94       * @param inputX
95       * @param inputY
96       */
97      public void setCoordinates (int inputX, int inputY) {
98        xPosition  = inputX;
99        yPosition  = inputY;
100     }
101
102     /**
103      * Set position number
104      * @param inputNumber
105      */
106     public void setNumber (String inputNumber) {
107       nodeNumber  = inputNumber;
108     }
109
110     /**
111      * Set position description
112      * @param inputName
113      */
114     public void setName (String inputName) {
115       nodeName  = inputName;
116     }
117
118     /**
119      * get x coordinate
120      * @return
121      */
122     public int getX ( ) {
123       return xPosition;
124     }
125
126     /**
127      * get y coordinate
128      * @return
129      */
130     public int getY ( ) {
131       return yPosition;
132     }
133
134     /**
135      * get node numner
136      * @return
137      */
138     public String getNumber ( ) {
139       return nodeNumber;
140     }
141
142     /**
```

```
143      * Get node name
144      * @return
145      */
146     public String getName ( ) {
147       return nodeName;
148     }
149
150     /**
151      * Compares this node to another
152      * @param node
153      * @return
154      */
155     public boolean compare (Position node) {
156       if(this.xPosition == node.xPosition && this.yPosition == node.yPosition && this.
          nodeNumber.equals(node.nodeNumber) && this.nodeName.equals(node.nodeName))
157         return true;
158       return false;
159     }
160
161     /**
162      * Not Used but required???
163      */
164     public int compareTo (Object node) {
165       Position temp = (Position)node;
166           return (int)(fScore - temp.fScore);
167     }
168
169
170     /**
171      * Testing methods for the POSITION class. These methods are provided
172      * for testing and debugging purposes capable of printing variables to the log
173      */
174     public void printCoordinates ( ) {
175       Log.d("POSITION CLASS", "Coordinates: (" + xPosition + "," + yPosition + ")");
176     }
177
178     public void printNumber ( ) {
179       Log.d("POSITION CLASS", "Node Number: " + nodeNumber);
180     }
181
182     public void printName ( ) {
183       Log.d("POSITION CLASS", "Node Name: " + nodeName);
184     }
185   }
```

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/menu_search"
        android:title="@string/menu_search"
        android:onClick="displaySearchDialog"
        android:titleCondensed="@string/menu_search">
    </item>
    <item
        android:id="@+id/menu_get_directions"
        android:title="@string/menu_get_directions"
        android:onClick="displayGetDirectionsDialog"
        android:titleCondensed="@string/menu_get_directions">
    </item>
  <!--   <item
       android:id="@+id/menu_update_position"
        android:title="@string/menu_update_position"
        android:onClick="updatePosition"
        android:titleCondensed="@string/menu_update_position">
    </item>-->
    <item
        android:id="@+id/menu_exit_maps"
        android:title="@string/menu_exit_maps"
        android:onClick="exitMaps"
        android:titleCondensed="@string/menu_exit_maps">
    </item>
    <item
        android:id="@+id/menu_show_maps_help"
        android:title="@string/menu_show_maps_help"
        android:onClick="showHelp">
    </item>

</menu>
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/maps_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapActivity" >

    <edu.seaaddicts.brockbutler.maps.MapsTouchImageView
        android:id="@+id/imgv_map"
        android:layout_alignParentTop="true"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:src="@drawable/mch_maps"
        android:contentDescription="@string/test_image" />

    <TextView
        android:id="@+id/txtv_count"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/imgv_map"
        android:contentDescription="@string/temp"
        android:textColor="#fff" />


</RelativeLayout>
```

```
 1  package edu.seaaddicts.brockbutler.scheduler;
 2
 3  import java.text.SimpleDateFormat;
 4  import java.util.ArrayList;
 5  import java.util.Calendar;
 6  import java.util.Date;
 7  import java.util.Locale;
 8
 9  import android.app.Activity;
10  import android.app.DatePickerDialog;
11  import android.app.Dialog;
12  import android.os.Bundle;
13  import android.util.Log;
14  import android.view.View;
15  import android.view.View.OnClickListener;
16  import android.view.WindowManager;
17  import android.widget.ArrayAdapter;
18  import android.widget.Button;
19  import android.widget.DatePicker;
20  import android.widget.EditText;
21  import android.widget.Spinner;
22  import android.widget.TextView;
23  import edu.seaaddicts.brockbutler.R;
24  import edu.seaaddicts.brockbutler.coursemanager.Course;
25  import edu.seaaddicts.brockbutler.coursemanager.CourseHandler;
26
27  public class AddTaskActivity extends Activity {
28    private static final String TAG = "AddTaskActivity";
29    private static final int DATE_DIALOG_ID = 0;
30
31    private Button mDueDateButton;
32    private Button mSaveButton;
33    private Button mCancelButton;
34
35    private ArrayList<Course> mRegCourses;
36    private TextView mDueDateTextView;
37
38    private EditText mTaskTitle;
39    private EditText mTaskWeight;
40    private EditText mTaskBase;
41    private EditText mTaskMark;
42
43    private Spinner mCourseSpinner;
44    private Spinner mPrioritySpinner;
45
46    private CourseHandler mCourseHandle = null;
47
48    private int mYear;
49    private int mMonth;
50    private int mDay;
51
52    @Override
53    protected void onCreate(Bundle savedInstanceState) {
54      super.onCreate(savedInstanceState);
55      setContentView(R.layout.activity_add_task);
56      this.getWindow().setSoftInputMode(
57          WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);
58      init();
59    }
60
61    /*
62     * Initialize all views and sets Button OnClickListeners.
63     */
64    private void init() {
65      /*
66       * CourseHandler
67       */
68      mCourseHandle = new CourseHandler(this);
69
70      /*
71       * Buttons
```

```
72          */
73         mDueDateButton = (Button) findViewById(R.id.add_task_due_date_button);
74         mSaveButton = (Button) findViewById(R.id.add_task_save_button);
75         mCancelButton = (Button) findViewById(R.id.add_task_cancel_button);
76 
77         /*
78          * TextViews
79          */
80         mDueDateTextView = (TextView) findViewById(R.id.add_task_date_textview);
81 
82         /*
83          * Spinners
84          */
85         mPrioritySpinner = (Spinner) findViewById(R.id.add_task_priority_spinner);
86         mCourseSpinner = (Spinner) findViewById(R.id.add_task_course_spinner);
87 
88         /*
89          * EditTexts
90          */
91         mTaskTitle = (EditText) findViewById(R.id.add_task_title);
92         mTaskWeight = (EditText) findViewById(R.id.task_weight);
93         mTaskBase = (EditText) findViewById(R.id.task_base);
94         mTaskMark = (EditText) findViewById(R.id.task_mark);
95 
96         mRegCourses = mCourseHandle.getRegisteredCourses();
97         ArrayList<String> cs = new ArrayList<String>();
98 
99         /*
100         * ArrayList of Course Strings in format: SUBJECT CODE
101         */
102        for (int i = 0; i < mRegCourses.size(); i++) {
103          cs.add(mRegCourses.get(i).mSubject + " " + mRegCourses.get(i).mCode);
104        }
105 
106        /*
107         * Set the course Spinner
108         */
109        mCourseSpinner.setAdapter(new ArrayAdapter<String>(this,
110            android.R.layout.simple_spinner_dropdown_item, cs));
111 
112        /*
113         * DatePicker stuff.
114         */
115        final Calendar cal = Calendar.getInstance();
116        mYear = cal.get(Calendar.YEAR);
117        mMonth = cal.get(Calendar.MONTH);
118        mDay = cal.get(Calendar.DAY_OF_MONTH);
119 
120        /*
121         * OnClickListeners
122         */
123 
124        // Deprecated but easier than alternative.
125        mDueDateButton.setOnClickListener(new OnClickListener() {
126          @SuppressWarnings("deprecation")
127          public void onClick(View v) {
128            showDialog(DATE_DIALOG_ID);
129          }
130        });
131 
132        mSaveButton.setOnClickListener(new OnClickListener() {
133          public void onClick(View v) {
134            Task t = new Task();
135            t.mSubj = mRegCourses.get(mCourseSpinner
136                .getSelectedItemPosition()).mSubject;
137            t.mCode = mRegCourses.get(mCourseSpinner
138                .getSelectedItemPosition()).mCode;
139            t.mName = mTaskTitle.getText().toString();
140            // Lower == higher priority
141            t.mPriority = mPrioritySpinner.getSelectedItemPosition();
142            t.mDueDate = mDueDateTextView.getText().toString();
```

```
143
144           try {
145             t.mWeight = Float.parseFloat(mTaskWeight.getText()
146                 .toString());
147             t.mBase = Float.parseFloat(mTaskBase.getText().toString());
148             t.mMark = Float.parseFloat(mTaskMark.getText().toString());
149           } catch (NumberFormatException e) {
150             Log.e(TAG,
151                 "It is OK, we just had a blank field when parsing a float.");
152           }
153
154           // Get current date
155           SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd", Locale.
              CANADA);
156           Date date = new Date();
157           t.mCreationDate = dateFormat.format(date);
158           mCourseHandle.addTask(t);
159           onBackPressed();
160         }
161       });
162
163       mCancelButton.setOnClickListener(new OnClickListener() {
164         public void onClick(View v) {
165           // Do nothing.
166           onBackPressed();
167         }
168       });
169     }
170
171     /**
172      * (non-Javadoc)
173      *
174      * @see android.app.Activity#onCreateDialog(int)
175      *
176      *      Overridden method to handle multiple dialogs that may be created
177      *      from this activity.
178      */
179     @Override
180     protected Dialog onCreateDialog(int id) {
181       switch (id) {
182       case DATE_DIALOG_ID:
183         return new DatePickerDialog(this, mDatePickerListener, mYear,
184             mMonth, mDay);
185       }
186       return null;
187     }
188
189     /**
190      * Retrieves date set from DatePickerDialog and sets TextView in this
191      * activity.
192      */
193     private DatePickerDialog.OnDateSetListener mDatePickerListener = new
        DatePickerDialog.OnDateSetListener() {
194
195       public void onDateSet(DatePicker view, int selectedYear,
196           int selectedMonth, int selectedDay) {
197         mYear = selectedYear;
198         mMonth = selectedMonth + 1; // Since index starts at 0 and there is
199                         // no 0th month.
200         mDay = selectedDay;
201
202         // Set TextView in this activity.
203         if (mMonth < 10) {
204           if (mDay < 10) {
205             mDueDateTextView.setText(new StringBuilder().append(mYear)
206                 .append("/").append(0).append(mMonth).append("/")
207                 .append(0).append(mDay));
208           } else
209             mDueDateTextView.setText(new StringBuilder().append(mYear)
210                 .append("/").append(0).append(mMonth).append("/")
211                 .append(mDay));
```

```
212         } else if (mMonth > 9) {
213           if (mDay < 10) {
214             mDueDateTextView.setText(new StringBuilder().append(mYear)
215                 .append("/").append(mMonth).append("/").append(0)
216                 .append(mDay));
217           } else
218             mDueDateTextView.setText(new StringBuilder().append(mYear)
219                 .append("/").append(mMonth).append("/")
220                 .append(mDay));
221         }
222 
223       }
224     };
225 
226   }
227 
```

```
 1  package edu.seaaddicts.brockbutler.scheduler;
 2
 3  import java.text.SimpleDateFormat;
 4  import java.util.ArrayList;
 5  import java.util.Date;
 6  import java.util.Locale;
 7
 8  import android.app.Activity;
 9  import android.app.DatePickerDialog;
10  import android.app.Dialog;
11  import android.os.Bundle;
12  import android.util.Log;
13  import android.view.View;
14  import android.view.View.OnClickListener;
15  import android.view.WindowManager;
16  import android.widget.ArrayAdapter;
17  import android.widget.Button;
18  import android.widget.DatePicker;
19  import android.widget.EditText;
20  import android.widget.Spinner;
21  import android.widget.TextView;
22  import edu.seaaddicts.brockbutler.R;
23  import edu.seaaddicts.brockbutler.coursemanager.Course;
24  import edu.seaaddicts.brockbutler.coursemanager.CourseHandler;
25
26  public class ModifyTaskActivity extends Activity {
27    private static final String TAG = "ModifyTaskActivity";
28    private static final int DATE_DIALOG_ID = 0;
29
30    private Button mDueDateButton;
31    private Button mSaveButton;
32    private Button mCancelButton;
33
34    private ArrayList<Course> mRegCourses;
35    private TextView mDueDateTextView;
36
37    private EditText mTaskTitle;
38    private EditText mTaskWeight;
39    private EditText mTaskBase;
40    private EditText mTaskMark;
41
42    private Spinner mCourseSpinner;
43    private Spinner mPrioritySpinner;
44
45    private CourseHandler mCourseHandle = null;
46
47    private int mYear;
48    private int mMonth;
49    private int mDay;
50    private int mAssign;
51    private int mIsDone;
52
53    @Override
54    protected void onCreate(Bundle savedInstanceState) {
55      super.onCreate(savedInstanceState);
56      setContentView(R.layout.activity_add_task);
57      this.getWindow().setSoftInputMode(WindowManager.LayoutParams.
        SOFT_INPUT_STATE_ALWAYS_HIDDEN);
58      init();
59    }
60
61    /*
62     * Initialize all views and sets Button OnClickListeners.
63     */
64    private void init() {
65      Bundle bundle = this.getIntent().getExtras();
66      mAssign = bundle.getInt("assign");
67      mIsDone = bundle.getInt("is_done");
68      /*
69       * CourseHandler
70       */
```

```
71        mCourseHandle = new CourseHandler(this);
72
73        /*
74         * Buttons
75         */
76        mDueDateButton = (Button) findViewById(R.id.add_task_due_date_button);
77        mSaveButton = (Button) findViewById(R.id.add_task_save_button);
78        mCancelButton = (Button) findViewById(R.id.add_task_cancel_button);
79
80        /*
81         * TextViews
82         */
83        mDueDateTextView = (TextView) findViewById(R.id.add_task_date_textview);
84
85        /*
86         * Spinners
87         */
88        mPrioritySpinner = (Spinner) findViewById(R.id.add_task_priority_spinner);
89        mPrioritySpinner.setSelection(bundle.getInt("priority"));
90        mCourseSpinner = (Spinner) findViewById(R.id.add_task_course_spinner);
91
92        /*
93         * EditTexts
94         */
95        mTaskTitle = (EditText) findViewById(R.id.add_task_title);
96        mTaskTitle.setText(bundle.getString("title"));
97        mTaskWeight = (EditText) findViewById(R.id.task_weight);
98        mTaskWeight.setText(bundle.getFloat("weight")+"");
99        mTaskBase = (EditText) findViewById(R.id.task_base);
100       mTaskBase.setText(bundle.getFloat("base")+"");
101       mTaskMark = (EditText) findViewById(R.id.task_mark);
102       mTaskMark.setText(bundle.getFloat("mark")+"");
103
104       mRegCourses = mCourseHandle.getRegisteredCourses();
105       ArrayList<String> cs = new ArrayList<String>();
106
107       /*
108        * ArrayList of Course Strings in format: SUBJECT CODE
109        */
110       String course = bundle.getString("course");
111       int pos = 0;
112       for (int i = 0; i < mRegCourses.size(); i++) {
113         String c = mRegCourses.get(i).mSubject + " " + mRegCourses.get(i).mCode;
114         cs.add(c);
115         if (c.equals(course))
116           pos = i;
117       }
118
119       /*
120        * Set the course Spinner
121        */
122       mCourseSpinner.setAdapter(new ArrayAdapter<String>(this,
123           android.R.layout.simple_spinner_dropdown_item, cs));
124
125       mCourseSpinner.setSelection(pos);
126
127       /*
128        * DatePicker stuff.
129        */
130       mYear = bundle.getInt("year");
131       mMonth = bundle.getInt("month");
132       mDay = bundle.getInt("day");
133
134       /*
135        * OnClickListeners
136        */
137
138       // Deprecated but easier than alternative.
139       mDueDateButton.setOnClickListener(new OnClickListener() {
140         @SuppressWarnings("deprecation")
141         public void onClick(View v) {
```

```
142           showDialog(DATE_DIALOG_ID);
143         }
144       });
145
146       mSaveButton.setOnClickListener(new OnClickListener() {
147         public void onClick(View v) {
148           Task t = new Task();
149           t.mAssign = mAssign;
150           t.mIsDone = mIsDone;
151           t.mSubj = mRegCourses.get(mCourseSpinner
152               .getSelectedItemPosition()).mSubject;
153           t.mCode = mRegCourses.get(mCourseSpinner
154               .getSelectedItemPosition()).mCode;
155           t.mName = mTaskTitle.getText().toString();
156           // Lower == higher priority
157           t.mPriority = mPrioritySpinner.getSelectedItemPosition();
158           t.mDueDate = mDueDateTextView.getText().toString();
159
160           try {
161             t.mWeight = Float.parseFloat(mTaskWeight.getText().toString());
162             t.mBase = Float.parseFloat(mTaskBase.getText().toString());
163             t.mMark = Float.parseFloat(mTaskMark.getText().toString());
164           } catch (NumberFormatException e) {
165             Log.e(TAG,"It is OK, we just had a blank field when parsing a float.");
166           }
167
168           // Get current date
169           SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd", Locale.
              CANADA);
170           Date date = new Date();
171           t.mCreationDate = dateFormat.format(date);
172           mCourseHandle.addTask(t);
173           onBackPressed();
174         }
175       });
176
177       mCancelButton.setOnClickListener(new OnClickListener() {
178         public void onClick(View v) {
179           // Do nothing.
180           onBackPressed();
181         }
182       });
183
184       setDueDateText();
185     }
186
187     /**
188      * (non-Javadoc)
189      *
190      * @see android.app.Activity#onCreateDialog(int)
191      *
192      *      Overridden method to handle multiple dialogs that may be created
193      *      from this activity.
194      */
195     @Override
196     protected Dialog onCreateDialog(int id) {
197       switch (id) {
198       case DATE_DIALOG_ID:
199         return new DatePickerDialog(this, mDatePickerListener, mYear,
200             mMonth, mDay);
201       }
202       return null;
203     }
204
205     /**
206      * Retrieves date set from DatePickerDialog and sets TextView in this
207      * activity.
208      */
209     private DatePickerDialog.OnDateSetListener mDatePickerListener = new
        DatePickerDialog.OnDateSetListener() {
210
```

```
211       public void onDateSet(DatePicker view, int selectedYear,
212           int selectedMonth, int selectedDay) {
213         mYear = selectedYear;
214         mMonth = selectedMonth + 1; // Since index starts at 0 and there is
215                       // no 0th month.
216         mDay = selectedDay;
217         setDueDateText();
218
219       }
220     };
221
222     private void setDueDateText(){
223       // Set TextView in this activity.
224       if (mMonth < 10) {
225         if (mDay < 10) {
226           mDueDateTextView.setText(new StringBuilder().append(mYear)
227               .append("/").append(0).append(mMonth).append("/")
228               .append(0).append(mDay));
229         } else
230           mDueDateTextView.setText(new StringBuilder().append(mYear)
231               .append("/").append(0).append(mMonth).append("/")
232               .append(mDay));
233       } else if (mMonth > 9) {
234         if (mDay < 10) {
235           mDueDateTextView.setText(new StringBuilder().append(mYear)
236               .append("/").append(mMonth).append("/").append(0)
237               .append(mDay));
238         } else
239           mDueDateTextView.setText(new StringBuilder().append(mYear)
240               .append("/").append(mMonth).append("/")
241               .append(mDay));
242       }
243     }
244   }
245
```

```
 1  package edu.seaaddicts.brockbutler.scheduler;
 2
 3  import java.util.ArrayList;
 4  import java.util.Calendar;
 5
 6  import android.app.Activity;
 7  import android.app.AlertDialog;
 8  import android.content.Context;
 9  import android.content.DialogInterface;
10  import android.content.Intent;
11  import android.graphics.Paint;
12  import android.net.Uri;
13  import android.os.Bundle;
14  import android.util.Log;
15  import android.view.Menu;
16  import android.view.MenuItem;
17  import android.view.View;
18  import android.view.View.OnClickListener;
19  import android.view.ViewGroup;
20  import android.widget.AdapterView;
21  import android.widget.ArrayAdapter;
22  import android.widget.CheckBox;
23  import android.widget.CompoundButton;
24  import android.widget.CompoundButton.OnCheckedChangeListener;
25  import android.widget.ImageButton;
26  import android.widget.Button;
27  import android.widget.LinearLayout;
28  import android.widget.ListView;
29  import android.widget.TextView;
30  import android.widget.Toast;
31  import edu.seaaddicts.brockbutler.R;
32  import edu.seaaddicts.brockbutler.animation.ExpandAnimation;
33  import edu.seaaddicts.brockbutler.coursemanager.Course;
34  import edu.seaaddicts.brockbutler.coursemanager.CourseHandler;
35  import edu.seaaddicts.brockbutler.coursemanager.Offering;
36  import edu.seaaddicts.brockbutler.coursemanager.OfferingTime;
37  import edu.seaaddicts.brockbutler.help.HelpActivity;
38
39  public class SchedulerActivity extends Activity {
40    private static final String TAG = "SchedulerActivity";
41
42    private static final int VISIBLE = 0;
43    private static final int GONE = 8;
44
45    private ArrayList<Course> mRegisteredCoursesList = null;
46    private CourseHandler mCourseHandle = null;
47    private ListView mRegisteredCoursesListView = null;
48    ArrayAdapter<String> mListAdaptor;
49    View mView;
50    int mCurTaskTextViewId;
51    int mCurTaskCheckBoxId;
52    int mCurTaskNum;
53    ArrayList<Task> mCurTasks;
54
55    @Override
56    protected void onCreate(Bundle savedInstanceState) {
57      super.onCreate(savedInstanceState);
58      setContentView(R.layout.activity_scheduler);
59      mCourseHandle = new CourseHandler(this);
60      populateCoursesLayout();
61    }
62
63    @Override
64    protected void onResume() {
65      super.onResume();
66      populateCoursesLayout();
67    }
68
69    /**
70     *
71     */
```

```
72      private void populateCoursesLayout() {
73        TextView tvNoCourses = (TextView) findViewById(R.id.tv_no_courses);
74        mRegisteredCoursesListView = (ListView) findViewById(R.id.sched_list);
75        mRegisteredCoursesList = mCourseHandle.getRegisteredCourses();
76        if (mRegisteredCoursesList.size() == 0) {
77          // There are no registered courses so set message.
78          mRegisteredCoursesListView.setVisibility(GONE);
79          tvNoCourses.setVisibility(VISIBLE);
80        } else {
81          // We have registered courses so populate ListView.
82          // Creating the list adapter and populating the list
83          mListAdaptor = new CustomListAdapter(this, R.layout.sched_list_item);
84          for (int i = 0; i < mRegisteredCoursesList.size(); i++)
85            mListAdaptor.add(mRegisteredCoursesList.get(i).mSubject + " "
86                + mRegisteredCoursesList.get(i).mCode);
87          mRegisteredCoursesListView.setAdapter(mListAdaptor);
88          tvNoCourses.setVisibility(GONE);
89          mRegisteredCoursesListView.setVisibility(VISIBLE);
90          mRegisteredCoursesListView
91              .setOnItemClickListener(new AdapterView.OnItemClickListener() {
92                public void onItemClick(AdapterView<?> parent,
93                    final View view, int position, long id) {
94                  showHideToolbar(view, position);
95                }
96              });
97          registerForContextMenu(mRegisteredCoursesListView);
98        }
99      }
100
101     private void populateCourseView(View view, int position) {
102
103       mView = view;
104       // Get current course info.
105       String offering = "";
106       ArrayList<Offering> offs = mRegisteredCoursesList.get(position).mOfferings;
107       ArrayList<OfferingTime> offTimes;
108       ArrayList<Task> tasks = mRegisteredCoursesList.get(position).mTasks;
109       Log.d(TAG, "NUMBER OF TASKS: " + tasks.size());
110
111       // Set Instructor
112       ((TextView) view.findViewById(R.id.tv_prof_name))
113           .setText(mRegisteredCoursesList.get(position).mInstructor);
114       final String e= mRegisteredCoursesList.get(position).mInstructor_email;
115       ((Button) view.findViewById(R.id.prof_email_button))
116       .setOnClickListener(new OnClickListener() {
117         public void onClick(View v) {
118           sendEmail(e);
119         }
120       });
121
122       // Add Offerings registered for.
123       for (int i = 0; i < offs.size(); i++) {
124         String what = offs.get(i).mType.substring(0, 3).trim();
125         offTimes = offs.get(i).mOfferingTimes;
126         offering = "";
127
128         // Loop through OfferingTimes for each Offering to populate
129         for (int j = 0; j < offTimes.size(); j++) {
130           offering += offTimes.get(j).mDay + " "
131               + offTimes.get(j).mStartTime + " - "
132               + offTimes.get(j).mEndTime + " @ "
133               + offTimes.get(j).mLocation + "\n";
134
135         // Check for type of Offering and add as appropriate
136         if (what.equalsIgnoreCase("lec")) {
137           TextView tv = ((TextView) view
138               .findViewById(R.id.tv_lecture));
139           tv.setText(offering);
140         }
141
142         // ...a lab?
```

```
143           else if (what.equalsIgnoreCase("lab"))
144             ((TextView) view.findViewById(R.id.tv_lab))
145                 .setText(offering);
146
147           // ..a tutorial?
148           else if (what.equalsIgnoreCase("tut"))
149             ((TextView) view.findViewById(R.id.tv_tutorial))
150                 .setText(offering);
151
152           // ...a seminar?
153           else if (what.equalsIgnoreCase("sem"))
154             ((TextView) view.findViewById(R.id.tv_seminar))
155                 .setText(offering);
156         }
157       }
158
159       /*
160        * Add Tasks to view
161        */
162       if (tasks.size() > 0) {
163         mCurTasks = tasks;
164         // Set visibility for Tasks
165         view.findViewById(R.id.sched_tasks_title).setVisibility(VISIBLE);
166         view.findViewById(R.id.sched_tasks).setVisibility(VISIBLE);
167         ((LinearLayout) view.findViewById(R.id.sched_tasks))
168             .removeAllViews();
169         for (int j = 0; j < tasks.size(); j++) {
170           mCurTaskNum = j;
171           // LinearLayout for each task
172           LinearLayout ll = new LinearLayout(this);
173
174           // TextView for task info
175           TextView tv = new TextView(this);
176
177           ll.addView(tv);
178
179           // Each TextView will need an ID when added to the layout
180           mCurTaskTextViewId = 1000+j;
181           tv.setId(mCurTaskTextViewId);
182
183           // Cosmetics
184           tv.setPadding(20, 20, 30, 20);
185           tv.setLayoutParams(new LinearLayout.LayoutParams(425,
186               LinearLayout.LayoutParams.WRAP_CONTENT));
187
188           // Set the TextView
189           tv.setText(tasks.get(j).mName + "\n\tDue: " + tasks.get(j).mDueDate);
190
191           // Checkbox for Task status
192           CheckBox cb = new CheckBox(this);
193
194           // Buttons for Modify/Delete
195           ImageButton ib_mod = new ImageButton(this);
196           ImageButton ib_del = new ImageButton(this);
197           ib_mod.setFocusable(false);
198           ib_mod.setFocusableInTouchMode(false);
199           ib_mod.setBackgroundResource(android.R.drawable.ic_dialog_info);
200           final Task edit = mCurTasks.get(mCurTaskNum);
201           ib_mod.setOnClickListener(new OnClickListener() {
202
203             public void onClick(View v) {
204               modifyTask(edit);
205             }
206           });
207           ib_del.setFocusable(false);
208           ib_del.setFocusableInTouchMode(false);
209           ib_del.setBackgroundResource(android.R.drawable.ic_delete);
210           ib_del.setOnClickListener(new OnClickListener() {
211
212             public void onClick(View v) {
213               displayTaskRemoveDialog(edit);
```

```
214             }
215           });
216           ll.addView(ib_del);
217           ll.addView(ib_mod);
218
219
220           // Each TextView will need an ID when added to the layout
221           mCurTaskCheckBoxId = 1000 + j + 1;
222           cb.setId(mCurTaskCheckBoxId);
223           final Task currentTask = tasks.get(j);
224           final View fView = mView;
225           cb.setOnCheckedChangeListener(new OnCheckedChangeListener() {
226
227             public void onCheckedChanged(CompoundButton buttonView,
228                 boolean isChecked) {
229               Log.d(TAG, "mCurTaskTextViewId: " + (buttonView.getId()-1));
230               TextView tv = ((TextView) fView.findViewById(buttonView.getId()-1));
231               if (isChecked) {
232                 tv.setPaintFlags(tv.getPaintFlags()
233                     | Paint.STRIKE_THRU_TEXT_FLAG);
234                 currentTask.mIsDone = 1;
235                 mCourseHandle.addTask(currentTask);
236               } else {
237                 tv.setPaintFlags(0);
238                 currentTask.mIsDone = 0;
239                 mCourseHandle.addTask(currentTask);
240               }
241             }
242           });
243           cb.setPadding(20, 0, 0, 0);
244           cb.setFocusable(false);
245           cb.setFocusableInTouchMode(false);
246           ll.addView(cb);
247           ((LinearLayout) view.findViewById(R.id.sched_tasks)).addView(ll);
248           cb.setChecked(currentTask.mIsDone != 0);  //HUZZAH!!!
249         }
250       }
251
252       float mark  = mCourseHandle.getMark(mRegisteredCoursesList.get(position));
253       float base  = mCourseHandle.getBase(mRegisteredCoursesList.get(position));
254       float total=0;
255       if(base != 0){
256         total= (mark/base)*100;
257       }
258
259       String grade= ""+(int)mark+"/"+(int)base+" = "+ (int)total+"%";
260       ((TextView) view.findViewById(R.id.course_grade_grade))
261       .setText(grade);
262
263       /*
264        * Hide class type if none available
265        */
266       if (((TextView) view.findViewById(R.id.tv_lecture)).getText()
267           .toString().equalsIgnoreCase("none"))
268         view.findViewById(R.id.row_lec).setVisibility(GONE);
269       if (((TextView) view.findViewById(R.id.tv_lab)).getText().toString()
270           .equalsIgnoreCase("none"))
271         view.findViewById(R.id.row_lab).setVisibility(GONE);
272       if (((TextView) view.findViewById(R.id.tv_tutorial)).getText()
273           .toString().equalsIgnoreCase("none"))
274         view.findViewById(R.id.row_tut).setVisibility(GONE);
275       if (((TextView) view.findViewById(R.id.tv_seminar)).getText()
276           .toString().equalsIgnoreCase("none"))
277         view.findViewById(R.id.row_sem).setVisibility(GONE);
278     }
279
280     private void displayTaskRemoveDialog(final Task t) {
281       AlertDialog.Builder editalert = new AlertDialog.Builder(this);
282
283       editalert.setTitle("Delete Task?");
284       editalert.setMessage("Are you sure you wish to delete this task?");
```

```
285
286       editalert.setPositiveButton("Yes",
287           new DialogInterface.OnClickListener() {
288             public void onClick(DialogInterface dialog, int whichButton) {
289               mCourseHandle.removeTask(t);
290               populateCoursesLayout();
291             }
292           });
293       editalert.setNegativeButton("No",
294           new DialogInterface.OnClickListener() {
295             public void onClick(DialogInterface dialog, int whichButton) {
296               // do nothing
297             }
298           });
299
300       editalert.show();
301     }
302
303     private void showHideToolbar(View view, int position) {
304       View toolbar = view.findViewById(R.id.sched_toolbar);
305       populateCourseView(view, position);
306
307       // Creating the expand animation for the item
308       ExpandAnimation expandAni = new ExpandAnimation(toolbar,
309           ExpandAnimation.ANIMATE_SHORT);
310
311       // Start the animation on the toolbar
312       toolbar.startAnimation(expandAni);
313
314     }
315
316     public void showHelp(MenuItem item) {
317       Intent intent = new Intent(SchedulerActivity.this, HelpActivity.class);
318       Bundle bundle = new Bundle();
319       bundle.putString("activity", "scheduler");
320       intent.putExtras(bundle);
321       startActivity(intent);
322     }
323
324     public void modifyTask(Task task){
325       Intent intent = new Intent(SchedulerActivity.this, ModifyTaskActivity.class);
326       Bundle bundle = new Bundle();
327       bundle.putInt("assign", task.mAssign);
328       bundle.putString("title",task.mName);
329       bundle.putFloat("weight",task.mWeight);
330       bundle.putFloat("base",task.mBase);
331       bundle.putFloat("mark",task.mMark);
332       bundle.putString("course",task.mSubj + " " + task.mCode);
333       bundle.putInt("priority",task.mPriority);
334       bundle.putInt("is_done",task.mIsDone);
335       if (task.mDueDate.equals(getResources().getString(R.string.activity_add_task_date
          ))){
336         final Calendar cal = Calendar.getInstance();
337         bundle.putInt("year",cal.get(Calendar.YEAR));
338         bundle.putInt("month",cal.get(Calendar.MONTH));
339         bundle.putInt("day",cal.get(Calendar.DAY_OF_MONTH));
340       }
341       else {
342         int year = Integer.parseInt(task.mDueDate.substring(0,4));
343         int month = Integer.parseInt(task.mDueDate.substring(5,7));
344         int day = Integer.parseInt(task.mDueDate.substring(8));
345         bundle.putInt("year",year);
346         bundle.putInt("month",month);
347         bundle.putInt("day",day);
348       }
349       intent.putExtras(bundle);
350       startActivity(intent);
351     }
352
353     public void removeTask(Task task){
354       //REMOVE THE TASK
```

```
355     }
356
357     @Override
358     public boolean onCreateOptionsMenu(Menu menu) {
359       // Inflate the menu; this adds items to the action bar if it is present.
360       getMenuInflater().inflate(R.menu.activity_scheduler, menu);
361       return true;
362     }
363
364     private void sendEmail(String instr_email) {
365       Intent i = new Intent(Intent.ACTION_SENDTO);
366       i.setType("message/rfc822");
367       i.setData(Uri.parse("mailto:"+instr_email));
368       try {
369         startActivity(Intent.createChooser(i, "Send mail..."));
370       } catch (android.content.ActivityNotFoundException ex) {
371         Toast.makeText(SchedulerActivity.this,
372             "There are no email clients installed.", Toast.LENGTH_SHORT)
373             .show();
374       }
375     }
376
377     public void addTask(MenuItem item) {
378       Intent i = new Intent(SchedulerActivity.this, AddTaskActivity.class);
379       startActivity(i);
380     }
381
382     /**
383      * A simple implementation of list adapter.
384      */
385     class CustomListAdapter extends ArrayAdapter<String> {
386
387       public CustomListAdapter(Context context, int textViewResourceId) {
388         super(context, textViewResourceId);
389       }
390
391       @Override
392       public View getView(int position, View convertView, ViewGroup parent) {
393
394         if (convertView == null) {
395           convertView = getLayoutInflater().inflate(
396               R.layout.sched_list_item, null);
397         }
398
399         ((TextView) convertView.findViewById(R.id.sched_list_item_title))
400             .setText(getItem(position));
401
402         // Resets the toolbar to be closed
403         View toolbar = convertView.findViewById(R.id.sched_toolbar);
404         ((LinearLayout.LayoutParams) toolbar.getLayoutParams()).bottomMargin = -(toolbar
405             .getHeight());
406         toolbar.setVisibility(View.GONE);
407
408         return convertView;
409       }
410     }
411
412   }
413
```

```
 1  /**
 2   * Tasks.java
 3   * Brock Butler
 4   * A wrapper class for tasks information
 5   * Created by James Grisdale on 2013-02-24
 6   * Copyright (c) 2013 Sea Addicts. All rights reserved.
 7  **/
 8
 9  package edu.seaaddicts.brockbutler.scheduler;
10
11  import java.util.ArrayList;
12
13  import edu.seaaddicts.brockbutler.contacts.Contact;
14
15  public class Task {
16    public String mSubj; //course name
17    public String mCode; //course code
18    public boolean mIsPastDue; //is the task late
19    public float mMark; //mark on task
20    public float mBase; //base mark of task
21    public float mWeight; //weight of the mark
22    public int mAssign;  //assignment number
23    public String mName; //name of task
24    public String mCreationDate; //creation date
25    public String mDueDate; //due date
26    public int mPriority; //priority of task
27    public int mIsDone; //is the task complete
28    public ArrayList<Contact> mContacts; //task contacts
29
30    //contructor - initializes the contacts arraylist
31    public Task(){
32      mContacts = new ArrayList<Contact>();
33    }
34
35    //isPastDue - calculates if the task is past the due date
36    public boolean isPastDueDate(){
37      return true;
38    }
39
40  }
41
```

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10sp"
    android:orientation="vertical"
    tools:context=".AddTaskActivity" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

            <TextView
                android:layout_width="85sp"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:text="@string/activity_add_task_course" />

            <Spinner
                android:id="@+id/add_task_course_spinner"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_marginLeft="25sp"
                android:prompt="@string/task_priority_prompt" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

            <TextView
                android:layout_width="85sp"
                android:layout_height="wrap_content"
                android:text="@string/activity_add_task_task_title" />

            <EditText
                android:id="@+id/add_task_title"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="25sp"
                android:hint="@string/activity_add_task_hint_task_title"
                android:inputType="text" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

            <TextView
                android:layout_width="85sp"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:text="@string/activity_add_task_priority" />

            <Spinner
                android:id="@+id/add_task_priority_spinner"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_marginLeft="25sp"
                android:entries="@array/task_priority_array"
                android:prompt="@string/task_priority_prompt" />
```

```
72          </LinearLayout>
73
74          <LinearLayout
75              android:layout_width="fill_parent"
76              android:layout_height="wrap_content"
77              android:orientation="horizontal" >
78
79              <TextView
80                  android:layout_width="85sp"
81                  android:layout_height="wrap_content"
82                  android:text="@string/activity_add_task_weight" />
83
84              <EditText
85                  android:id="@+id/task_weight"
86                  android:layout_width="fill_parent"
87                  android:layout_height="wrap_content"
88                  android:layout_marginLeft="25sp"
89                  android:hint="@string/activity_add_task_hint_weight"
90                  android:inputType="numberDecimal" />
91          </LinearLayout>
92
93          <LinearLayout
94              android:layout_width="fill_parent"
95              android:layout_height="wrap_content"
96              android:orientation="horizontal" >
97
98              <TextView
99                  android:layout_width="85sp"
100                 android:layout_height="wrap_content"
101                 android:text="@string/activity_add_task_base" />
102
103             <EditText
104                 android:id="@+id/task_base"
105                 android:layout_width="fill_parent"
106                 android:layout_height="wrap_content"
107                 android:layout_marginLeft="25sp"
108                 android:hint="@string/activity_add_task_hint_base"
109                 android:inputType="numberDecimal" />
110         </LinearLayout>
111
112         <LinearLayout
113             android:layout_width="fill_parent"
114             android:layout_height="wrap_content"
115             android:orientation="horizontal" >
116
117             <TextView
118                 android:layout_width="85sp"
119                 android:layout_height="wrap_content"
120                 android:text="@string/activity_add_task_mark" />
121
122             <EditText
123                 android:id="@+id/task_mark"
124                 android:layout_width="fill_parent"
125                 android:layout_height="wrap_content"
126                 android:layout_marginLeft="25sp"
127                 android:hint="@string/activity_add_task_hint_mark"
128                 android:inputType="numberDecimal" />
129         </LinearLayout>
130
131         <LinearLayout
132             android:layout_width="fill_parent"
133             android:layout_height="wrap_content"
134             android:layout_marginTop="10sp"
135             android:gravity="center_horizontal"
136             android:orientation="vertical" >
137
138             <Button
139                 android:id="@+id/add_task_due_date_button"
140                 android:layout_width="150sp"
141                 android:layout_height="wrap_content"
142                 android:gravity="center"
```

```
143                     android:text="@string/activity_add_task_due_date" />
144
145                 <TextView
146                     android:id="@+id/add_task_date_textview"
147                     android:layout_width="wrap_content"
148                     android:layout_height="wrap_content"
149                     android:layout_marginTop="10sp"
150                     android:text="@string/activity_add_task_date"
151                     android:textSize="20sp" />
152             </LinearLayout>
153
154             <LinearLayout
155                 android:layout_width="fill_parent"
156                 android:layout_height="wrap_content"
157                 android:layout_margin="10sp"
158                 android:gravity="center_horizontal"
159                 android:orientation="horizontal" >
160
161                 <Button
162                     android:id="@+id/add_task_save_button"
163                     android:layout_width="100sp"
164                     android:layout_height="wrap_content"
165                     android:text="@string/button_save" />
166
167                 <Button
168                     android:id="@+id/add_task_cancel_button"
169                     android:layout_width="100sp"
170                     android:layout_height="wrap_content"
171                     android:layout_marginLeft="20sp"
172                     android:text="@string/button_cancel" />
173             </LinearLayout>
174         </LinearLayout>
175
176     </ScrollView>
```

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="10sp"
    android:orientation="vertical"
    tools:context=".ModifyTaskActivity" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

            <TextView
                android:layout_width="85sp"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:text="@string/activity_add_task_course" />

            <Spinner
                android:id="@+id/add_task_course_spinner"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_marginLeft="25sp"
                android:prompt="@string/task_priority_prompt" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

            <TextView
                android:layout_width="85sp"
                android:layout_height="wrap_content"
                android:text="@string/activity_add_task_task_title" />

            <EditText
                android:id="@+id/add_task_title"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="25sp"
                android:hint="@string/activity_add_task_hint_task_title"
                android:inputType="text" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

            <TextView
                android:layout_width="85sp"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:text="@string/activity_add_task_priority" />

            <Spinner
                android:id="@+id/add_task_priority_spinner"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_marginLeft="25sp"
                android:entries="@array/task_priority_array"
                android:prompt="@string/task_priority_prompt" />
```

```
72          </LinearLayout>
73
74          <LinearLayout
75              android:layout_width="fill_parent"
76              android:layout_height="wrap_content"
77              android:orientation="horizontal" >
78
79              <TextView
80                  android:layout_width="85sp"
81                  android:layout_height="wrap_content"
82                  android:text="@string/activity_add_task_weight" />
83
84              <EditText
85                  android:id="@+id/task_weight"
86                  android:layout_width="fill_parent"
87                  android:layout_height="wrap_content"
88                  android:layout_marginLeft="25sp"
89                  android:hint="@string/activity_add_task_hint_weight"
90                  android:inputType="numberDecimal" />
91          </LinearLayout>
92
93          <LinearLayout
94              android:layout_width="fill_parent"
95              android:layout_height="wrap_content"
96              android:orientation="horizontal" >
97
98              <TextView
99                  android:layout_width="85sp"
100                 android:layout_height="wrap_content"
101                 android:text="@string/activity_add_task_base" />
102
103             <EditText
104                 android:id="@+id/task_base"
105                 android:layout_width="fill_parent"
106                 android:layout_height="wrap_content"
107                 android:layout_marginLeft="25sp"
108                 android:hint="@string/activity_add_task_hint_base"
109                 android:inputType="numberDecimal" />
110         </LinearLayout>
111
112         <LinearLayout
113             android:layout_width="fill_parent"
114             android:layout_height="wrap_content"
115             android:orientation="horizontal" >
116
117             <TextView
118                 android:layout_width="85sp"
119                 android:layout_height="wrap_content"
120                 android:text="@string/activity_add_task_mark" />
121
122             <EditText
123                 android:id="@+id/task_mark"
124                 android:layout_width="fill_parent"
125                 android:layout_height="wrap_content"
126                 android:layout_marginLeft="25sp"
127                 android:hint="@string/activity_add_task_hint_mark"
128                 android:inputType="numberDecimal" />
129         </LinearLayout>
130
131         <LinearLayout
132             android:layout_width="fill_parent"
133             android:layout_height="wrap_content"
134             android:layout_marginTop="10sp"
135             android:gravity="center_horizontal"
136             android:orientation="vertical" >
137
138             <Button
139                 android:id="@+id/add_task_due_date_button"
140                 android:layout_width="150sp"
141                 android:layout_height="wrap_content"
142                 android:gravity="center"
```

```
143                     android:text="@string/activity_add_task_due_date" />
144
145                 <TextView
146                     android:id="@+id/add_task_date_textview"
147                     android:layout_width="wrap_content"
148                     android:layout_height="wrap_content"
149                     android:layout_marginTop="10sp"
150                     android:text="@string/activity_add_task_date"
151                     android:textSize="20sp" />
152             </LinearLayout>
153
154             <LinearLayout
155                 android:layout_width="fill_parent"
156                 android:layout_height="wrap_content"
157                 android:layout_margin="10sp"
158                 android:gravity="center_horizontal"
159                 android:orientation="horizontal" >
160
161                 <Button
162                     android:id="@+id/add_task_save_button"
163                     android:layout_width="100sp"
164                     android:layout_height="wrap_content"
165                     android:text="@string/button_save" />
166
167                 <Button
168                     android:id="@+id/add_task_cancel_button"
169                     android:layout_width="100sp"
170                     android:layout_height="wrap_content"
171                     android:layout_marginLeft="20sp"
172                     android:text="@string/button_cancel" />
173             </LinearLayout>
174         </LinearLayout>
175
176     </ScrollView>
177
```

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/menu_add_task"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:onClick="addTask"
        android:title="@string/menu_add_task"/>
<!--    <item
        android:id="@+id/menu_expand_all"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/scheduler_menu_showhide_completed"/> -->
<!--     <item -->
<!--         android:id="@+id/menu_collapse_all" -->
<!--         android:orderInCategory="100" -->
<!--         android:showAsAction="never" -->
<!--         android:title="@string/scheduler_menu_show_completed"/> -->
    <item
        android:id="@+id/menu_show_scheduler_help"
         android:orderInCategory="100"
         android:showAsAction="never"
        android:onClick="showHelp"
        android:title="@string/menu_show_scheduler_help"/>
</menu>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_scheduler"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SchedulerActivity" >

    <ListView
        android:id="@+id/sched_list"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:divider="@drawable/grad_course_title" >
    </ListView>

    <TextView
        android:id="@+id/tv_no_courses"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:text="@string/sched_no_courses"
        android:textColor="#ddd"
        android:textStyle="italic"
        android:textSize="20sp"
        android:gravity="center"
        android:visibility="gone" />

</LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/sched_list_item"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_marginBottom="10dip"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/sched_list_item_title"
        android:layout_width="fill_parent"
        android:layout_height="60sp"
        android:layout_marginBottom="5dip"
        android:background="@drawable/grad_course_title"
        android:focusable="false"
        android:focusableInTouchMode="false"
        android:padding="10dip"
        android:textSize="30sp"
        android:textStyle="bold" />

    <!-- Toolbar -->

    <LinearLayout
        android:id="@+id/sched_toolbar"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="10dip"
        android:visibility="gone" >

        <include
            android:id="@+id/row_prof"
            layout="@layout/row_prof_simple" />

        <include
            android:id="@+id/row_lec"
            layout="@layout/row_task_lecture" />

        <include
            android:id="@+id/row_lab"
            layout="@layout/row_task_lab" />

        <include
            android:id="@+id/row_tut"
            layout="@layout/row_task_tutorial" />

        <include
            android:id="@+id/row_sem"
            layout="@layout/row_task_seminar" />

        <include
            android:id="@+id/row_grade"
            layout="@layout/row_grade" />

        <TextView
            android:id="@+id/sched_tasks_title"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginLeft="20dip"
            android:background="@drawable/grad_task_title"
            android:paddingLeft="5sp"
            android:paddingRight="10sp"
            android:paddingTop="2sp"
            android:text="@string/title_tasks"
            android:textSize="16sp"
            android:textStyle="bold"
            android:typeface="sans"
            android:visibility="gone" />

        <LinearLayout
```

```
72              android:id="@+id/sched_tasks"
73              android:layout_width="fill_parent"
74              android:layout_height="wrap_content"
75              android:layout_marginLeft="10dip"
76              android:orientation="vertical"
77              android:paddingLeft="10sp"
78              android:paddingRight="10sp"
79              android:visibility="gone" >
80          </LinearLayout>
81      </LinearLayout>
82
83  </LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">BrockButler</string>

    <!-- AddTaskActivity -->
    <string name="activity_add_task">Add Task</string>
    <string name="activity_add_task_task_title">Title</string>
    <string name="activity_add_task_hint_task_title">Task title</string>
    <string name="activity_add_task_course">Course</string>
    <string name="activity_add_task_priority">Priority</string>
    <string name="activity_add_task_weight">Weight</string>
    <string name="activity_add_task_hint_weight">0 for no mark</string>
    <string name="activity_add_task_description">Description</string>
    <string name="activity_add_task_hint_description">Add description</string>
    <string name="activity_add_task_base">Base</string>
    <string name="activity_add_task_hint_base">Enter base mark</string>
    <string name="activity_add_task_mark">Mark</string>
    <string name="activity_add_task_hint_mark">Awarded mark</string>
    <string name="activity_add_task_due_date">Edit Due Date</string>
    <string name="activity_add_task_contact">Add Contact</string>
    <string name="activity_add_task_date">No due date set</string>
    <string name="activity_add_task_contacts">No contacts set</string>

    <!-- CourseManagerActivity -->
    <string name="activity_coursemanager">Course Manager</string>
    <string name="add_course_remove">Remove Course</string>
    <string name="add_course_update_master_list">Update Master List</string>
    <string name="loading_courses_registrar">Updating Course Database</string>
    <string name="no_courses">No registered courses.\n\nMenu>Add Course</string>
    <string-array name="course_manager_context_menu">
        <item>Modify</item>
        <item>Remove</item>
    </string-array>
    <string name="loading_courses_registrar_msg">The course database is updating
    from the Brock Registrar.\n\nThis may take a few moments.</string>
    <string name="menu_show_coursemanager_help">Show Help</string>
    <string name="course_grade">Course Grade</string>

    <!-- AddContactActivity -->
    <string name="activity_add_contact">Add Contact</string>
    <string name="activity_add_contact_fname">First Name</string>
    <string name="activity_add_contact_hint_fname">Enter first name</string>
    <string name="activity_add_contact_lname">Last Name</string>
    <string name="activity_add_contact_hint_lname">Enter last name</string>
    <string name="activity_add_contact_email">Email</string>
    <string name="activity_add_contact_hint_email">Enter email address</string>
    <string name="activity_add_contact_courses">Courses</string>
    <string name="activity_add_contact_add_courses">Add Course</string>
    <string name="activity_add_contact_remove">Remove</string>

    <!-- AddCourseActivity -->
    <string name="title_add_course">Add Course</string>
    <string name="add_course_subject">Subject</string>
    <string name="add_course_add">Add Course</string>
    <string name="add_course_code">Code</string>
    <string name="add_course_add_lecs">Lectures</string>
    <string name="add_course_add_labs">Labs</string>
    <string name="add_course_add_sems">Seminars</string>
    <string name="add_course_add_tuts">Tutorials</string>

    <!-- Professors -->
    <string name="prof_name">Professor Name</string>
    <string name="prof_office">Office</string>
    <string name="prof_email">Email</string>

    <!-- SchedulerActivity -->
    <string name="menu_add_task">Add Task</string>
    <string name="menu_remove_task">Remove Task</string>
    <string name="menu_modify_task">Modify Task</string>
    <string name="menu_expand_collapse">Expand/Collapse All</string>
```

```
    <string name="sched_no_courses">No registered courses.\n\nGo to Course Manager to add courses.</string>
     <string name="menu_show_scheduler_help">Show Help</string>
    <string name="scheduler_menu_showhide_completed">Show//Hide Completed Tasks</string>

    <!-- row_task_* -->
    <string name="title_tasks">Tasks</string>
    <string name="title_lecture">Lectures</string>
    <string name="title_lab">Labs</string>
    <string name="title_tutorial">Tutorials</string>
    <string name="title_seminar">Seminar</string>
    <string name="title_assignment">Assignments</string>
    <string name="title_test">Tests</string>

    <!-- MapsActivity -->
    <string name="menu_search">Search for Location</string>
    <string name="menu_get_directions">Get Directions</string>
    <string name="menu_update_position">Update Position</string>
    <string name="menu_exit_maps">Exit Maps</string>
    <string name="menu_show_maps_help">Show Help</string>

    <!-- TourActivity -->
    <string name="menu_go_back">Go Back</string>
    <string name="menu_turn_around">Turn Around</string>
    <string name="menu_teleport">Teleport!</string>
    <string name="menu_end_tour">End Tour</string>
    <string name="a_block">A Block</string>
    <string name="b_block">B Block</string>
    <string name="c_block">C Block</string>
    <string name="d_block">D Block</string>
    <string name="e_block">E Block</string>
    <string name="f_block">F Block</string>
    <string name="g_block">G Block</string>
    <string name="h_block">H Block</string>
    <string name="j_block">J Block</string>
    <string name="go_left">Go Left</string>
    <string name="go_forward">Go Forward</string>
    <string name="go_right">Go Right</string>
    <string name="menu_settings">Settings</string>

    <!-- LoginActivity -->
    <string name="title_activity_login">Login Activity</string>
    <string name="login_email">Login</string>
    <string name="login_password">Password</string>
    <string name="login_skip">Skip</string>
    <string name="login_remember">Remember me</string>
    <string name="login_skip_msg">Always skip login</string>
    <string name="login">Login</string>
    <string name="temp">Logo Here</string>
    <string name="msg_skip">By logging in BrockButler attempts to automatically update your timetable, grades, and other information from my.brocku.\n\nDo you still want to skip the login process?</string>

    <!-- MainActivity -->
    <string name="title_activity_main">Brock Butler</string>
    <string name="title_activity_map">Brock Maps</string>
    <string name="test_image">This is a test image.</string>
    <string name="title_scheduler">Scheduler</string>
    <string name="title_maps">Maps</string>
    <string name="title_contacts">Contacts</string>
    <string name="title_tour">Tour</string>
    <string name="title_help">User Manual</string>
    <string name="title_activity_help">User Manual</string>
    <string name="title_activity_scheduler">Scheduler</string>
    <string name="title_activity_contacts">Contacts</string>
    <string name="title_activity_tour">Brock Butler Tour</string>
    <string name="title_activity_add_task">Add Task</string>
    <string name="title_modify_task">Modify Task</string>
    <string name="title_modify_course">Modify Course</string>
    <string name="task_priority_prompt">Task Priority</string>
```

```
138
139         <!-- Miscellaneous -->
140         <string name="contentdesc">Content Description</string>
141         <string name="button_save">Save</string>
142         <string name="button_cancel">Cancel</string>
143
144         <string-array name="task_priority_array">
145             <item>High</item>
146             <item>Normal</item>
147             <item>Low</item>
148         </string-array>
149
150         <string-array name="empty">
151             <item>None</item>
152         </string-array>
153
154         <string name="none">None</string>
155
156     </resources>
```