```
 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
```